GeoNode Documentation

Release 2.0

GeoNode Development Team

February 14, 2017

Welcome to GeoNode's Documentation.

GeoNode is an Open Source, Content Management System (CMS) for geospatial data. It is a web-based application and platform for developing geospatial information systems (GIS) and for deploying spatial data infrastructures (SDI).

CHAPTER 1

First Steps

The following sections give an overview of GeoNode from different perspectives, they are targeted at a non-technical audience and the quick installation guide at people who just want to get it installed and will come back later to the complete documentation.

- About GeoNode
- `Quick Installation Guide`
- GeoNode Users Quickstart Manual

CHAPTER 2

How To Use The Documentation

The documentation is geared toward three distinct types of users:

1. **Users**: Are people who log into a GeoNode website and use its functionality.
2. **Administrators**: Are people who install and deploy GeoNode websites in production for their Users.
3. **Developers**: Are people who write code to add functionality, integrate with other systems, fix bugs, and potentially help an Administrator setup a server and deploy a GeoNode instance for production.

The documentation is divided into three sections:

1. Tutorials: Step-by-step instructions in workshop format that help a user to accomplish a set of tasks.
2. Reference: Architecture, component information, API descriptions etc.
3. Organizational: About the project, how to contribute, links, resources, other info.

CHAPTER 3

Table of contents

3.1 Tutorials

The Tutorials section contains step-by-step workshops that are oriented around performing particular sets of tasks, like adding data or publishing maps, setting up and maintaining a server, or setting up a project to extend from GeoNode. These tutorials are written in a workshop like format and are broken into three groups: Users, Administrators and Developers.

3.1.1 Tutorials

The tutorials are based around performing tasks, like adding data or publishing maps. The tutorials are written in a workshop like format and are broken into three groups *Users*, *Administrators* and *Developers*.

Users Workshop In the user workshop you will learn how to create an account on GeoNode, add layers and maps to your account as well as publishing those.

Administrators Workshop The administrator workshop will guide you through the installation and configuration of GeoNode, as well as explore further possibilities with GeoNode.

Developers Workshop The developers workshop will show you how to set up your own GeNode project, start developing, customizing, integrating, bug fixing etc.

Users Workshop

Welcome to the GeoNode Users Workshop! This workshop will teach how to use the GeoNode software application.

Introduction An introduction to GeoNode, what it is, what it does, including a brief tour of the application.

GeoNode Quickstart Click to get started with the GeoNode quickstart

Accounts and users Create an account in GeoNode and interact with other users

Managing layers Create and manage GeoNode layers

Managing maps Create and manage a GeoNode map

Using GeoNode with other applications Learn how to integrate GeoNode with other applications and systems.

Spatial Processing with GeoNode Learn how to use the data in your GeoNode to do Spatial data processing with various tools.

Introduction

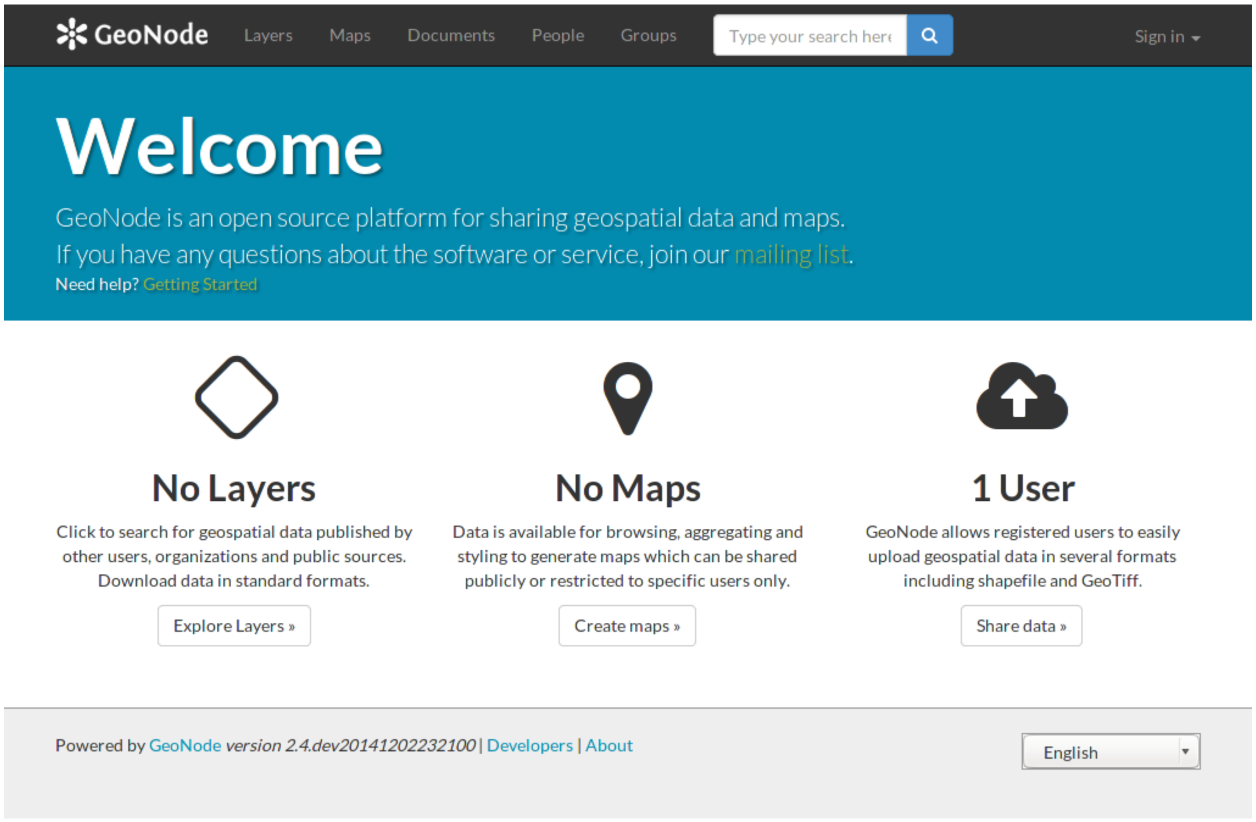
This section will give a brief introduction to GeoNode and tour its web-based interface.

A tour of GeoNode In order to get started, let's look at the GeoNode interface and get a feel for how to navigate around it.

The GeoNode web interface is the primary method of interacting with GeoNode as a user. From this interface, one can view and modify existing spatial layers and maps, as well as find information on other GeoNode users.

Without being logged in, you are limited to read-only access of public layers.

1. Navigate to your GeoNode instance, available here:



Welcome

GeoNode is an open source platform for sharing geospatial data and maps.
If you have any questions about the software or service, join our mailing list.
Need help? Getting Started

No Layers

Click to search for geospatial data published by other users, organizations and public sources. Download data in standard formats.

Explore Layers »

No Maps

Data is available for browsing, aggregating and styling to generate maps which can be shared publicly or restricted to specific users only.

Create maps »

1 User

GeoNode allows registered users to easily upload geospatial data in several formats including shapefile and GeoTiff.

Share data »

Powered by GeoNode *version 2.4.dev20141202232100* | Developers | About

English

Fig. 3.1: *Welcome page*

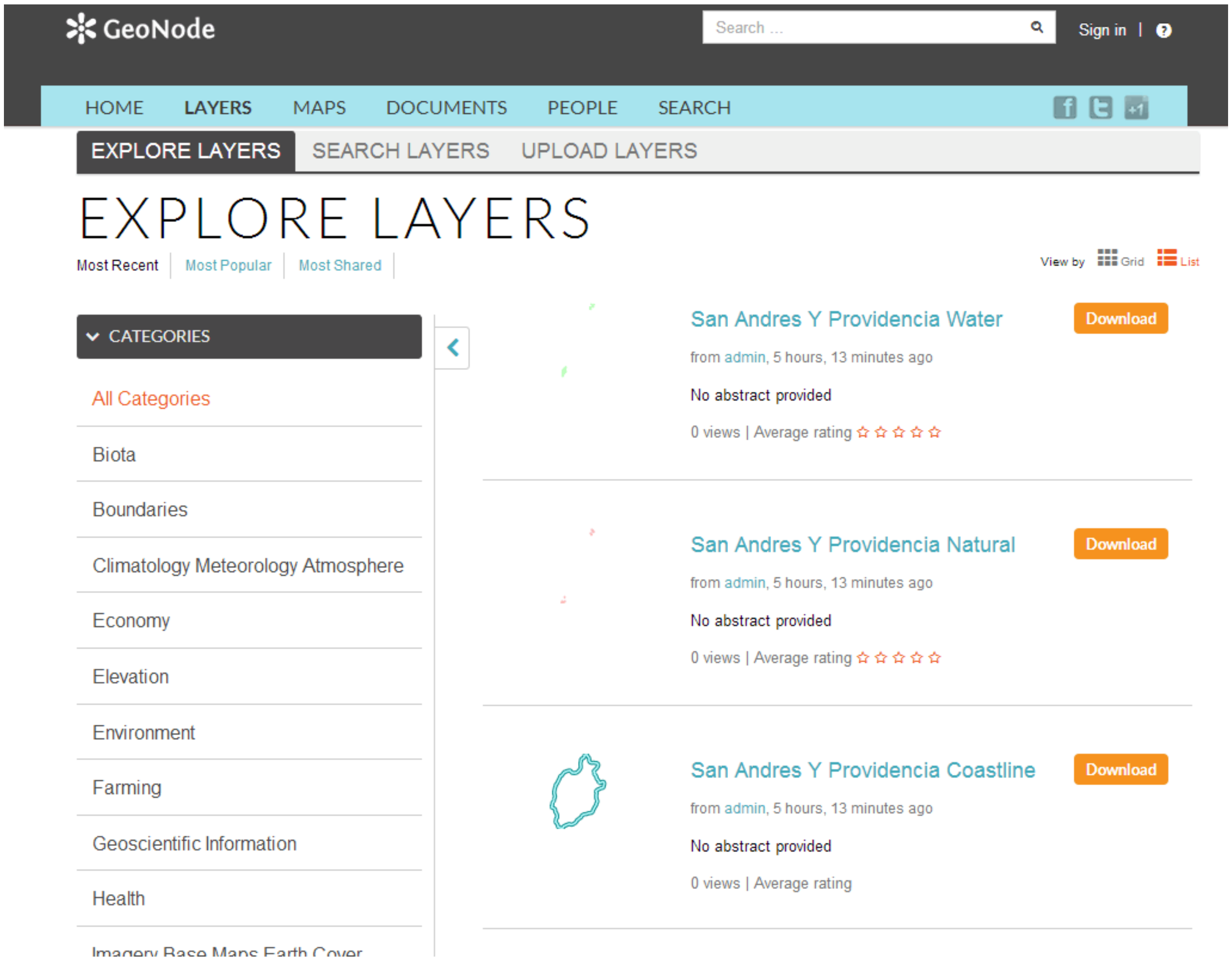
This page shows a variety of information about the current GeoNode instance. At the top of the page is a toolbar showing quick links to view *layers*, *maps*, documents (metadata), people ADD LINK, and a search field. Below this is a listing of recently updated layers, including abstract, owner, rating, and download button (if available).

2. Click the *Layers* link in the toolbar to go to the *Explore Layers* page.

   This page shows all layers known to GeoNode, available in either List or Grid viewing. Layers can be sorted by *Most Recent*, *Most Popular*, or *Most Shared*. Also available are a list of categories, with which layers can be connected with.

3. Find a layer and click on its name.

4. A layer viewing page will display, with the layer itself superimposed on a hosted base layer (in this case MapQuest OpenStreetMap). Explore this page, noting the various options available to you.



EXPLORE LAYERS

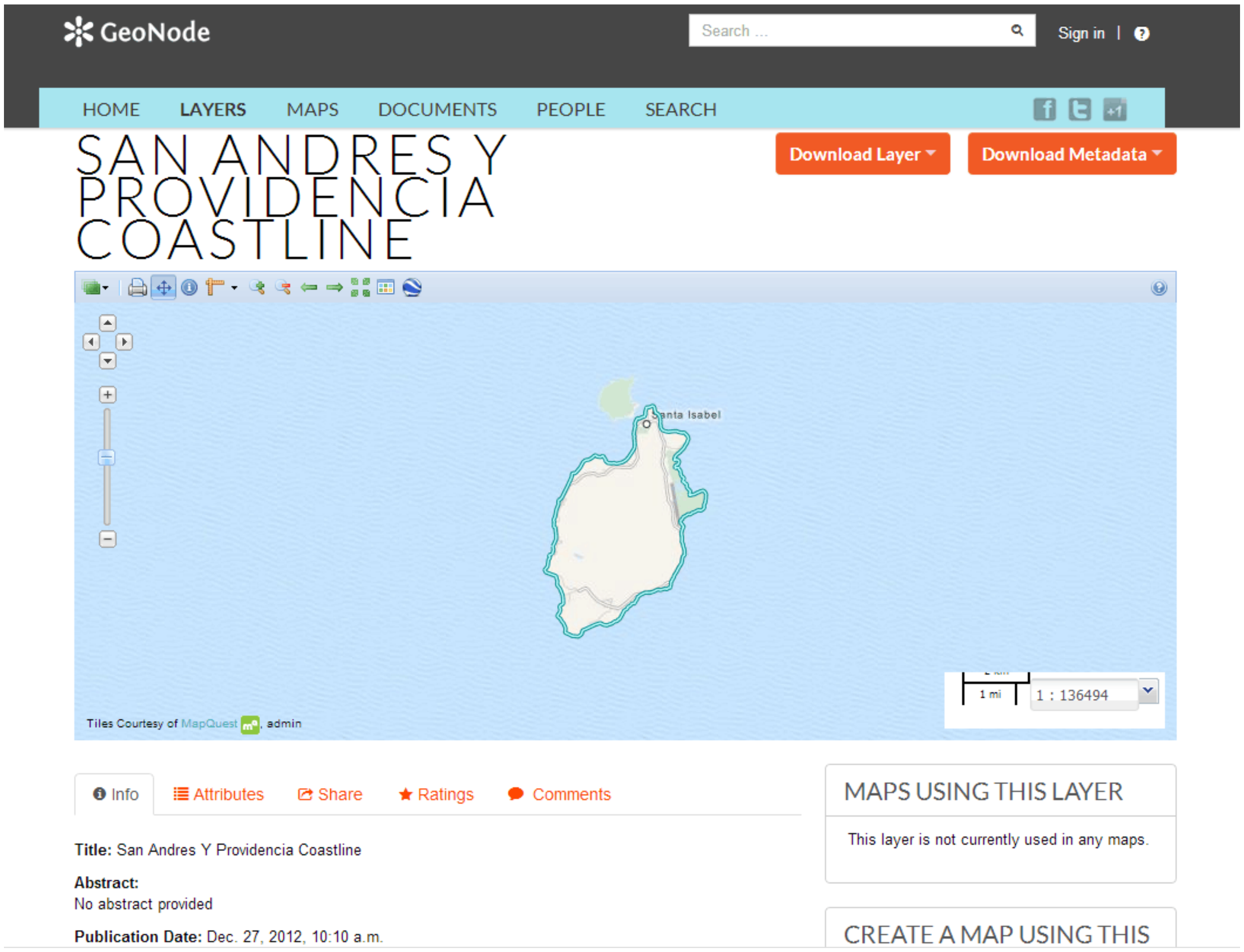
All Categories
Biota
Boundaries
Climatology Meteorology Atmosphere
Economy
Elevation
Environment
Farming
Geoscientific Information
Health
Imagery Base Maps Earth Cover

San Andres Y Providencia Water
from admin, 5 hours, 13 minutes ago
No abstract provided
0 views | Average rating

San Andres Y Providencia Natural
from admin, 5 hours, 13 minutes ago
No abstract provided
0 views | Average rating

San Andres Y Providencia Coastline
from admin, 5 hours, 13 minutes ago
No abstract provided
0 views | Average rating

Fig. 3.2: *Explore Layers page*



SAN ANDRES Y PROVIDENCIA COASTLINE

Title: San Andres Y Providencia Coastline

Abstract:
No abstract provided

Publication Date: Dec. 27, 2012, 10:10 a.m.

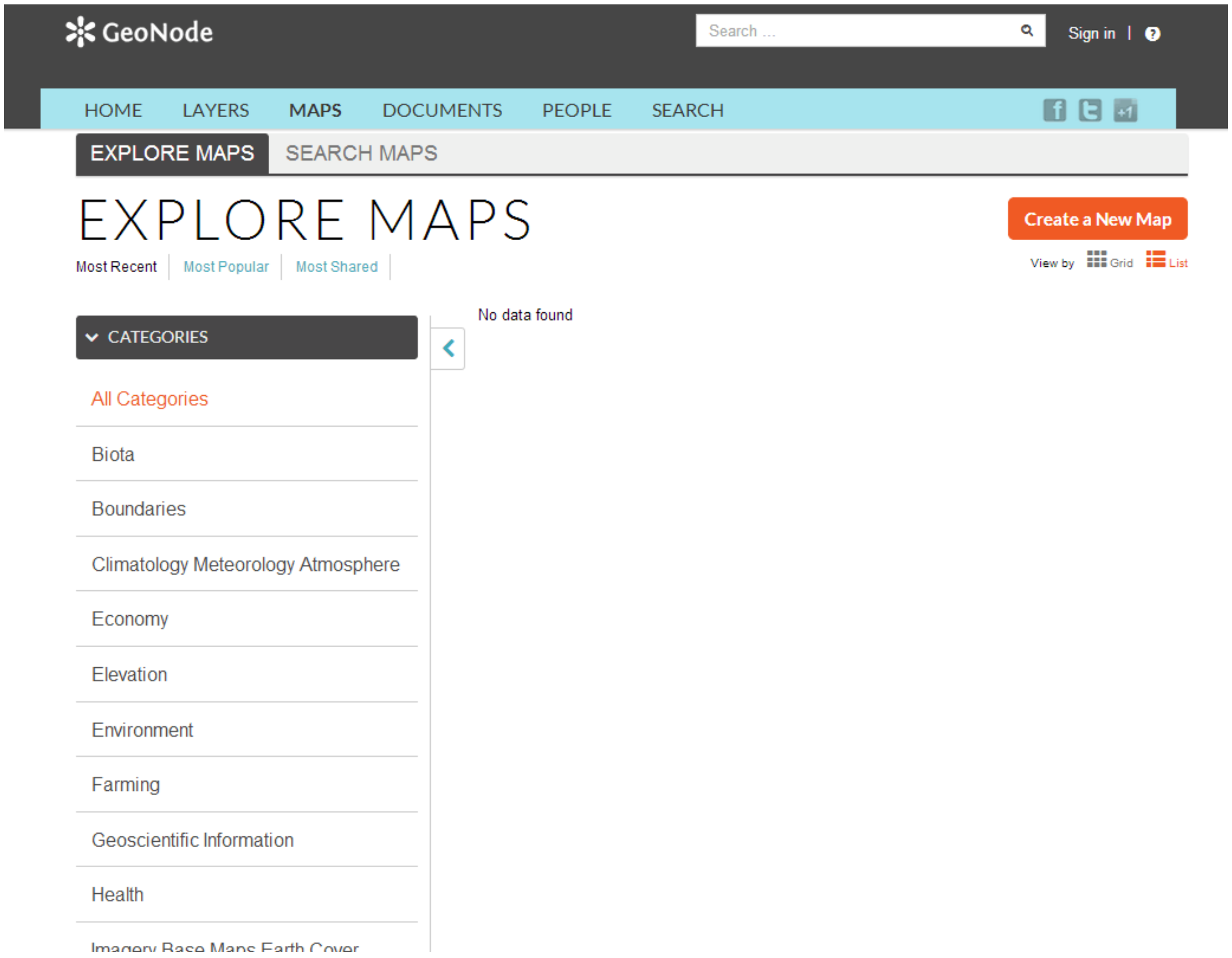
MAPS USING THIS LAYER

This layer is not currently used in any maps.

CREATE A MAP USING THIS

Fig. 3.3: *Viewing a layer*

5. Now click the *Maps* link in the tool bar to go to the *Explore Maps* page.



No data found

Fig. 3.4: *Explore Maps page*

This page shows all maps known to GeoNode, available with similar viewing options as with the layers. Currently, there are no maps here, but we will create one later on in the workshop.

6. Click the *Search* link in the toolbar to bring up the *Search* page.

   This page contains a wealth of options for customizing a search for various information on this GeoNode instance. While a simple search box is available at the top of every page, this search form allows for much more fine-tuned searches.

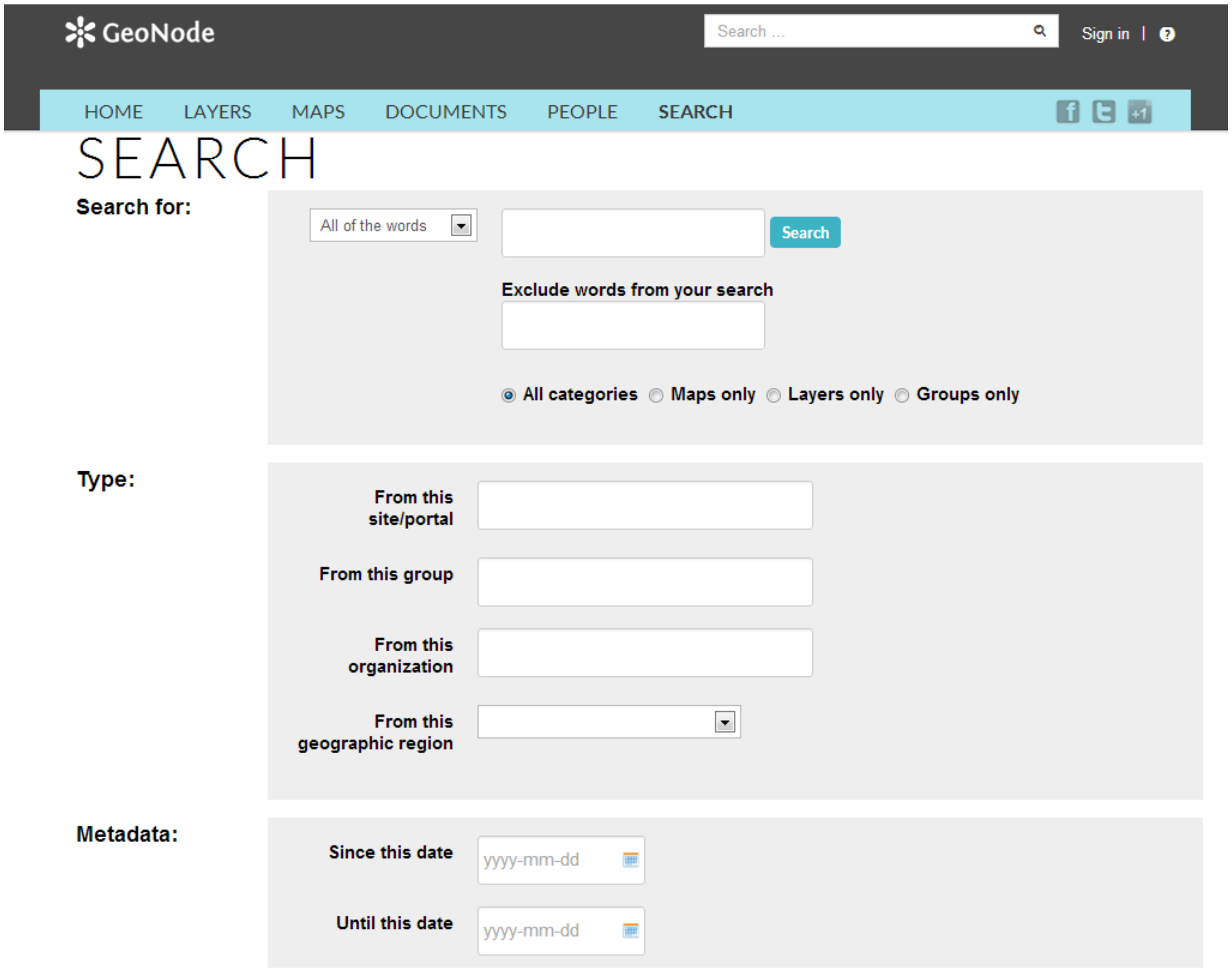
Now that you are familiar with the basic interface, the next step is to create your own account so you manage some GeoNode resources of your own.

GeoNode Quickstart

Open Source Geospatial Content Management System

GeoNode is a web-based application and platform for developing geospatial information systems (GIS) and for deploying spatial data infrastructures (SDI).

In this Quickstart guide you will learn the following:



SEARCH

Search for: All of the words | Search

Exclude words from your search

(● All categories) Maps only Layers only Groups only

Type:

From this site/portal

From this group

From this organization

From this geographic region

Metadata:

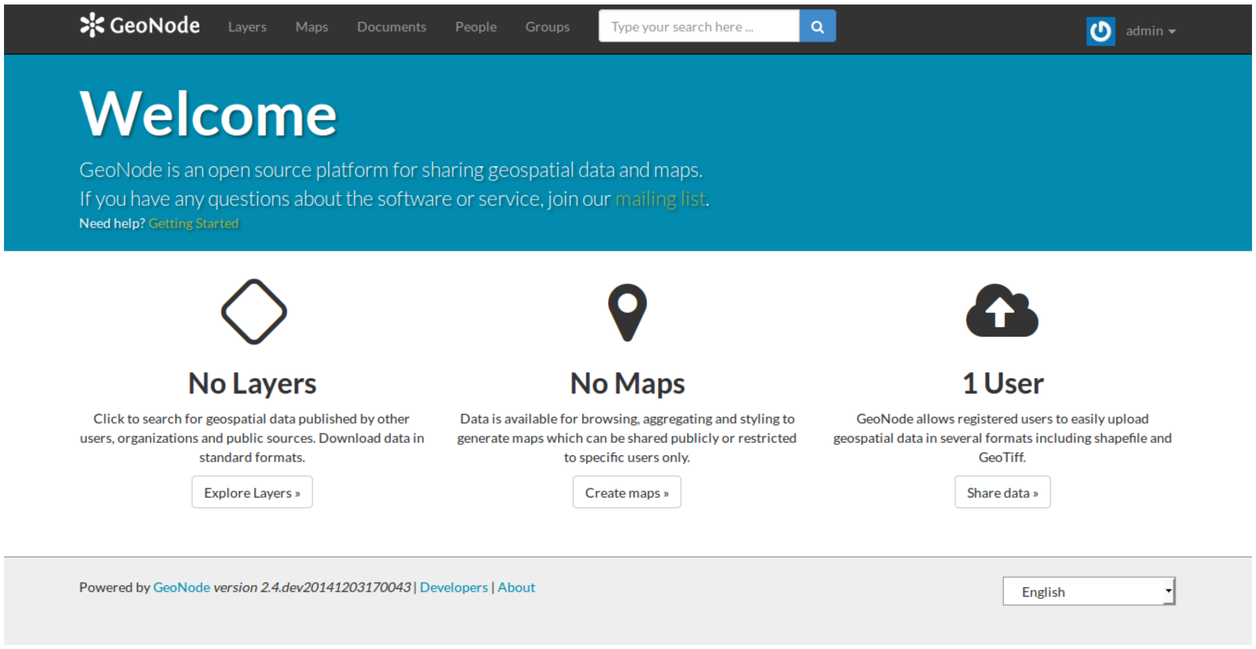
Since this date yyyy-mm-dd

Until this date yyyy-mm-dd

Fig. 3.5: *Search page*

1. to register a new account to get started
2. add a new layer
3. create a map using your new layer
4. share your map with others

To start GeoNode on your OSGeoLive DVD you have to choose *Geospatial* => *Browser Clients* => *Start GeoNode* and the GeoNode web page will automatically be opened at http://localhost:8000/ (I assume!). The page will look like shown in the image below.



Welcome

GeoNode is an open source platform for sharing geospatial data and maps.
If you have any questions about the software or service, join our mailing list.
Need help? Getting Started

No Layers

Click to search for geospatial data published by other users, organizations and public sources. Download data in standard formats.

Explore Layers »

No Maps

Data is available for browsing, aggregating and styling to generate maps which can be shared publicly or restricted to specific users only.

Create maps »

1 User

GeoNode allows registered users to easily upload geospatial data in several formats including shapefile and GeoTiff.

Share data »

Powered by GeoNode *version 2.4.dev20141203170043* | Developers | About

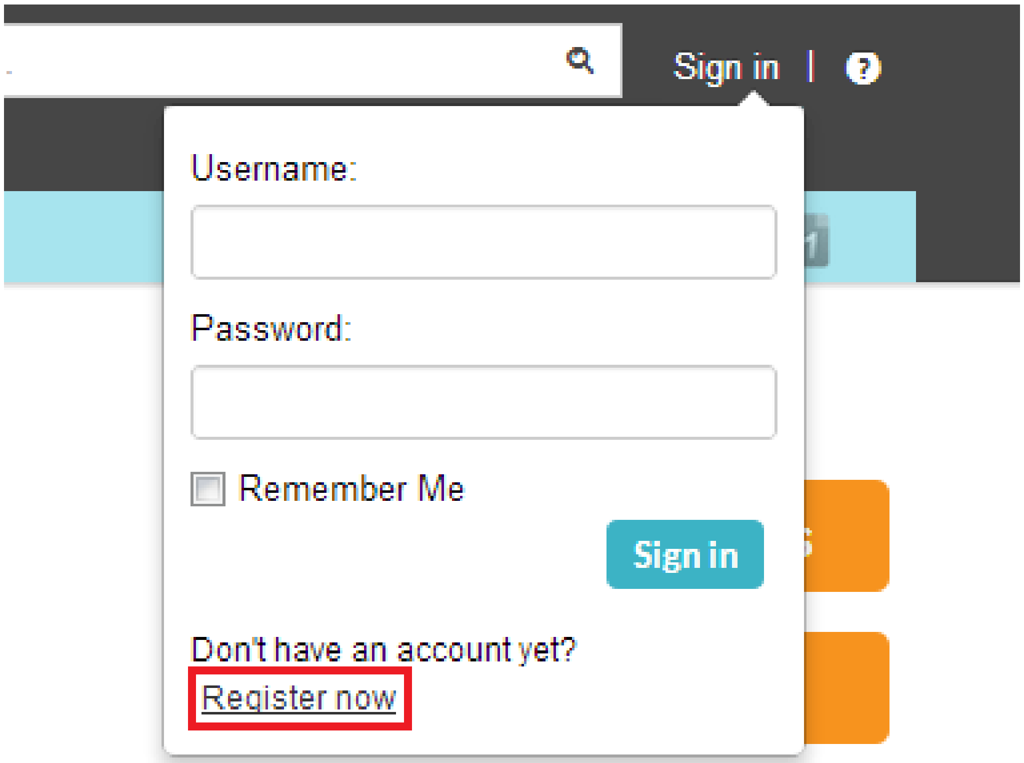
English

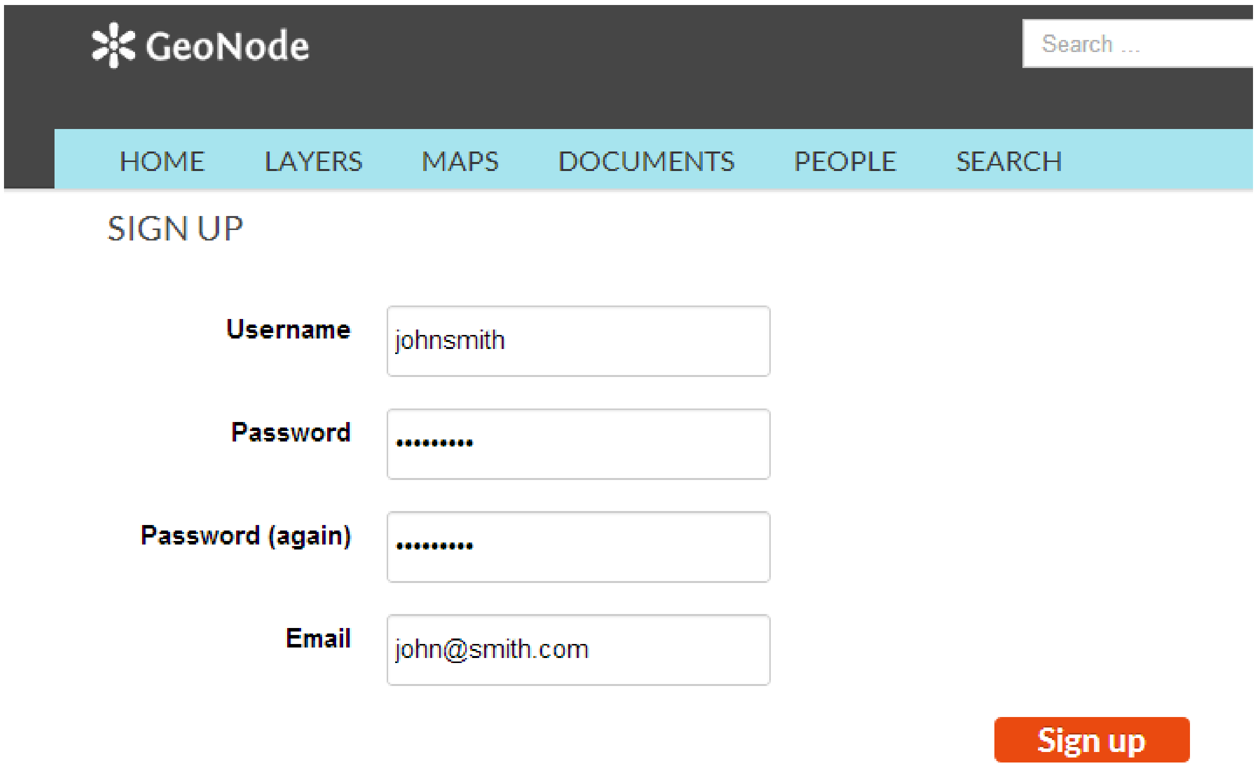
1. Register a new account From the interface shown above, one can view and modify existing spatial layers and maps, as well as find information on other GeoNode users. But, without being logged in, you are limited to read-only access of public layers. In order to create a map and add layers to it, you have to have create an account first.

1. From any page in the web interface, you will see a *Sign in* link. Click that link, and in the dialog that displays, click the *Register now* link.
2. On the next page, fill out the form. Enter a user name and password in the fields. Also, enter your email address for verification.
3. You will be returned to the welcome page. An email will be sent confirming that you have signed up. While you are now logged in, you will need to confirm your account. Navigate to the link that was sent in the email.
4. By clicking *Confirm* you will be returned to the homepage. Now you've registered an account, you are able to add layers to it as well as create maps and share those with other users.

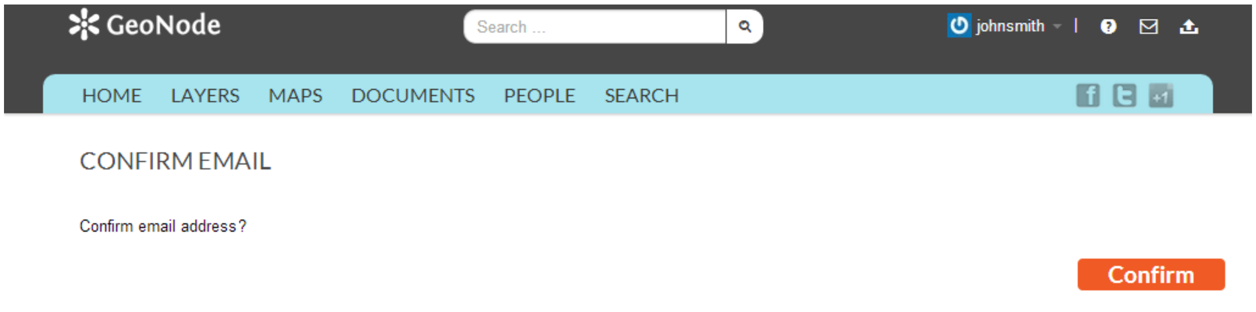
2. Add a new layer Layers are a published resource representing a raster or vector spatial data source. Layers also can be associated with metadata, ratings, and comments.

1. To add a layer to your account, navigate to the welcome page. There the following toolbar can be seen:



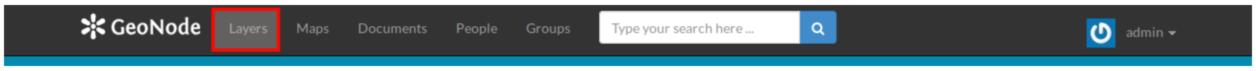


Username	johnsmith
Password	•••••••••
Password (again)	•••••••••
Email	john@smith.com

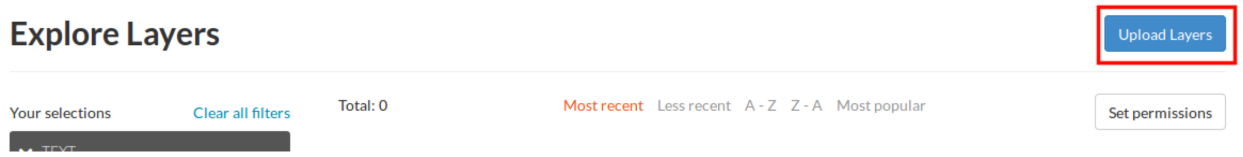


CONFIRM EMAIL

Confirm email address?



2. By clicking the *Layers* link you will be brought to the *Layers* menu where a new subtoolbar can be seen. This toolbar allows you to *Explore*, *Search* and *Upload* layers.

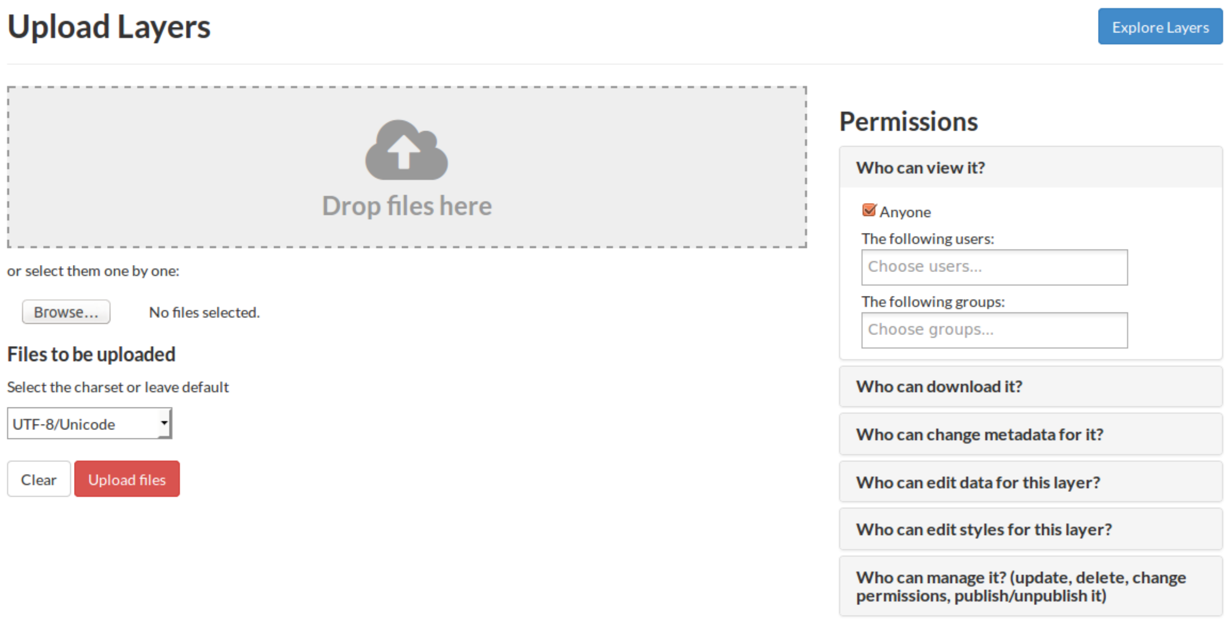


3. Now click *Upload Layers* and you'll see the upload form.
4. You have two possibilities to add your files. You can either do that by using *drag & drop* or you choose to *browse* them. Be aware that you have to upload a complete set of files, consisting of a *shp*, a *prj*, a *dbf* and a *shx* file. If one of them is missing, GeoNode will warn you before you upload them.
5. You shold now be able to see all the files you want to upload.
6. GeoNode has the ability to restrict who can view, edit, and manage layers. On the right side of the page you can see the *Permission* section, where you can limit the access on your layer. Under *Who can view and download this data*, select *Any registered user*. This will ensure that anonymous view access is disabled. In the same area, under *Who can edit this data*, select your username. This will ensure that only you are able to edit the data in the layer.
7. To upload data, click the *Upload* button at the bottom.

3. Create a new map The next step for you is to create a map and add the newly created layers to this map.

1. Click the *Maps* link on the top toolbar. This will bring up the list of maps.
2. Currently, there aren't any maps here. To add one click the *Create a New Map* button and a map composition interface will display.

   In this interface there is a toolbar, layer list, and map window. The map window contains the MapQuest OpenStreetMap layer by default. There are other service layers available here as well: Blue Marble, Bing Aerial With Labels, MapQuest, and OpenStreetMap.
3. Click on the *New Layers* button and select *Add Layers*.
4. Now you should be able to see all the availabel layers. In your case, this should only be the ones you've added before (San Andreas?).



Upload Layers

Explore Layers

Drop files here

or select them one by one:

Browse... No files selected.

Files to be uploaded

Select the charset or leave default

UTF-8/Unicode

Clear Upload files

Permissions

Who can view it?

☑ Anyone

The following users:

Choose users...

The following groups:

Choose groups...

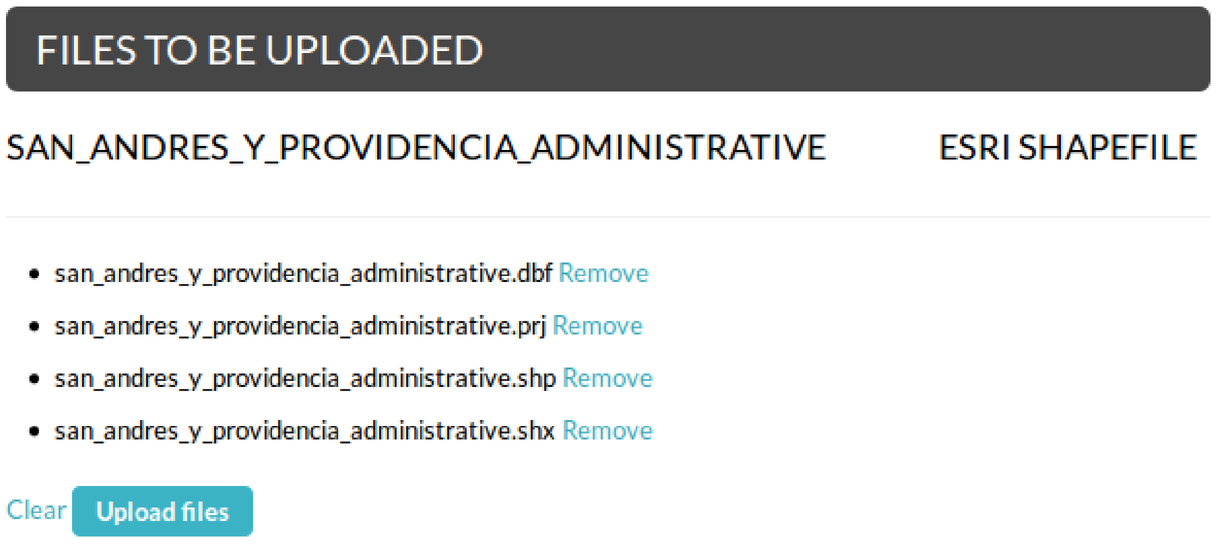
Who can download it?

Who can change metadata for it?

Who can edit data for this layer?

Who can edit styles for this layer?

Who can manage it? (update, delete, change permissions, publish/unpublish it)

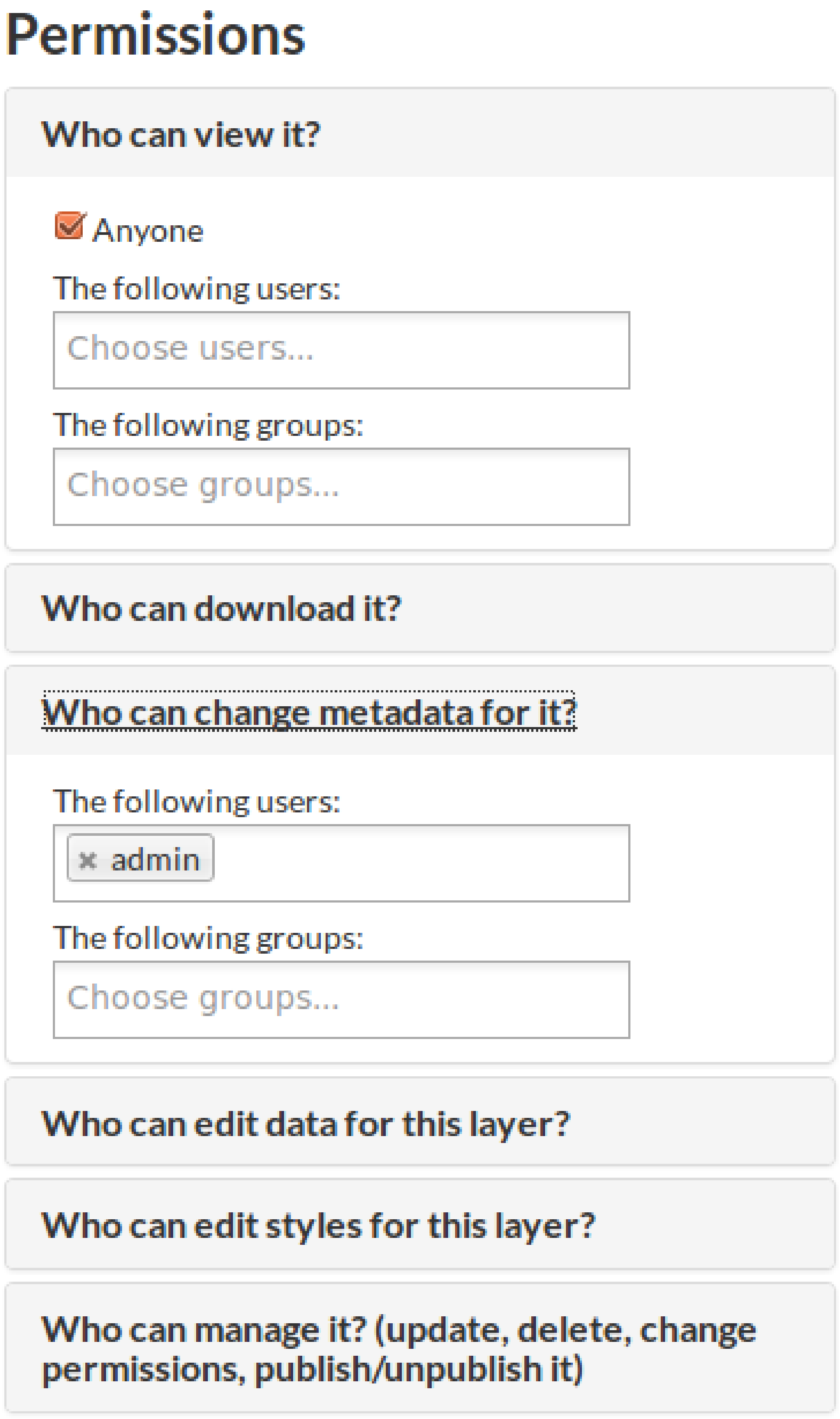


FILES TO BE UPLOADED

SAN_ANDRES_Y_PROVIDENCIA_ADMINISTRATIVE ESRI SHAPEFILE

- san_andres_y_providencia_administrative.dbf Remove
- san_andres_y_providencia_administrative.prj Remove
- san_andres_y_providencia_administrative.shp Remove
- san_andres_y_providencia_administrative.shx Remove

Clear Upload files



Permissions

Who can view it?

[✓] Anyone

The following users: Choose users...

The following groups: Choose groups...

Who can download it?

Who can change metadata for it?

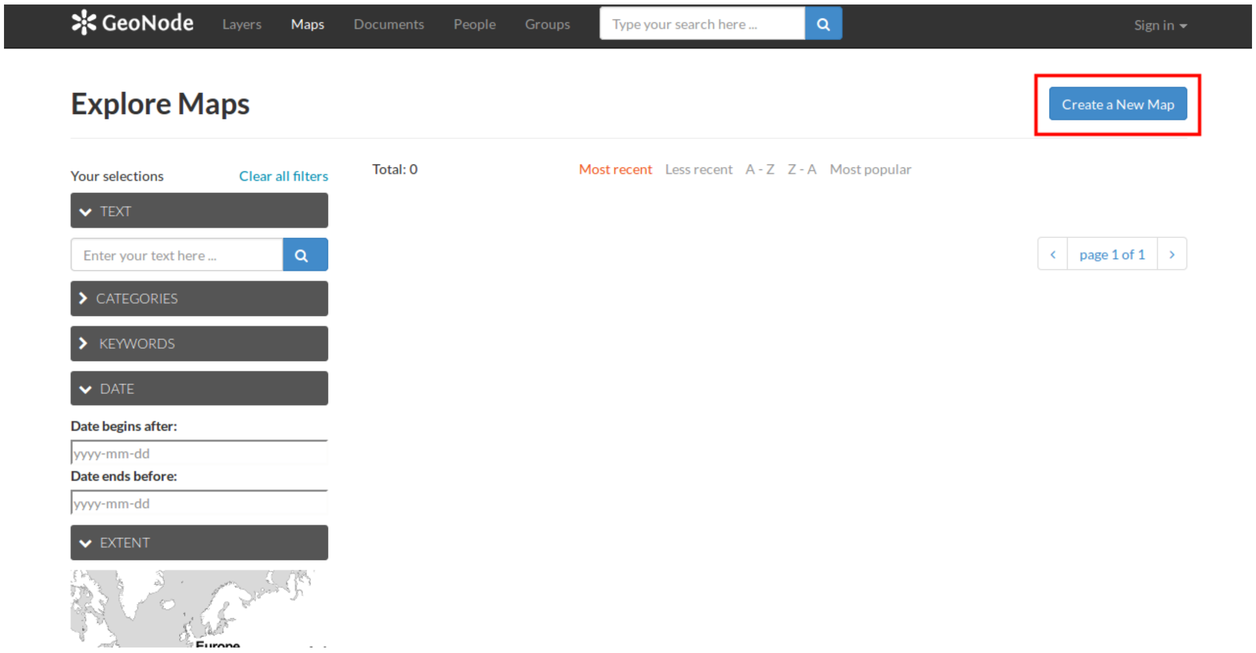
The following users: × admin

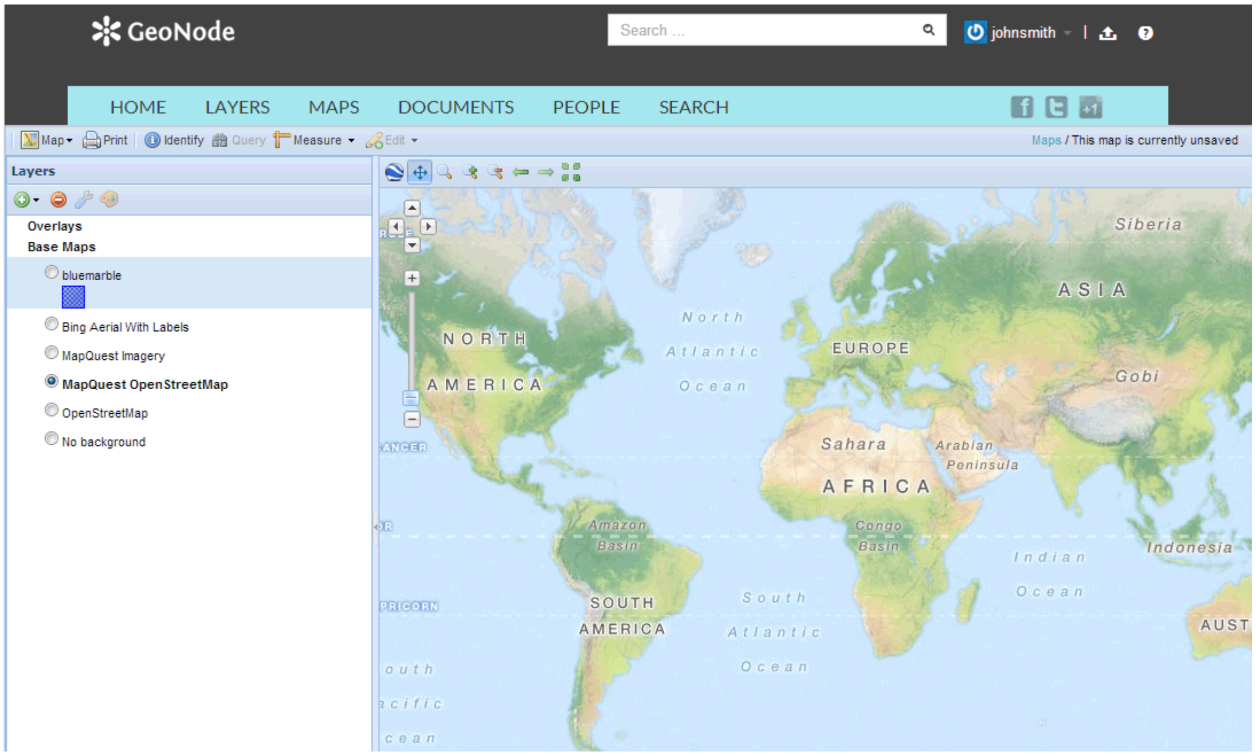
The following groups: Choose groups...

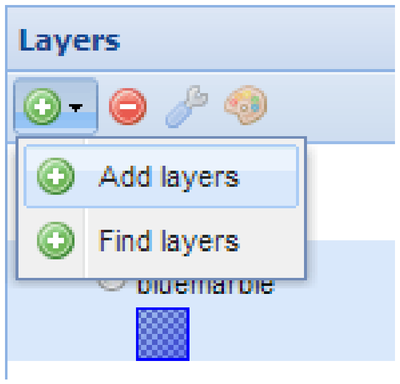
Who can edit data for this layer?

Who can edit styles for this layer?

Who can manage it? (update, delete, change permissions, publish/unpublish it)


GeoNode
Layers
Maps
Documents
People
Groups
Type your search here ...
Sign in
Explore Maps
Create a New Map
Your selections
Clear all filters
Total: 0
Most recent
Less recent
A - Z
Z - A
Most popular
TEXT
Enter your text here ...
page 1 of 1
CATEGORIES
KEYWORDS
DATE
Date begins after:
yyyy-mm-dd
Date ends before:
yyyy-mm-dd
EXTENT
Europe


GeoNode
Search ...
johnsmith
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
Map
Print
Identify
Query
Measure
Edit
Maps / This map is currently unsaved
Layers
Overlays
Base Maps
bluemarble
Bing Aerial With Labels
MapQuest Imagery
MapQuest OpenStreetMap
OpenStreetMap
No background
Siberia
ASIA
NORTH AMERICA
North Atlantic Ocean
EUROPE
Gobi
Sahara
Arabian Peninsula
AFRICA
Amazon Basin
Congo Basin
Indian Ocean
Indonesia
SOUTH AMERICA
South Atlantic Ocean
AUST



5. Select all of the layers by clicking the top entry and Shift-clicking the bottom one. Click *Add Layers* to add them all to the map.
6. The layers will be added to the map. Click *Done* (right next to *Add Layers* at the bottom) to return to the main layers list.
7. To save the map click on the *Map* button in the toolbar, and select *Save Map*.
8. Enter a title and abstract for your map.
9. Click *Save*. Notice that the link on the top right of the page changed to reflect the map's name.

   This link contains a permalink to your map. If you open this link in a new window, your map will appear exactly as it was saved.

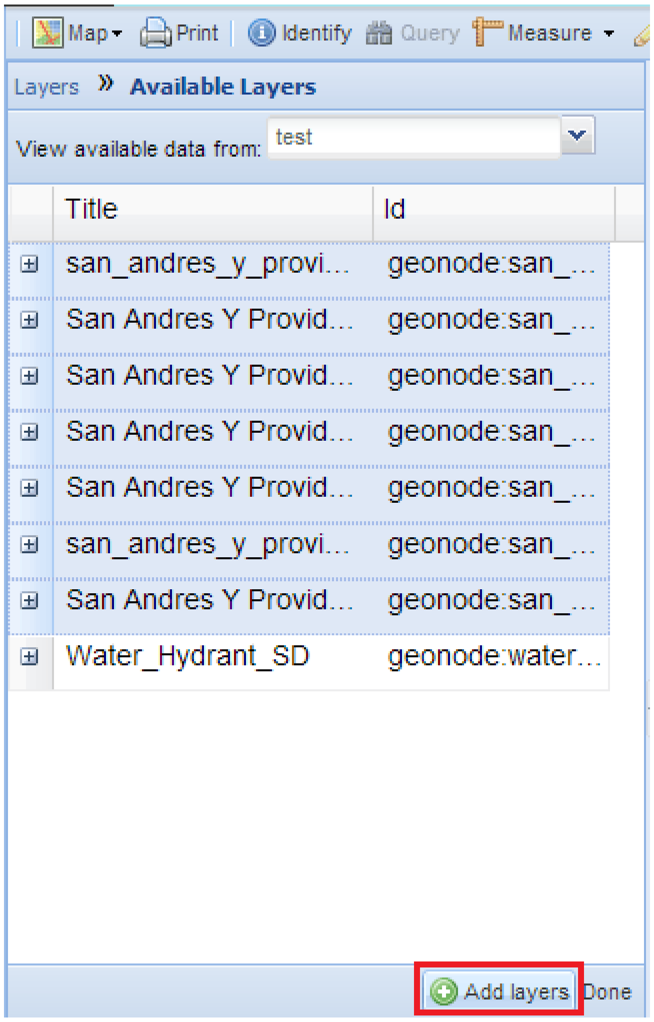
4. Share your map Now let's finish our map.

1. Check the box next to the *highway* layer to activate it. If it is not below the *POI* layer in the list, click and drag it down.
2. Make any final adjustments to the map composition as desired, including zoom and pan settings.
3. Click the *Map* button in the toolbar, and then click *Publish Map*.
4. The title and abstract as previously created should still be there. Make any adjustments as necessary, and click *Save*.
5. A new dialog will appear with instructions on how to embed this map in a web page, including a code snippet. You can adjust the parameters as necessary.

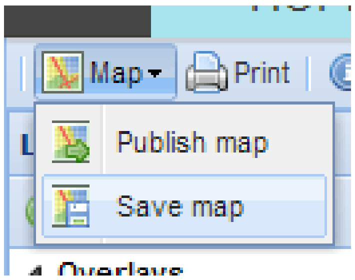
Your map can now be shared!

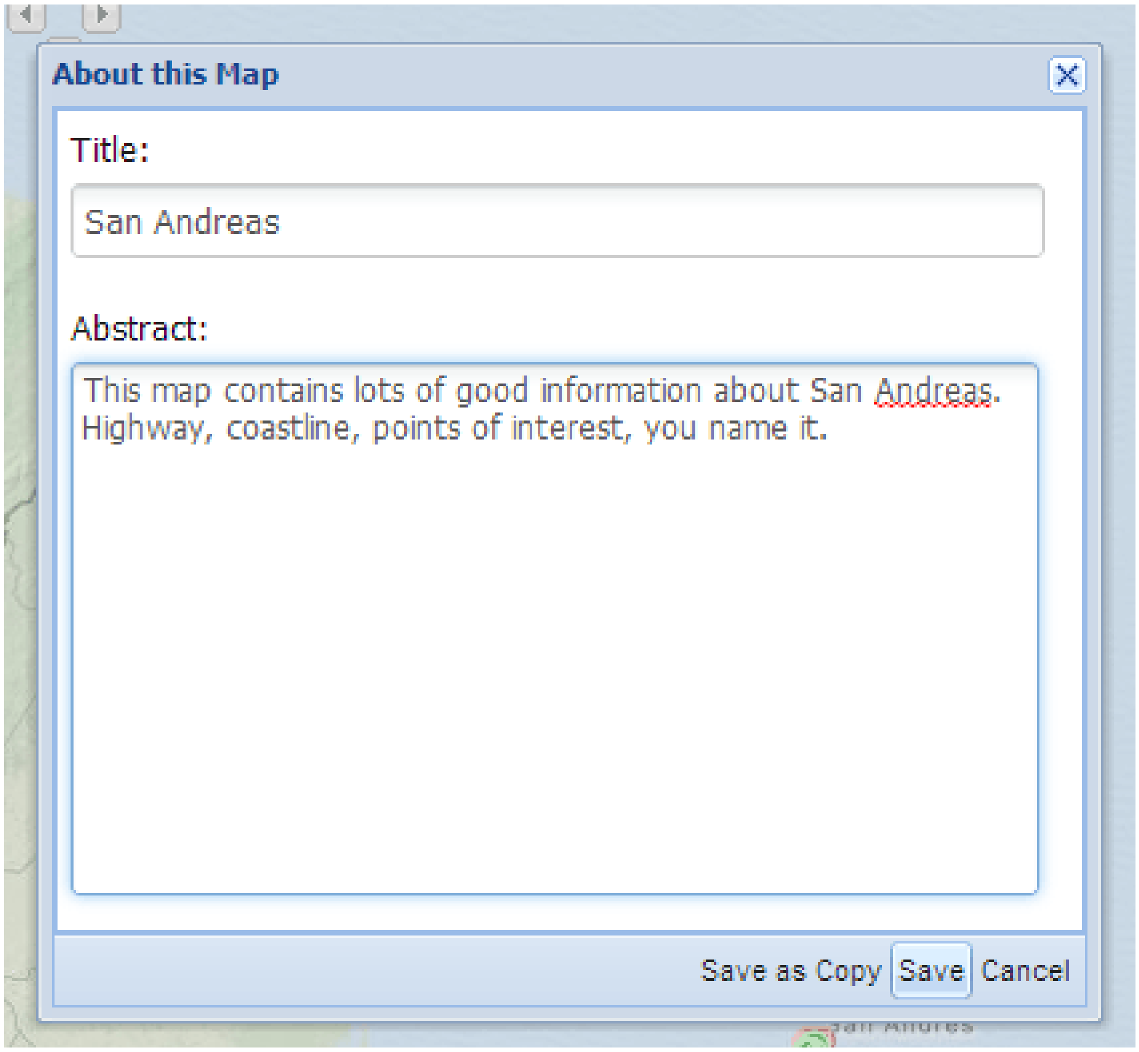
To be continued Now you've gotten a quick insight in the possibilities of GeoNode. To learn more about GeoNode and its features, visit our webpage www.geonode.org. To install GeoNode on your own server, follow our Quick Installation Guide or the Complete Installation Guide. In order to get started with GeoNode our documentation might be useful.

If you need help or want to get some information about a specific topic please don't hesitate to ask us! You can do this through the #geonode IRC channel using http://webchat.freenode.net/ or by asking your question in our google group !



Title	Id
san_andres_y_provi...	geonode:san_...
San Andres Y Provid...	geonode:san_...
San Andres Y Provid...	geonode:san_...
San Andres Y Provid...	geonode:san_...
San Andres Y Provid...	geonode:san_...
san_andres_y_provi...	geonode:san_...
San Andres Y Provid...	geonode:san_...
Water_Hydrant_SD	geonode:water...



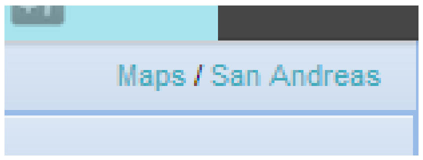


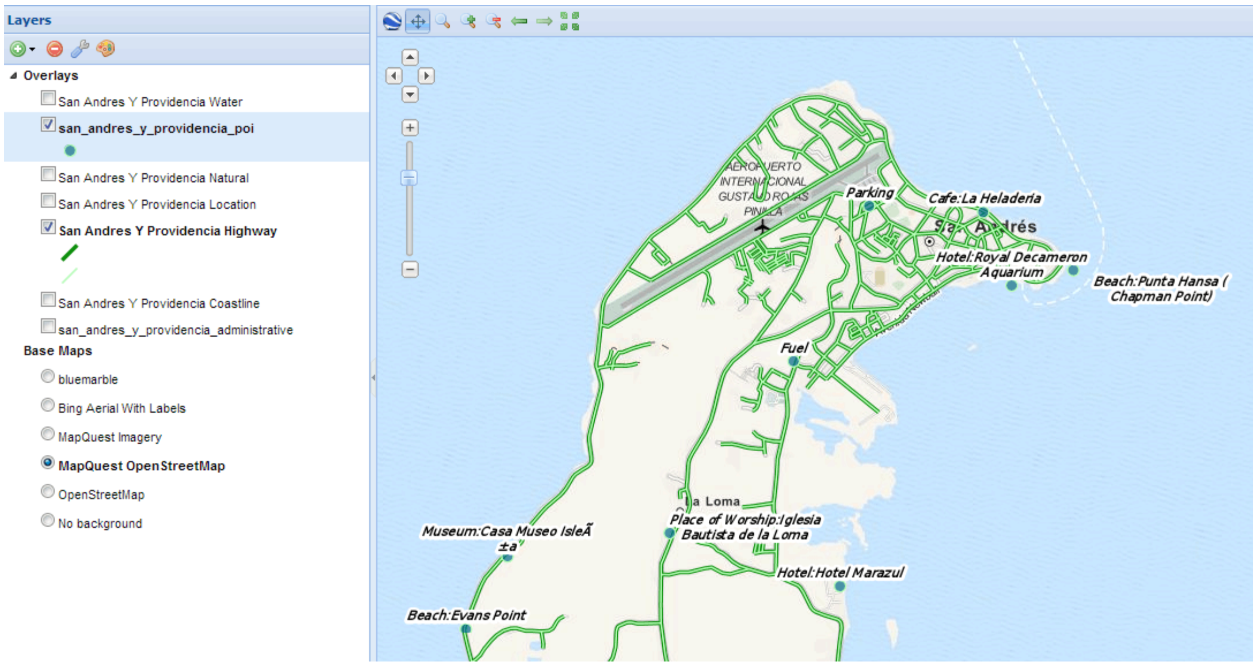
Title:

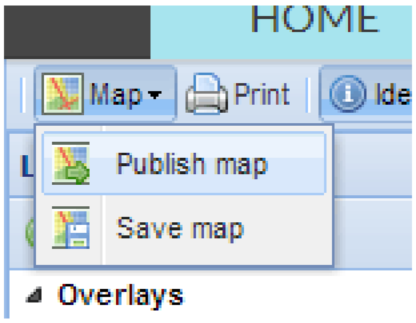
San Andreas

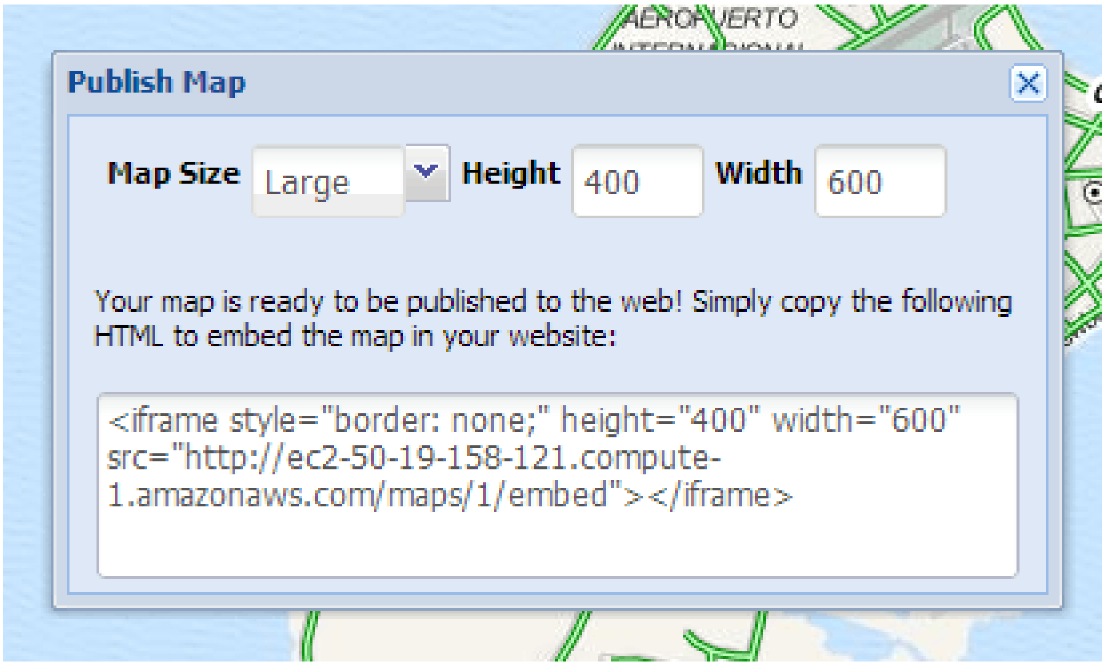
Abstract:

This map contains lots of good information about San Andreas. Highway, coastline, points of interest, you name it.




Layers
Overlays
San Andres Y Providencia Water
san_andres_y_providencia_poi
San Andres Y Providencia Natural
San Andres Y Providencia Location
San Andres Y Providencia Highway
San Andres Y Providencia Coastline
san_andres_y_providencia_administrative
Base Maps
bluemarble
Bing Aerial With Labels
MapQuest Imagery
MapQuest OpenStreetMap
OpenStreetMap
No background
Parking
Cafe:La Heladeria
Hotel:Royal Decameron
Aquarium
Beach:Punta Hansa (Chapman Point)
Fuel
Place of Worship:Iglesia Bautista de la Loma
Museum:Casa Museo IsleÃ±a
Hotel:Hotel Marazul
Beach:Evans Point





Map Size: Large Height: 400 Width: 600

Your map is ready to be published to the web! Simply copy the following HTML to embed the map in your website:

```
<iframe style="border: none;" height="400" width="600"
src="http://ec2-50-19-158-121.compute-
1.amazonaws.com/maps/1/embed"></iframe>
```

Accounts and users

GeoNode is primarily a *social* platform, and thus a primary component of any GeoNode instance is the user account. This section will guide you through account registration, updating your account information, and viewing other user accounts.

Creating a new account Before you can save or edit any layers on a GeoNode instance, you need to create an account.

1. From any page in the web interface, you will see a *Register* link. Click that link, and the register form will appear

Note: The registrations in GeoNode must be open, in case you don't see the register link then it's not possible to register unless the addministrator of the side does that for you.

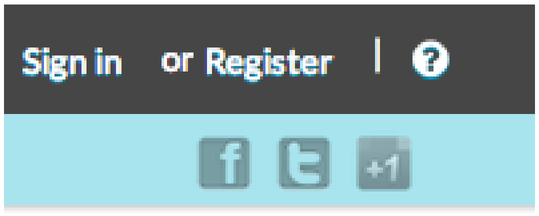
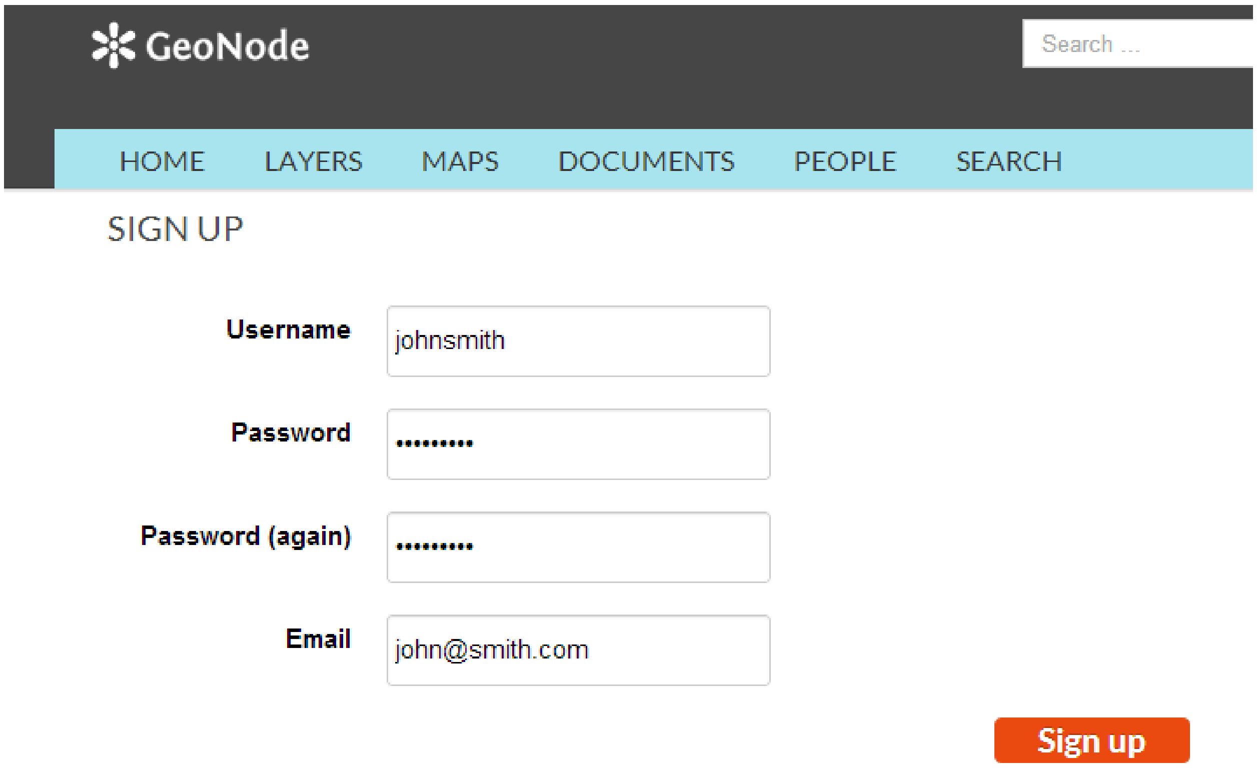


Fig. 3.6: *Sign in screen*

1. On the next page, fill out the form. Enter a user name and password in the fields. Also, enter your email address for verification.
2. You will be returned to the welcome page. An email will be sent confirming that you have signed up. While you are now logged in, you will need to confirm your account. Navigate to the link that was sent in the email.
3. Click *Confirm*. You will be returned to the homepage.

Managing your profile Your profile contains personal information.

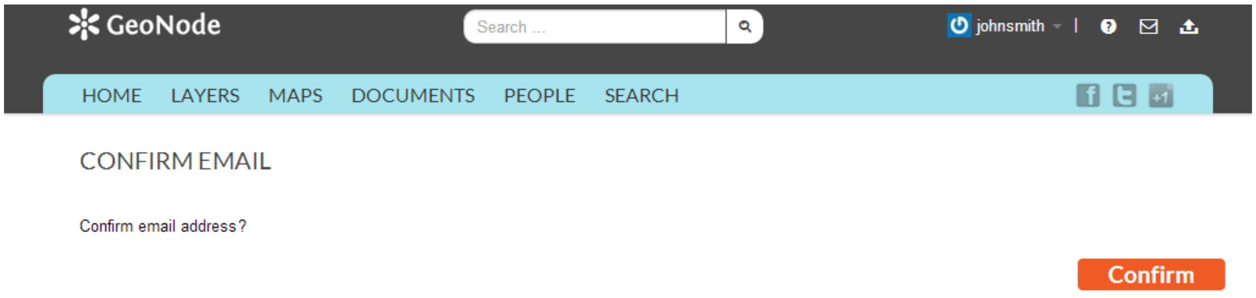
1. Click on your user name in the top right of the screen. A drop-down list will show. Click on *Profile* to enter the Profile settings page.
2. The next page shows your profile, which is currently empty.
3. Click the *Edit profile information* link.
4. On this page, your personal information can be set, including your avatar. Enter some details in the *Profile* box as well as your city and country info.
5. When finished, click *Update profile*.
6. You will be returned to the main profile page. Now click *Account settings*.
7. On this page you can change your email address, time zone, and language. Your email should be populated already, but set the timezone to your current location.
8. When finished, click *Save*.



SIGN UP

Username	johnsmith
Password	•••••••••
Password (again)	•••••••••
Email	john@smith.com

Fig. 3.7: *Registering for a new account*



CONFIRM EMAIL

Confirm email address?

Fig. 3.8: *Confirming your email address*

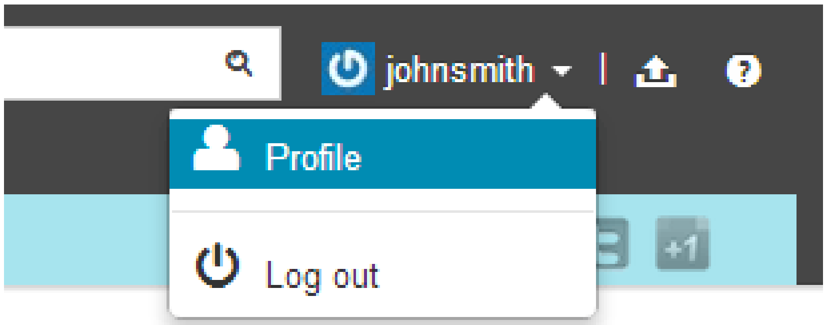


Fig. 3.9: *Link to your profile*

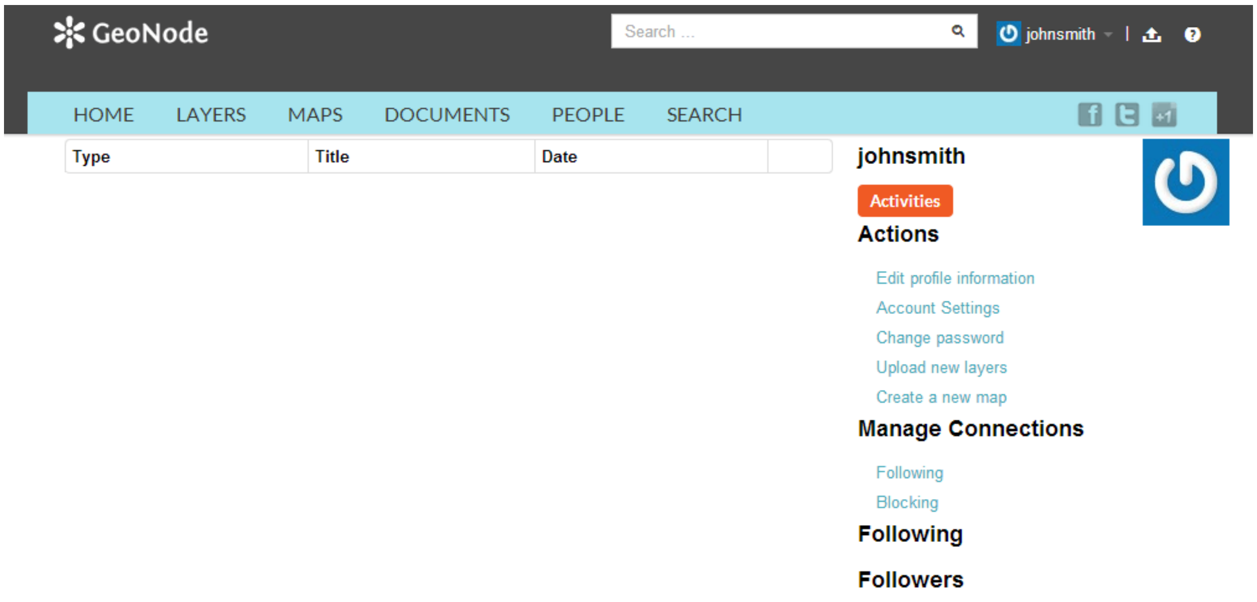


Fig. 3.10: *Profile page*

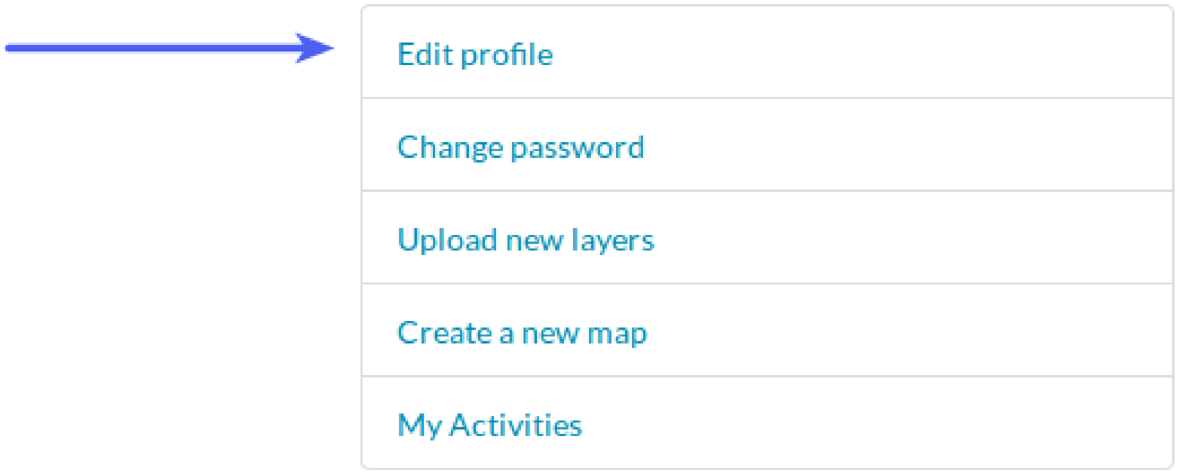
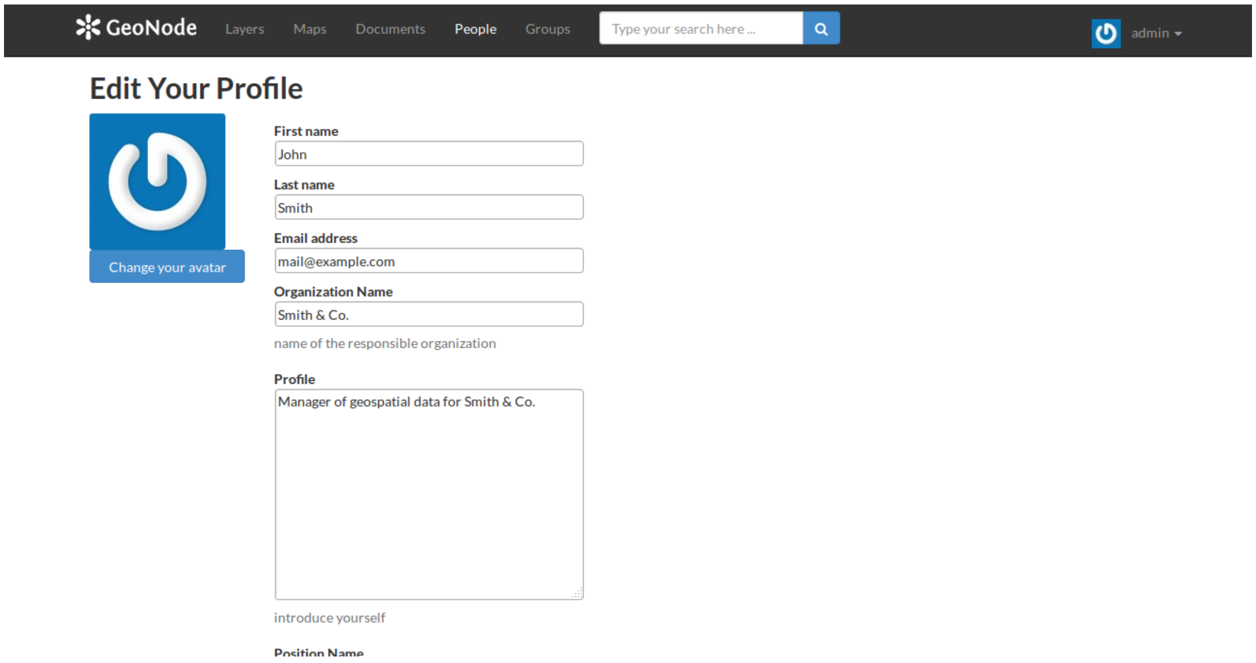


Fig. 3.11: *Link to edit your profile*



Edit Your Profile

First name
John

Last name
Smith

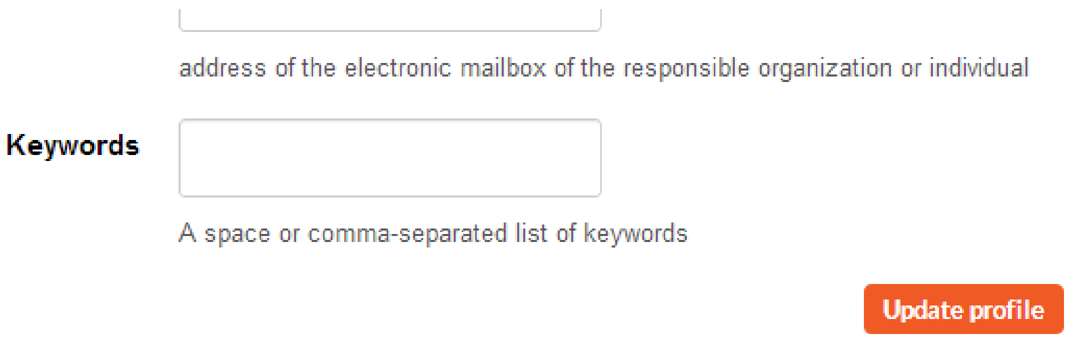
Email address
mail@example.com

Organization Name
Smith & Co.
name of the responsible organization

Profile
Manager of geospatial data for Smith & Co.
introduce yourself

Position Name

Fig. 3.12: *Editing your profile*


address of the electronic mailbox of the responsible organization or individual

Keywords

A space or comma-separated list of keywords

Fig. 3.13: *Link to save your profile updates*

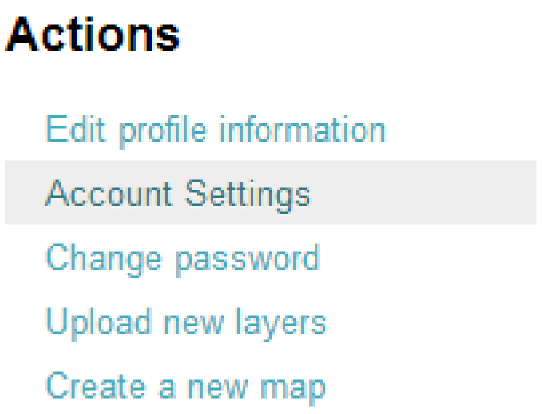
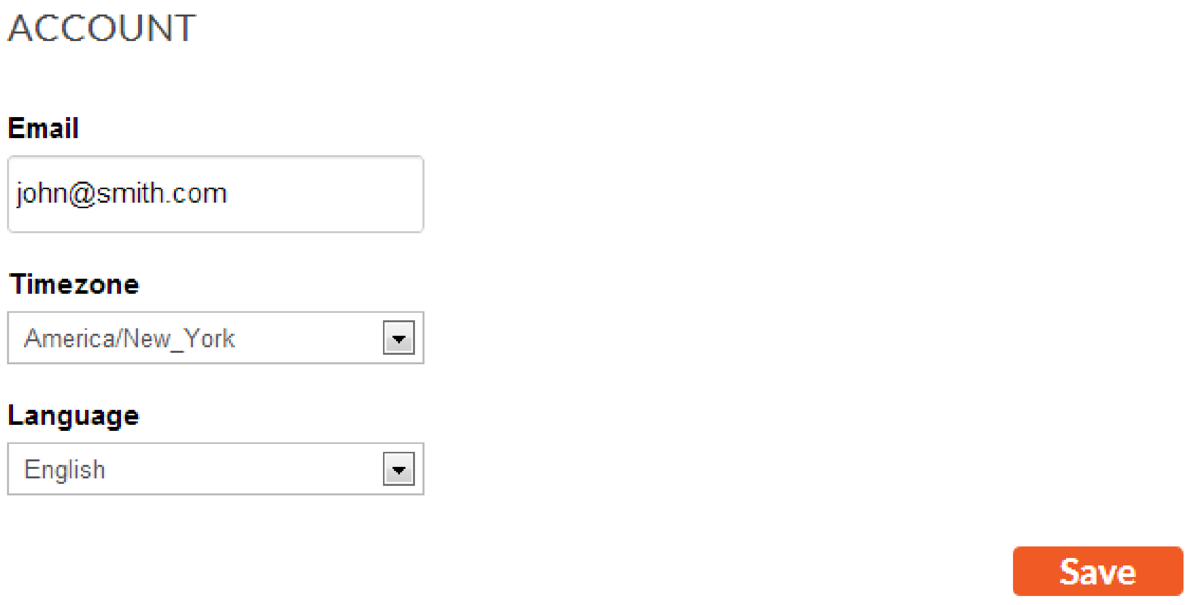


Fig. 3.14: *Link to edit your account settings*


ACCOUNT

Email

john@smith.com

Timezone

America/New_York

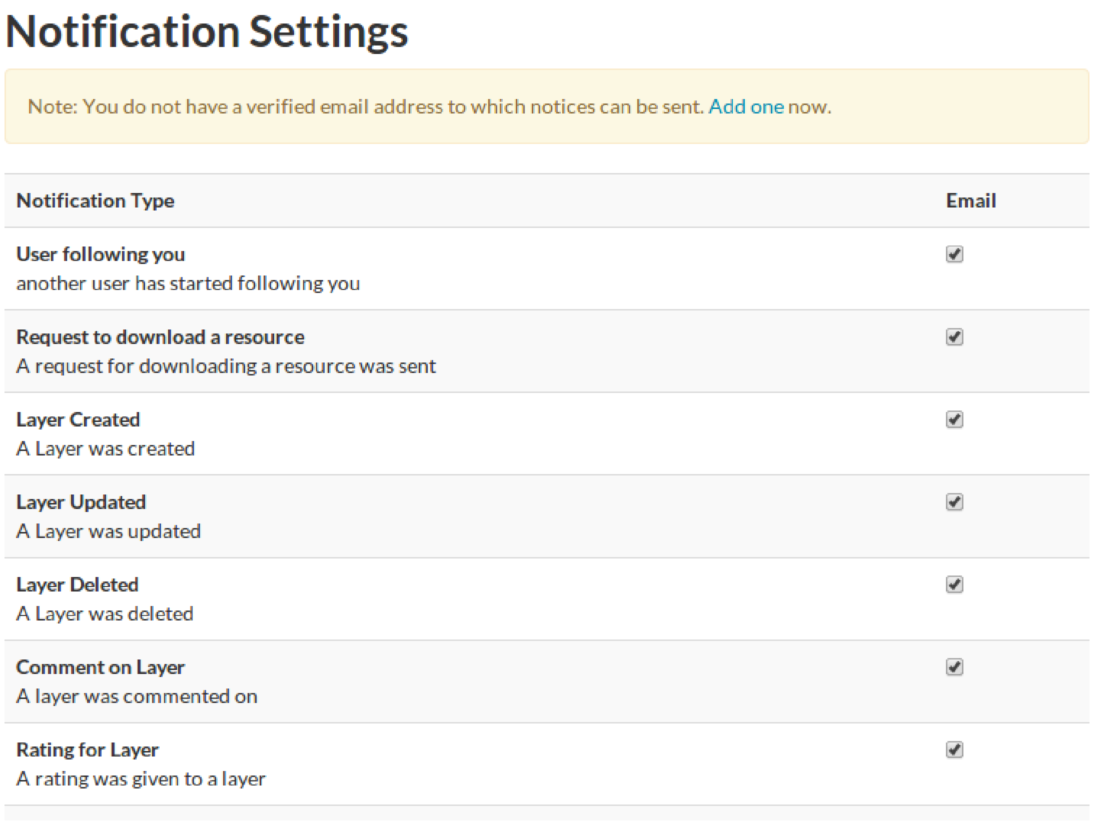
Language

English

Fig. 3.15: *Editing your account*

Setting notification preferences By default GeoNode sends notifications to the users for events that the users could be subscribed such as a new layer uploaded or a new rate added to a map.

1. you can adjust your notification settings by clicking on your user name in the top right of the screen. A drop-down list will show. Click on *Notifications* to enter the Notifications Settings page.


Notification Settings

Note: You do not have a verified email address to which notices can be sent. Add one now.

Notification Type	Email
User following you another user has started following you	✓
Request to download a resource A request for downloading a resource was sent	✓
Layer Created A Layer was created	✓
Layer Updated A Layer was updated	✓
Layer Deleted A Layer was deleted	✓
Comment on Layer A layer was commented on	✓
Rating for Layer A rating was given to a layer	✓

2. make sure to have a verified email address to which notices can be sent. If not, click on the proposed link to add one
3. now check/uncheck the notification types you wish to receive or not receive. It is possible to be notified for the following events:

- Layer Created
- Layer Updated
- Layer Deleted
- Rating for Layer
- Comment for Layer
- Map Created
- Map Updated
- Map Deleted
- Rating for Map
- Comment for Map
- Document Created

- Document Updated
- Document Deleted
- Rating for Document
- Comment for Document
- User following you
- Request to donwload a resource

Viewing other user accounts Now that your account is created, you can view other accounts on the system. Note that on the main profile page there are options for following (and blocking) other users.

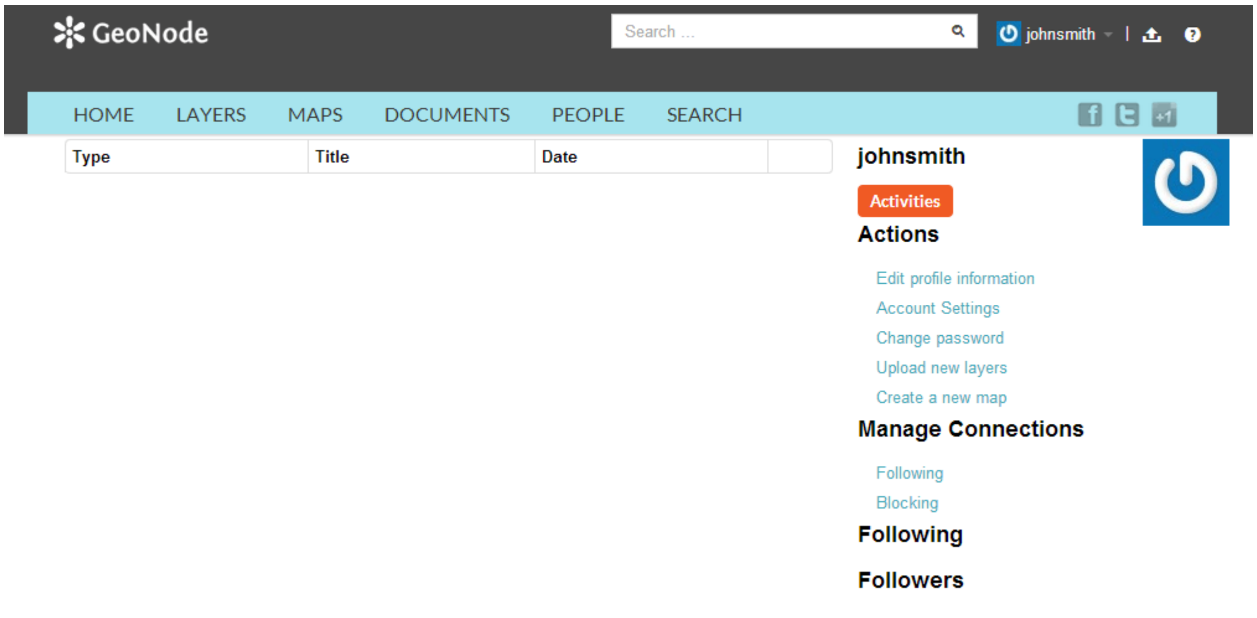


Fig. 3.16: *Profile page*

1. To see information about other users on the system, click the *People* link on the top toolbar. You will see a list of users registered on this system.
2. Click on the user name for a particular user. You will see the layers owned by this user.

1. You can also click *Activities* to see the activity feed.
2. If you are interested in keeping track of what this user does, go back to the previous page and click the *Follow* button.
3. A confirmation page will display. Click *Confirm*.
4. You will now be following this user, and your profile page will note this.

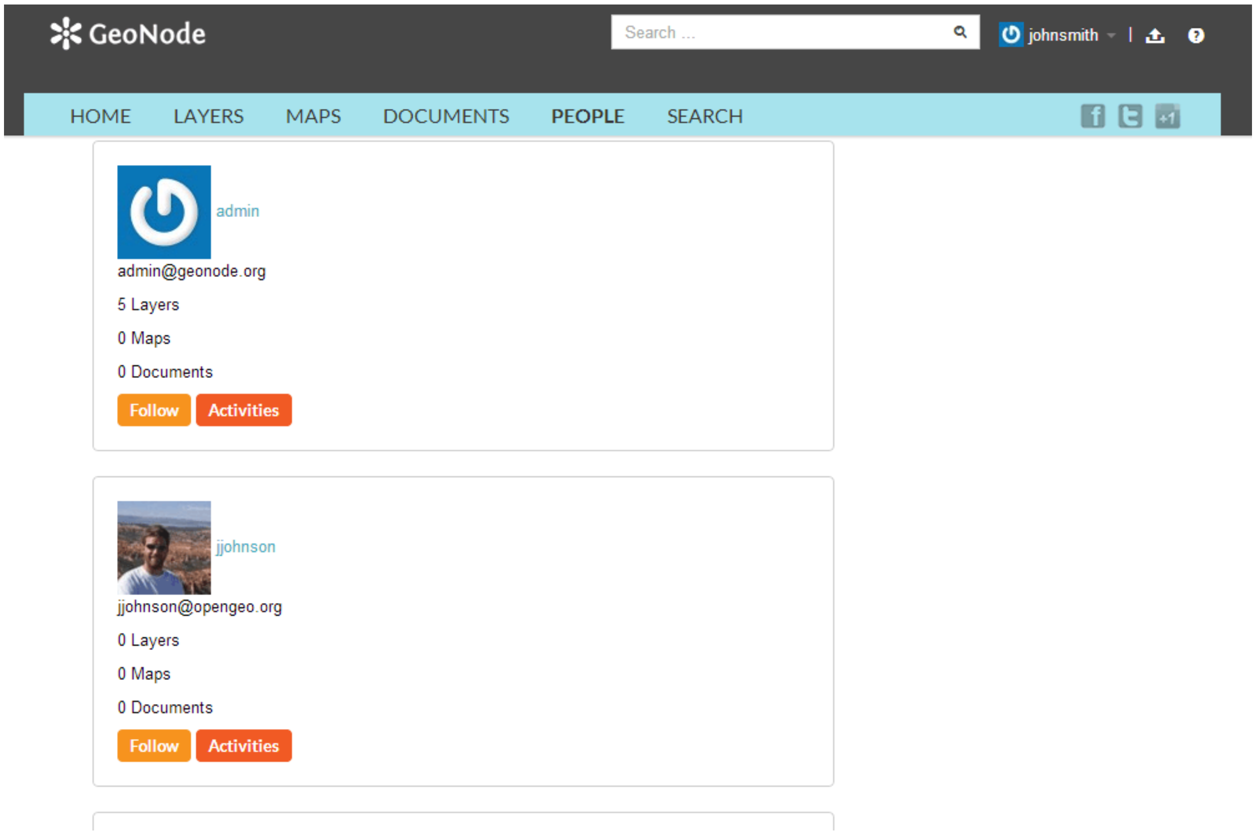
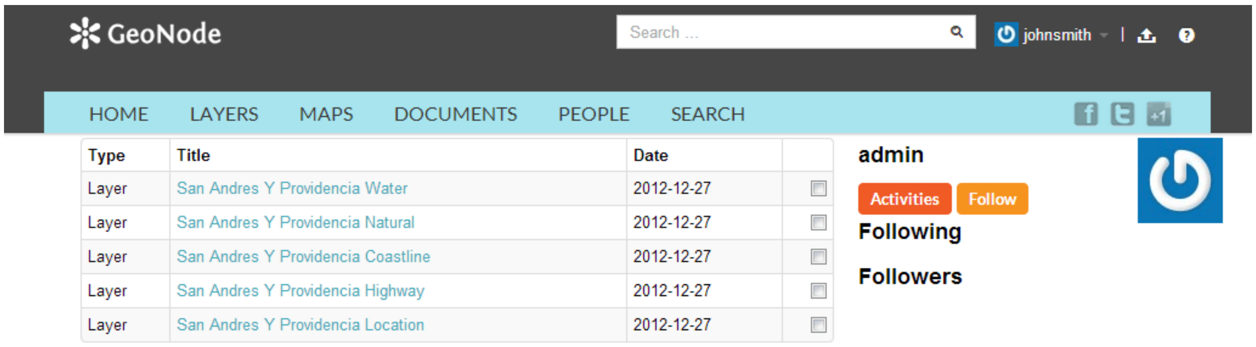
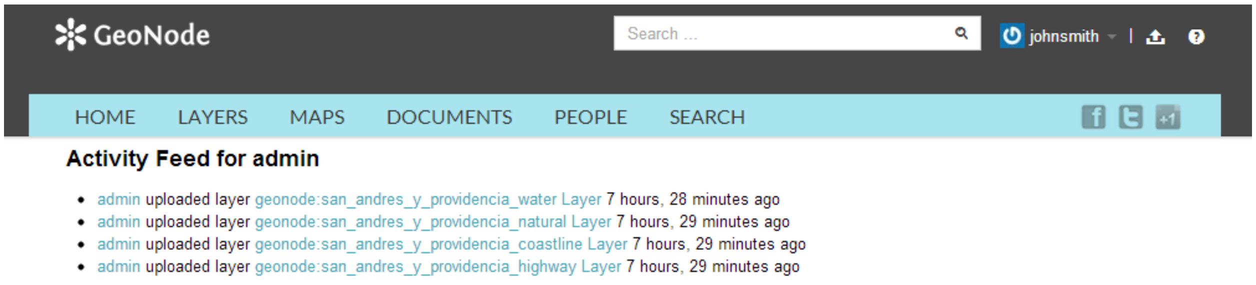


Fig. 3.17: *List of users*



Type	Title	Date	
Layer	San Andres Y Providencia Water	2012-12-27	☐
Layer	San Andres Y Providencia Natural	2012-12-27	☐
Layer	San Andres Y Providencia Coastline	2012-12-27	☐
Layer	San Andres Y Providencia Highway	2012-12-27	☐
Layer	San Andres Y Providencia Location	2012-12-27	☐

Fig. 3.18: *List of layers owned by a user*



Activity Feed for admin

- admin uploaded layer geonode:san_andres_y_providencia_water Layer 7 hours, 28 minutes ago
- admin uploaded layer geonode:san_andres_y_providencia_natural Layer 7 hours, 29 minutes ago
- admin uploaded layer geonode:san_andres_y_providencia_coastline Layer 7 hours, 29 minutes ago
- admin uploaded layer geonode:san_andres_y_providencia_highway Layer 7 hours, 29 minutes ago

Fig. 3.19: *List of users*

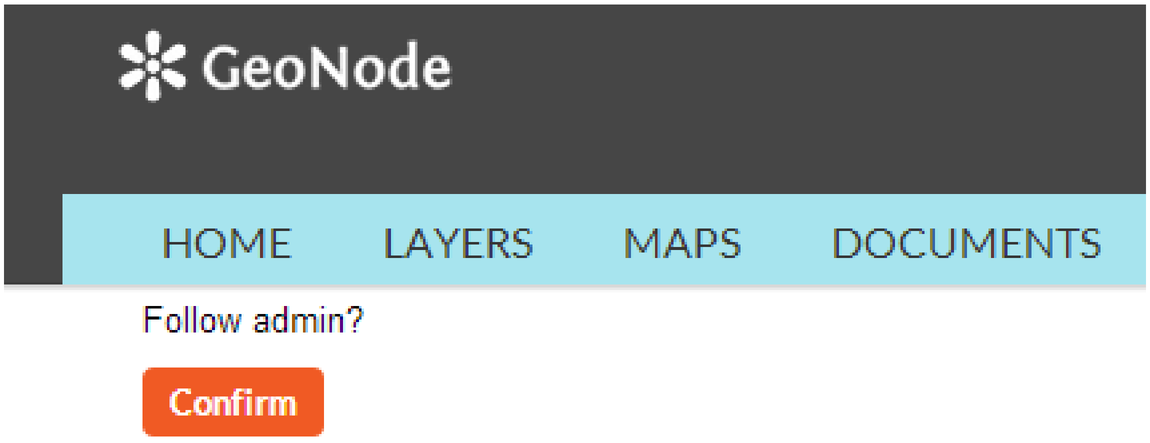


Fig. 3.20: *Confirming following a user*

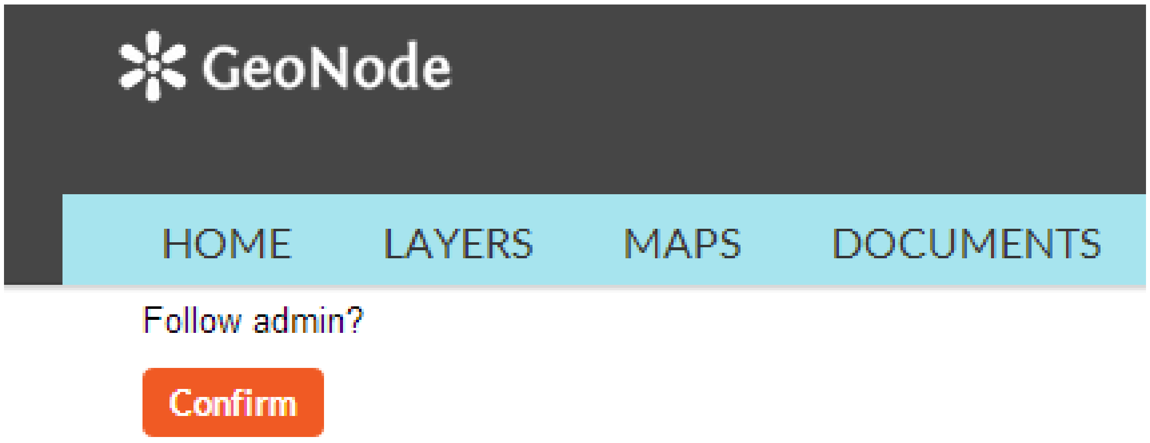


Fig. 3.21: *Success following a user*

Managing layers

After user accounts, the next primary component of GeoNode is the **layer**. Layers are a published resource representing a raster or vector spatial data source. Layers also can be associated with metadata, ratings, and comments.

In this section, you will learn how to create a new layer by uploading a local data set, add layer info, change the style of the layer, and share the results.

Uploading a layer Now that we have taken a tour of GeoNode and viewed existing layers, the next step is to upload our own.

In your data pack is a directory called `data`. Inside that directory is a shapefile called `san_andres_y_providencia_administrative.shp`. This is a data set containing administrative boundaries for the San Andres Province. This will be the first layer that we will upload to GeoNode.

1. Navigate to the GeoNode welcome page.
2. Click the *Layers* link on the top toolbar. This will bring up the Layers menu.

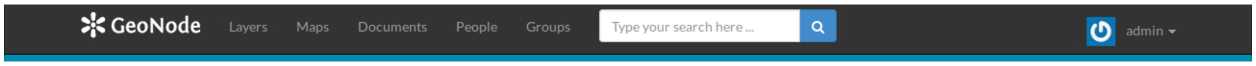


Fig. 3.22: *Main toolbar for GeoNode*

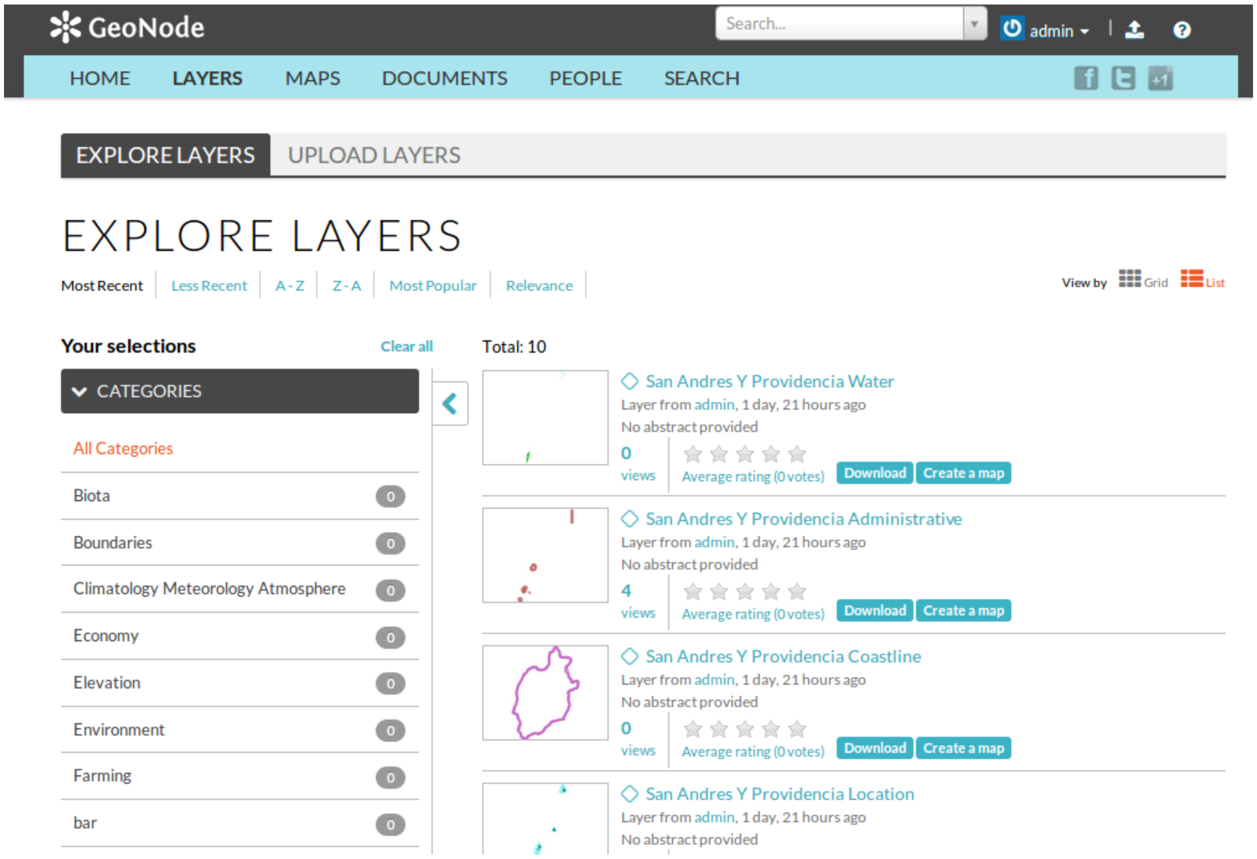


Fig. 3.23: *Layers menu*

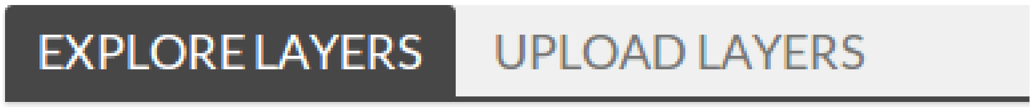
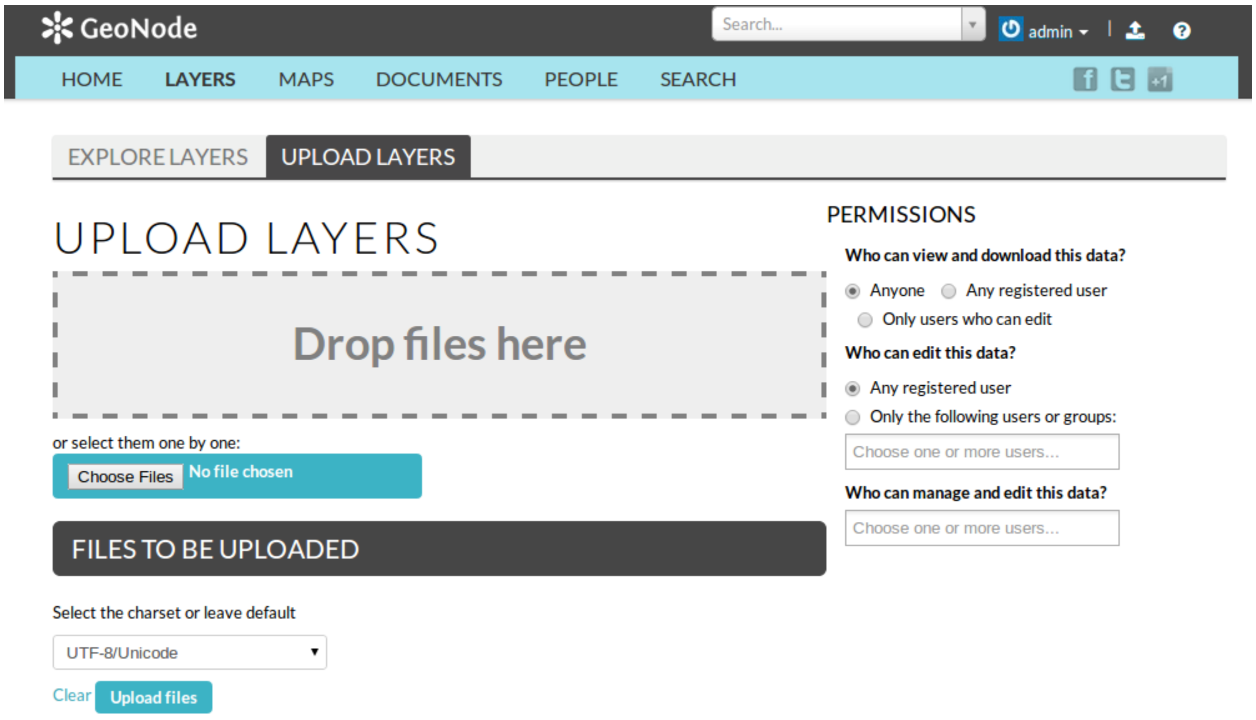


Fig. 3.24: *Layers toolbar*



or select them one by one: No file chosen

Select the charset or leave default: UTF-8/Unicode

Who can view and download this data?

[✓ Anyone] Any registered user Only users who can edit

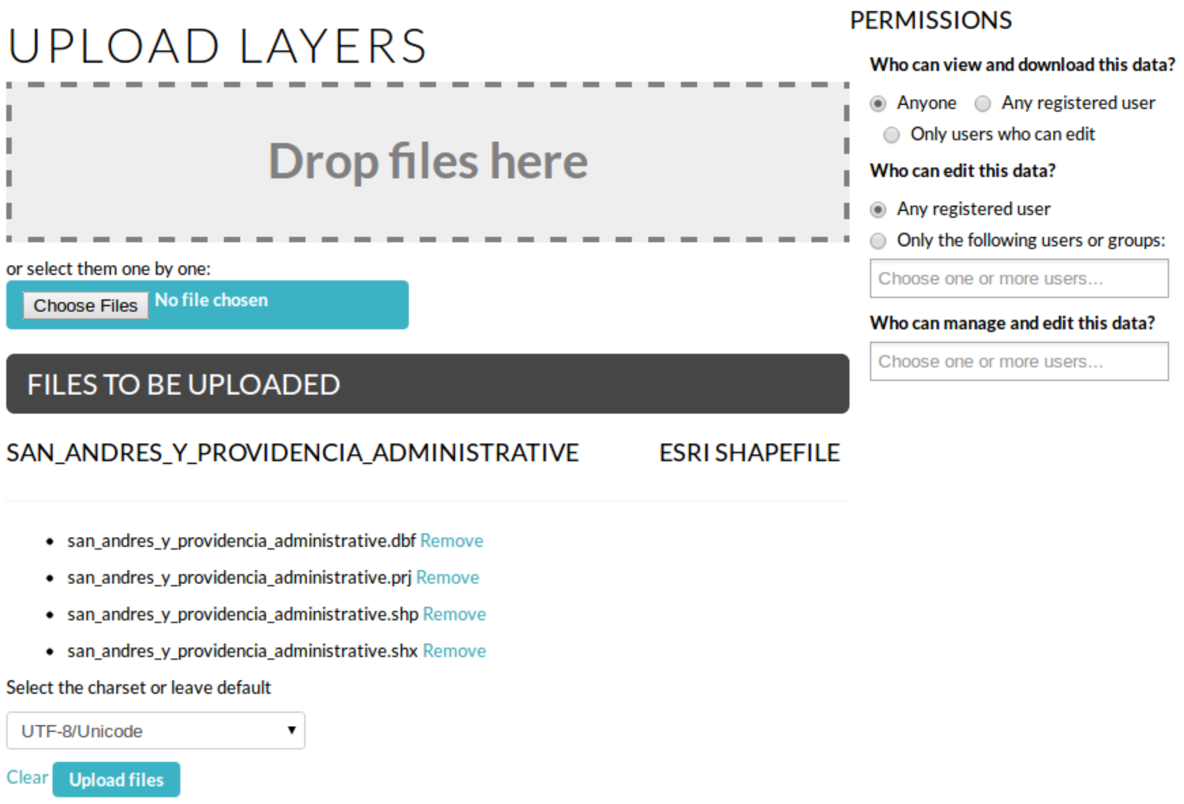
Who can edit this data?

[✓ Any registered user] Only the following users or groups: Choose one or more users...

Who can manage and edit this data? Choose one or more users...

Fig. 3.25: *Upload Layers form*

3. Click *Upload Layers* in the Layers toolbar. This will bring up the upload form
4. Fill out the form.
   - Click on the *Browse...* button. This will bring up a local file dialog. Navigate to your data folder and select all of the four files composing the shapefile (`san_andres_y_providencia_administrative.shp`, `san_andres_y_providencia_administrative.dbf`, `san_andres_y_providencia_administrative` `san_andres_y_providencia_administrative.prj`). Alternatively you could drag and drop the four files in the *Drop files here* area.
   - The upload form should appear like this now:



UPLOAD LAYERS

Drop files here

or select them one by one:

Choose Files No file chosen

FILES TO BE UPLOADED

SAN_ANDRES_Y_PROVIDENCIA_ADMINISTRATIVE	ESRI SHAPEFILE

- san_andres_y_providencia_administrative.dbf Remove
- san_andres_y_providencia_administrative.prj Remove
- san_andres_y_providencia_administrative.shp Remove
- san_andres_y_providencia_administrative.shx Remove

Select the charset or leave default

UTF-8/Unicode

Clear Upload files

PERMISSIONS

Who can view and download this data?

[✓ Anyone] Any registered user Only users who can edit

Who can edit this data?

[✓ Any registered user]

Only the following users or groups:

Choose one or more users...

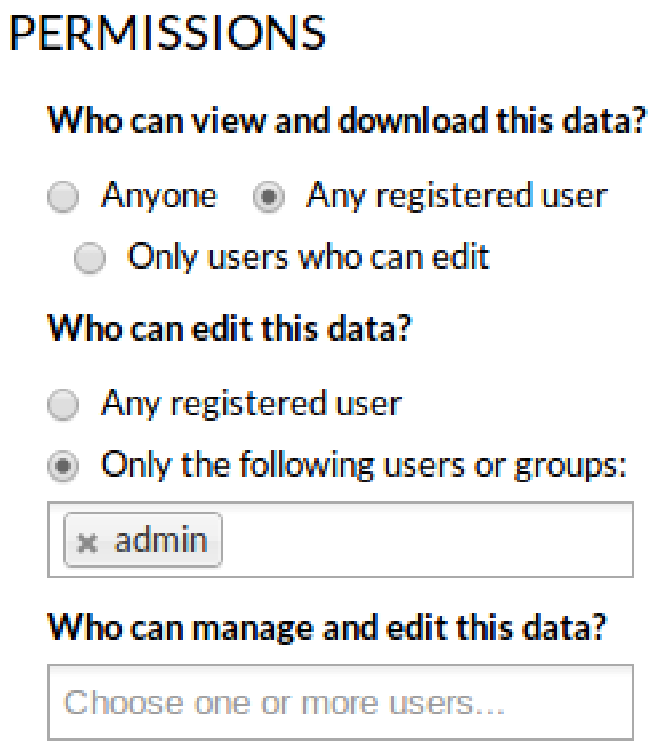
Who can manage and edit this data?

Choose one or more users...

Fig. 3.26: *Files ready for upload*

5. GeoNode has the ability to restrict who can view, edit, and manage layers. On the right side of the page, under *Who can view and download this data?*, select *Any registered user*. This will ensure that anonymous view access is disabled.
6. In the same area, under *Who can edit this data?*, select the *Only the following users or groups* option and type your username. This will ensure that only you are able to edit the data in the layer.
7. Click *Upload* to upload the data and create a layer. A dialog will display showing the progress of the upload.
8. Your layer has been uploaded to GeoNode. Now you will be able to access to the its info page (clicking on the *Layer Info* button), access to its metadata edit form (clicking on the *Edit Metadata* button) or to manage the styles for it (clicking on the *Manage Styles* button).

Layer information After upload, another form will displaying, containing metadata about the layer. Change any information as desired, and then click *Update* at the very bottom of the form.



PERMISSIONS

Who can view and download this data?

Anyone [✓ Any registered user] Only users who can edit

Who can edit this data?

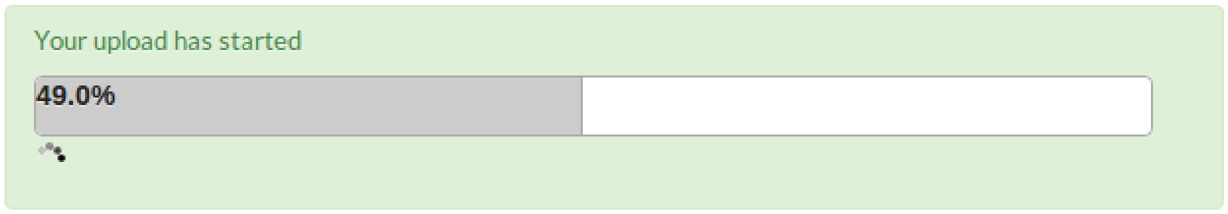
Any registered user [✓ Only the following users or groups:]

admin

Who can manage and edit this data?

Choose one or more users...

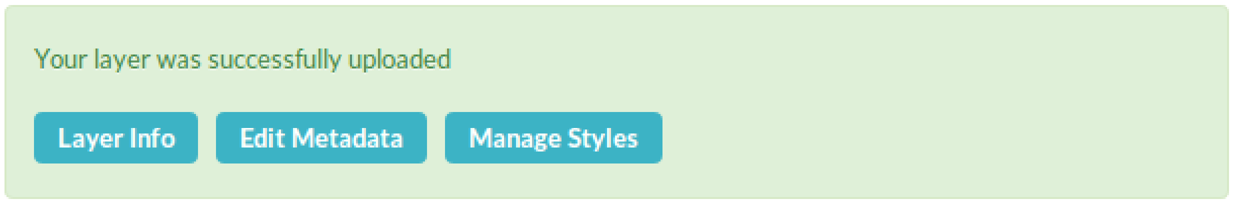
Fig. 3.27: *Permissions for new layer*



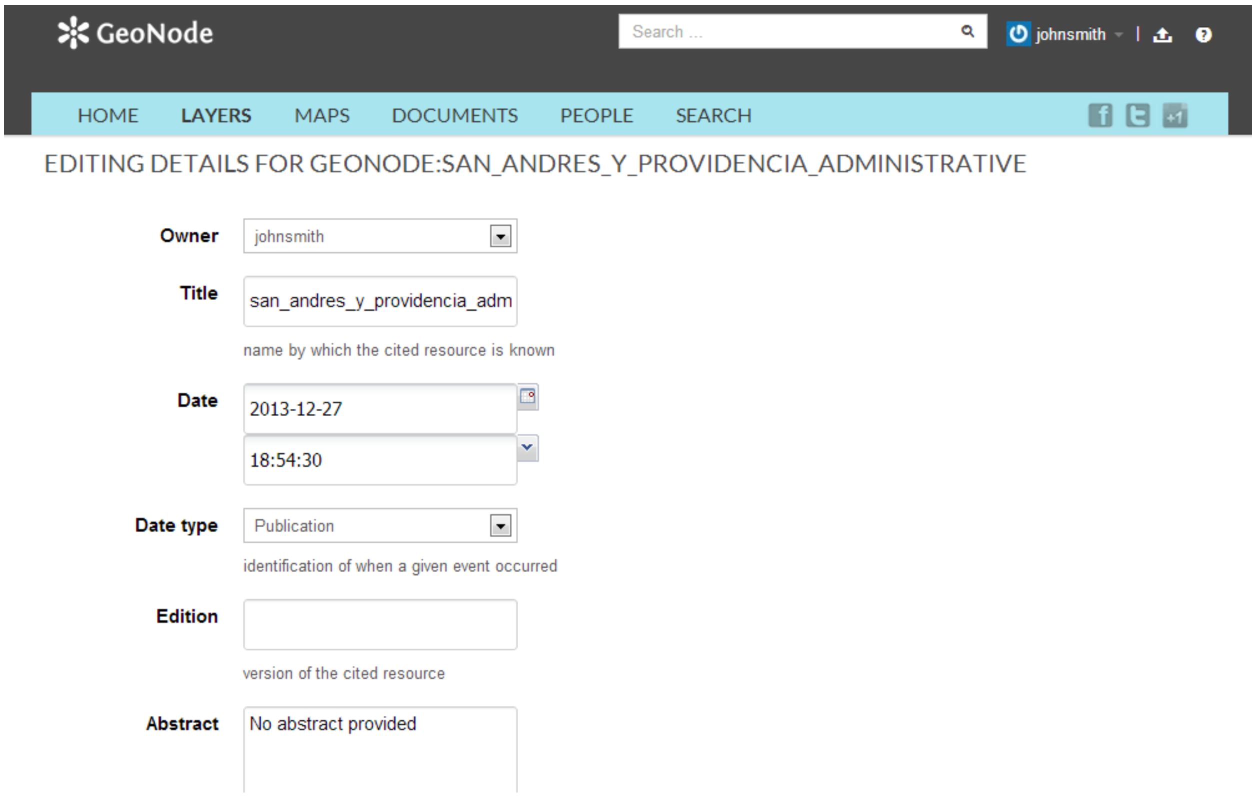
Your upload has started

49.0%

Fig. 3.28: *Upload in progress*



Your layer was successfully uploaded



EDITING DETAILS FOR GEONODE:SAN_ANDRES_Y_PROVIDENCIA_ADMINISTRATIVE

Owner	johnsmith
Title	san_andres_y_providencia_adm
	name by which the cited resource is known
Date	2013-12-27
	18:54:30
Date type	Publication
	identification of when a given event occurred
Edition	
	version of the cited resource
Abstract	No abstract provided

Fig. 3.29: *Layer metadata*

After the update, the layer will display in a preview window.

This page contains lots of options for managing this layer. Let's look at a few of them:

Downloads At the top of the page there are two buttons titled *Download Layer* and *Download Metadata*. These buttons provide access to the ability to extract geospatial data and metadata from within GeoNode. In this way, GeoNode allows for two way data and metadata access; one can import as well as export data.

Data

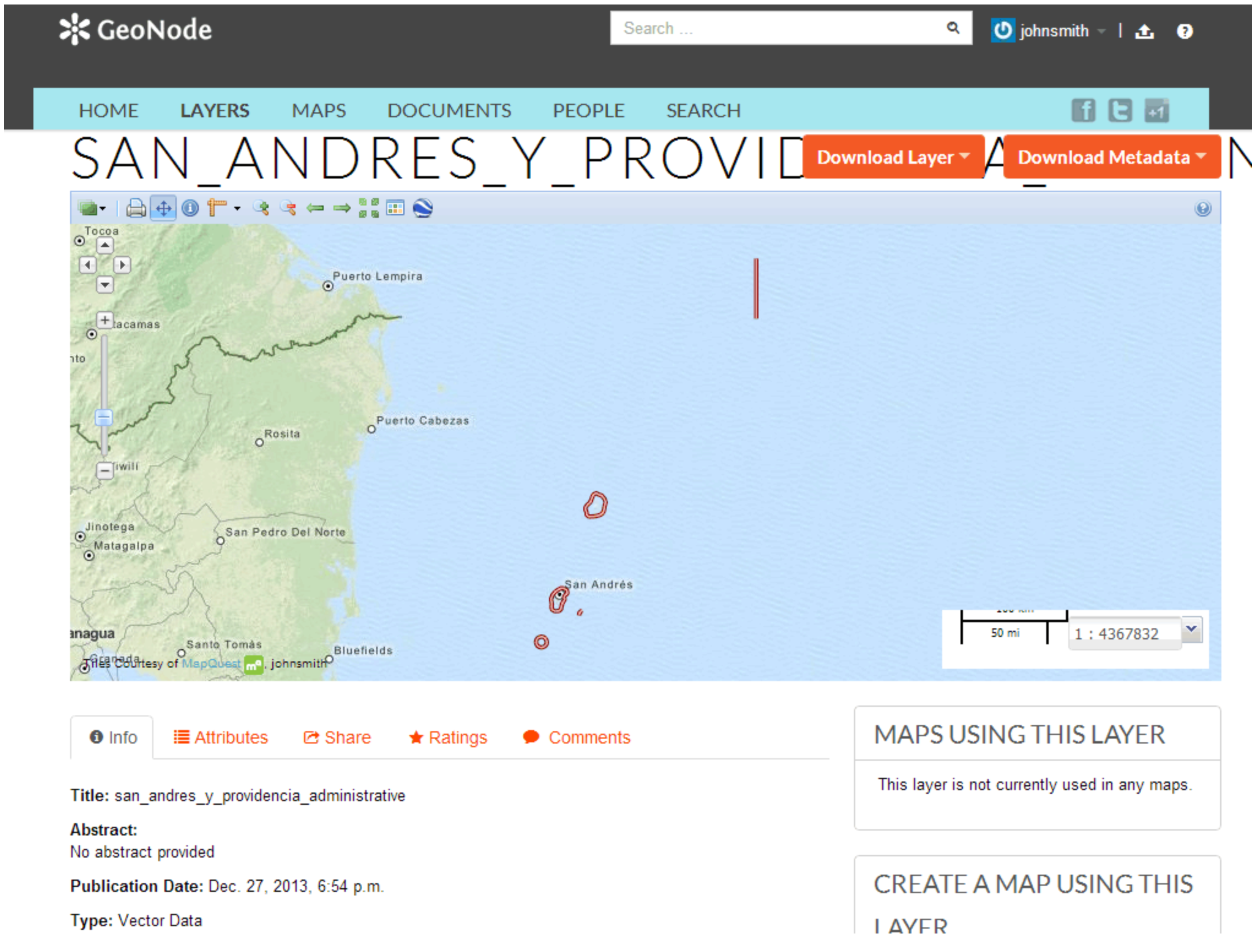
1. Click the *Download Layer* button. You will see a list of options of the supported export formats.

1. Click the option for *Zipped Shapefile*.
2. GeoNode will process the request and bring up a Save As dialog. Save this file to your computer, and note how it is the same content as was uploaded.

Metadata

1. Click the *Download Metadata* button. You will see a list of options of the supported export formats.

1. Click the option for *DUBLIN CORE*.
2. GeoNode will process the request and display XML metadata. Try clicking various metadata formats, and note how it is the same metadata content in various formats compatible with metadata and GIS packages.



This layer is not currently used in any maps.

Title: san_andres_y_providencia_administrative

Abstract:
No abstract provided

Publication Date: Dec. 27, 2013, 6:54 p.m.

Type: Vector Data

Fig. 3.30: *Layer preview*

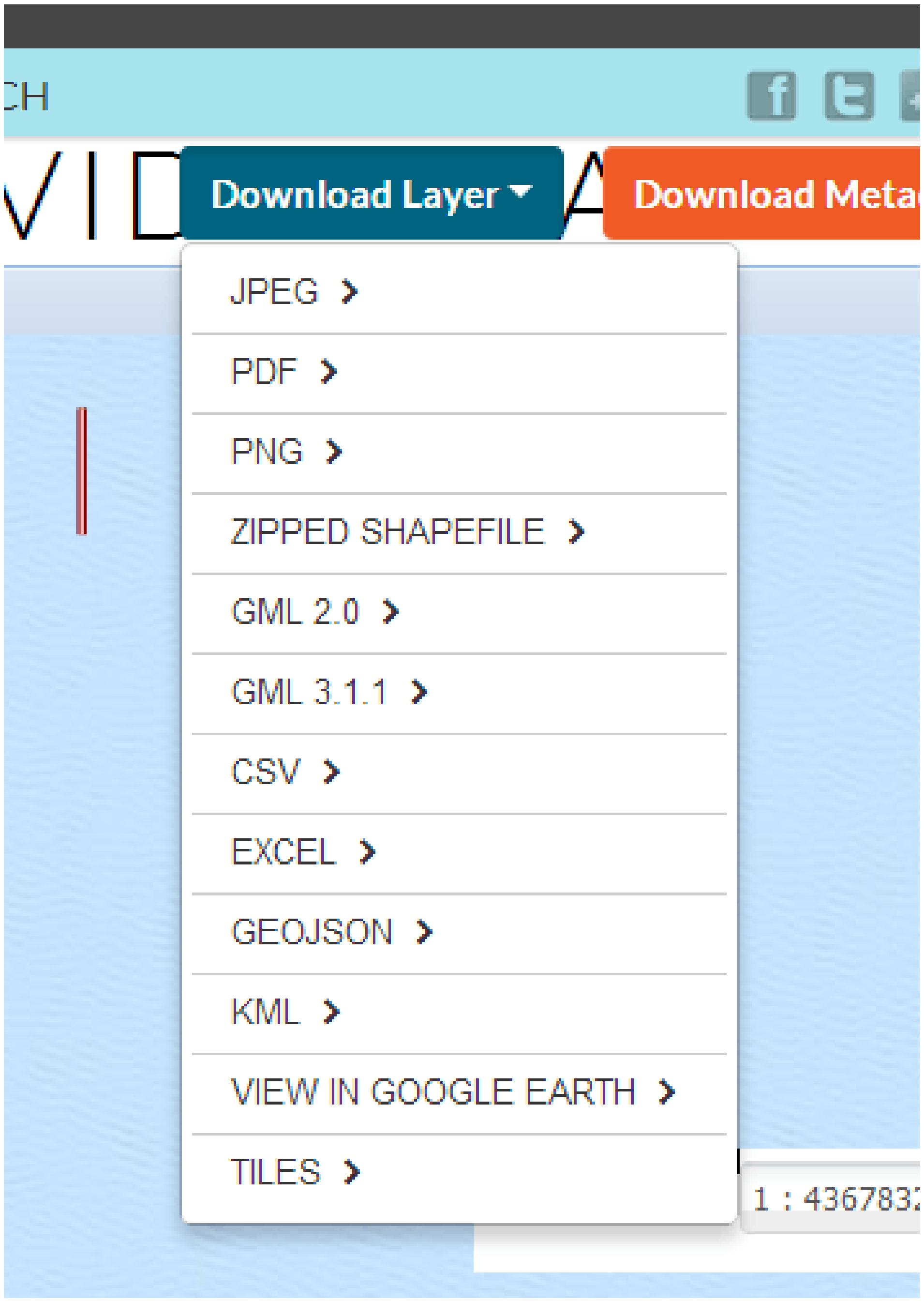


Fig. 3.31: *Available export formats*

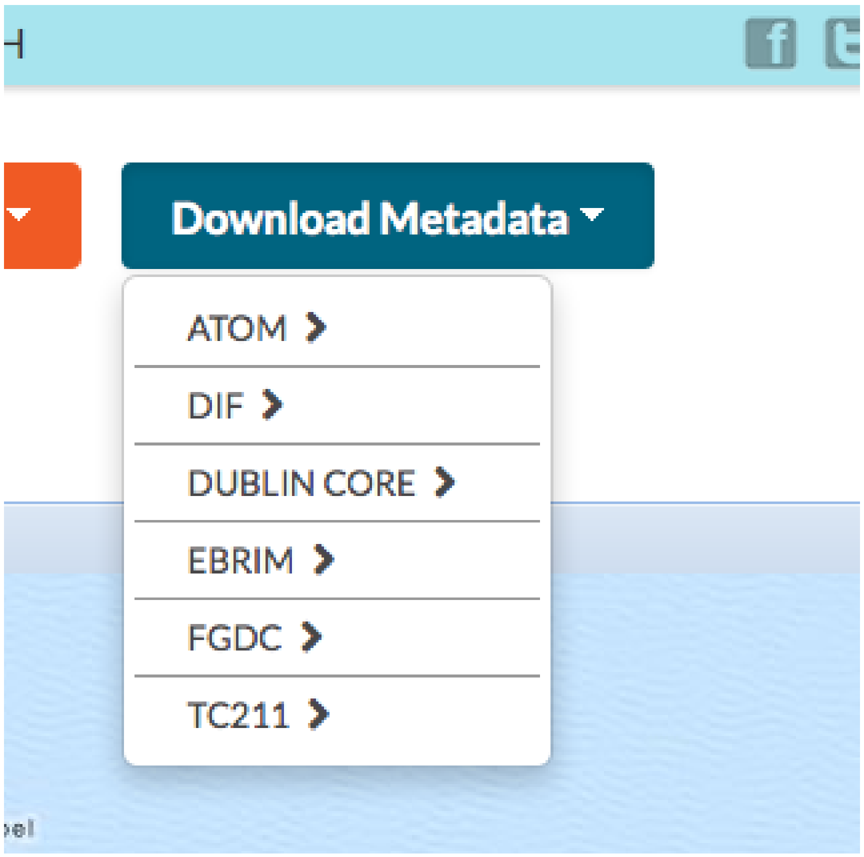
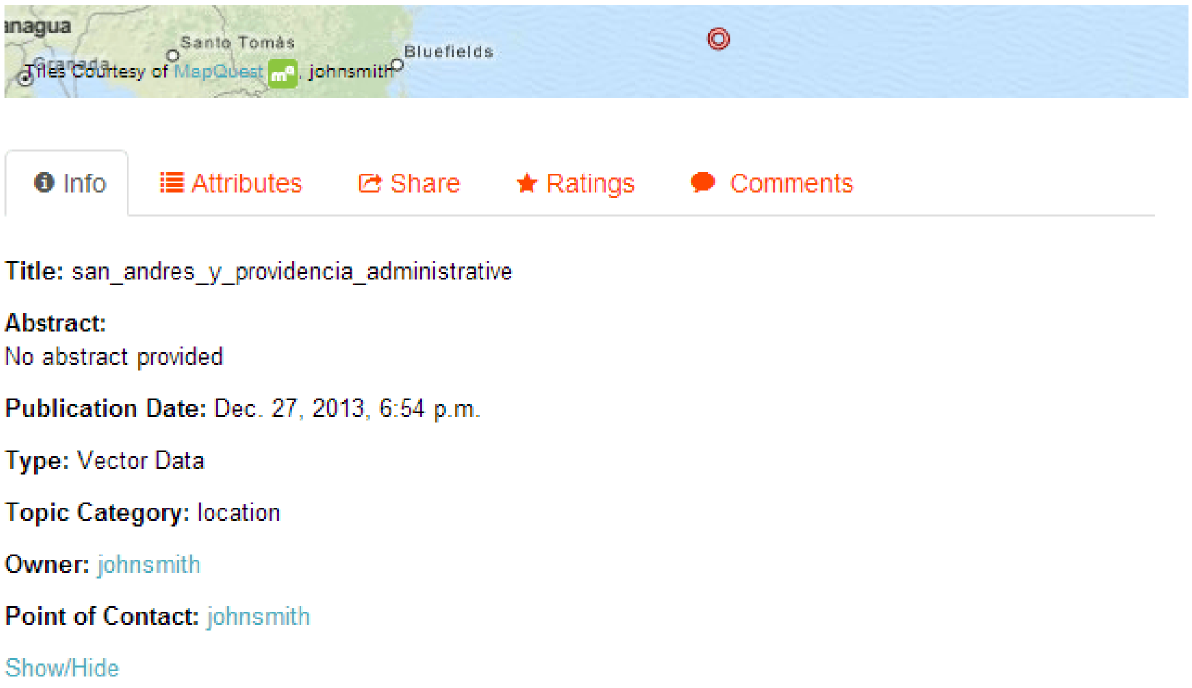


Fig. 3.32: *Available metadata export formats*

Layer Detail Tabs

1. Scroll down the page toward the bottom. Five tabs are available: *Info*, *Attributes*, *Share*, *Ratings*, and *Comments*. The info tab is already highlighted, and presents basic information about the layer, of the kind that was seen on the layer list page.



Title: san_andres_y_providencia_administrative

Abstract:
No abstract provided

Publication Date: Dec. 27, 2013, 6:54 p.m.

Type: Vector Data

Topic Category: location

Owner: johnsmith

Point of Contact: johnsmith

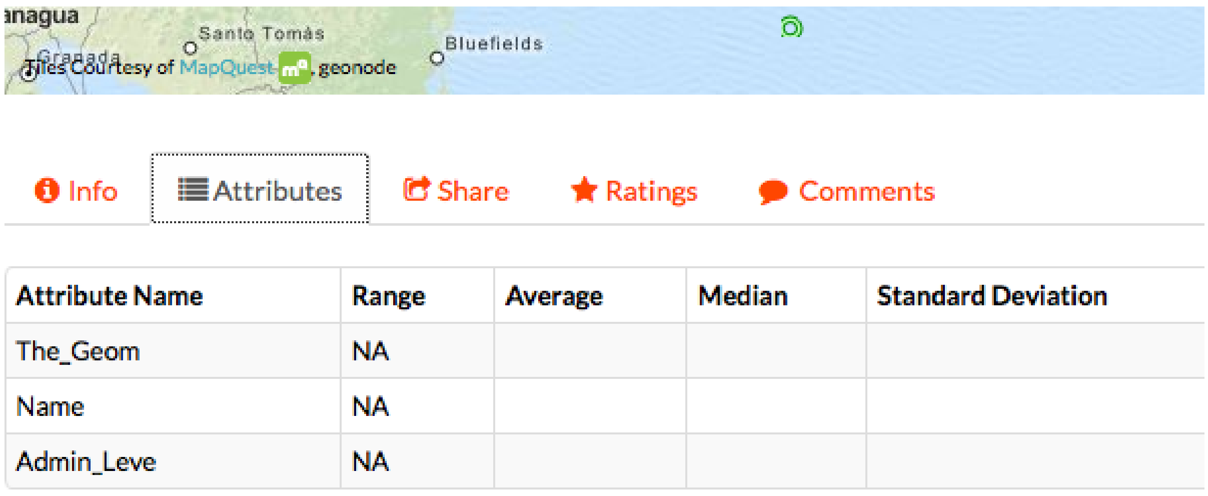
Fig. 3.33: *Layer Info tab*

2. Click the *Attributes* tab. This lists the attributes of the layer, including statistics (range, average, median and standard deviation). Layer attribute statistics are made available only for numeric attributes. As we can see, this layer's attributes are not numeric, so no statistics are calculated.
3. Click the *Ratings* tab. This tab allows you (and others viewing this page) to rate this layer. Ratings can be based on quality, accuracy, or any other metric. Click on the appropriate star to rate this layer.
4. Click the *Comments* tab. This tab allows you to leave a comment for other viewing this layer.
5. Click the *Add Comment* button and enter a comment.
6. When finished, click *Submit Comments*

Sharing layers GeoNode has the ability to restrict or allow other users to access a layer and share on social media.

Anonymous access

1. Go to the layer preview of the first layer uploaded, and copy the URL to that preview page.



Attribute Name	Range	Average	Median	Standard Deviation
The_Geom	NA			
Name	NA			
Admin_Leve	NA			

Fig. 3.34: *Attributes tab*

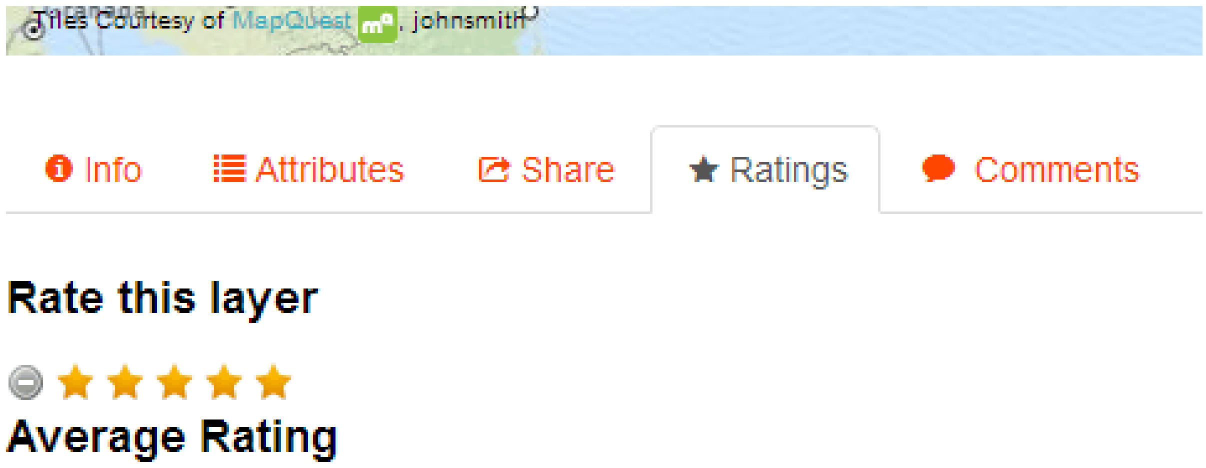


Fig. 3.35: *Layer Ratings tab*

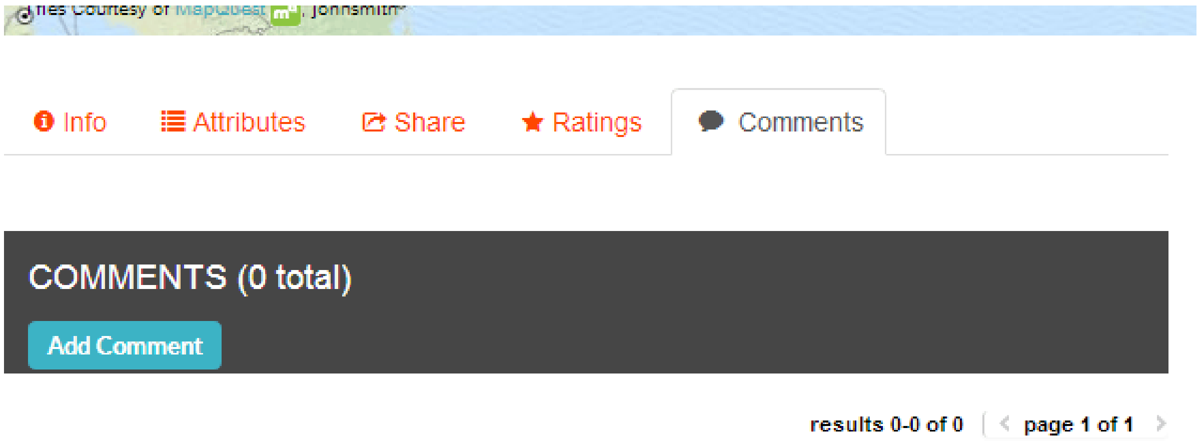
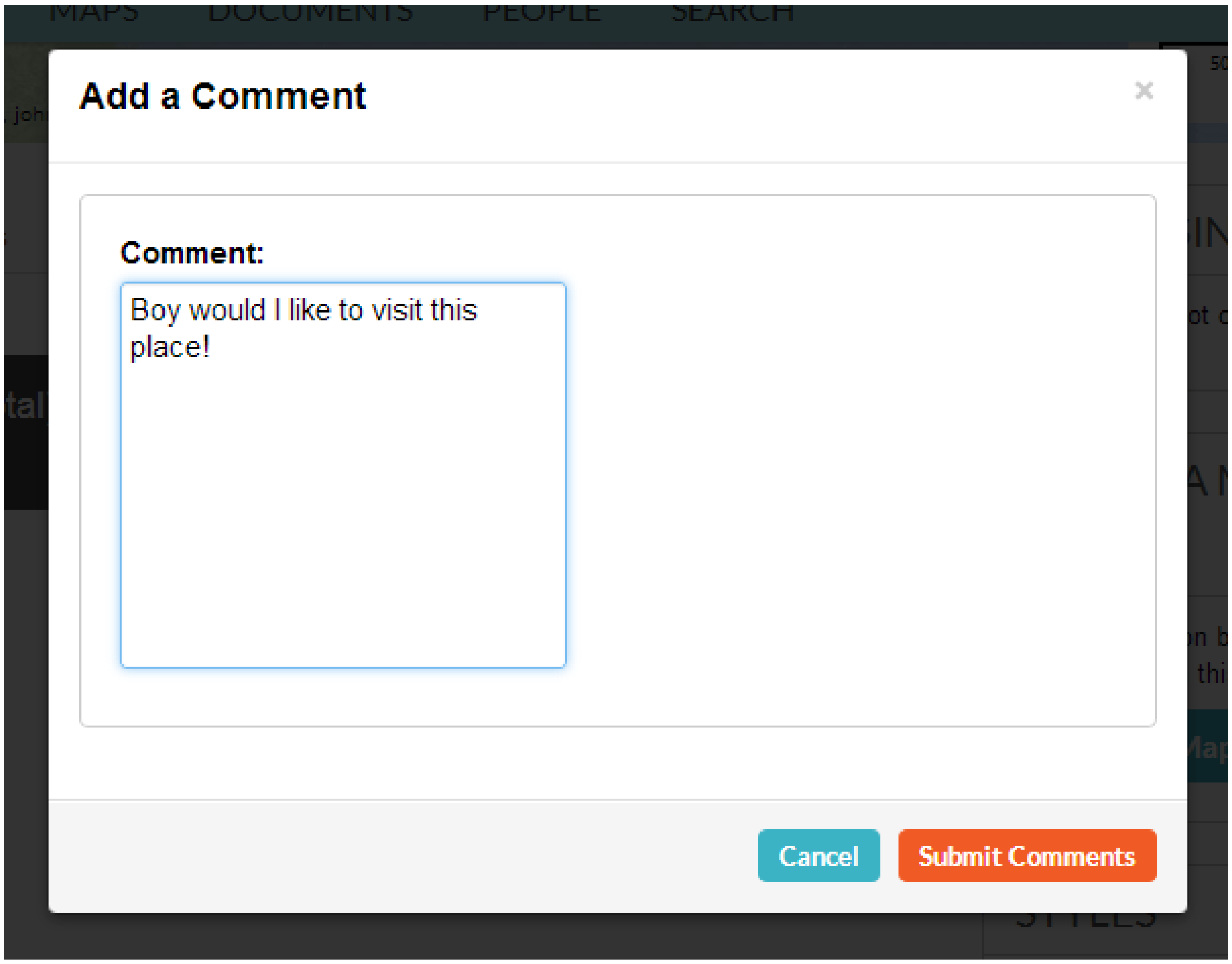
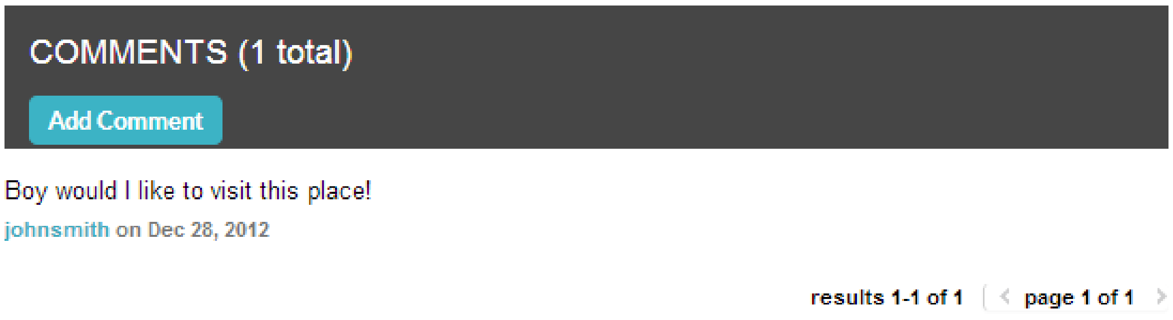


Fig. 3.36: *Layer Comments tab*



Boy would I like to visit this place!

Fig. 3.37: *Adding a new comment*


COMMENTS (1 total)

Add Comment

Boy would I like to visit this place!

johnsmith on Dec 28, 2012

results 1-1 of 1 | page 1 of 1

Fig. 3.38: *New comment posted*

Note: The URL should be something like: `http://GEONODE/layers/geonode:san_andres_y_providencia_adr`

2. Now log out of GeoNode by clicking on your profile name and selecting *Log out*.

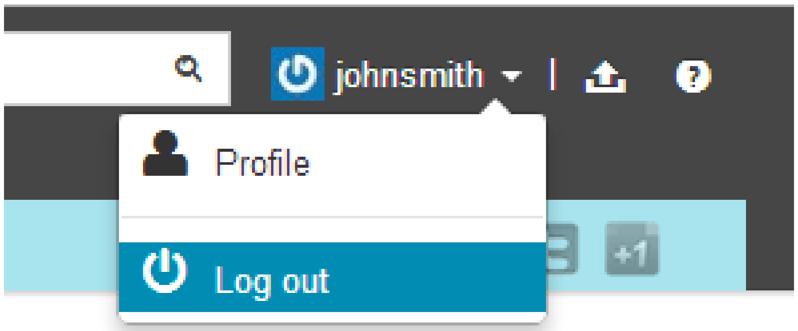
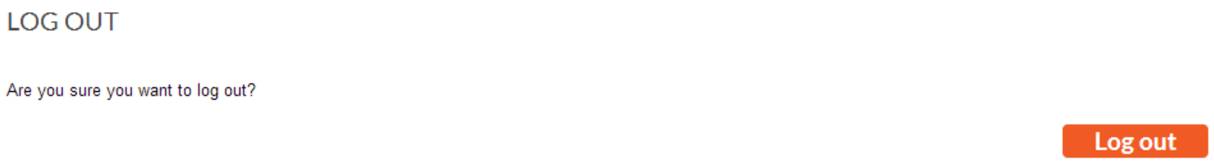


Fig. 3.39: *Log out*

3. When asked for confirmation, click the *Log out* button.


LOG OUT

Are you sure you want to log out?

Log out

Fig. 3.40: *Confirming log out*

4. Now paste the URL copied about into your browser address bar and navigate to that location.
5. You will be redirected to the Log In form. This is because when this layer was first uploaded, we set the view properties to be any registered user. Once logged out, we are no longer a registered user and so are not able to see or interact with the layer, unless we log in GeoNode again.

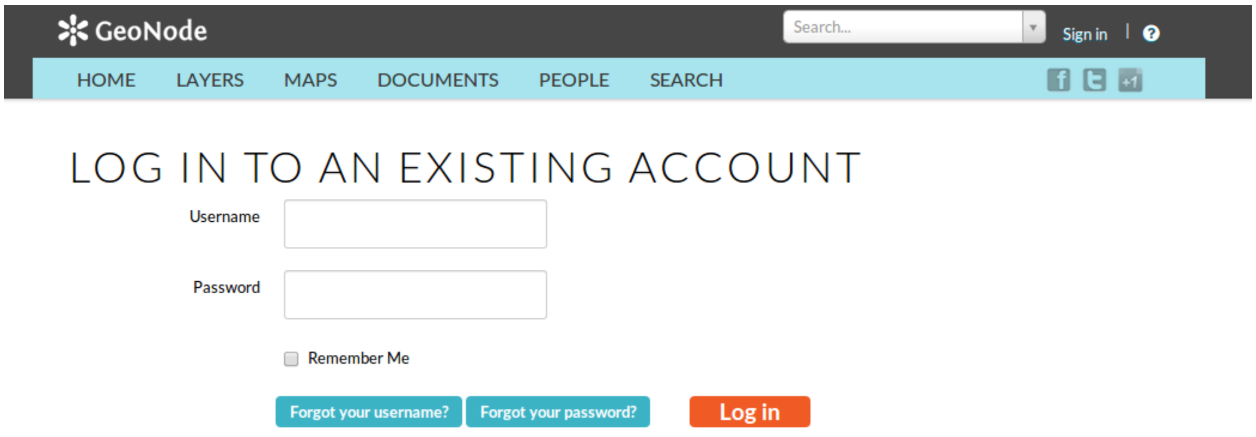


Fig. 3.41: *Unable to view this protected layer*

6. To stop this process from happening, you need to ensure that your permissions are set so *anyone* can view the layer for others to see it on social networks.

1. This is done by selecting *anyone* in the layer permissions tab, be aware this now means your layer is public!

Sharing with social media

1. On the taskbar below your username and profile picture there are three links to social media services, Twitter, Google Plus and Facebook.
2. Upon clicking on these icons you will be taken through the application's process for posting to the social network. Ensure the permissions are set so *anyone* can view the layer if you want unauthenticated to be able to access it.

Adding more layers We've uploaded one layer so far. There is one more layer in the `data` directory associated with this workshop called `san_andres_y_providencia_poi.shp`.

1. Upload this layer, referring to the directions on *uploading a layer*. As a difference, leave the permissions set to their default values.

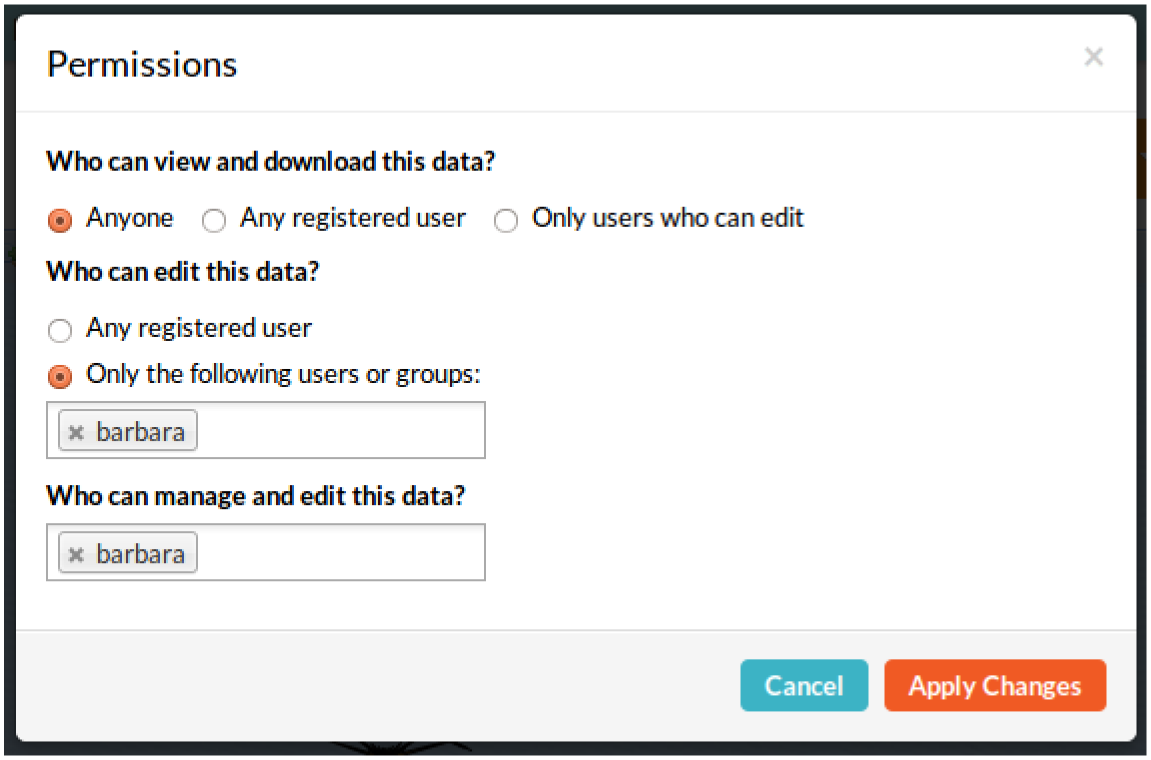
Managing maps

The next primary component of GeoNode is the **map**. Maps are comprised of various layers and their styles. Layers can be both local layers in GeoNode as well as remote layers either served from other WMS servers or by web service layers such as Google or MapQuest.

GeoNode maps also contain other information such as map zoom and extent, layer ordering, and style.

In this section, we'll create a map based on the layers uploaded in the previous section, combine them with some existing layers and a remote web service layer, and then share the resulting map for public viewing.

Creating a map



Permissions

Who can view and download this data?

[✓ Anyone] Any registered user Only users who can edit

Who can edit this data?

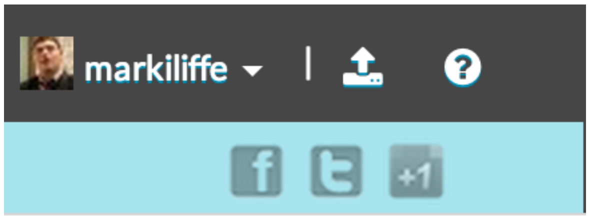
Any registered user

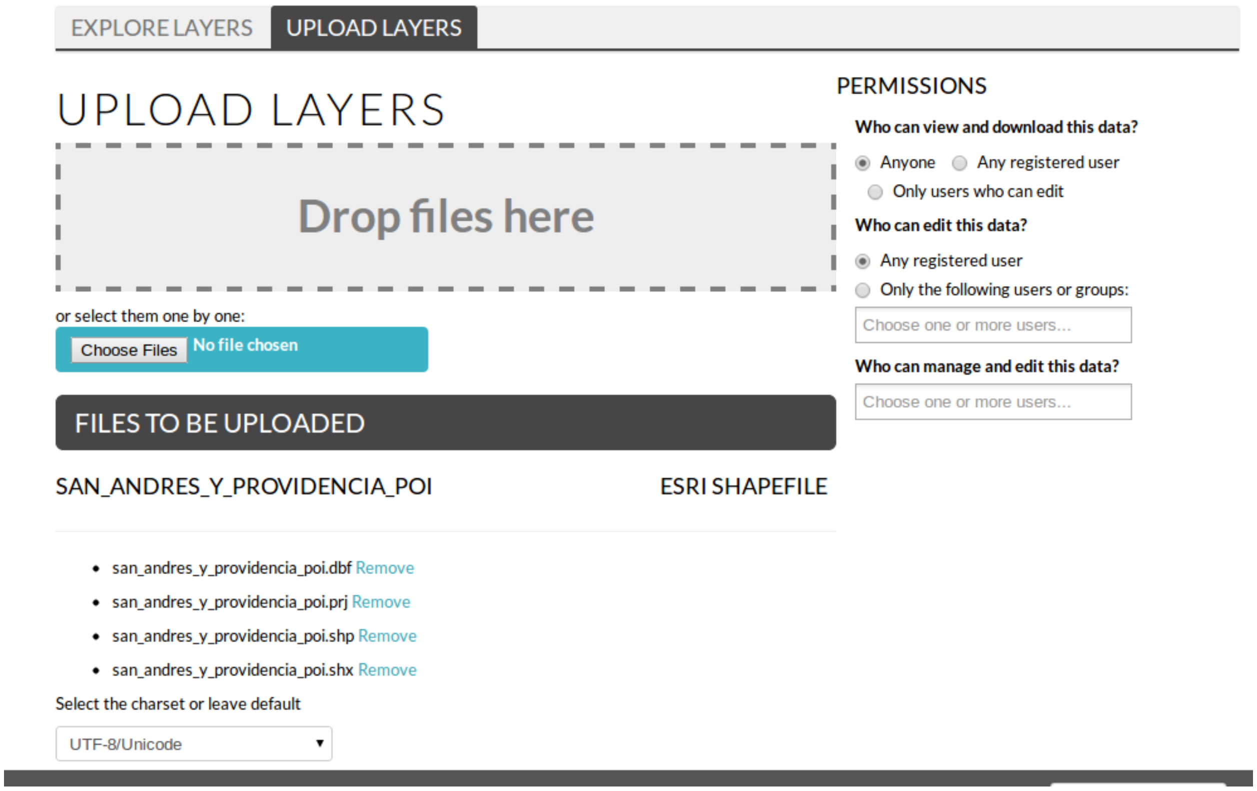
[✓ Only the following users or groups:]

barbara

Who can manage and edit this data?

barbara





UPLOAD LAYERS

or select them one by one:

FILES TO BE UPLOADED

SAN_ANDRES_Y_PROVIDENCIA_POI ESRI SHAPEFILE

- san_andres_y_providencia_poi.dbf Remove
- san_andres_y_providencia_poi.prj Remove
- san_andres_y_providencia_poi.shp Remove
- san_andres_y_providencia_poi.shx Remove

Select the charset or leave default

UTF-8/Unicode

PERMISSIONS

Who can view and download this data?

[✓ Anyone] Any registered user Only users who can edit

Who can edit this data?

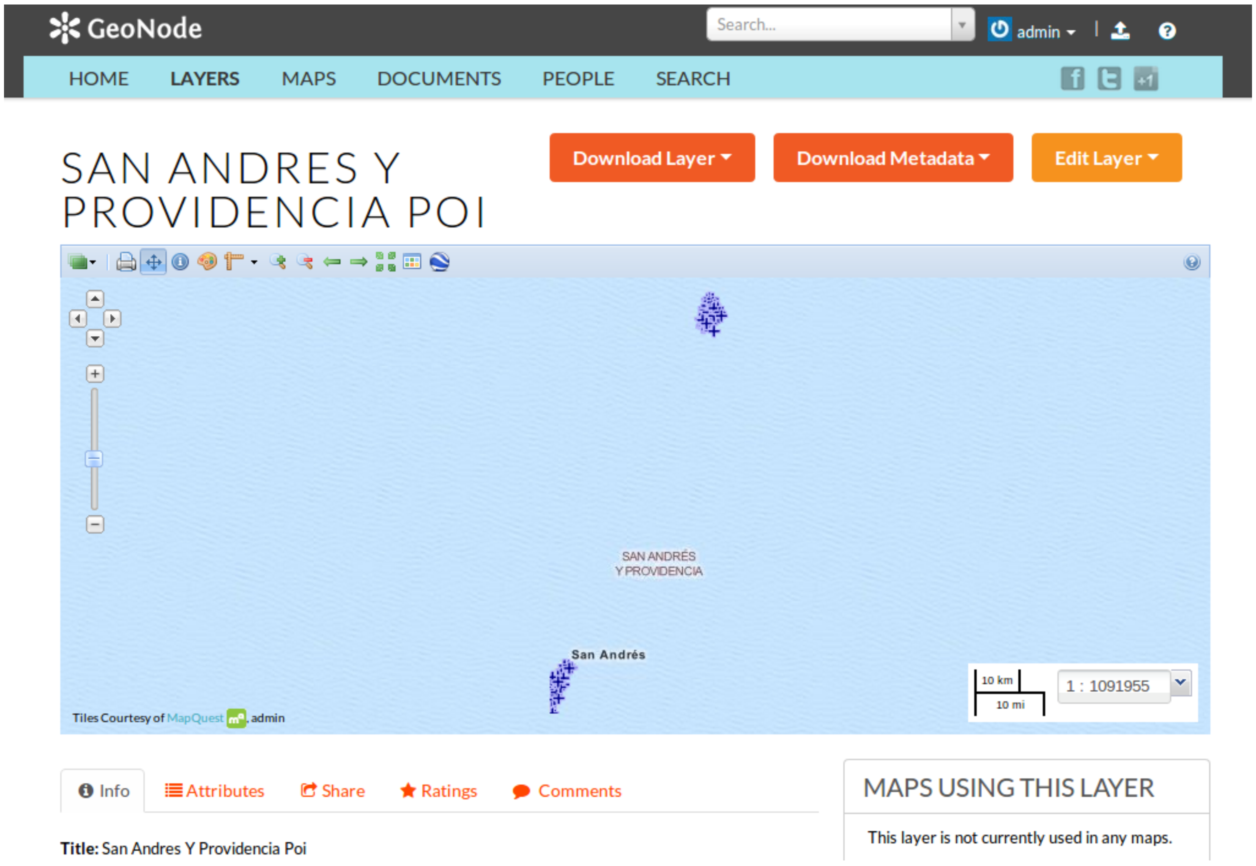
[✓ Any registered user] Only the following users or groups:

Choose one or more users...

Who can manage and edit this data?

Choose one or more users...

Fig. 3.42: *Uploading the layer*



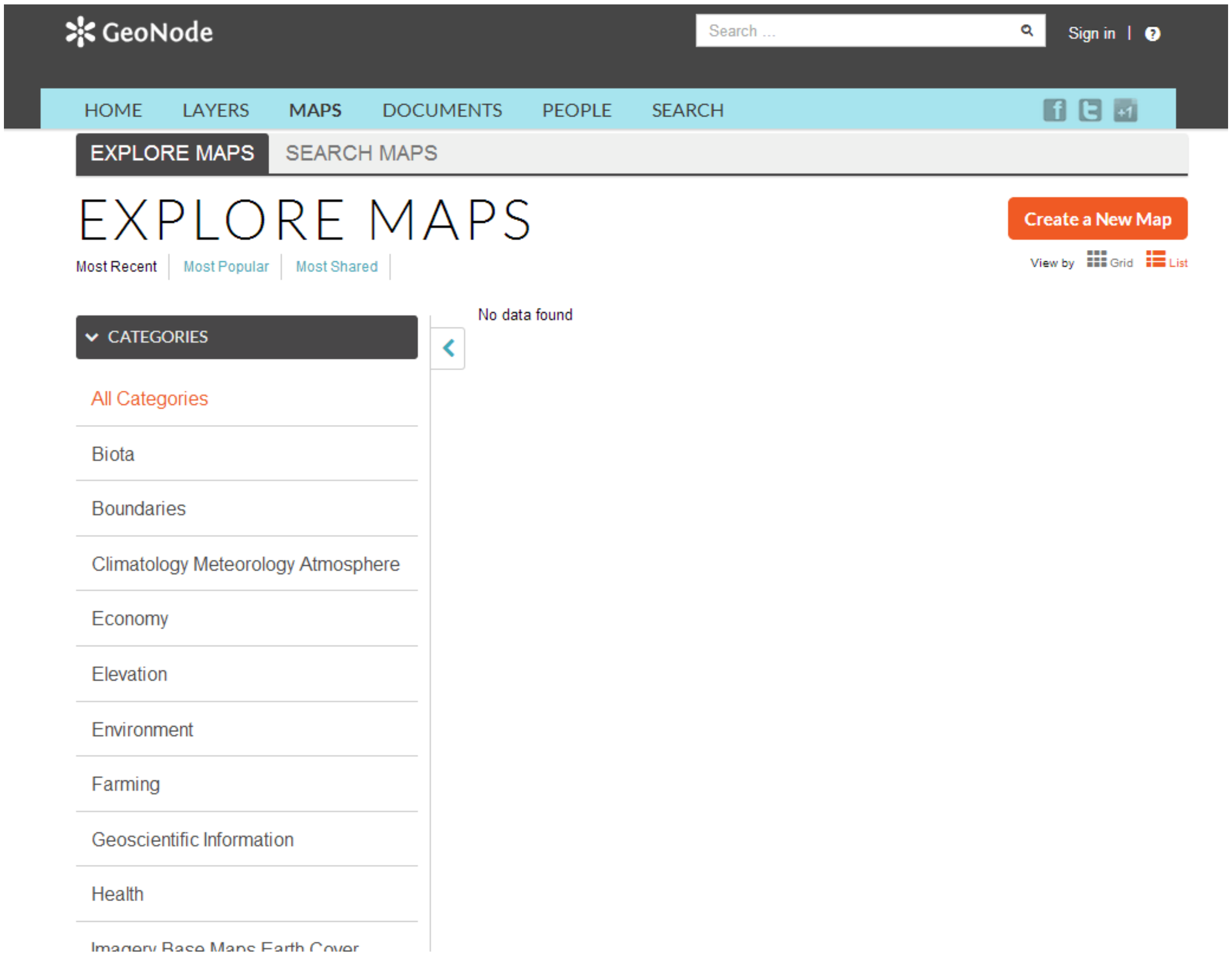
This layer is not currently used in any maps.

Title: San Andres Y Providencia Poi

Fig. 3.43: *Finished upload*

Adding layers

1. Click the *Maps* link on the top toolbar. This will bring up the list of maps.



No data found

Fig. 3.44: *Maps page*

2. Currently, there aren't any maps here, so let's add one. Click the *Create a New Map* button.
3. A map composition interface will display.

   In this interface there is a toolbar, layer list, and map window. The map window contains the MapQuest OpenStreetMap layer by default. There are other service layers available here as well: Blue Marble, Bing Aerial With Labels, MapQuest, and OpenStreetMap.
4. Click on the New Layers button and select Add Layers.
5. Select all of the San Andreas layers by clicking the top entry and Shift-clicking the bottom one. Click *Add Layers* to add them all to the map.

   Note: This selection includes not only the two layers uploaded in the previous section, but also the layers that were already hosted on GeoNode at the beginning of the workshop.

6. The layers will be added to the map. Click *Done* (right next to *Add Layers* at the bottom) to return to the main layers list.

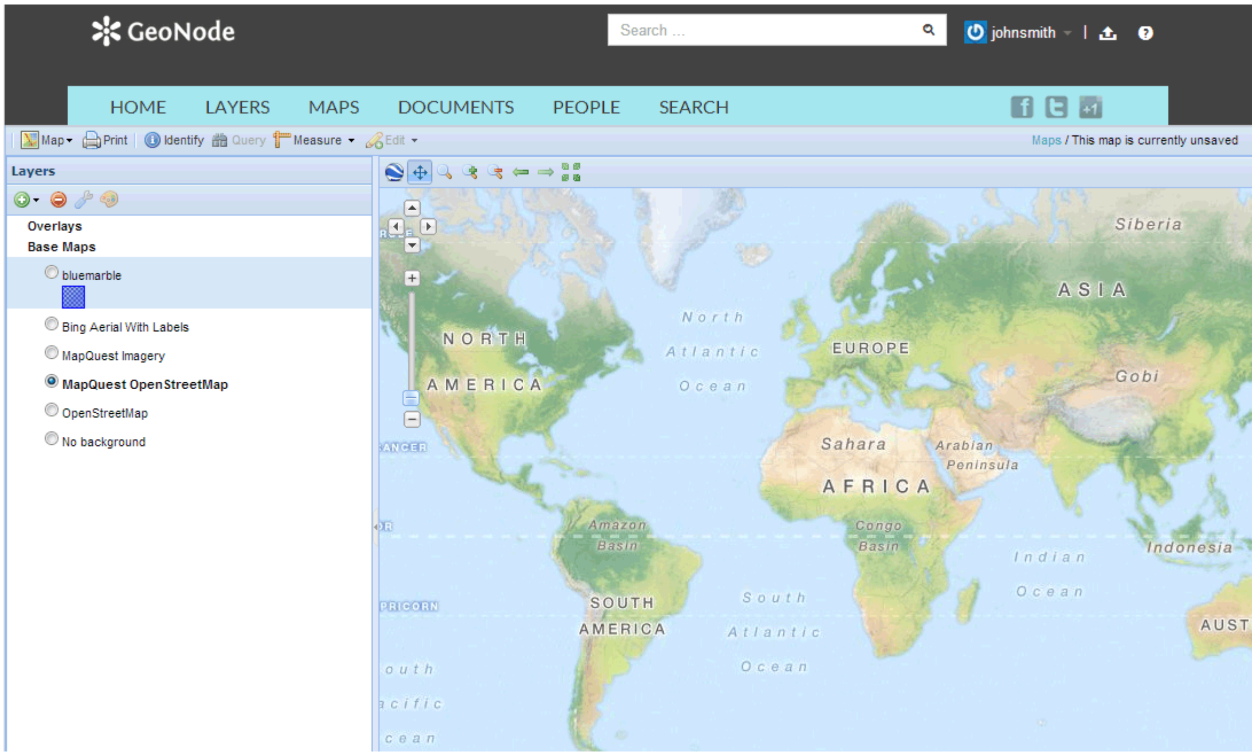


Fig. 3.45: *Create maps interface*

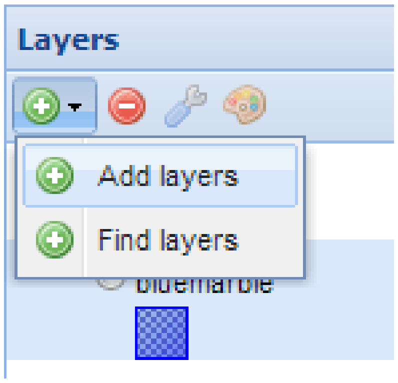
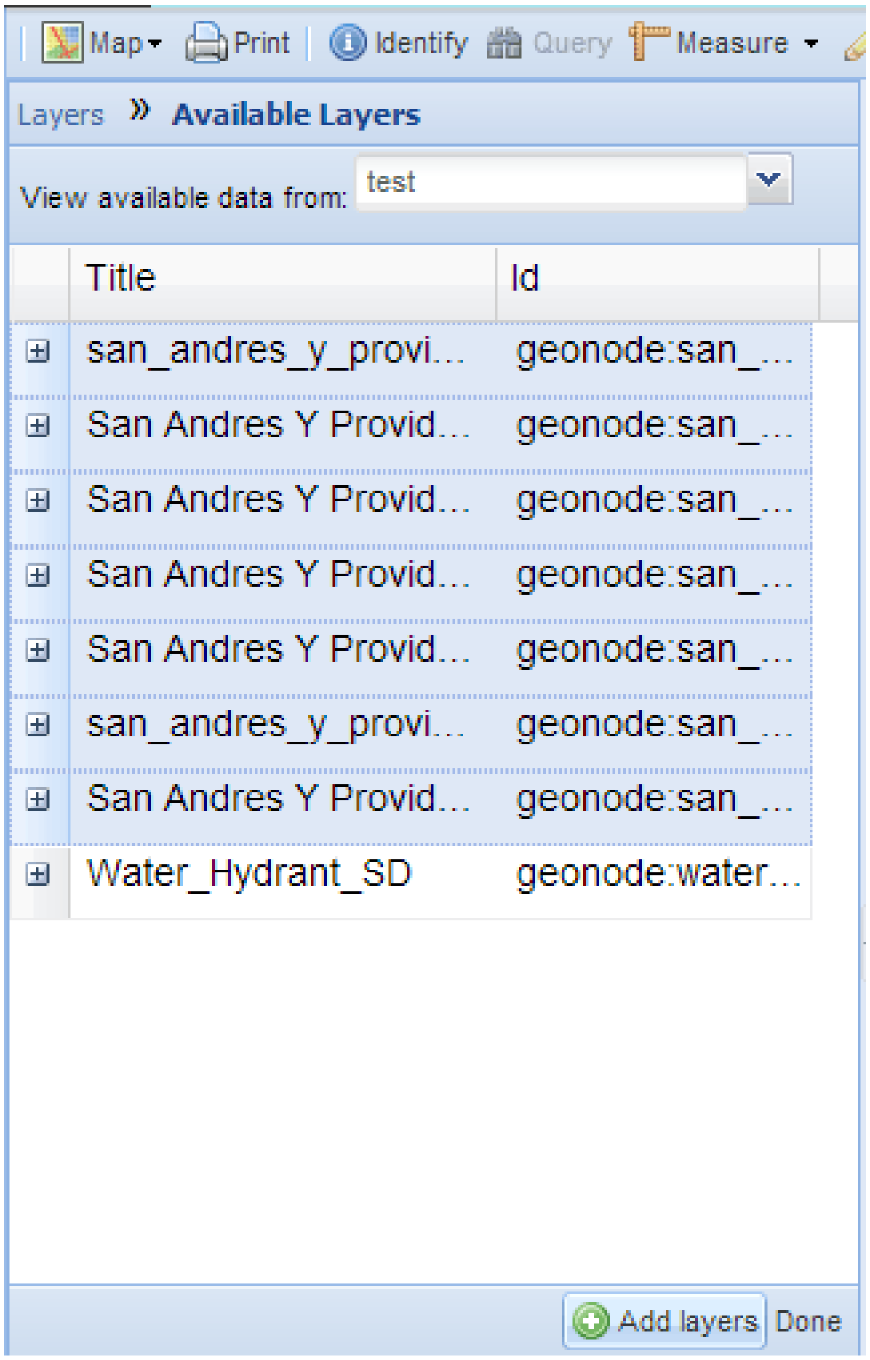


Fig. 3.46: *Add layers link*



View available data from: test

Title	Id
san_andres_y_provi...	geonode:san_...
San Andres Y Provid...	geonode:san_...
San Andres Y Provid...	geonode:san_...
San Andres Y Provid...	geonode:san_...
San Andres Y Provid...	geonode:san_...
san_andres_y_provi...	geonode:san_...
San Andres Y Provid...	geonode:san_...
Water_Hydrant_SD	geonode:water...

Fig. 3.47: *Selecting layers*

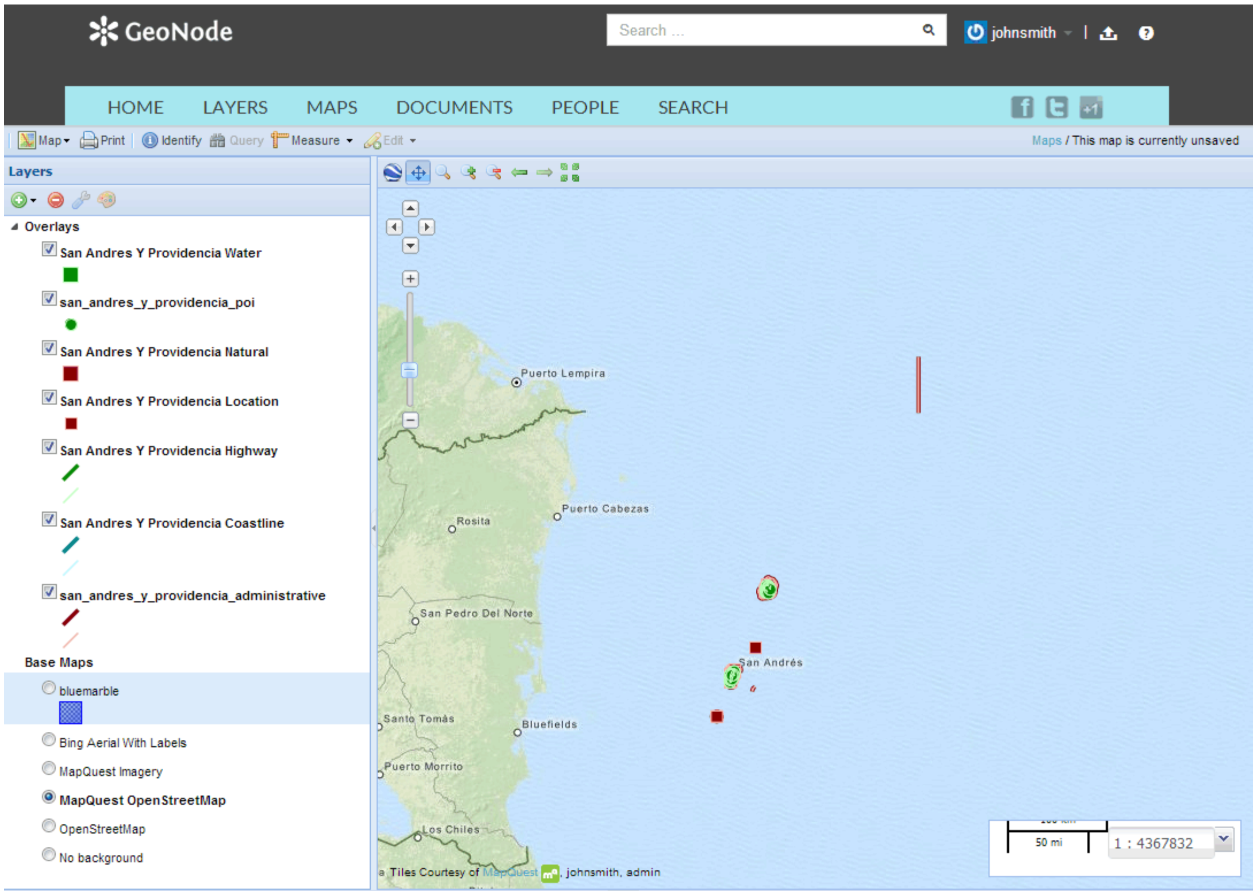


Fig. 3.48: *Layers added to the map*

Adding external layers

1. Once again, click on the New Layers button and select Add Layers.

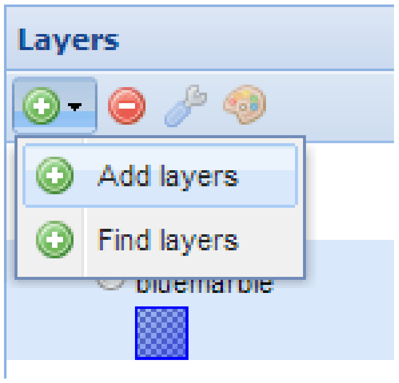


Fig. 3.49: *Add layers link*

2. From the top dropdown list, select *Add a New Server...*

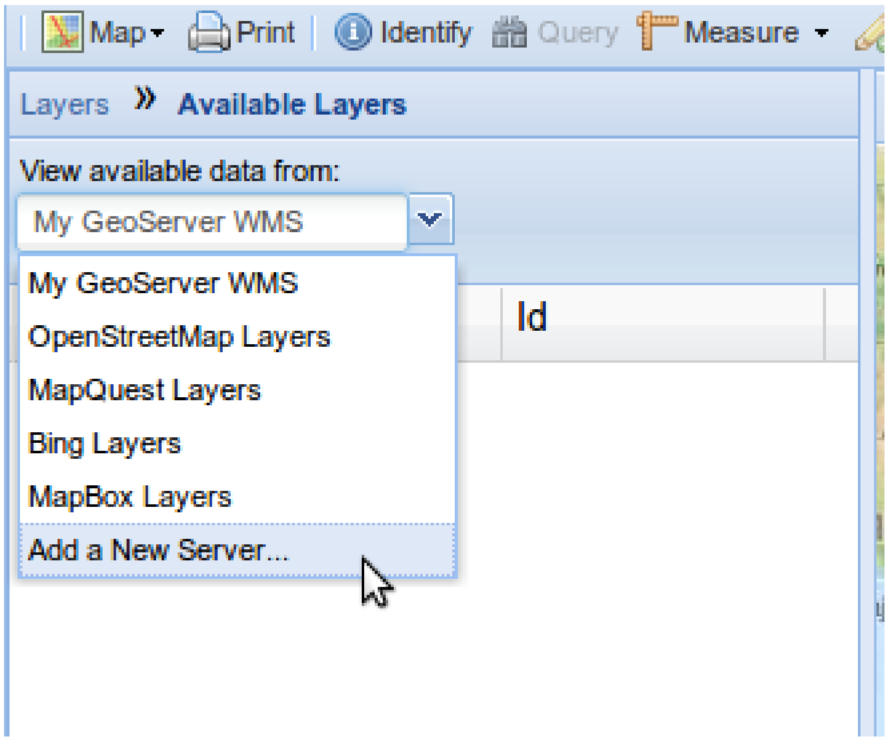
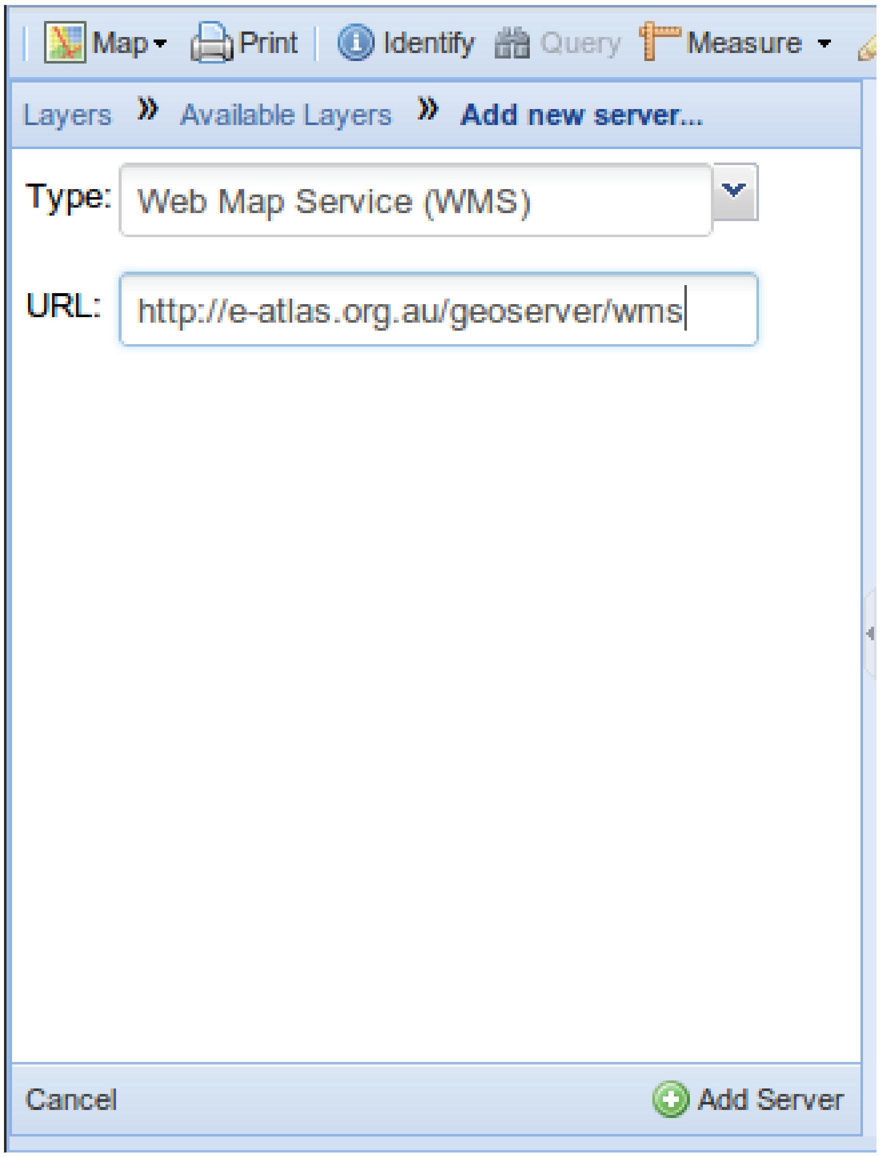


Fig. 3.50: *Add a New Server*

3. Enter the URL of the server, and select the correct type of server from the dropdown (WMS, TMS, or ArcGIS). For example, enter *http://e-atlas.org.au/geoserver/wms* for the URL and select *Web Map Service* as the type. Then click the *Add Server* button.
4. Note - for security purposes, the URL you enter must be on a list of pre-approved external services set up by the GeoNode admininistrator. Otherwise you will receive a 403 error when trying to add the server.
5. A list of layers available from that server should appear momentarily. The layers must be available in the Web

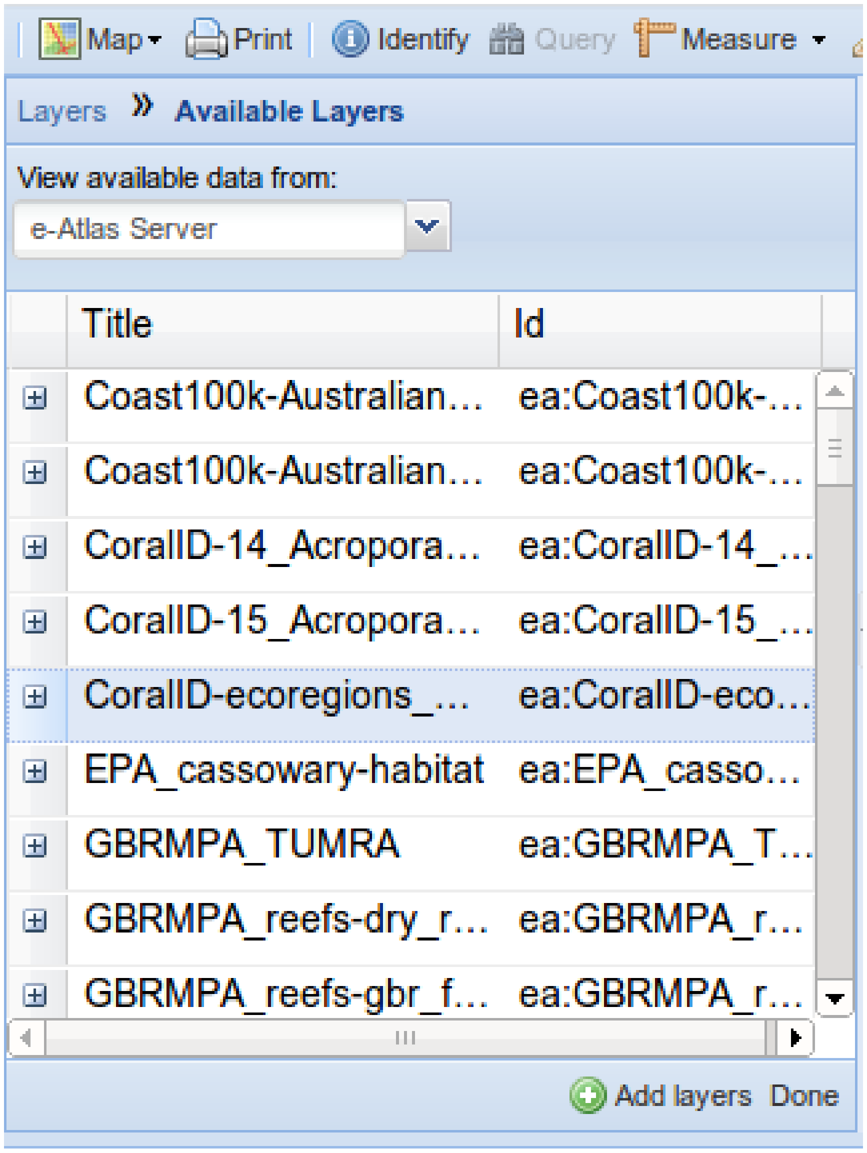


Type: Web Map Service (WMS)

URL: http://e-atlas.org.au/geoserver/wms

Fig. 3.51: *New Server URL and Type*

Mercator projection or they will not show up in the list. Select the layers you want to add to the map. Click *Add Layers* to add them all to the map.



Title	Id
Coast100k-Australian…	ea:Coast100k-…
Coast100k-Australian…	ea:Coast100k-…
CoralID-14_Acropora…	ea:CoralID-14_…
CoralID-15_Acropora…	ea:CoralID-15_…
CoralID-ecoregions_…	ea:CoralID-eco…
EPA_cassowary-habitat	ea:EPA_casso…
GBRMPA_TUMRA	ea:GBRMPA_T…
GBRMPA_reefs-dry_r…	ea:GBRMPA_r…
GBRMPA_reefs-gbr_f…	ea:GBRMPA_r…

Fig. 3.52: *Add layers*

6. The layers will be added to the map. Click *Done* (right next to *Add Layers* at the bottom) to return to the main layers list.

Saving the map

1. While we still have some work to do on our map, let's save it so that we can come back to it later. Click on the *Map* button in the toolbar, and select *Save Map*.
2. Enter a title and abstract for your map.
3. Click *Save*. Notice that the link on the top right of the page changed to reflect the map's name.

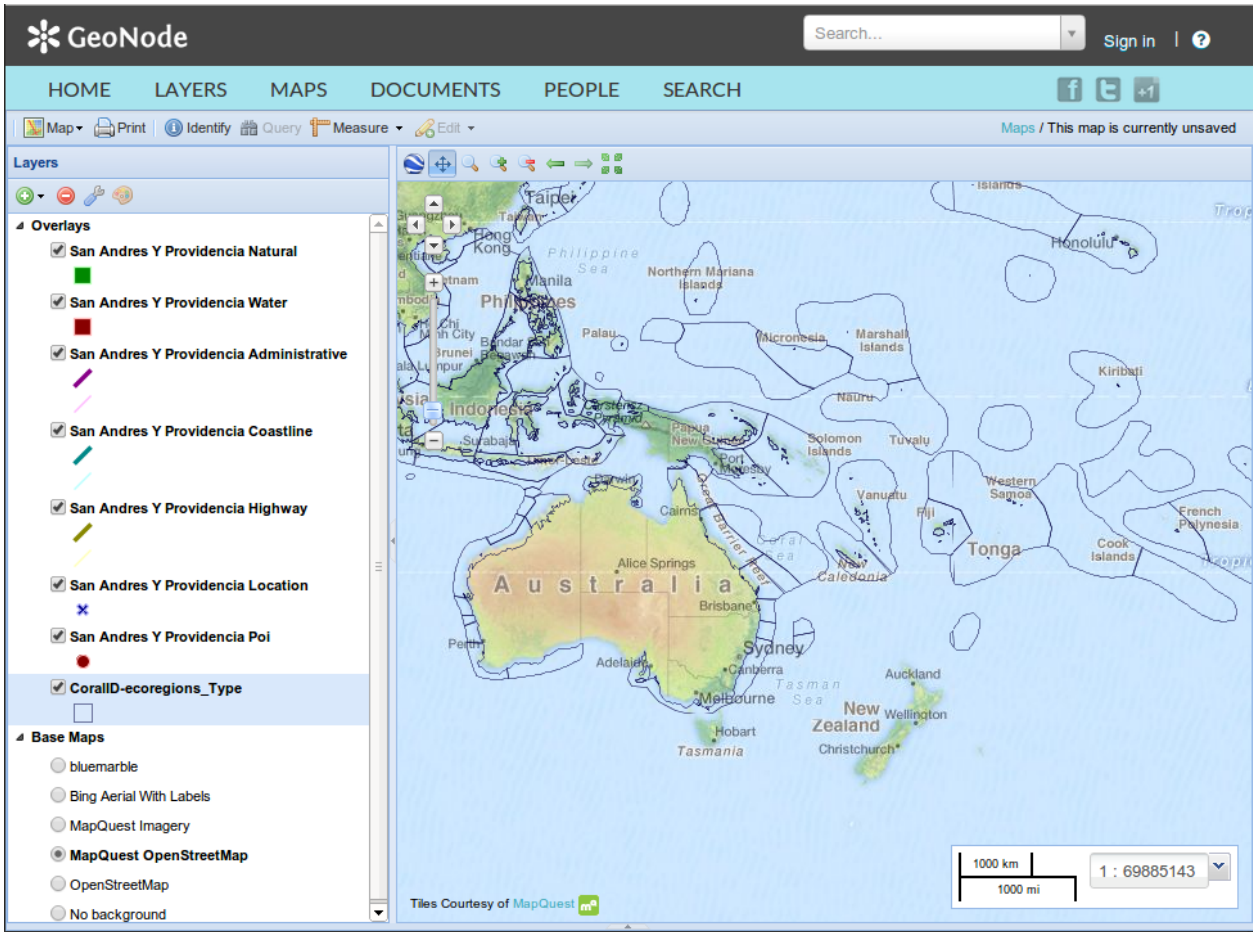

GeoNode
Search...
Sign in
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
Map
Print
Identify
Query
Measure
Edit
Maps / This map is currently unsaved
Layers
Overlays
San Andres Y Providencia Natural
San Andres Y Providencia Water
San Andres Y Providencia Administrative
San Andres Y Providencia Coastline
San Andres Y Providencia Highway
San Andres Y Providencia Location
San Andres Y Providencia Poi
CoralID-ecoregions_Type
Base Maps
bluemarble
Bing Aerial With Labels
MapQuest Imagery
MapQuest OpenStreetMap
OpenStreetMap
No background
1000 km
1000 mi
1 : 69885143
Tiles Courtesy of MapQuest

Fig. 3.53: *Layers added to the map*

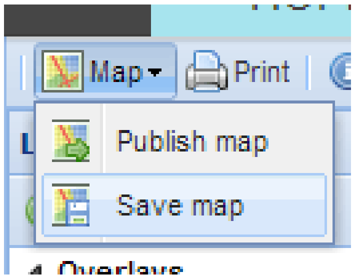
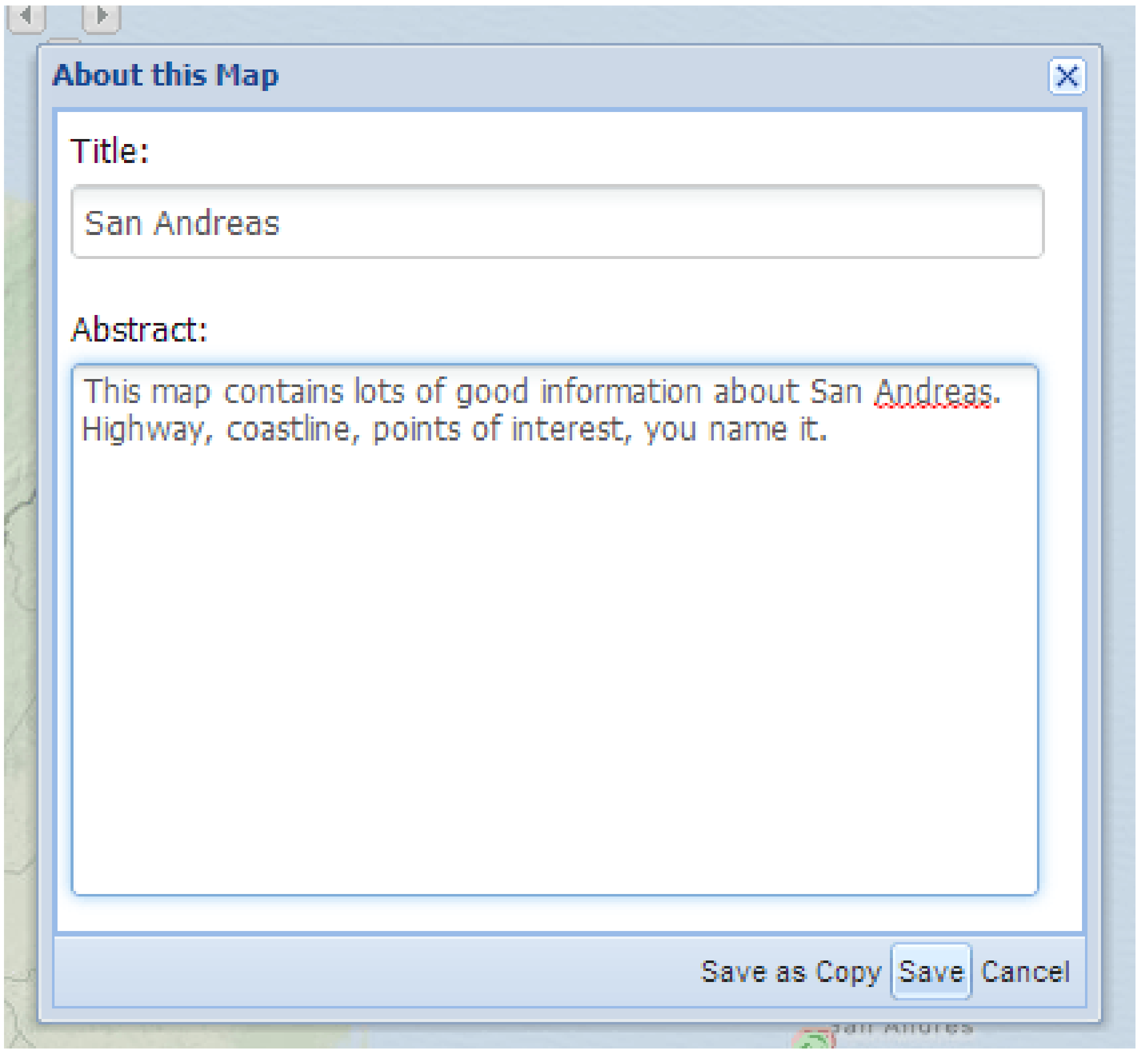


Fig. 3.54: *Save map link*


About this Map

Title:

San Andreas

Abstract:

This map contains lots of good information about San Andreas. Highway, coastline, points of interest, you name it.

Fig. 3.55: *Save map dialog*

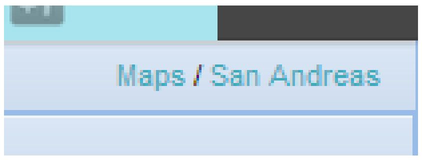


Fig. 3.56: *Saved map name*

This link contains a permalink to your map. If you open this link in a new window, your map will appear exactly as it was saved.

Styling layers In this interface, we can pause in our map creation and change the style of one of our uploaded layers. GeoNode allows you to edit layer styles graphically, without the need to resort to programming or requiring a technical background.

We'll be editing the `san_andres_y_providencia_poi` layer.

1. In the layer list, uncheck all of the layers except the above, so that only this one is visible (not including the base layer).

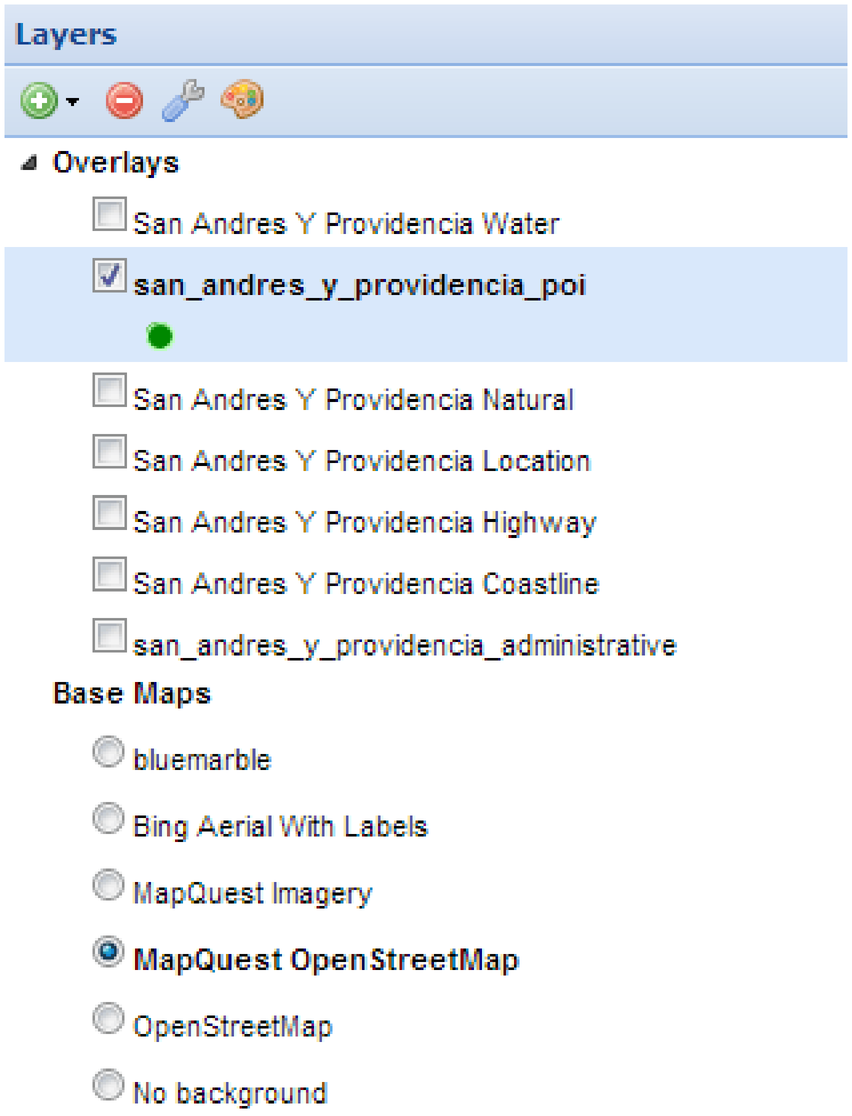

Layers
Overlays
San Andres Y Providencia Water
san_andres_y_providencia_poi
San Andres Y Providencia Natural
San Andres Y Providencia Location
San Andres Y Providencia Highway
San Andres Y Providencia Coastline
san_andres_y_providencia_administrative
Base Maps
bluemarble
Bing Aerial With Labels
MapQuest Imagery
MapQuest OpenStreetMap
OpenStreetMap
No background

Fig. 3.57: *Only one layer visible*

2. Zoom in closer using the toolbar or the mouse.
3. In the layer list, click to select the remaining layer and then click the palette icon (*Layer Styles*). This will bring up the style manager.

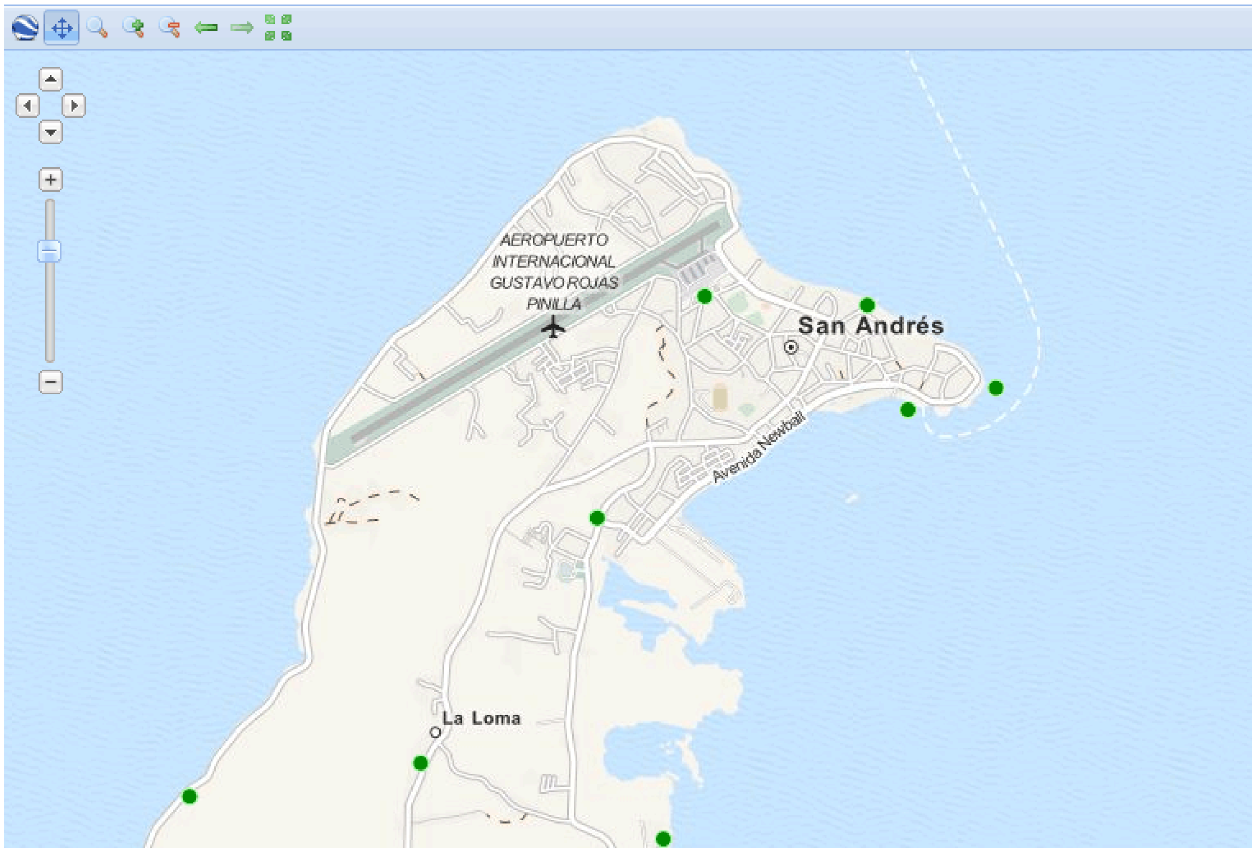


Fig. 3.58: *Zoomed in to see the layer better*

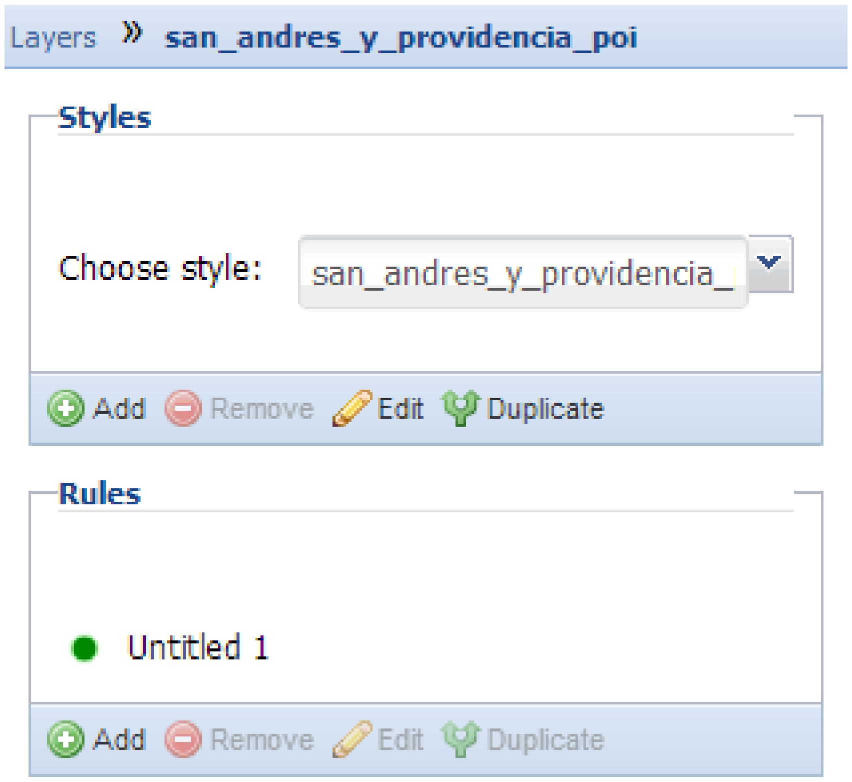


Fig. 3.59: *Styles manager*

4. This layer has one style (named the same as the layer) and one rule in that style. Click the rule (*Untitled 1*) to select it, and then click on *Edit* below it.

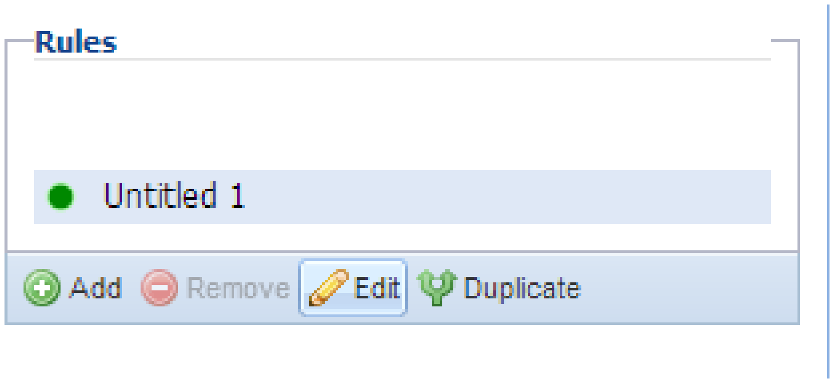


Fig. 3.60: *Edit style rule link*

5. Edit the style. You can choose from simple shapes, add labels, and even adjust the look of the points based on attribute values and scale.
6. When done, click *Save*, then click on the word *Layers* to return to the layer list.

Share your map Now let's finish our map.

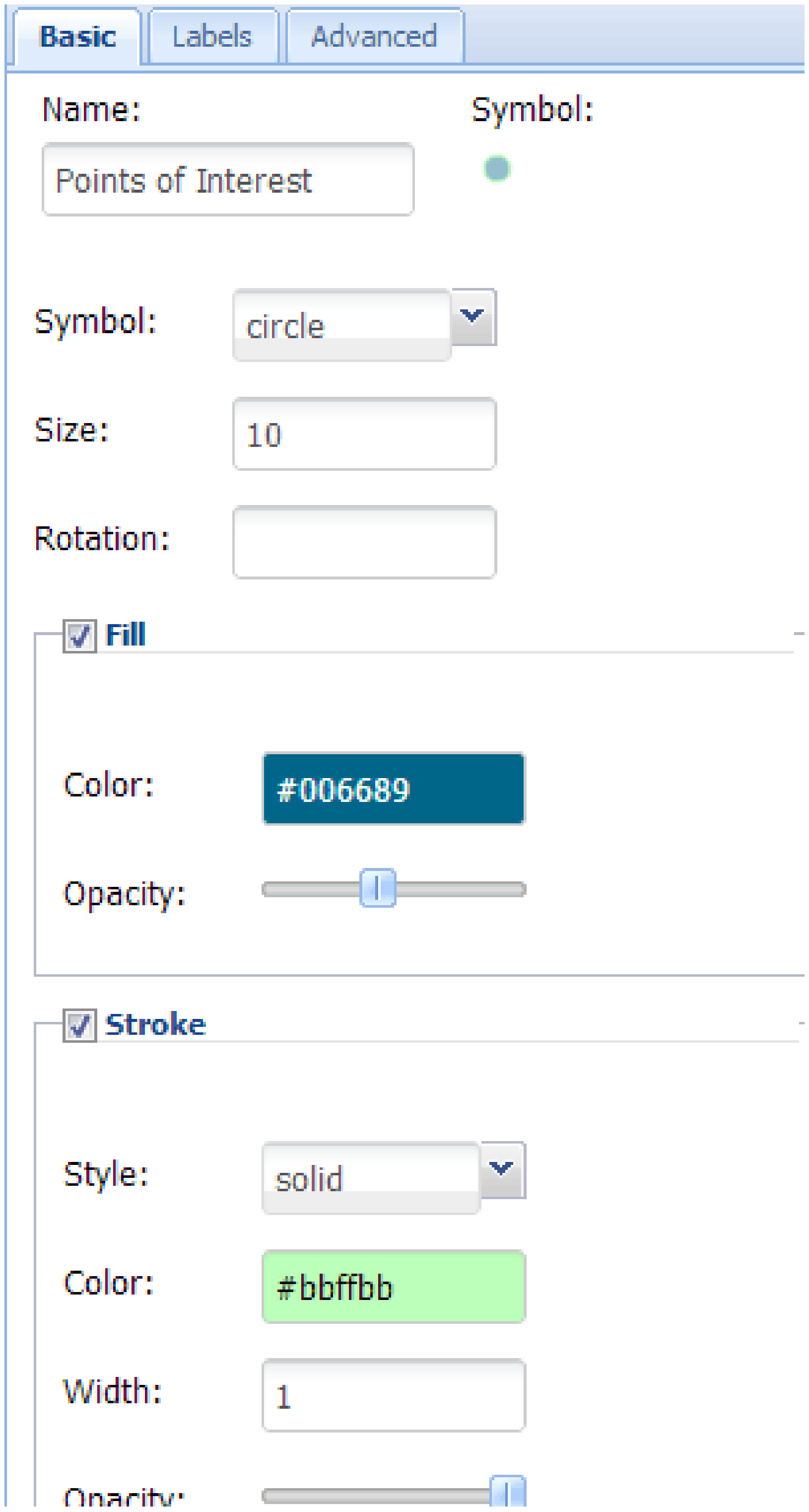
1. Check the box next to the *highway* layer to activate it. If it is not below the *POI* layer in the list, click and drag it down.
2. Make any final adjustments to the map composition as desired, including zoom and pan settings.
3. Click the *Map* button in the toolbar, and then click *Publish Map*.
4. The title and abstract as previously created should still be there. Make any adjustments as necessary, and click *Save*.
5. A new dialog will appear with instructions on how to embed this map in a webpage, including a code snippet. You can adjust the parameters as necessary.

Your map can now be shared.

Using GeoNode with other applications

Your GeoNode project is based on core components which are interoperable and as such, it is straightforward for you to integrate with external applications and services. This section will walk you through how to connect to your GeoNode instance from other applications and how to integrate other services into your GeoNode project. When complete, you should have a good idea about the possibilities for integration, and have basic knowledge about how to accomplish it. You may find it necessary to dive deeper into how to do more complex integration in order to accomplish your goals, but you should feel comfortable with the basics, and feel confident reaching out to the wider GeoNode community for help.

OGC services Since GeoNode is built on GeoServer which is heavily based on OGC services, the main path for integration with external services is via OGC Standards. A large number of systems, applications and services support adding WMS layers to them, but only a few key ones are covered below. WFS and WCS are also supported in a wide variety of clients and platforms and give you access to the actual data for use in GeoProcessing or to manipulate it to meet your requirements. GeoServer also bundles GeoWebCache which produces map tiles that can be added as layers



Name: Points of Interest

Symbol: ●

Symbol: circle

Size: 10

Rotation:

Color: #006689

Opacity:

Style: solid

Color: #bbffbb

Width: 1

Opacity:

Fig. 3.61: *Editing basic style rules*

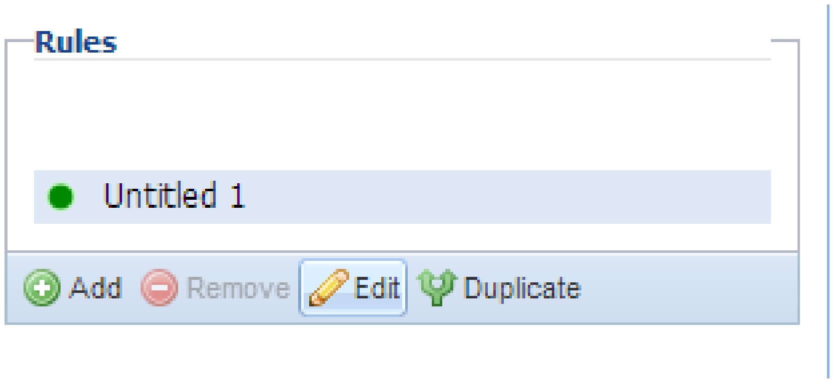


Fig. 3.62: *Editing style labels*

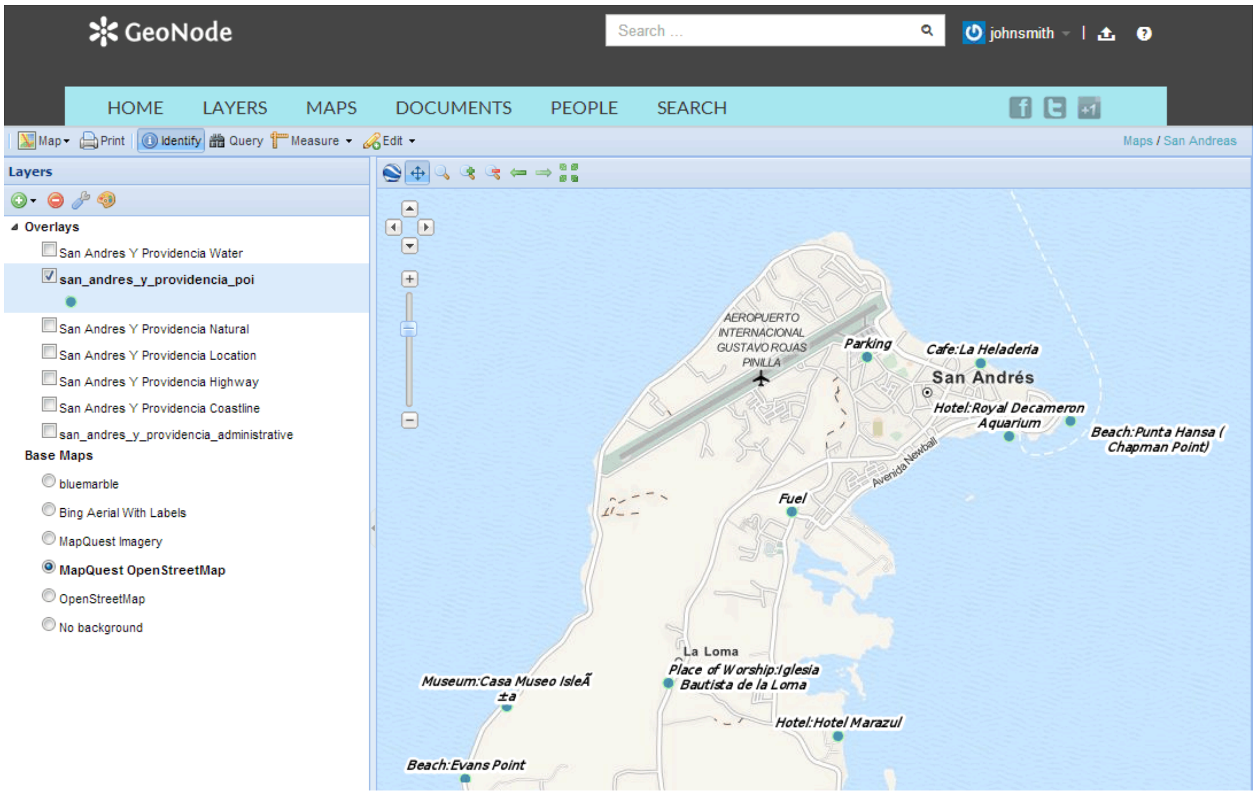

GeoNode
Search ...
johnsmith
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
Map
Print
Identify
Query
Measure
Edit
Maps / San Andreas
Layers
Overlays
San Andres Y Providencia Water
san_andres_y_providencia_poi
San Andres Y Providencia Natural
San Andres Y Providencia Location
San Andres Y Providencia Highway
San Andres Y Providencia Coastline
san_andres_y_providencia_administrative
Base Maps
bluemarble
Bing Aerial With Labels
MapQuest Imagery
MapQuest OpenStreetMap
OpenStreetMap
No background
AEROPUERTO INTERNACIONAL GUSTAVO ROJAS PINILLA
Parking
Cafe:La Heladeria
San Andrés
Hotel:Royal Decameron Aquarium
Beach:Punta Hansa (Chapman Point)
Avenida Newball
Fuel
La Loma
Place of Worship:Iglesia Bautista de la Loma
Museum:Casa Museo IsleÃ±a
Hotel:Hotel Marazul
Beach:Evans Point

Fig. 3.63: *Styled layer*

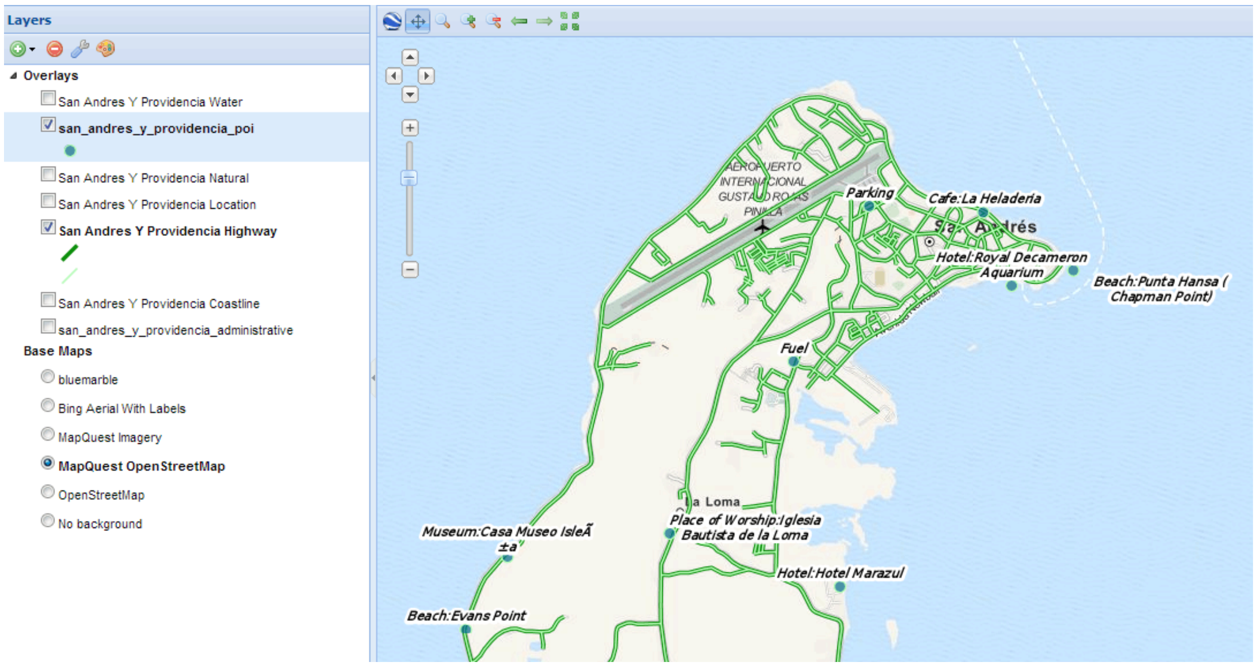

Layers
Overlays
San Andres Y Providencia Water
san_andres_y_providencia_poi
San Andres Y Providencia Natural
San Andres Y Providencia Location
San Andres Y Providencia Highway
San Andres Y Providencia Coastline
san_andres_y_providencia_administrative
Base Maps
bluemarble
Bing Aerial With Labels
MapQuest Imagery
MapQuest OpenStreetMap
OpenStreetMap
No background
Parking
Cafe:La Heladeria
Hotel:Royal Decameron Aquarium
Beach:Punta Hansa (Chapman Point)
Fuel
La Loma
Place of Worship:Iglesia Bautista de la Loma
Museum:Casa Museo IsleÃ±a
Hotel:Hotel Marazul
Beach:Evans Point

Fig. 3.64: *Adjusting map composition*

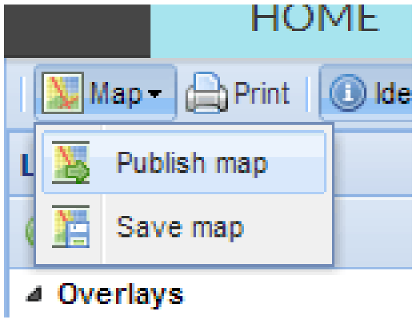
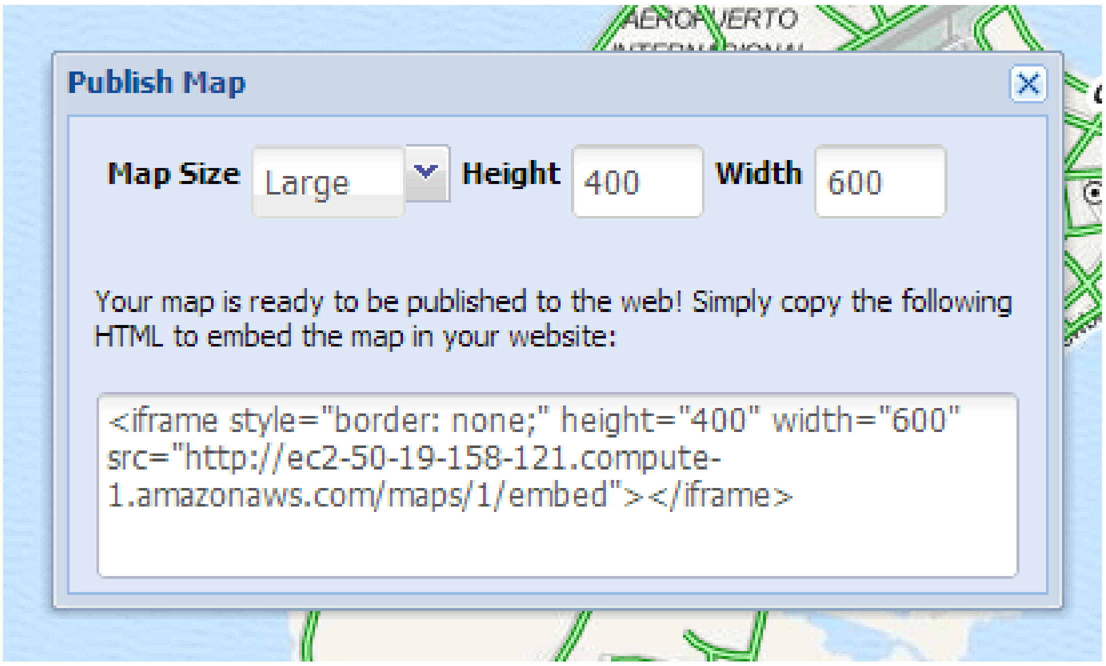


Fig. 3.65: *Publish map link*



Large 400 600

Your map is ready to be published to the web! Simply copy the following HTML to embed the map in your website:

```
<iframe style="border: none;" height="400" width="600"
src="http://ec2-50-19-158-121.compute-
1.amazonaws.com/maps/1/embed"></iframe>
```

Fig. 3.66: *Map publishing options*

in many popular web mapping tools including Google Maps, Leaflet, OpenLayers and others. You should review the reference material included in the first chapter to learn more about OGC Services and when evaluating external systems make sure that they are also OGC Compliant in order to integrate as seamlessly as possible.

Use GeoNode with:

ArcGIS ArcGIS Desktop (ArcMap) supports adding WMS layers to your map project. The following set of steps will walk you through how to configure a WMS Layer from your GeoNode within ArcMap.

First, you can start with a new empty project or add these layers to your existing project.

Next click the ArcCatalog button on the toolbar to bring up its interface.

From there, double click the “Add WMS Server” item in the tree to bring up the dialog that lets you enter the details for your WMS.

Next, enter the URL for your GeoNode’s WMS endpoint which is the base url with /geoserver/wms appended to the end of the URL. You can also enter your credentials into the optional Account section of this dialog to gain access to non-public layers that your user may have access to.

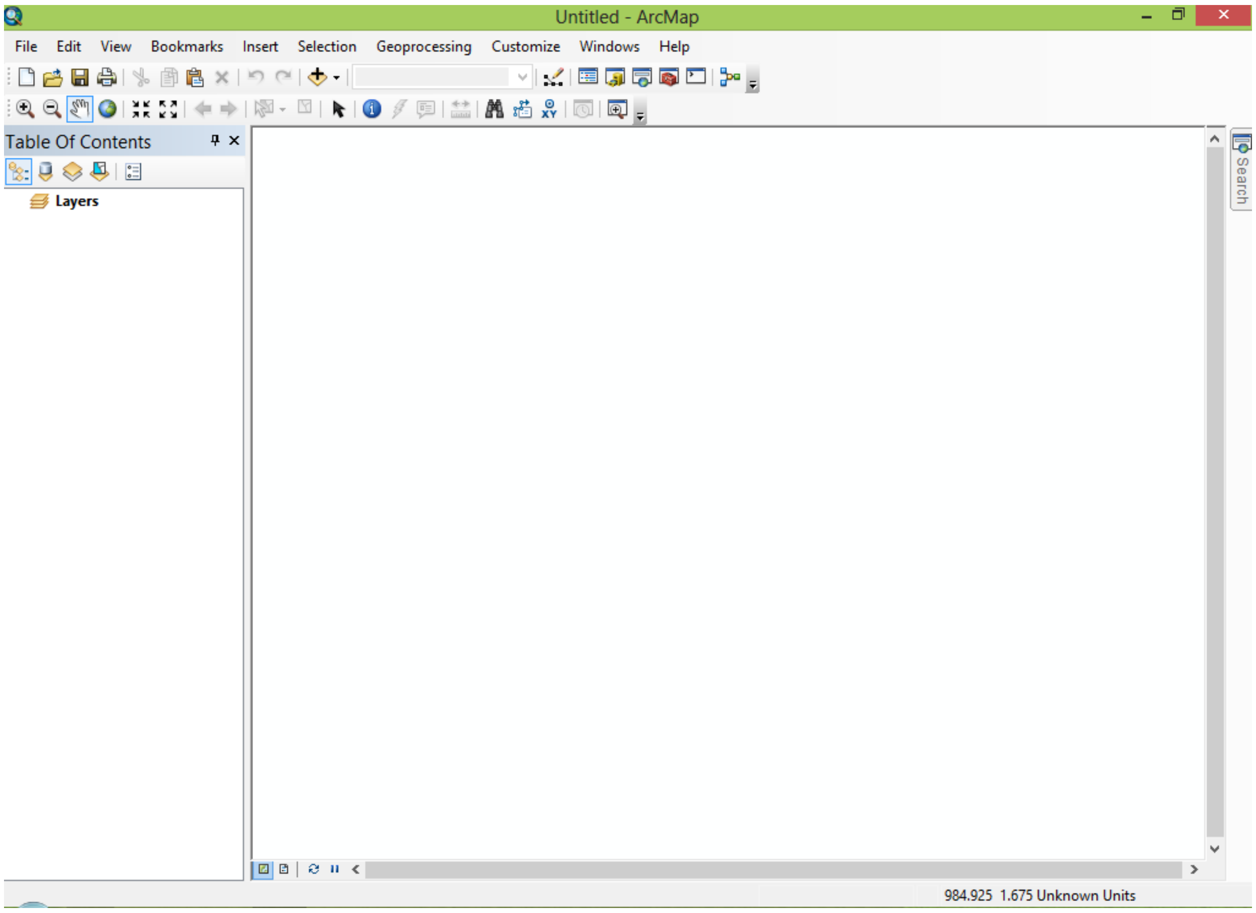
Click the “Get Layers” button to ask ArcMap to query your WMS’s GetCapabilities document to get the list of available layers.

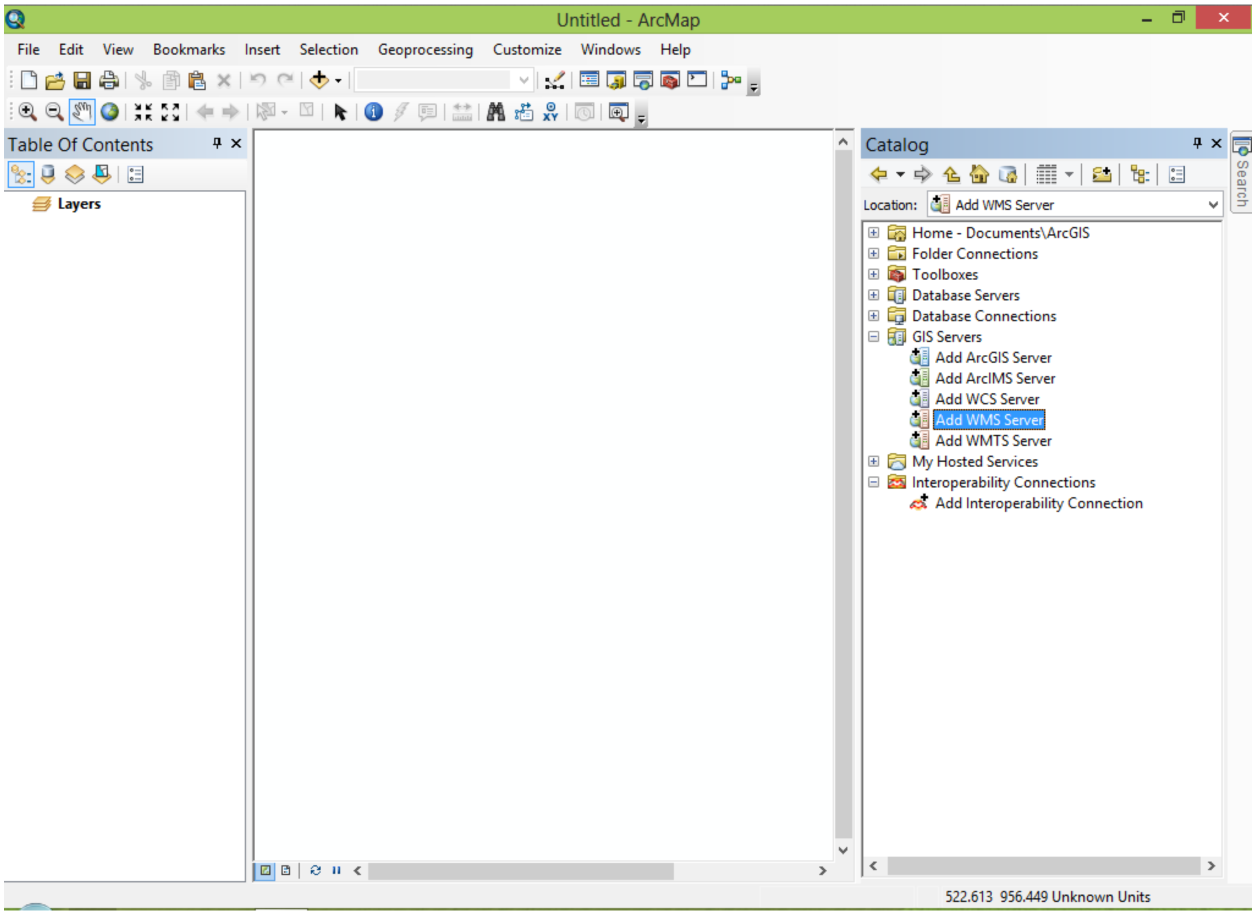
After you click the OK button, your GeoNode layers will appear in the ArcCatalog Interface.

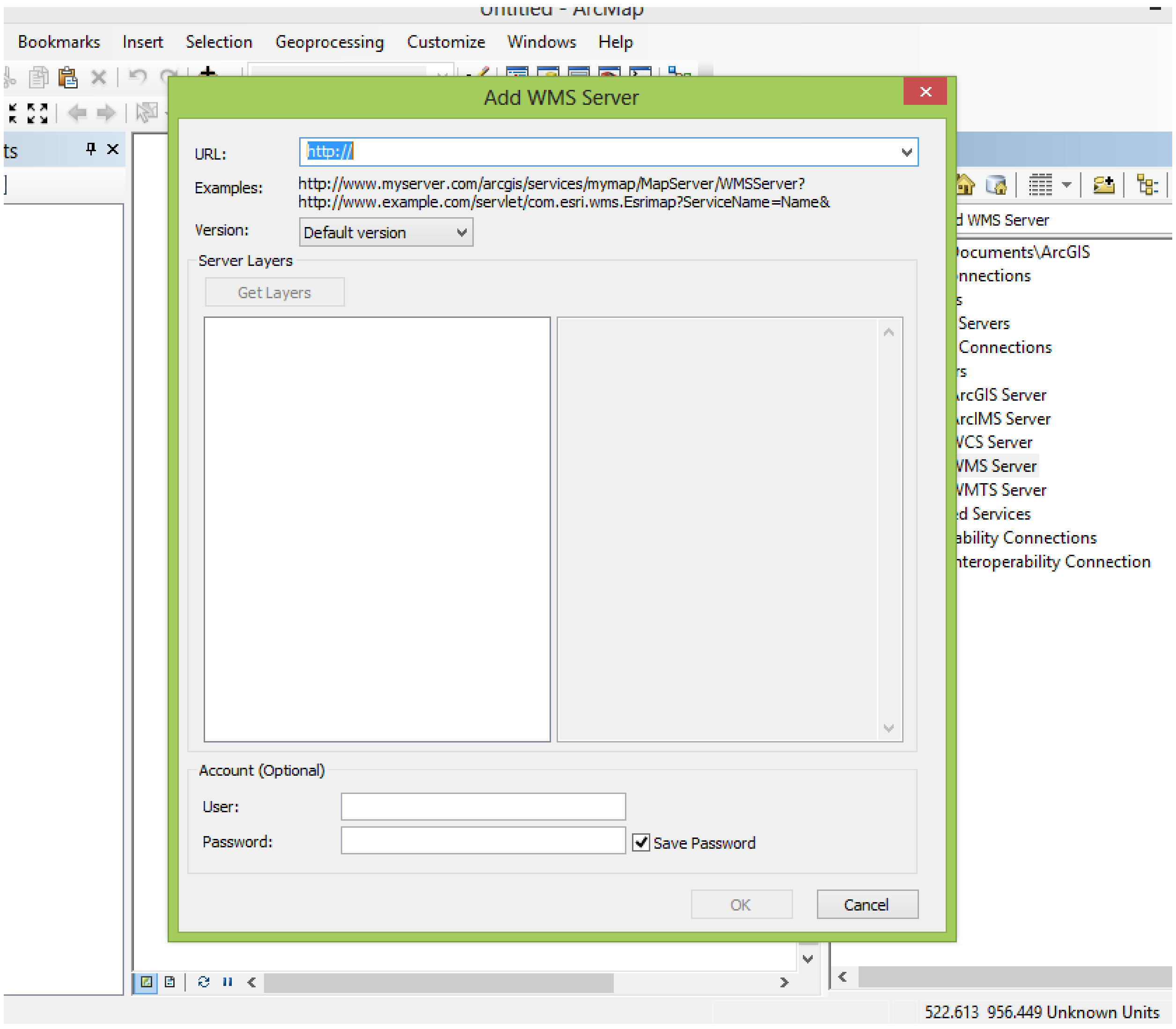
Once your server is configured in ArcMap, you can right click on one of the layers and investigate its properties.

In order to actually add the layer to your project, you can drag and drop it into the Table of Contents, or right click and select “Create Layer”. Your Layer will now be displayed in the map panel of your project.

Once the layer is in your projects Table of Contents, you can right click on it and select the Layer Properties option and select the Styles Tab to choose from the available styles for that layer.


Untitled - ArcMap
File
Edit
View
Bookmarks
Insert
Selection
Geoprocessing
Customize
Windows
Help
Table Of Contents
Layers
Search
984.925 1.675 Unknown Units


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
Table Of Contents
Layers
Catalog
Location: Add WMS Server
Home - Documents\ArcGIS
Folder Connections
Toolboxes
Database Servers
Database Connections
GIS Servers
Add ArcGIS Server
Add ArcIMS Server
Add WCS Server
Add WMS Server
Add WMTS Server
My Hosted Services
Interoperability Connections
Add Interoperability Connection
Search
522.613 956.449 Unknown Units



URL: http://

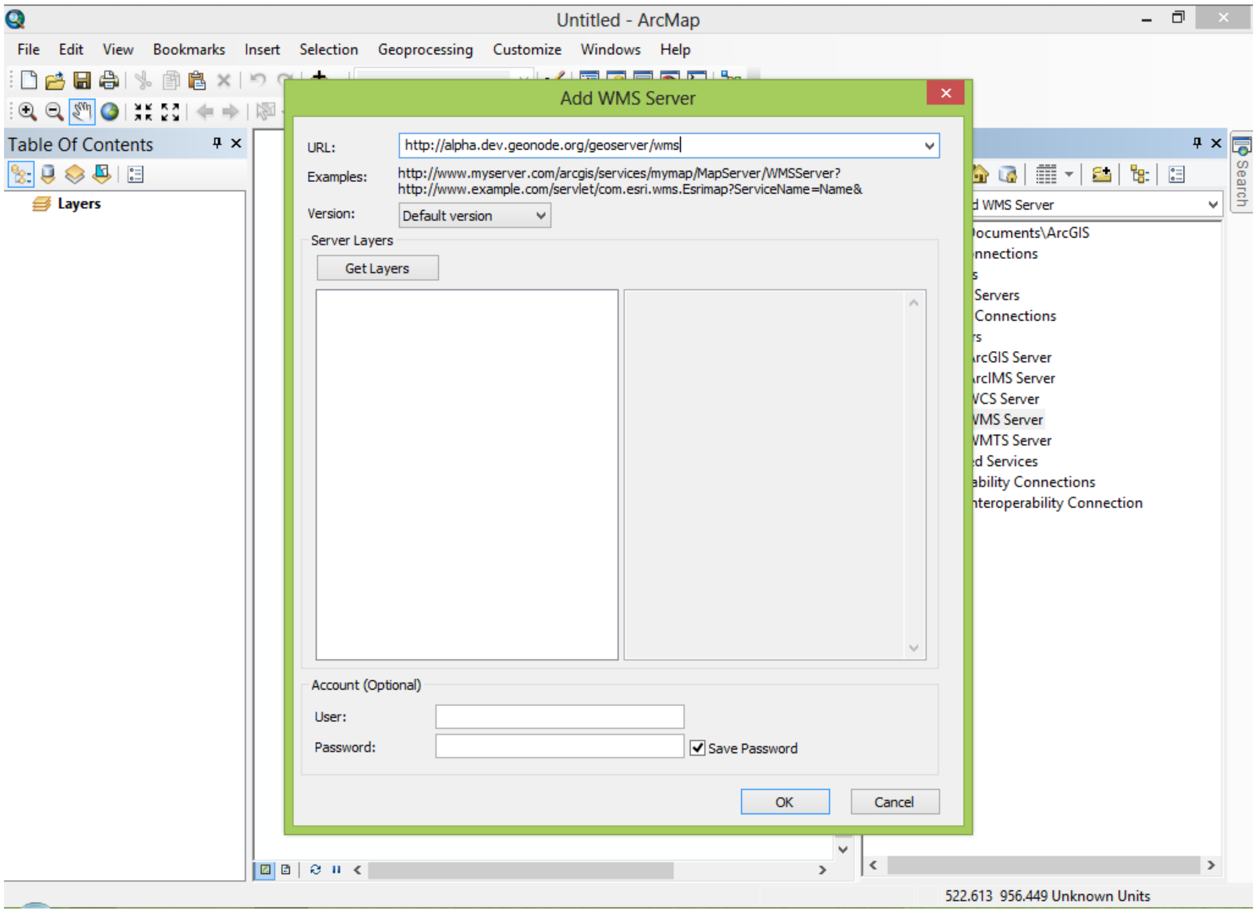
Examples: http://www.myserver.com/arcgis/services/mymap/MapServer/WMSServer?
http://www.example.com/servlet/com.esri.wms.Esrimap?ServiceName=Name&

Version: Default version

User:

Password: ☑ Save Password

522.613 956.449 Unknown Units



URL: http://alpha.dev.geonode.org/geoserver/wms

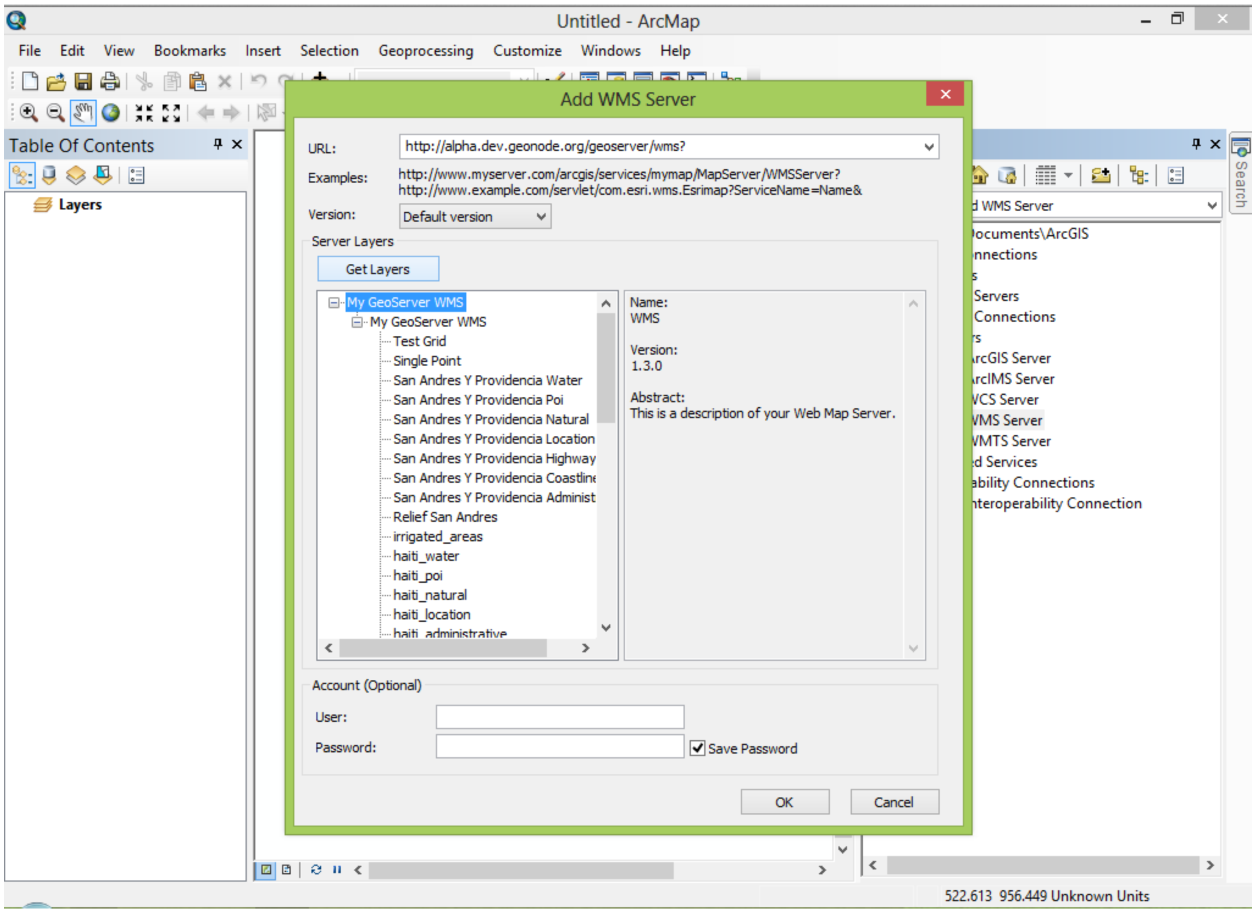
Examples: http://www.myserver.com/arcgis/services/mymap/MapServer/WMSServer?
http://www.example.com/servlet/com.esri.wms.Esrimap?ServiceName=Name&

Version: Default version

User:

Password: [✓ Save Password]

522.613 956.449 Unknown Units


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
Table Of Contents
Layers
Add WMS Server
Get Layers
OK
Cancel

URL: http://alpha.dev.geonode.org/geoserver/wms?

Examples: http://www.myserver.com/arcgis/services/mymap/MapServer/WMSServer?
http://www.example.com/servlet/com.esri.wms.Esrimap?ServiceName=Name&

Version: Default version

Server Layers

- My GeoServer WMS
  - My GeoServer WMS
    - Test Grid
    - Single Point
    - San Andres Y Providencia Water
    - San Andres Y Providencia Poi
    - San Andres Y Providencia Natural
    - San Andres Y Providencia Location
    - San Andres Y Providencia Highway
    - San Andres Y Providencia Coastline
    - San Andres Y Providencia Administ
    - Relief San Andres
    - irrigated_areas
    - haiti_water
    - haiti_poi
    - haiti_natural
    - haiti_location
    - haiti_administrative

Name:
WMS

Version:
1.3.0

Abstract:
This is a description of your Web Map Server.

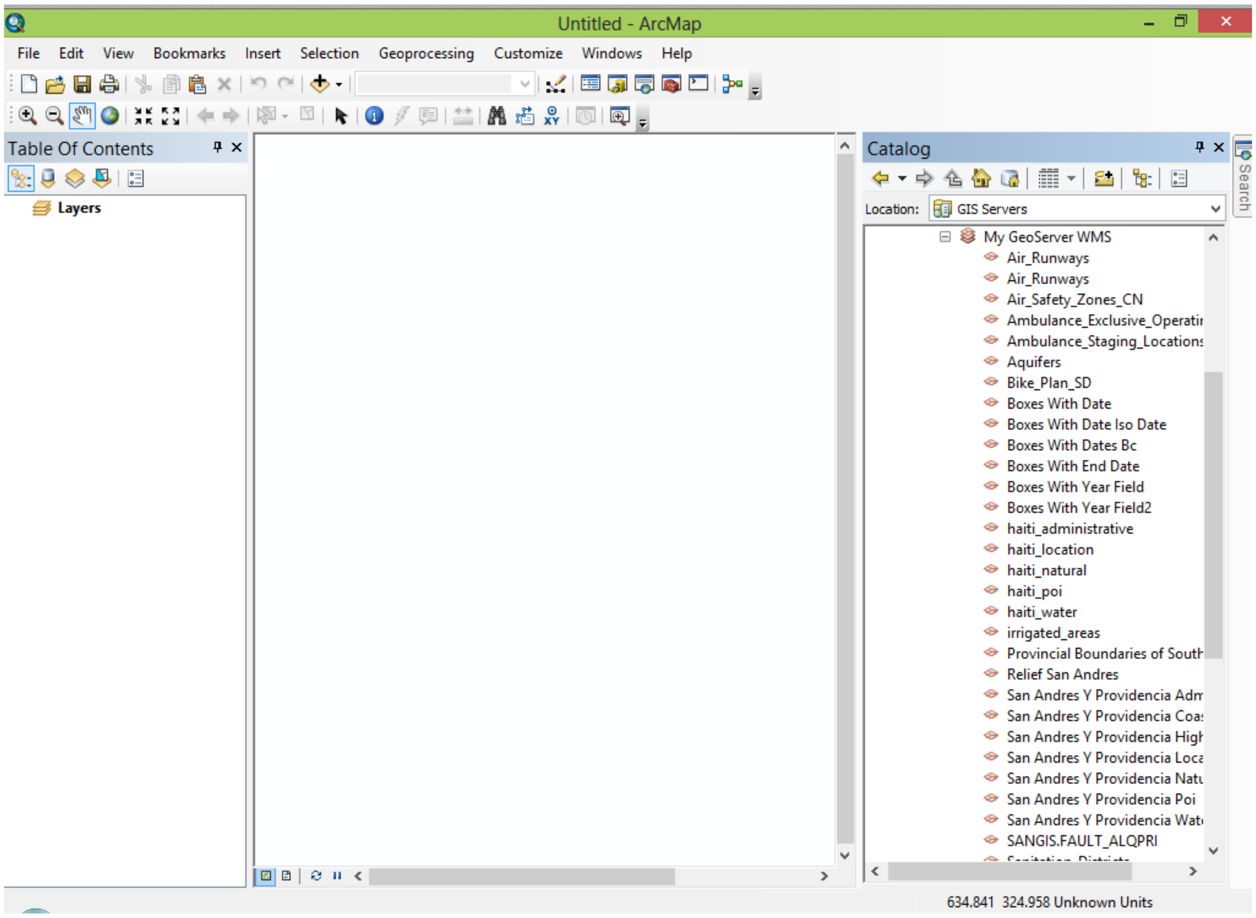
Account (Optional)

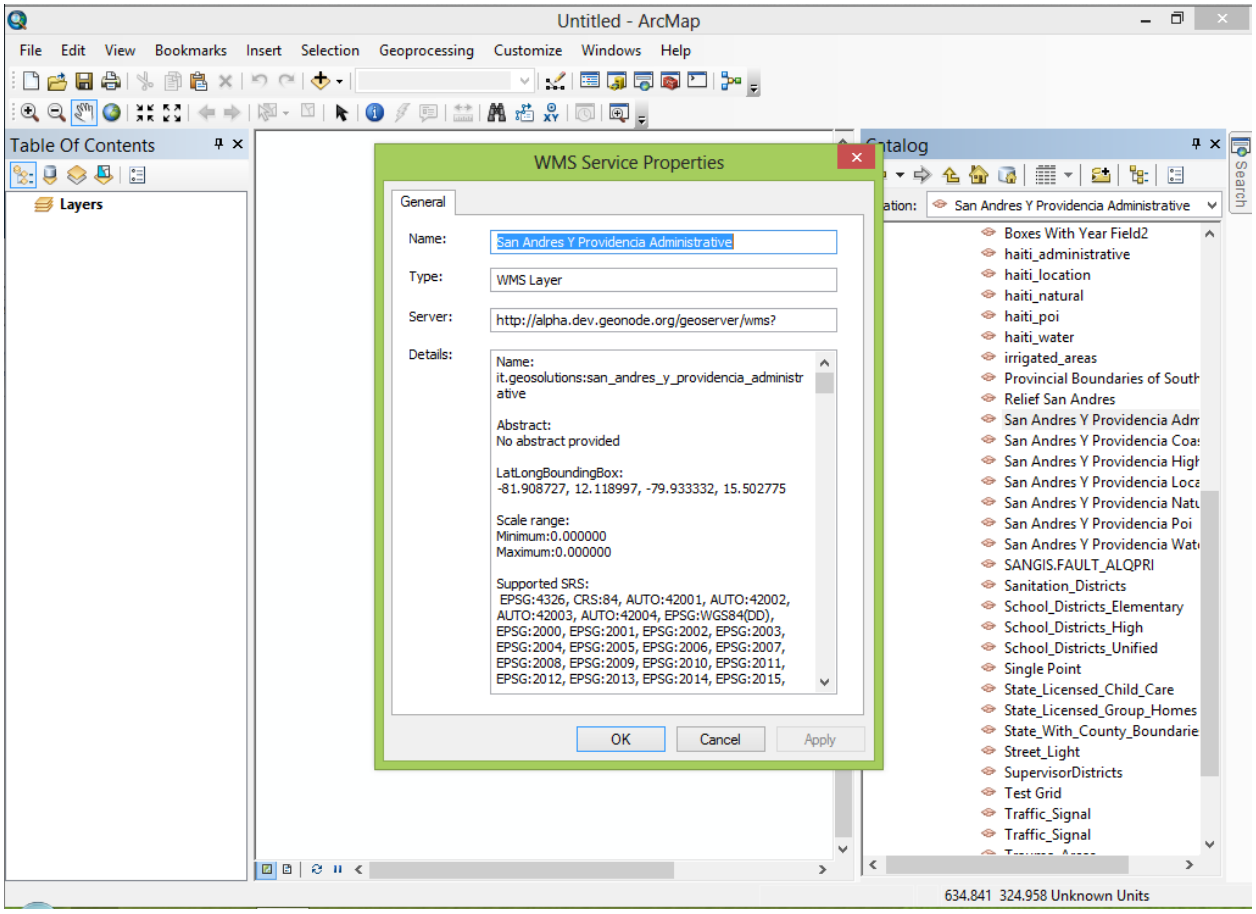
User:

Password:

[✓] Save Password

522.613 956.449 Unknown Units


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
Table Of Contents
Layers
Catalog
Location: GIS Servers
My GeoServer WMS
Air_Runways
Air_Runways
Air_Safety_Zones_CN
Ambulance_Exclusive_Operatir
Ambulance_Staging_Locations
Aquifers
Bike_Plan_SD
Boxes With Date
Boxes With Date Iso Date
Boxes With Dates Bc
Boxes With End Date
Boxes With Year Field
Boxes With Year Field2
haiti_administrative
haiti_location
haiti_natural
haiti_poi
haiti_water
irrigated_areas
Provincial Boundaries of South
Relief San Andres
San Andres Y Providencia Adm
San Andres Y Providencia Coas
San Andres Y Providencia High
San Andres Y Providencia Loca
San Andres Y Providencia Natu
San Andres Y Providencia Poi
San Andres Y Providencia Wat
SANGIS.FAULT_ALQPRI
Search
634.841 324.958 Unknown Units


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
Table Of Contents
Layers
Catalog
ation: San Andres Y Providencia Administrative
Boxes With Year Field2
haiti_administrative
haiti_location
haiti_natural
haiti_poi
haiti_water
irrigated_areas
Provincial Boundaries of South
Relief San Andres
San Andres Y Providencia Adm
San Andres Y Providencia Coa
San Andres Y Providencia High
San Andres Y Providencia Loca
San Andres Y Providencia Natu
San Andres Y Providencia Poi
San Andres Y Providencia Wat
SANGIS.FAULT_ALQPRI
Sanitation_Districts
School_Districts_Elementary
School_Districts_High
School_Districts_Unified
Single Point
State_Licensed_Child_Care
State_Licensed_Group_Homes
State_With_County_Boundarie
Street_Light
SupervisorDistricts
Test Grid
Traffic_Signal
Traffic_Signal
634.841 324.958 Unknown Units
WMS Service Properties
General
OK
Cancel
Apply

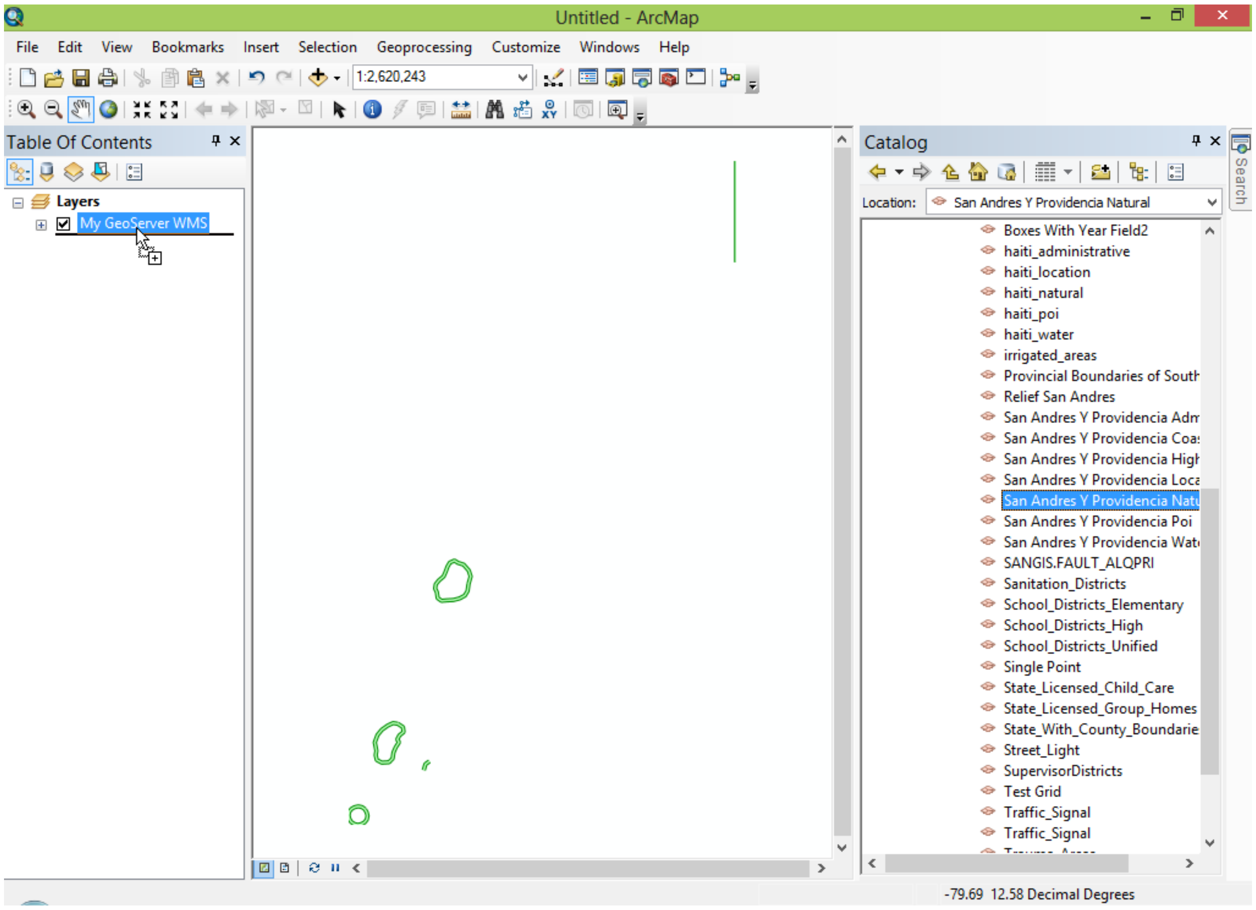
Name: San Andres Y Providencia Administrative

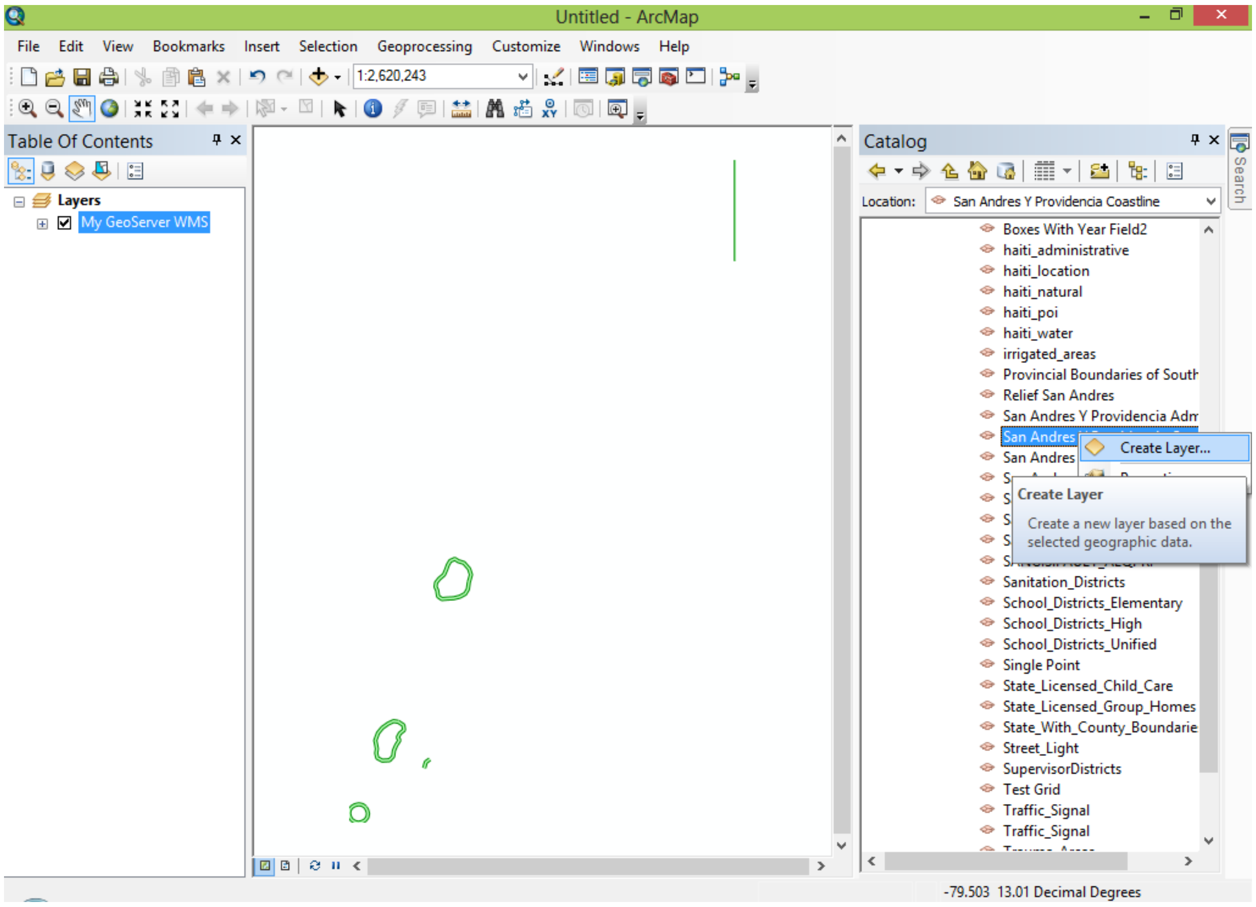
Type: WMS Layer

Server: http://alpha.dev.geonode.org/geoserver/wms?

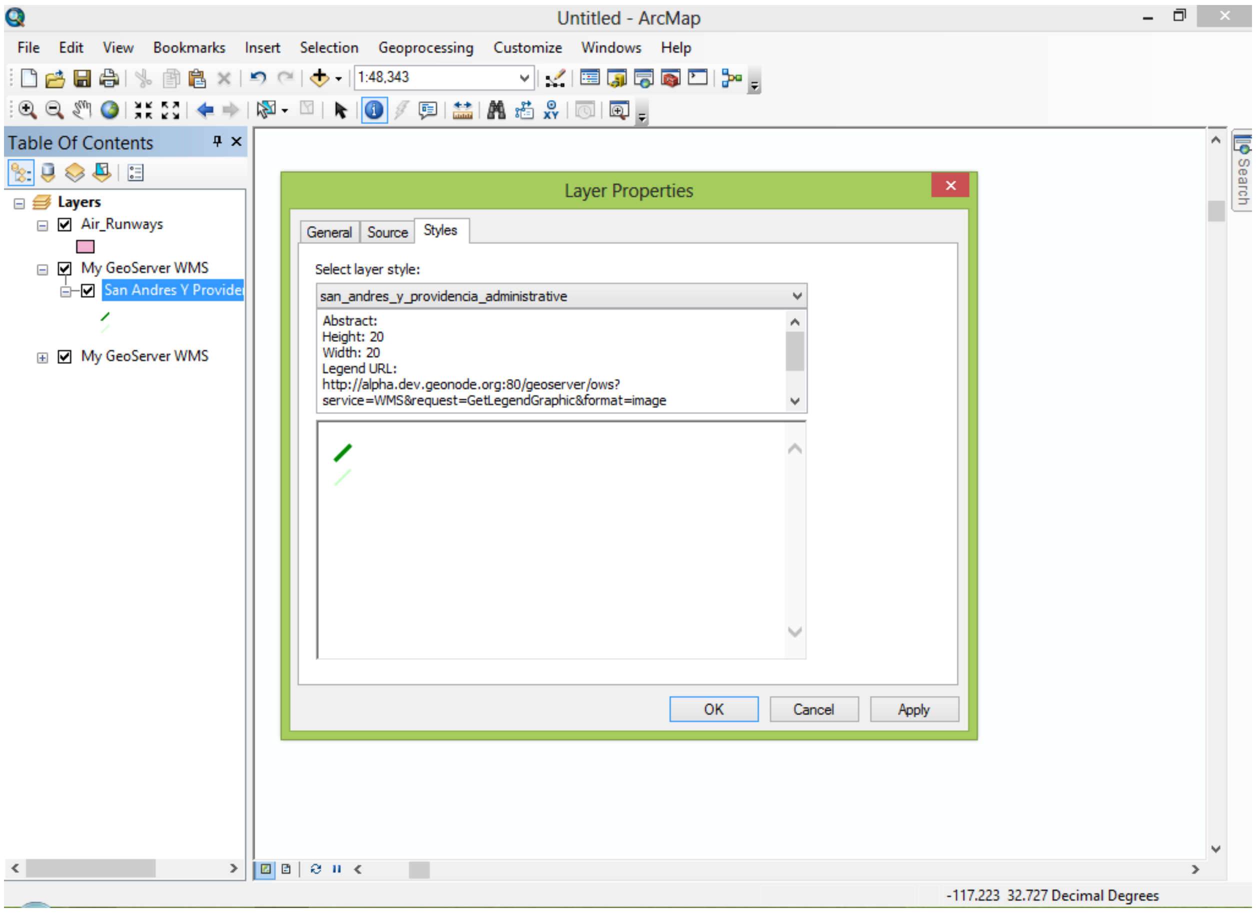
Details:

```
Name:
it.geosolutions:san_andres_y_providencia_administr
ative

Abstract:
No abstract provided

LatLongBoundingBox:
-81.908727, 12.118997, -79.933332, 15.502775

Scale range:
Minimum:0.000000
Maximum:0.000000

Supported SRS:
 EPSG:4326, CRS:84, AUTO:42001, AUTO:42002,
AUTO:42003, AUTO:42004, EPSG:WGS84(DD),
EPSG:2000, EPSG:2001, EPSG:2002, EPSG:2003,
EPSG:2004, EPSG:2005, EPSG:2006, EPSG:2007,
EPSG:2008, EPSG:2009, EPSG:2010, EPSG:2011,
EPSG:2012, EPSG:2013, EPSG:2014, EPSG:2015,
```


Untitled - ArcMap
File
Edit
View
Bookmarks
Insert
Selection
Geoprocessing
Customize
Windows
Help
1:2,620,243
Table Of Contents
Layers
My GeoServer WMS
Catalog
Location:
San Andres Y Providencia Natural
Boxes With Year Field2
haiti_administrative
haiti_location
haiti_natural
haiti_poi
haiti_water
irrigated_areas
Provincial Boundaries of South
Relief San Andres
San Andres Y Providencia Adm
San Andres Y Providencia Coa
San Andres Y Providencia High
San Andres Y Providencia Loca
San Andres Y Providencia Natu
San Andres Y Providencia Poi
San Andres Y Providencia Wat
SANGIS.FAULT_ALQPRI
Sanitation_Districts
School_Districts_Elementary
School_Districts_High
School_Districts_Unified
Single Point
State_Licensed_Child_Care
State_Licensed_Group_Homes
State_With_County_Boundarie
Street_Light
SupervisorDistricts
Test Grid
Traffic_Signal
Traffic_Signal
Search
-79.69 12.58 Decimal Degrees



Create a new layer based on the selected geographic data.



Select layer style: san_andres_y_providencia_administrative

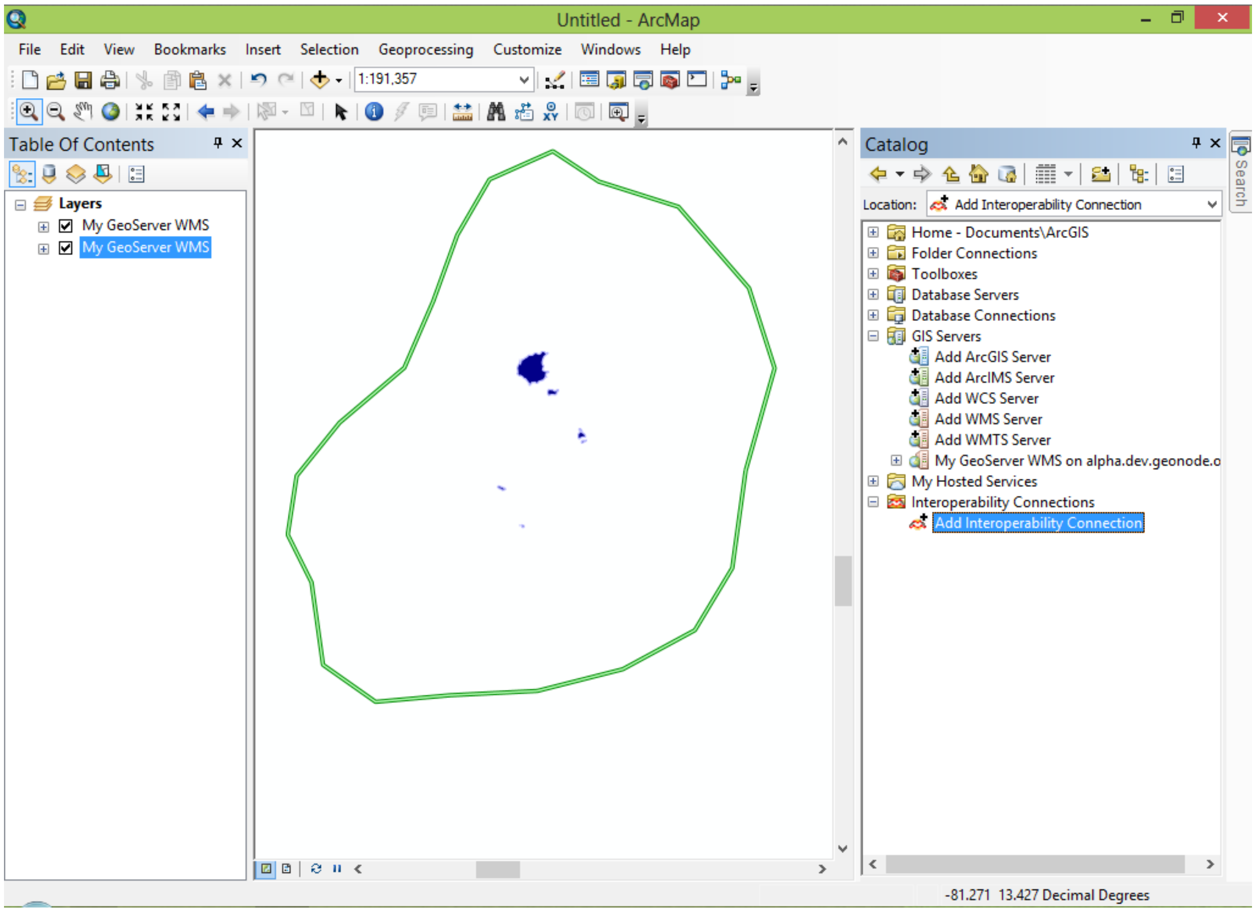
Abstract:
Height: 20
Width: 20
Legend URL:
http://alpha.dev.geonode.org:80/geoserver/ows?service=WMS&request=GetLegendGraphic&format=image

-117.223 32.727 Decimal Degrees

Now that we have seen how to add a WMS layer to our ArcMap project, lets walk through how to add the same layers as a WFS which retrieves the actual feature data from your GeoNode rather than a rendered map as you get with WMS. Adding layers as a WFS gives you more control over how the layers are styled within ArcMap and makes them available for you to use with other ArcGIS tools like the Geoprocessing toolbox.

Note: Adding WFS layers to ArcMap requires that you have the Data Interoperability Extension installed. This extension is not included in ArcMap by default and is licensed and installed separately.

Start by opening up the ArcCatalog Interface within ArcMap and make sure that you have the "Interoperability Connections" option listed in the list.



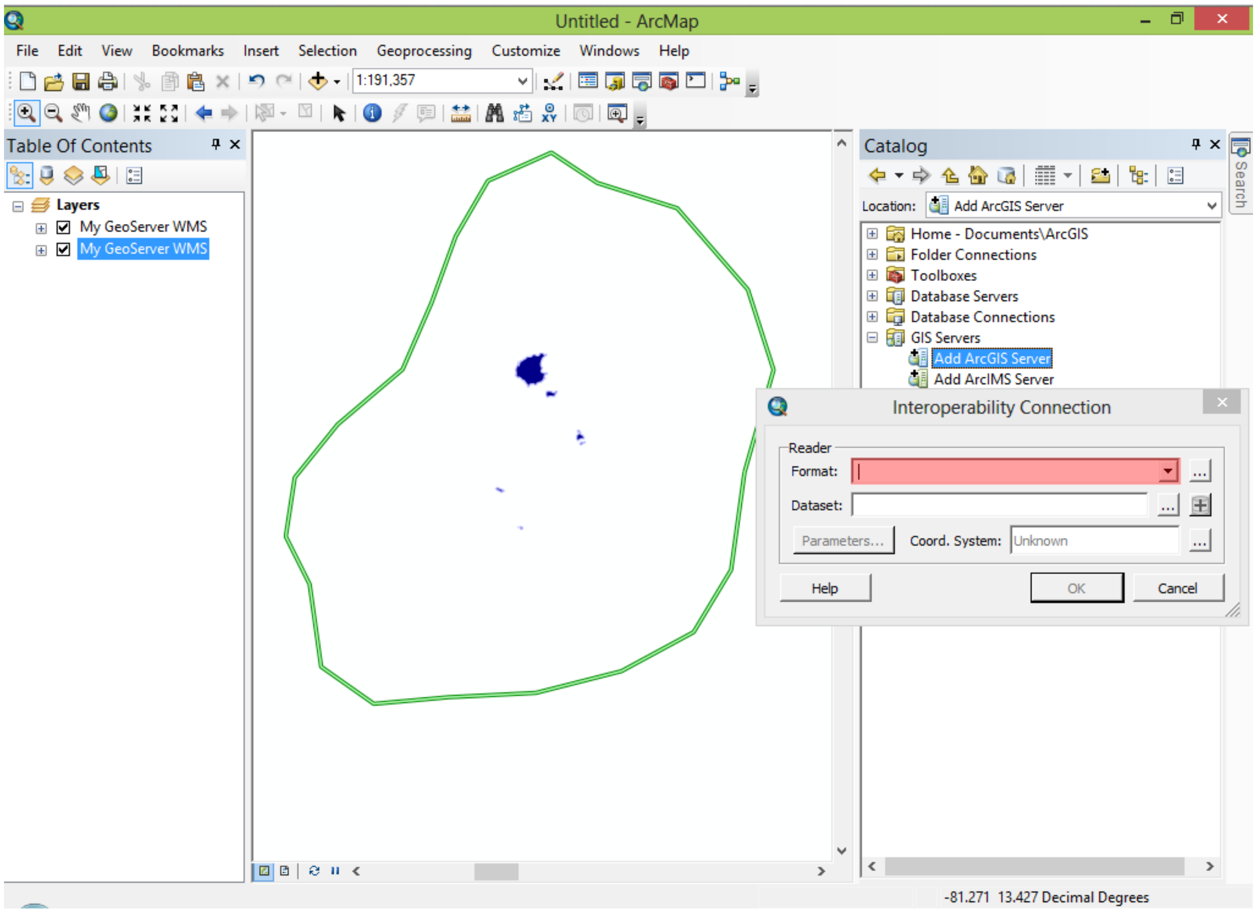
Next select "Add Interoperability Connection" to bring up the dialog that lets you add the WFS endpoint from your GeoNode.

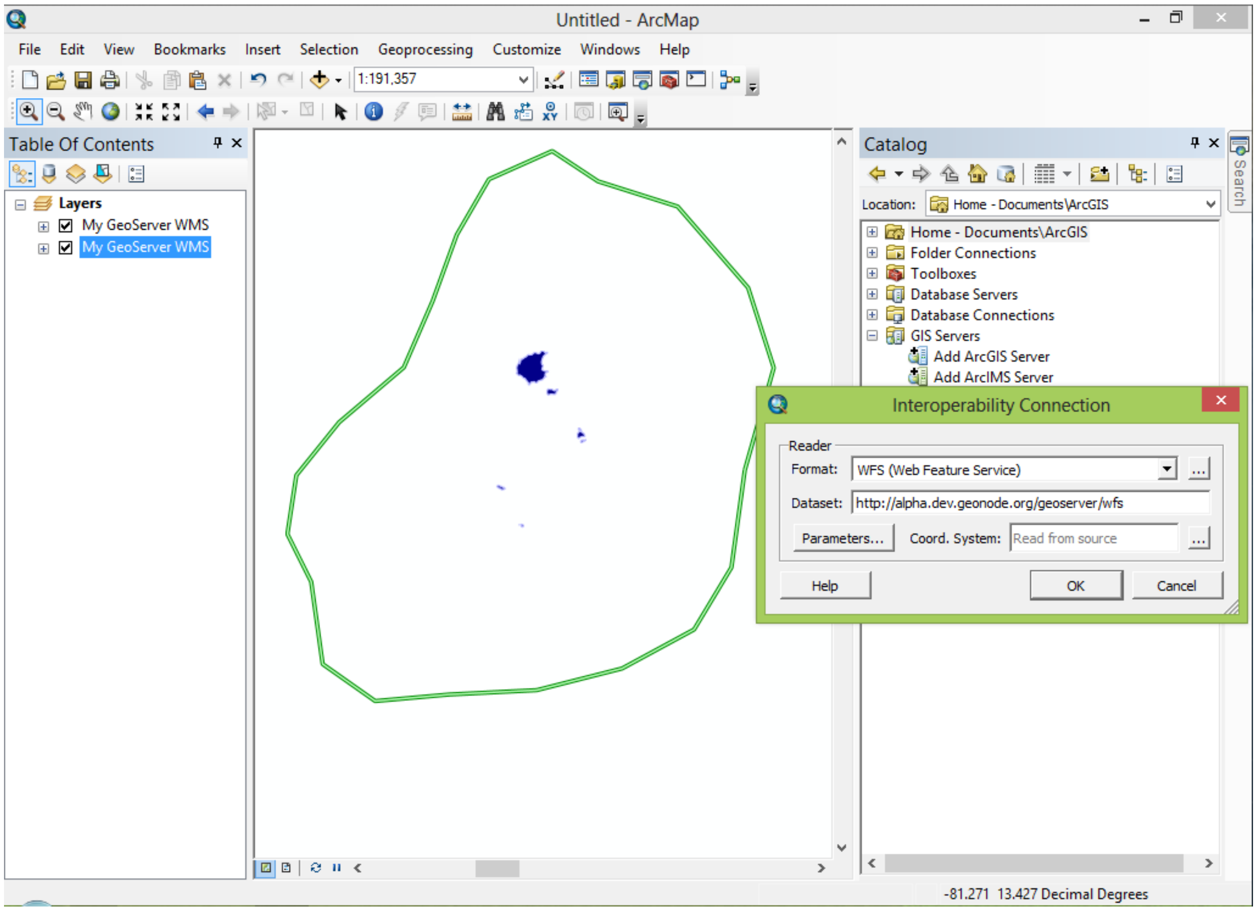
Select "WFS (Web Feature Service)" in the Format dropdown and enter the URL to the WFS endpoint for your GeoNode in the Dataset field. The WFS endpoint is your base URL + /geoserver/wfs

You will need to click the "Parameters" button to supply more connection information including your credentials which will give you the ability to use private layers that you have access to.

Select the Feature Types button to have ArcMap get a list of layers from the WFS Service of your GeoNode.

Select the layers that you want to add and click OK and ArcMap will import the features from your GeoNode into the system.


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:191,357
Table Of Contents
Layers
My GeoServer WMS
My GeoServer WMS
Catalog
Location: Add ArcGIS Server
Home - Documents\ArcGIS
Folder Connections
Toolboxes
Database Servers
Database Connections
GIS Servers
Add ArcGIS Server
Add ArcIMS Server
Interoperability Connection
Reader
Format:
Dataset:
Parameters...
Coord. System: Unknown
Help
OK
Cancel
-81.271 13.427 Decimal Degrees



Interoperability Connection

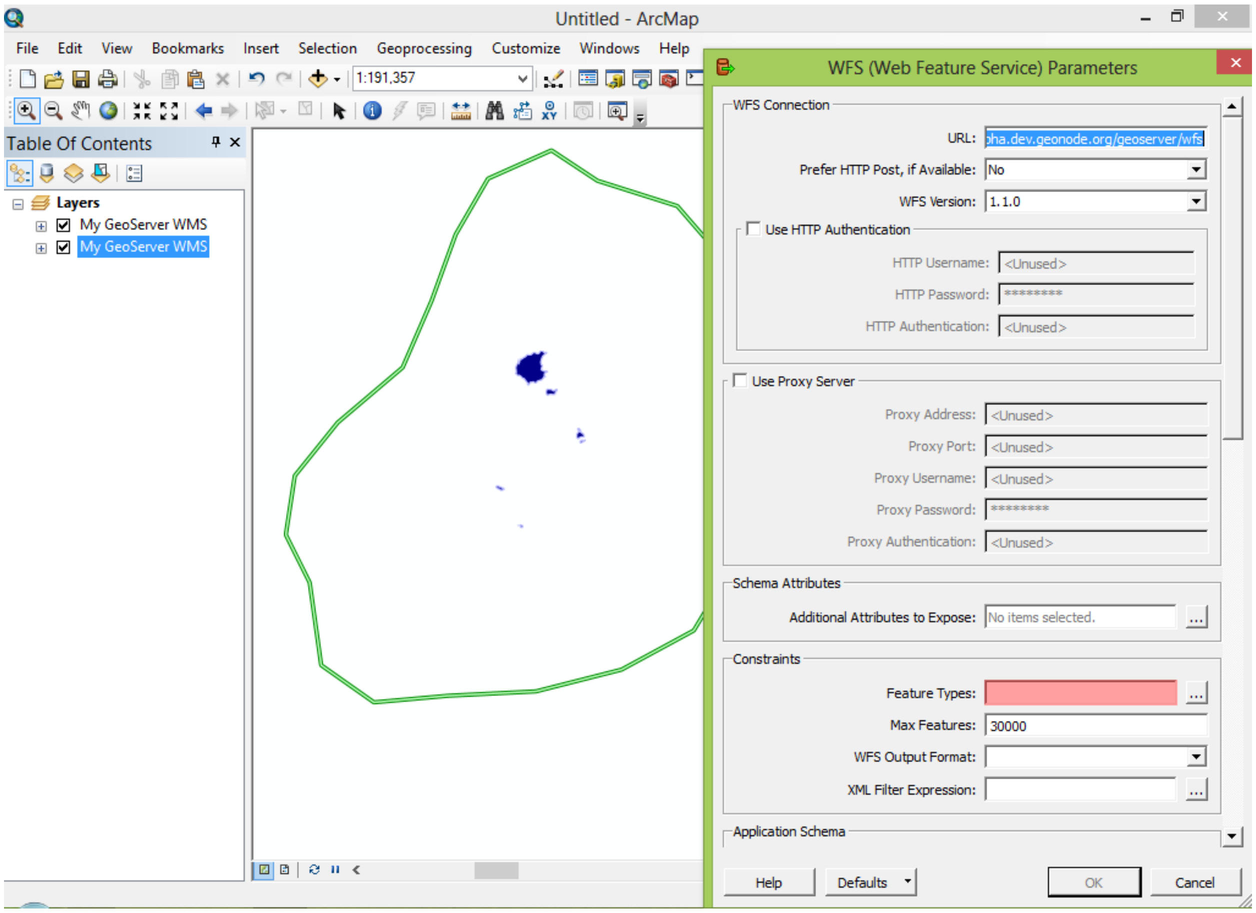
Reader

Format: WFS (Web Feature Service)

Dataset: http://alpha.dev.geonode.org/geoserver/wfs

Parameters... Coord. System: Read from source

Help OK Cancel



WFS (Web Feature Service) Parameters

WFS Connection

URL: pha.dev.geonode.org/geoserver/wfs

Prefer HTTP Post, if Available: No

WFS Version: 1.1.0

Use HTTP Authentication

HTTP Username: <Unused>

HTTP Password: ********

HTTP Authentication: <Unused>

Use Proxy Server

Proxy Address: <Unused>

Proxy Port: <Unused>

Proxy Username: <Unused>

Proxy Password: ********

Proxy Authentication: <Unused>

Schema Attributes

Additional Attributes to Expose: No items selected.

Constraints

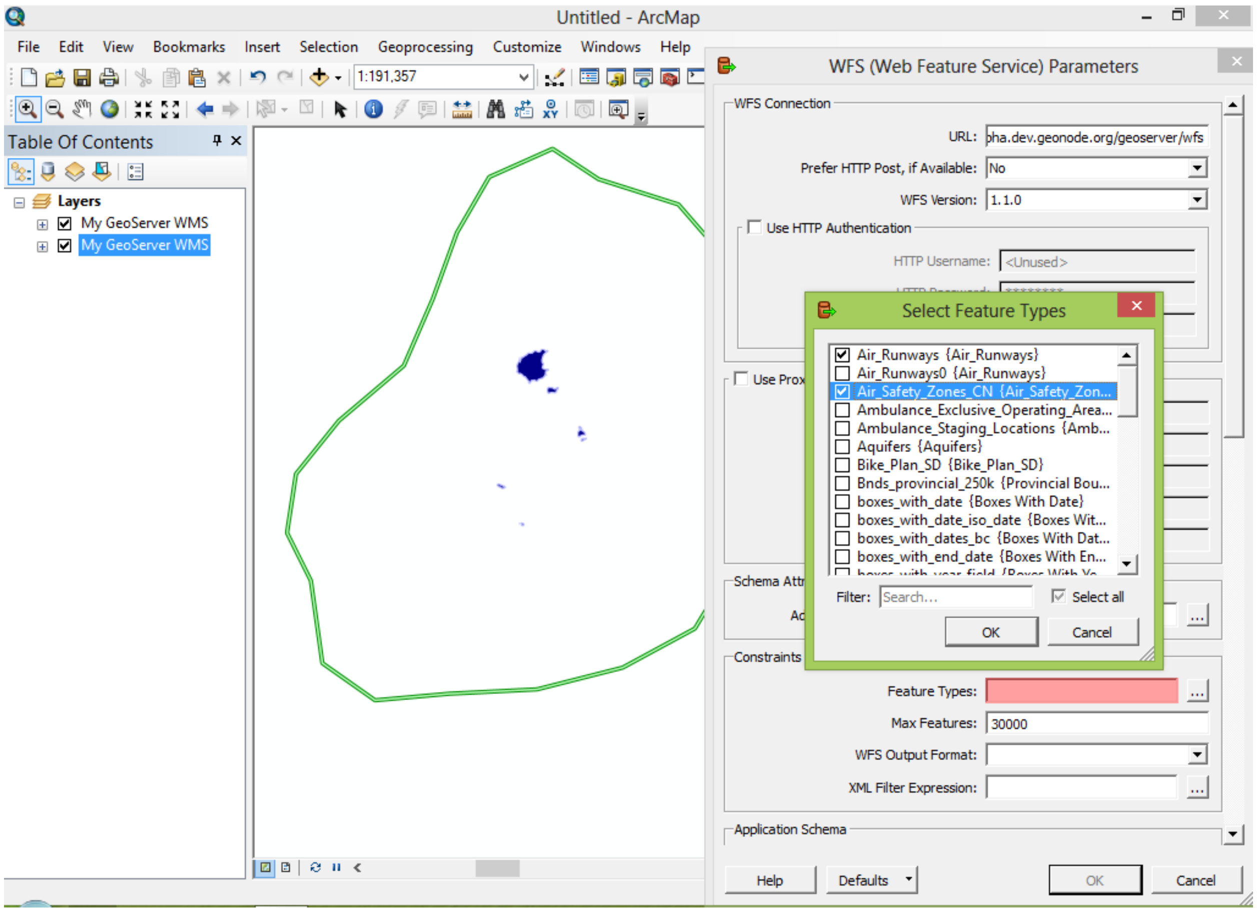
Feature Types:

Max Features: 30000

WFS Output Format:

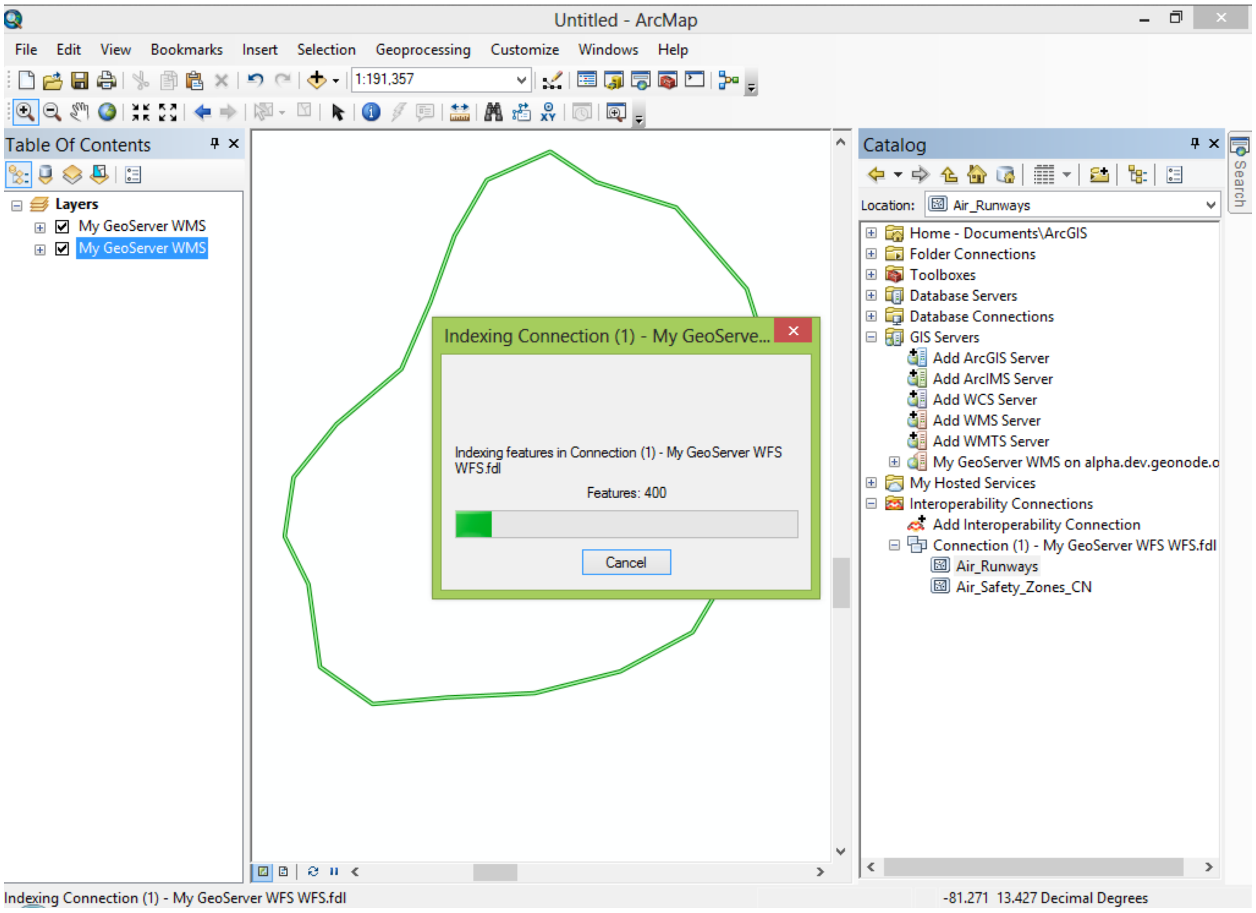
XML Filter Expression:

Application Schema



URL: pha.dev.geonode.org/geoserver/wfs
Prefer HTTP Post, if Available: No
WFS Version: 1.1.0
HTTP Username: <Unused>
Feature Types:
Max Features: 30000
WFS Output Format:
XML Filter Expression:

- [x] Air_Runways {Air_Runways}
- [ ] Air_Runways0 {Air_Runways}
- [x] Air_Safety_Zones_CN {Air_Safety_Zon...
- [ ] Ambulance_Exclusive_Operating_Area...
- [ ] Ambulance_Staging_Locations {Amb...
- [ ] Aquifers {Aquifers}
- [ ] Bike_Plan_SD {Bike_Plan_SD}
- [ ] Bnds_provincial_250k {Provincial Bou...
- [ ] boxes_with_date {Boxes With Date}
- [ ] boxes_with_date_iso_date {Boxes Wit...
- [ ] boxes_with_dates_bc {Boxes With Dat...
- [ ] boxes_with_end_date {Boxes With En...



Indexing Connection (1) - My GeoServe...

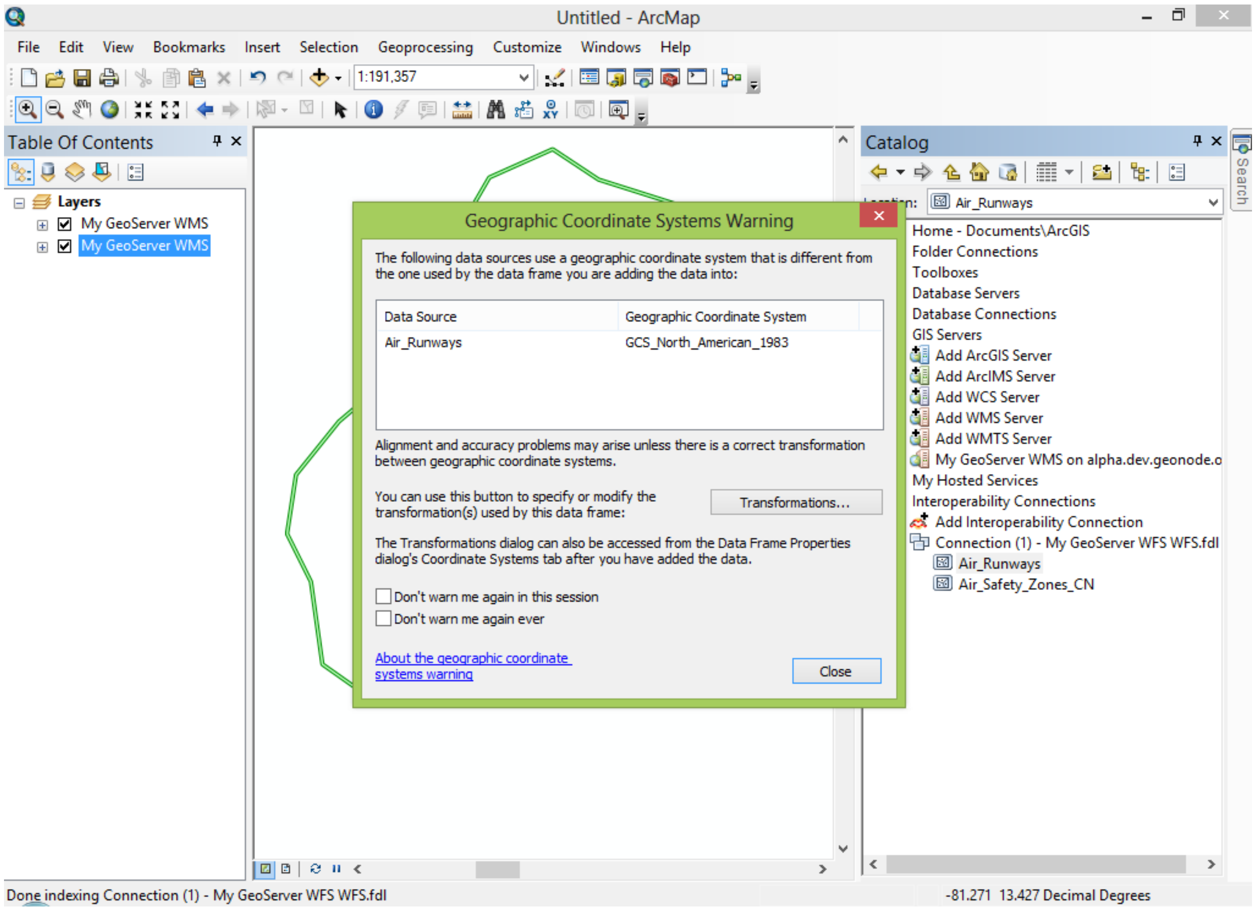
Indexing features in Connection (1) - My GeoServer WFS WFS.fdl

Features: 400

Indexing Connection (1) - My GeoServer WFS WFS.fdl

-81.271 13.427 Decimal Degrees

Depending on the projection of your data, you may receive a warning about Alignment and Accuracy of data transformations. You can specify the transformation manually or simply hit close to ignore this dialog. If you dont want to be warned again, use the checkboxes in this dialog to hide these warnings temporarily or permanently.



The following data sources use a geographic coordinate system that is different from the one used by the data frame you are adding the data into:

Data Source	Geographic Coordinate System
Air_Runways	GCS_North_American_1983

Alignment and accuracy problems may arise unless there is a correct transformation between geographic coordinate systems.

You can use this button to specify or modify the transformation(s) used by this data frame:

The Transformations dialog can also be accessed from the Data Frame Properties dialog's Coordinate Systems tab after you have added the data.

☐ Don't warn me again in this session

☐ Don't warn me again ever

About the geographic coordinate systems warning

Done indexing Connection (1) - My GeoServer WFS WFS.fdl

-81.271 13.427 Decimal Degrees

Your WFS Layer will be added to your map and you can view it in the Map Panel. If you need to, use the “Zoom to Layer Extent” or other zoom tools to zoom to the bounds of your layer.

You can now use the identify tool to inspect a feature in your layer, or perform any other function that you can normally use to work with Vector Layers in ArcMap.

Since your layer was imported as actual vector features, you can use normal ArcMap styling tools to style the layer to match how you want it to be displayed.

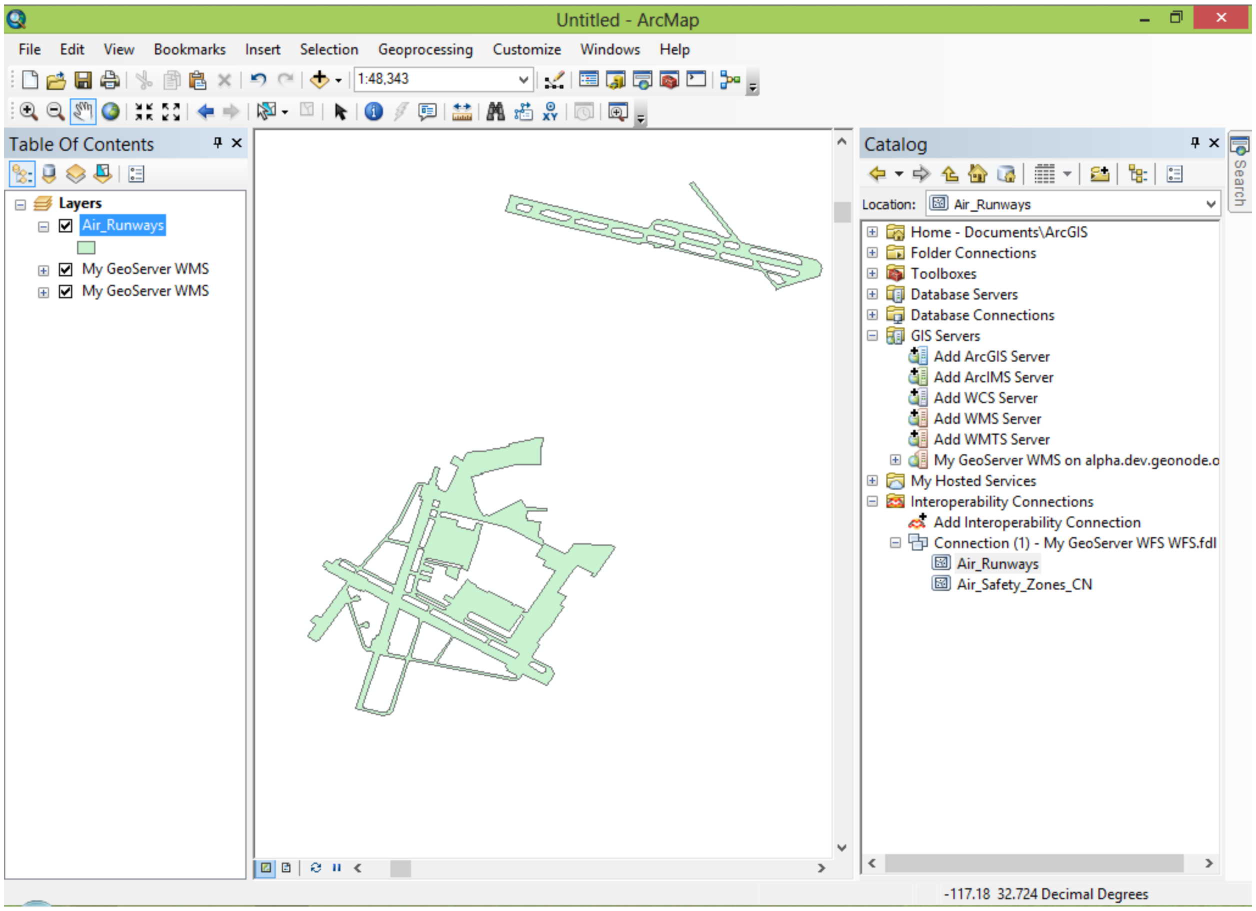
Now that you have added layers from your GeoNode as both WMS and WFS, you can explore the other options available to you with these layers within ArcMap.

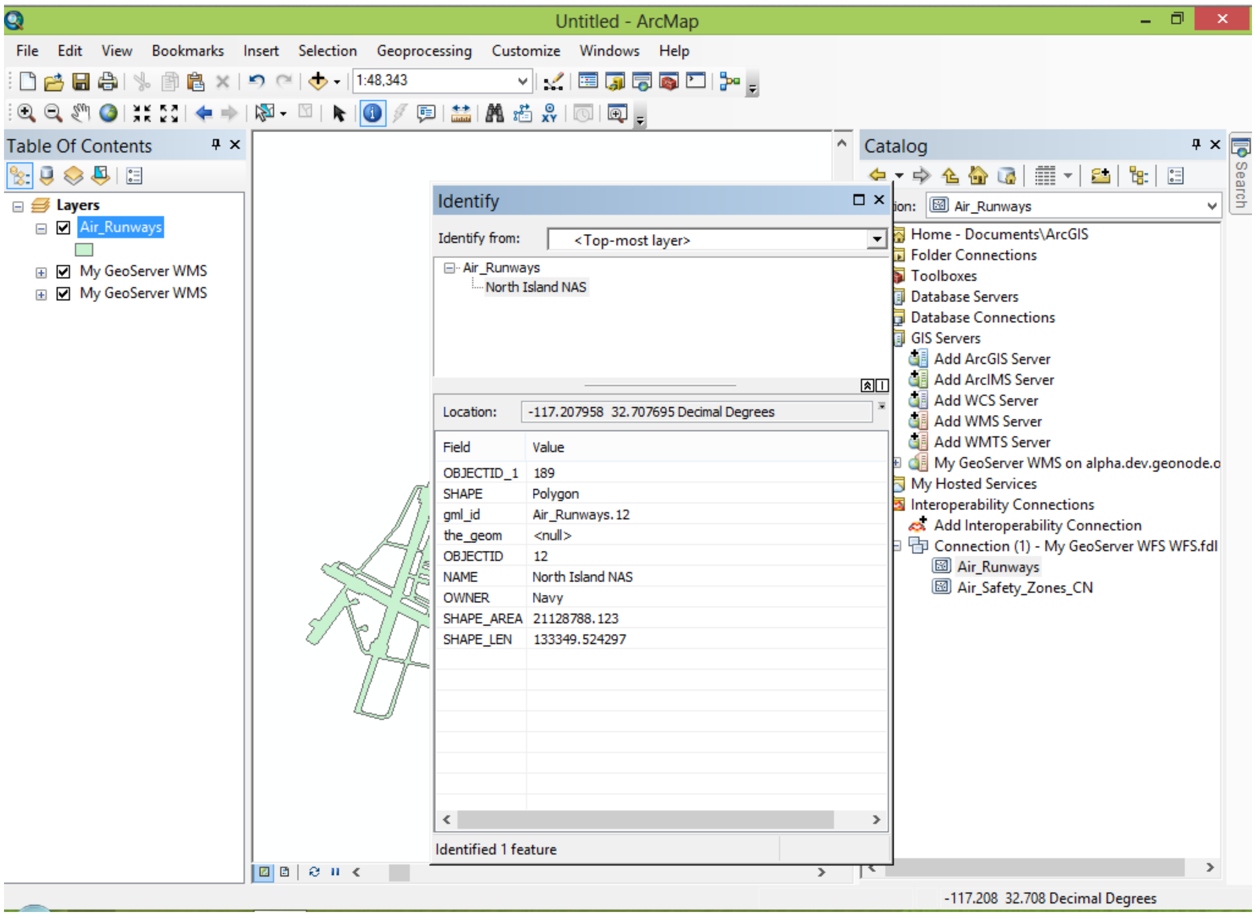
QGIS Quantum GIS or qGIS is an open source, cross platform desktop GIS app. It can also be used to add layers from your GeoNode instance as WMS or WFS. The process is very similar to how we add these same layers to ArcMap, and we will walk through the steps necessary in the following section.

First, select “Add WMS Layer” from the Layer menu.

The Add WMS Layer Dialog will be displayed where you are able to specify the parameters to connect to your WMS server.

Next, you need to fill in the parameters to connect to your GeoNode instance. The URL for your GeoNode’s WMS is the base URL + /geoserver/wms


Untitled - ArcMap
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:48,343
Table Of Contents
Layers
Air_Runways
My GeoServer WMS
My GeoServer WMS
Catalog
Location: Air_Runways
Home - Documents\ArcGIS
Folder Connections
Toolboxes
Database Servers
Database Connections
GIS Servers
Add ArcGIS Server
Add ArcIMS Server
Add WCS Server
Add WMS Server
Add WMTS Server
My GeoServer WMS on alpha.dev.geonode.o
My Hosted Services
Interoperability Connections
Add Interoperability Connection
Connection (1) - My GeoServer WFS WFS.fdl
Air_Runways
Air_Safety_Zones_CN
Search
-117.18 32.724 Decimal Degrees

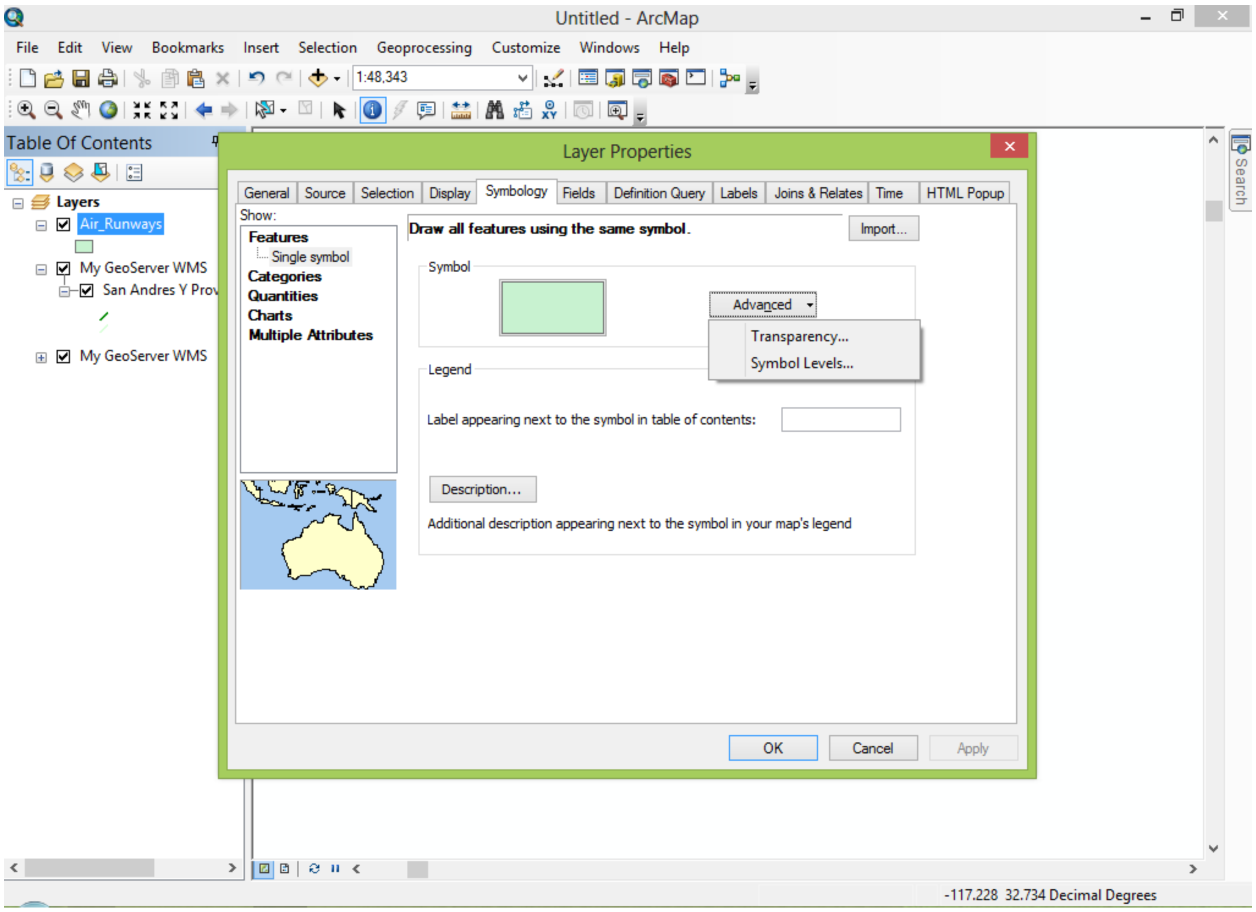


Location: -117.207958 32.707695 Decimal Degrees

Field	Value
OBJECTID_1	189
SHAPE	Polygon
gml_id	Air_Runways.12
the_geom	<null>
OBJECTID	12
NAME	North Island NAS
OWNER	Navy
SHAPE_AREA	21128788.123
SHAPE_LEN	133349.524297

Identified 1 feature

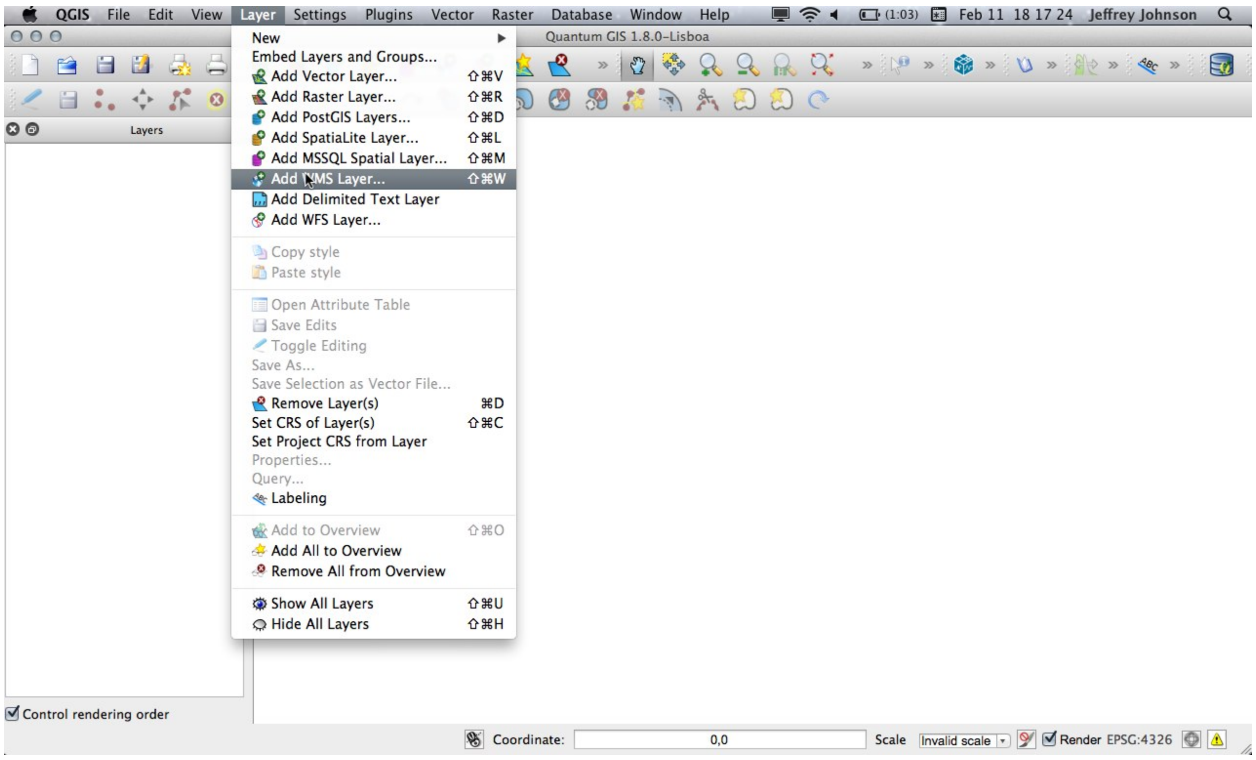
-117.208 32.708 Decimal Degrees

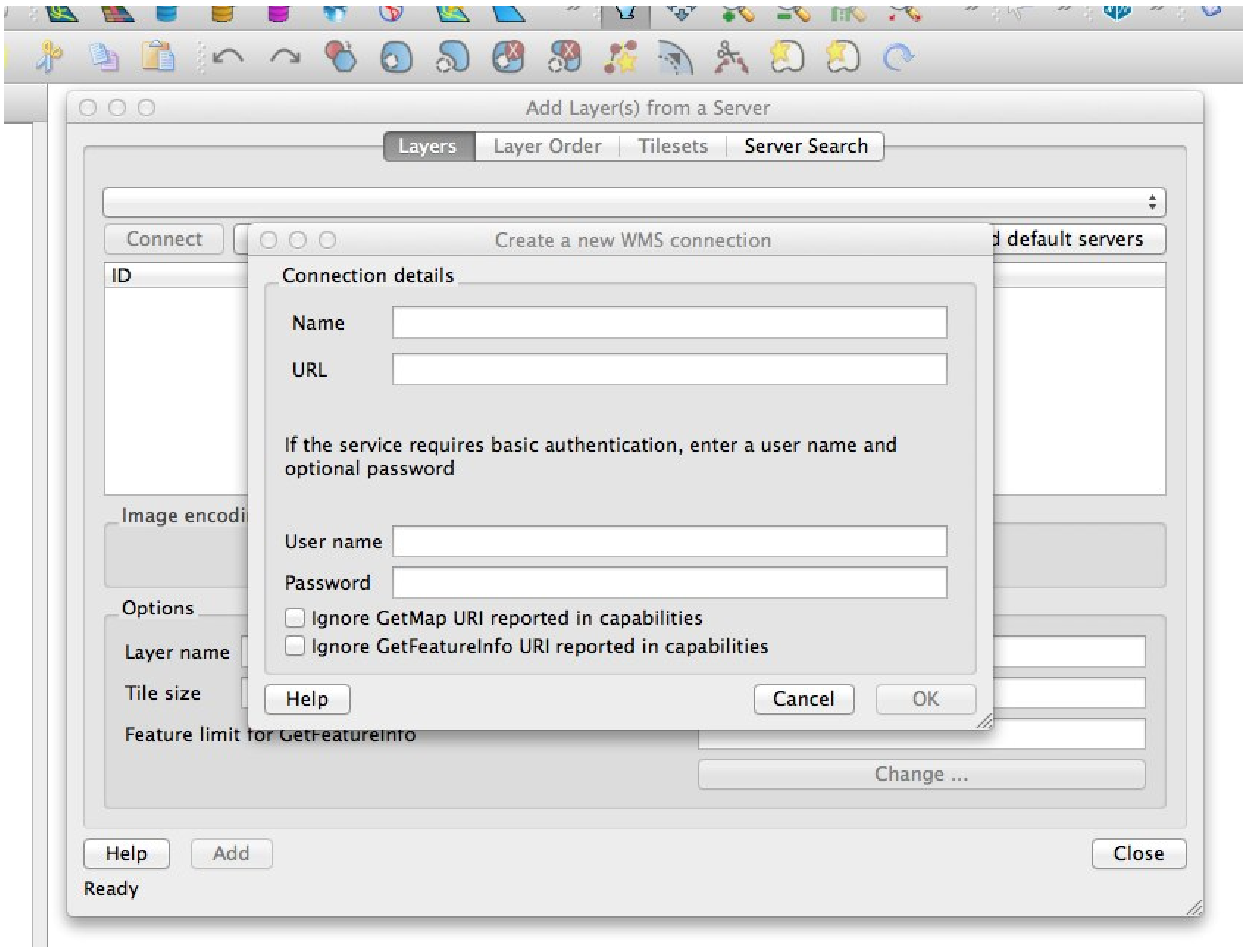


Draw all features using the same symbol.

Label appearing next to the symbol in table of contents:

Additional description appearing next to the symbol in your map's legend





Connection details

Name

URL

If the service requires basic authentication, enter a user name and optional password

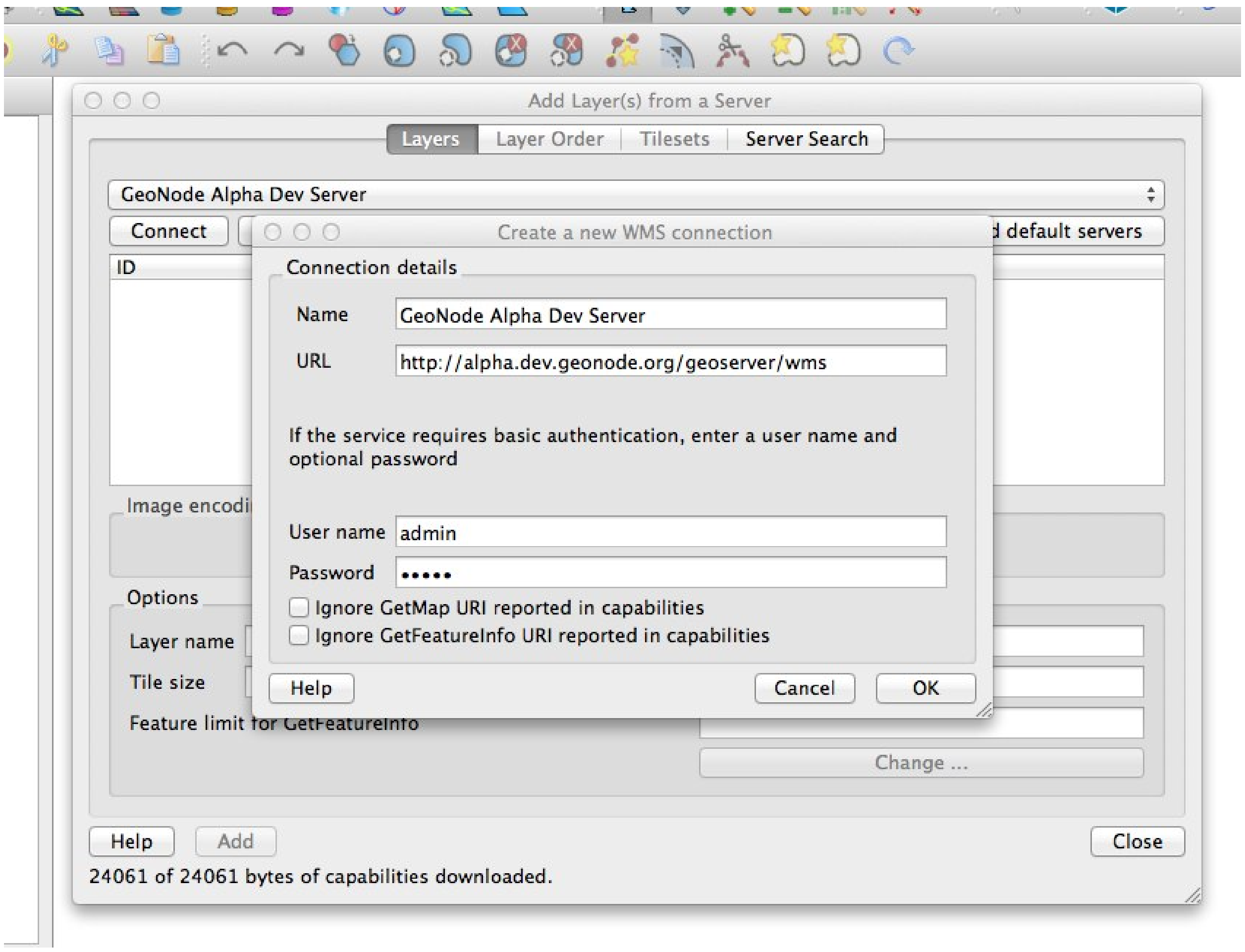
User name

Password

☐ Ignore GetMap URI reported in capabilities

☐ Ignore GetFeatureInfo URI reported in capabilities

Help Cancel OK



GeoNode Alpha Dev Server

Name: GeoNode Alpha Dev Server

URL: http://alpha.dev.geonode.org/geoserver/wms

If the service requires basic authentication, enter a user name and optional password

User name: admin

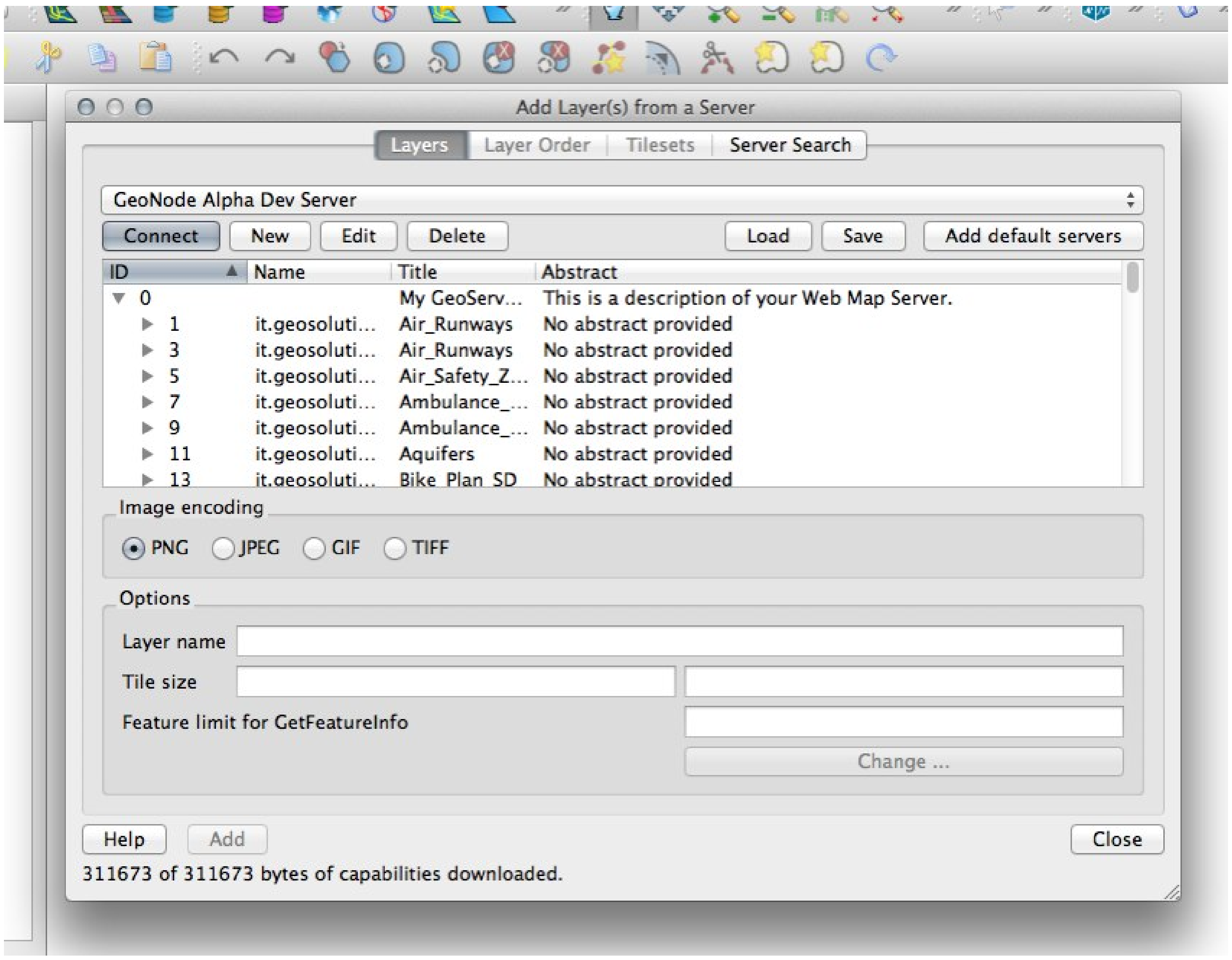
Password: •••••

☐ Ignore GetMap URI reported in capabilities

☐ Ignore GetFeatureInfo URI reported in capabilities

24061 of 24061 bytes of capabilities downloaded.

After clicking the OK button, your server will show up in the list of servers. Make sure its selected, then, click the connect button to have QGIS retrieve the list of layers from your GeoNode.



ID	Name	Title	Abstract
0		My GeoServ...	This is a description of your Web Map Server.
1	it.geosoluti...	Air_Runways	No abstract provided
3	it.geosoluti...	Air_Runways	No abstract provided
5	it.geosoluti...	Air_Safety_Z...	No abstract provided
7	it.geosoluti...	Ambulance_...	No abstract provided
9	it.geosoluti...	Ambulance_...	No abstract provided
11	it.geosoluti...	Aquifers	No abstract provided
13	it.geosoluti...	Bike_Plan_SD	No abstract provided

311673 of 311673 bytes of capabilities downloaded.

Select the layers you want to add to your QGIS project and click “Add”.

Your layer will be displayed in the map panel.

You can then zoom into your features in the Map.

From there, you can use the identify tool to inspect the attributes of one of the features on the map.

Or, you can look at the layer metadata by right clicking on the layer and selecting Layer Properties and selecting the metadata tab.

Adding WFS servers and layers to your QGIS project is very similar to adding WMS. Depending on your version of QGIS, you may need to add the WFS plugin. You can use the Plugin manager to add it.

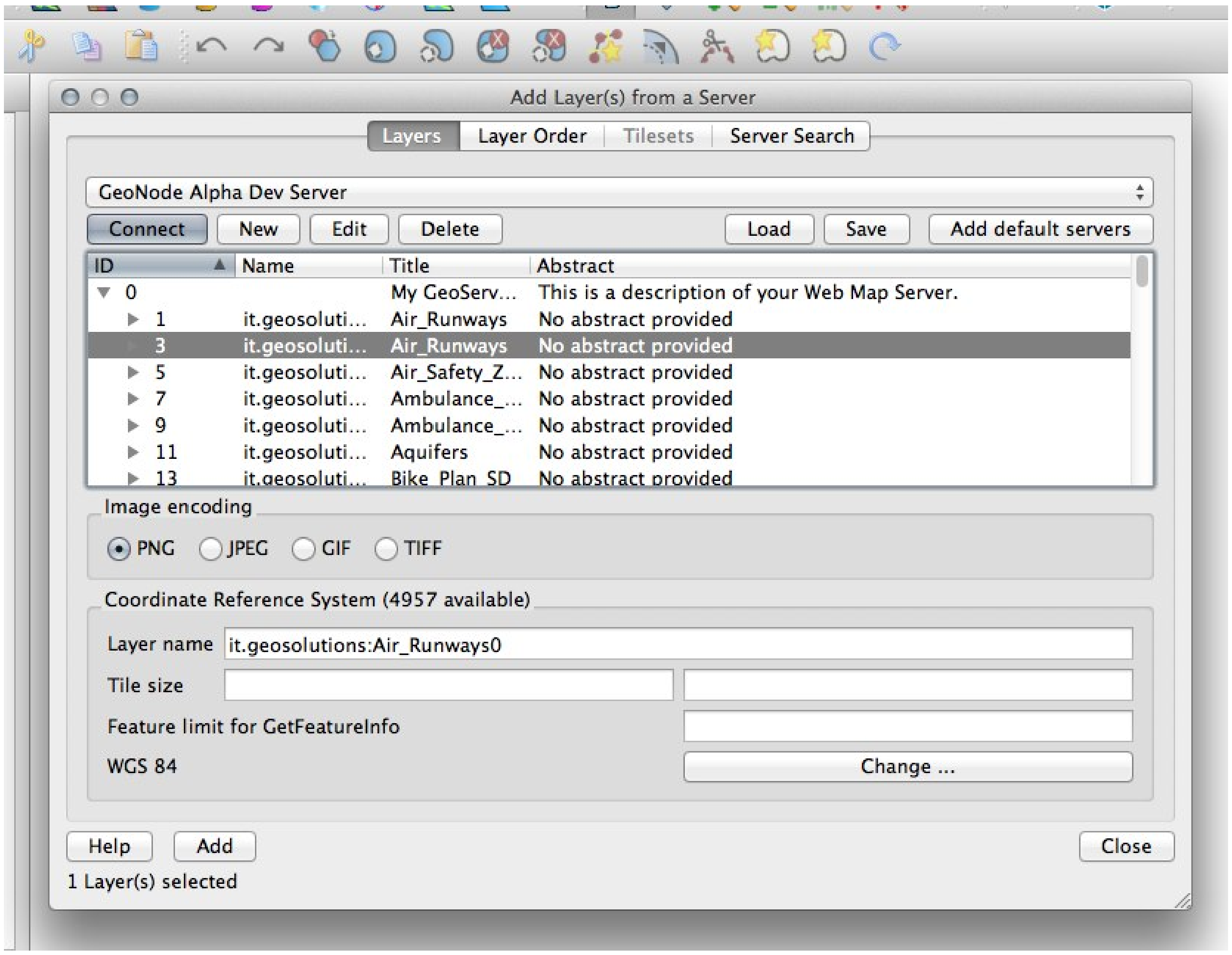
Once the plugin is installed, you can select the “Add WFS Layer” option from the Layer menu.

Step through the same process you did for WMS to create a new WFS connection. First specify server parameters and click OK.

Then click Connect to retrieve the list of layers on the server and select the layers you want to add and click Apply.

The layer(s) you selected will be displayed in the map panel.

You can use the same identify tool to inspect features in the map panel.



Add Layer(s) from a Server

GeoNode Alpha Dev Server

ID	Name	Title	Abstract
0		My GeoServ...	This is a description of your Web Map Server.
1	it.geosoluti...	Air_Runways	No abstract provided
3	it.geosoluti...	Air_Runways	No abstract provided
5	it.geosoluti...	Air_Safety_Z...	No abstract provided
7	it.geosoluti...	Ambulance_...	No abstract provided
9	it.geosoluti...	Ambulance_...	No abstract provided
11	it.geosoluti...	Aquifers	No abstract provided
13	it.geosoluti...	Bike_Plan_SD	No abstract provided

Image encoding

(● PNG) ○ JPEG ○ GIF ○ TIFF

Coordinate Reference System (4957 available)

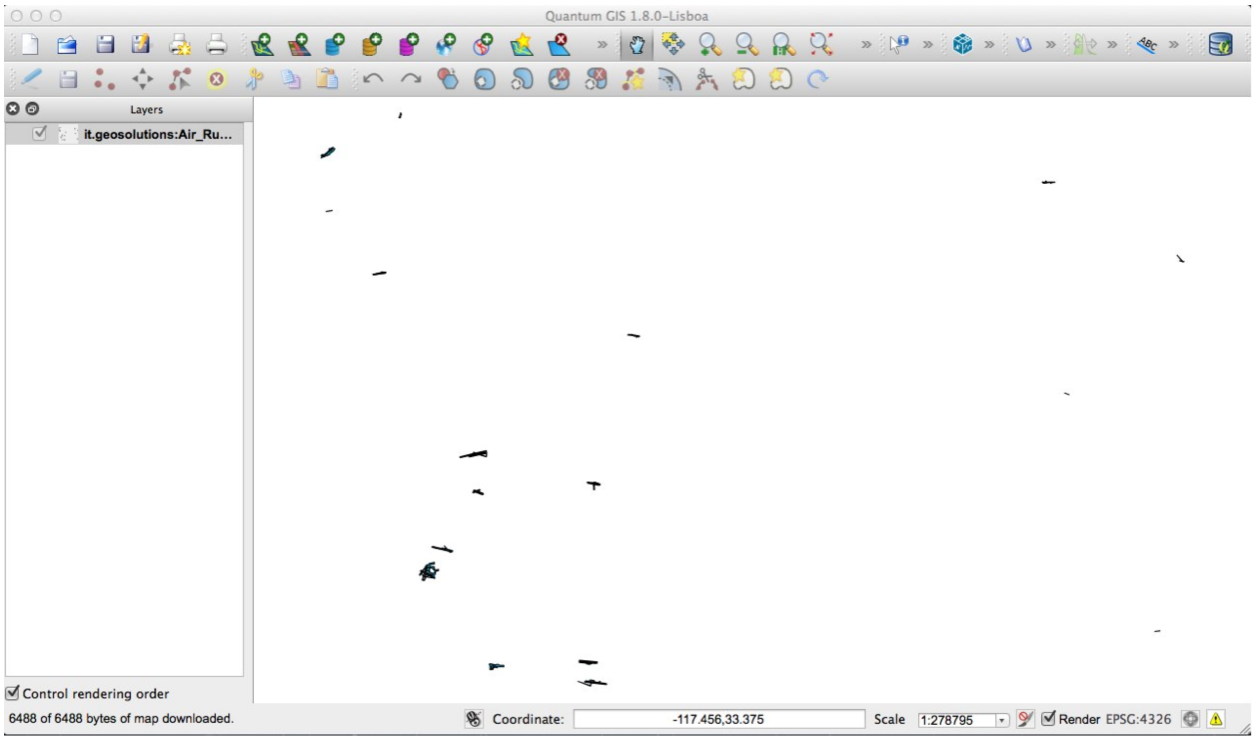
Layer name: it.geosolutions:Air_Runways0

Tile size

Feature limit for GetFeatureInfo

WGS 84

1 Layer(s) selected

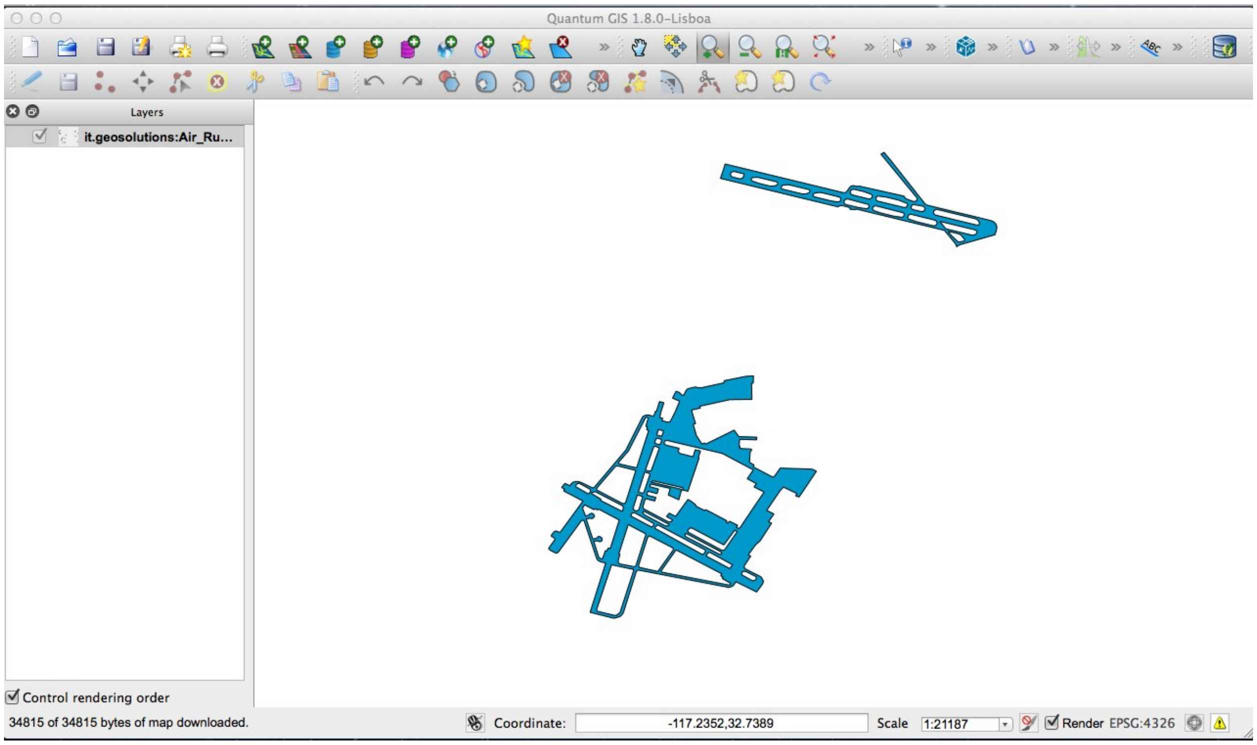


Quantum GIS 1.8.0-Lisboa

☑ Control rendering order

6488 of 6488 bytes of map downloaded.

Coordinate: -117.456,33.375 Scale 1:278795 ☑ Render EPSG:4326


Quantum GIS 1.8.0-Lisboa
Layers
it.geosolutions:Air_Ru...
Control rendering order
34815 of 34815 bytes of map downloaded.
Coordinate: -117.2352,32.7389
Scale 1:21187
Render EPSG:4326

To look at more information about your layer, right click the layer in the Table of Contents and select Layer Properties. You can look at the list of fields.

... or set a style to match how you want your data to be displayed.

You now know how to add layers from your GeoNode instance to a QGIS project. You can explore all of the other options available to you in QGIS by consulting its documentation.

Google Earth GeoNode's built in map interface lets you look at your layers and maps in the Google Earth plugin directly in your browser. You can switch to this 3D viewer directly in GeoNode by clicking the google earth icon in the map panel.

GeoServer will render your layer as an image until you are zoomed in sufficiently, and then it will switch to rendering it as a vector overlay that you can click on to view the attributes for the feature you clicked on.

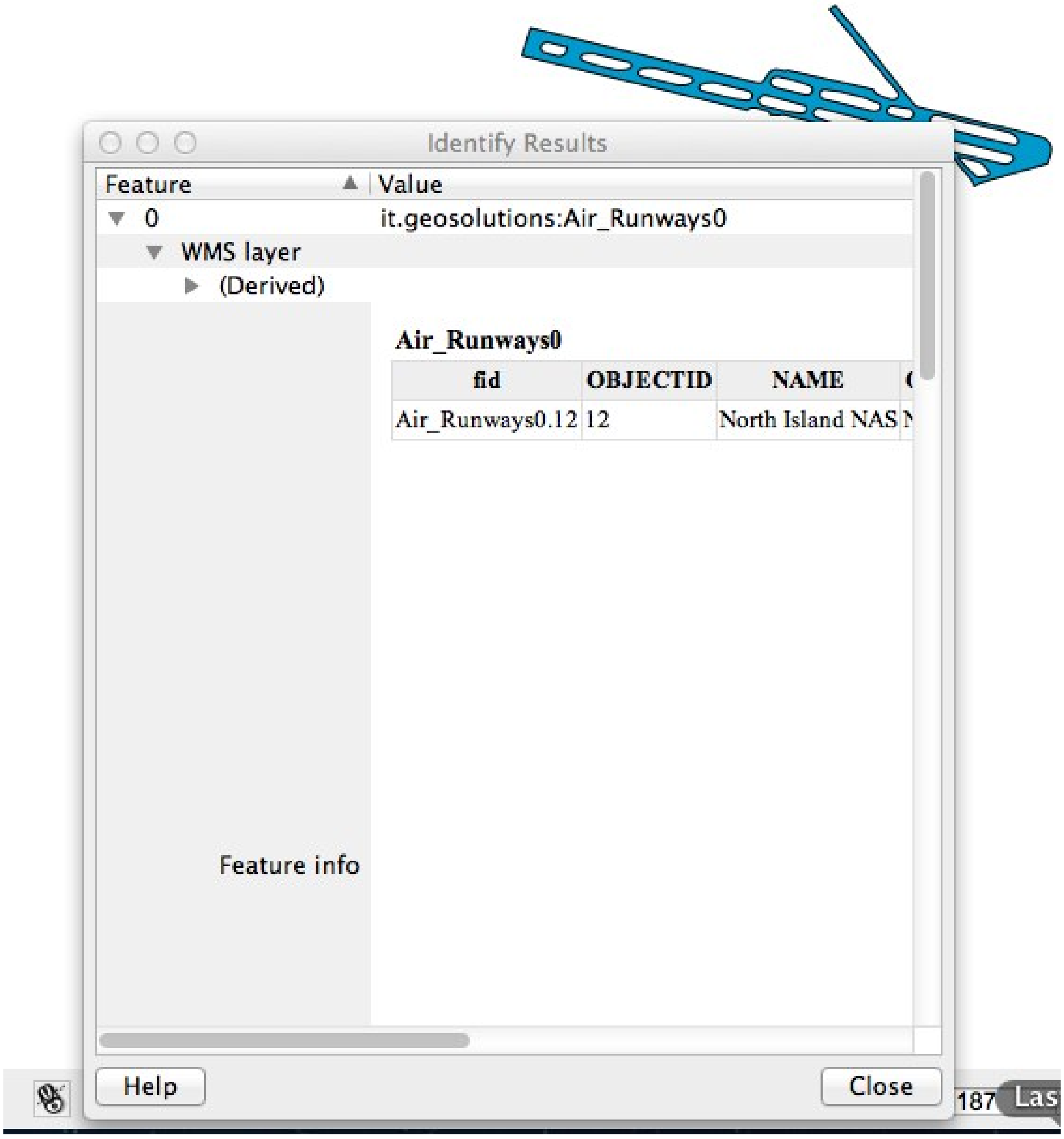
You can also use this option in the GeoExplorer client by clicking the same button.

Note: Some of the GeoExplorer options will not be available to you when you are in this mode, they will be grayed out an inaccessible.

If instead you want to use layers from your GeoNode in the Google Earth client itself, you have a few options available to you.

First, you can select the KML option from the Download Layer menu to download the entire layer in a single KML file. Depending on the size of the layer, your GeoNode could take several seconds or longer to generate this KML and return it to you.

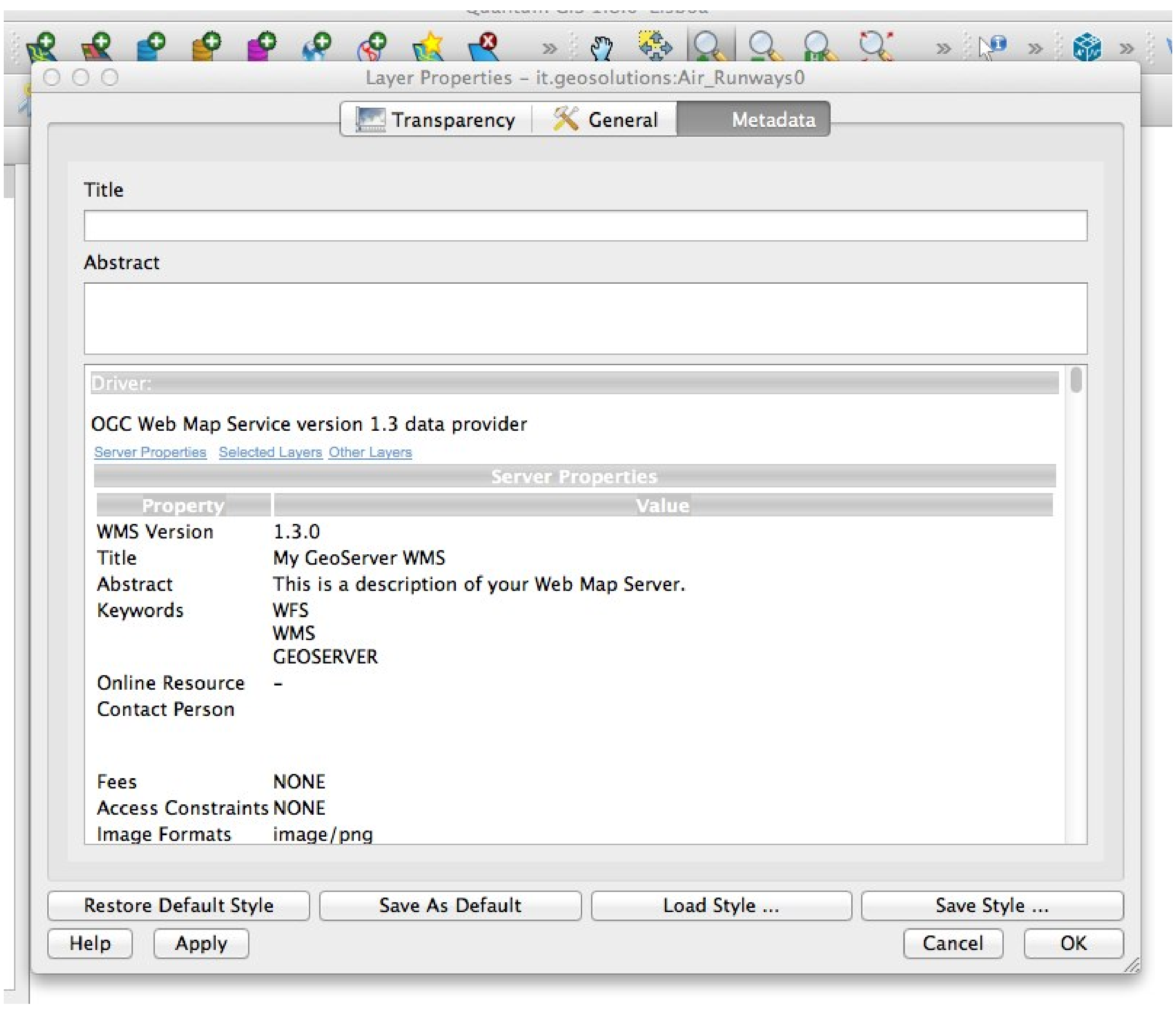
When the layer is generated, it will be downloaded to your desktop machine and you can simply double click it to open it in Google Earth.



Feature	Value
0	it.geosolutions:Air_Runways0
WMS layer	
(Derived)	
Feature info	

Air_Runways0

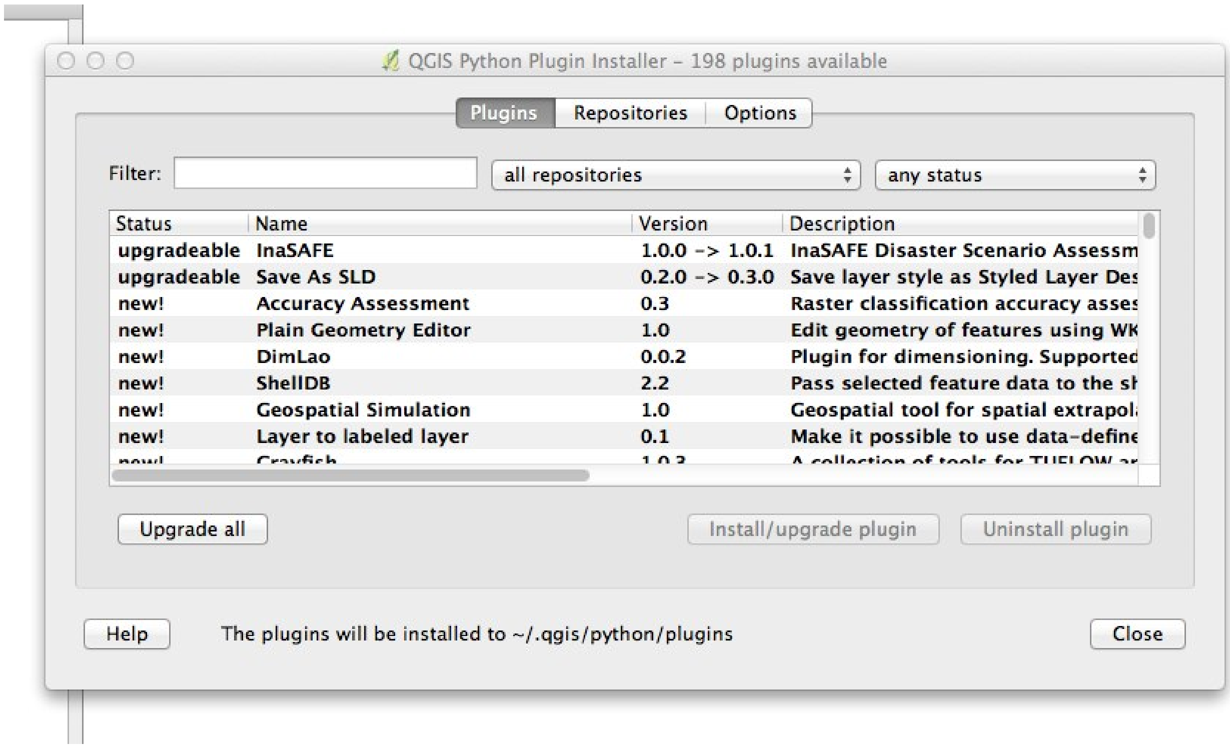
fid	OBJECTID	NAME
Air_Runways0.12	12	North Island NAS



OGC Web Map Service version 1.3 data provider

Server Properties

Property	Value
WMS Version	1.3.0
Title	My GeoServer WMS
Abstract	This is a description of your Web Map Server.
Keywords	WFS WMS GEOSERVER
Online Resource	-
Contact Person	
Fees	NONE
Access Constraints	NONE
Image Formats	image/png


QGIS Python Plugin Installer – 198 plugins available

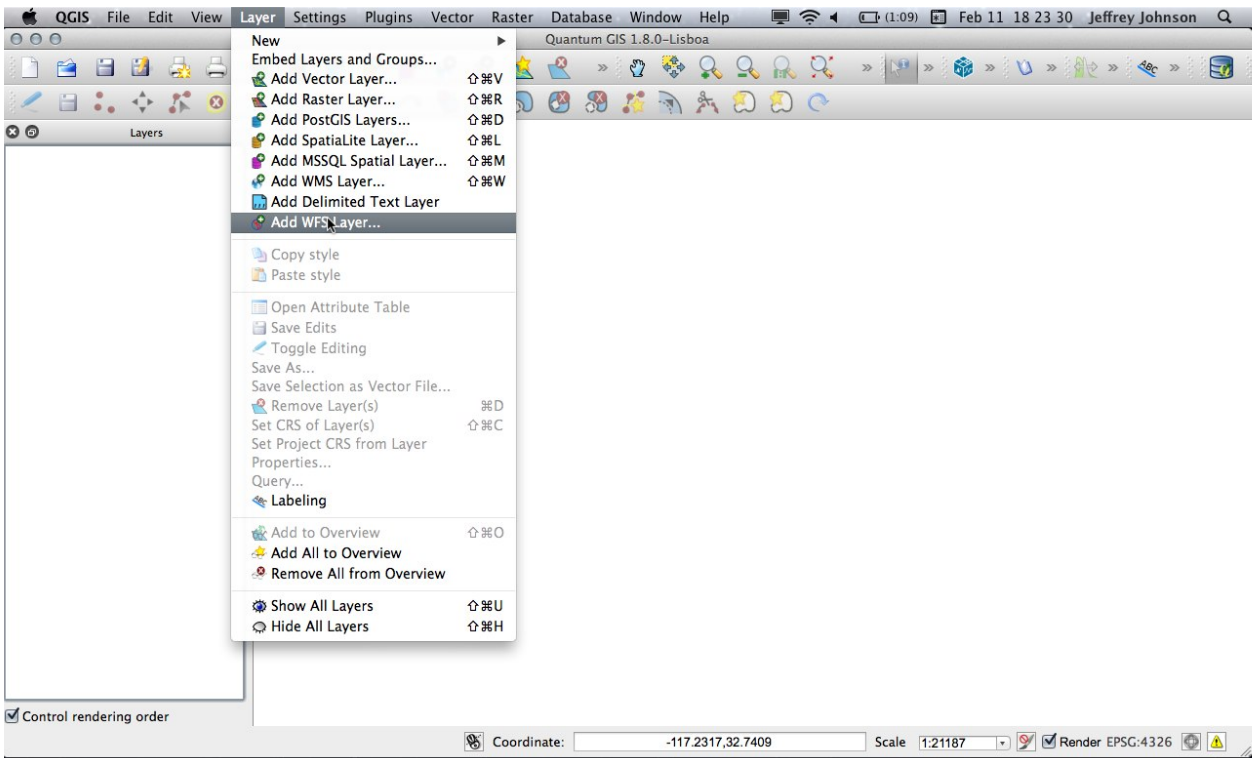
Plugins | Repositories | Options

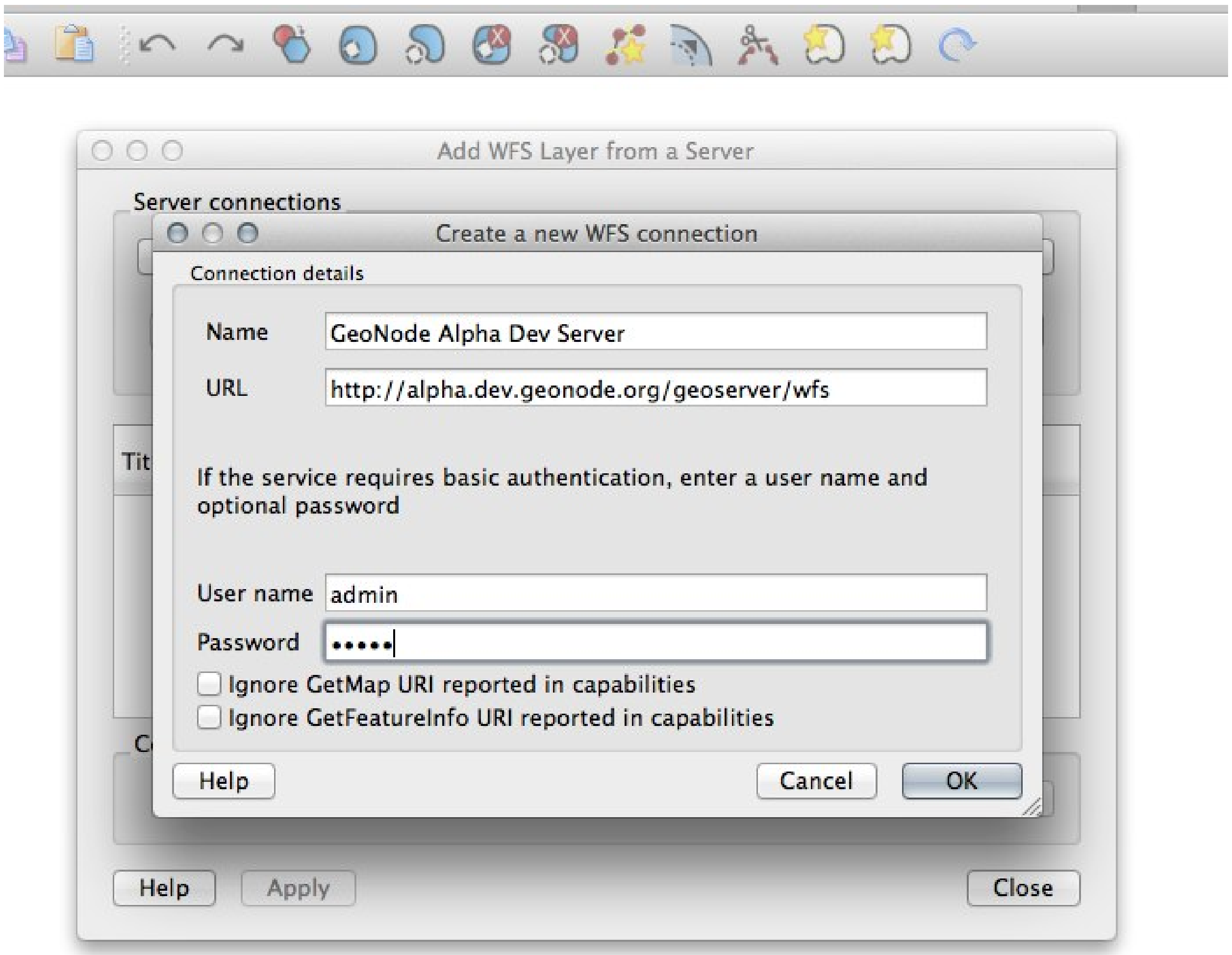
Filter: | all repositories | any status

Status	Name	Version	Description
upgradeable	InaSAFE	1.0.0 -> 1.0.1	InaSAFE Disaster Scenario Assessm
upgradeable	Save As SLD	0.2.0 -> 0.3.0	Save layer style as Styled Layer Des
new!	Accuracy Assessment	0.3	Raster classification accuracy asses
new!	Plain Geometry Editor	1.0	Edit geometry of features using WK
new!	DimLao	0.0.2	Plugin for dimensioning. Supported
new!	ShellDB	2.2	Pass selected feature data to the sh
new!	Geospatial Simulation	1.0	Geospatial tool for spatial extrapol
new!	Layer to labeled layer	0.1	Make it possible to use data-define
new!	Crayfish	1.0.3	A collection of tools for TUFLOW ar

Upgrade all | Install/upgrade plugin | Uninstall plugin

Help | The plugins will be installed to ~/.qgis/python/plugins | Close





Name: GeoNode Alpha Dev Server

URL: http://alpha.dev.geonode.org/geoserver/wfs

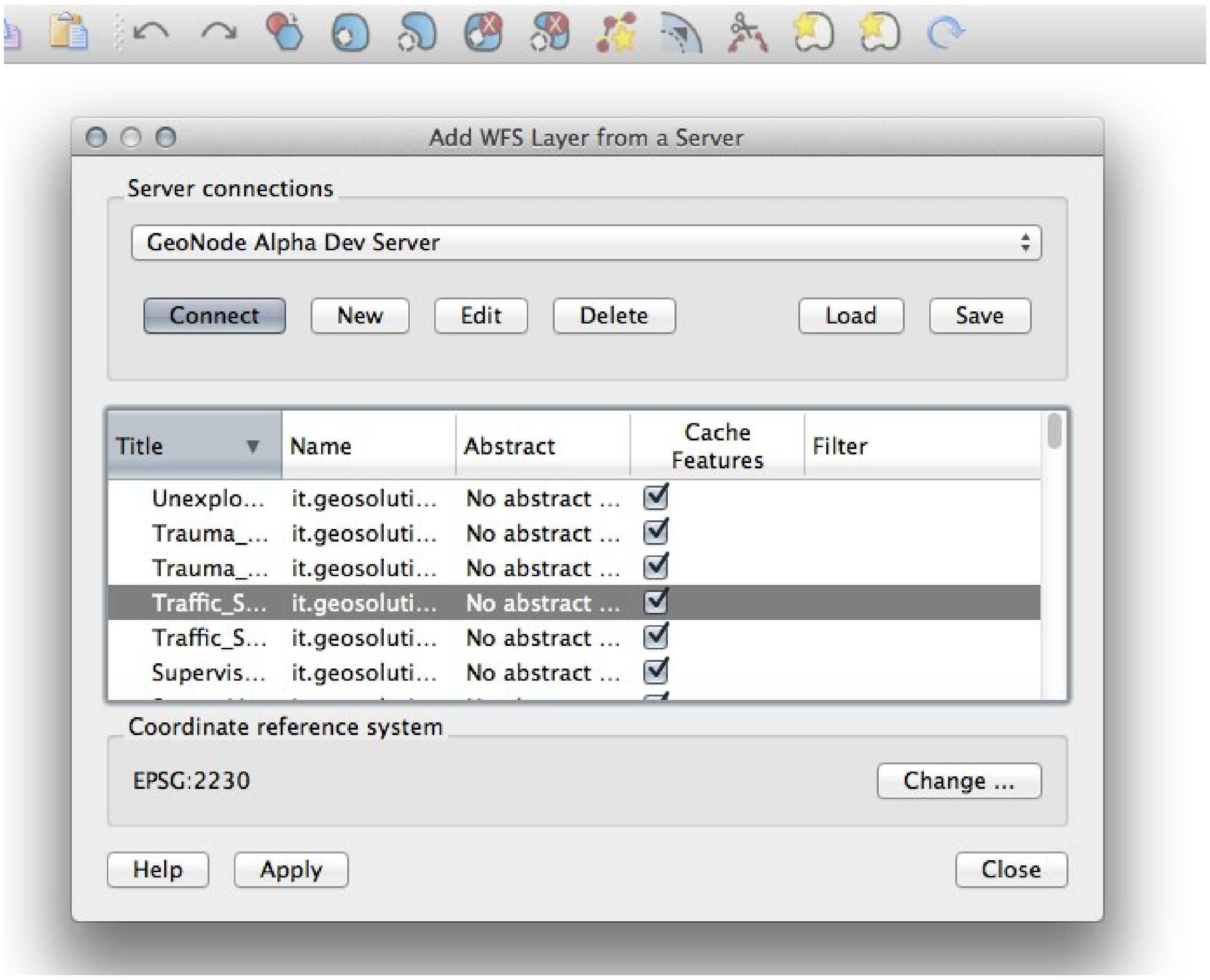
If the service requires basic authentication, enter a user name and optional password

User name: admin

Password: •••••

[ ] Ignore GetMap URI reported in capabilities

[ ] Ignore GetFeatureInfo URI reported in capabilities

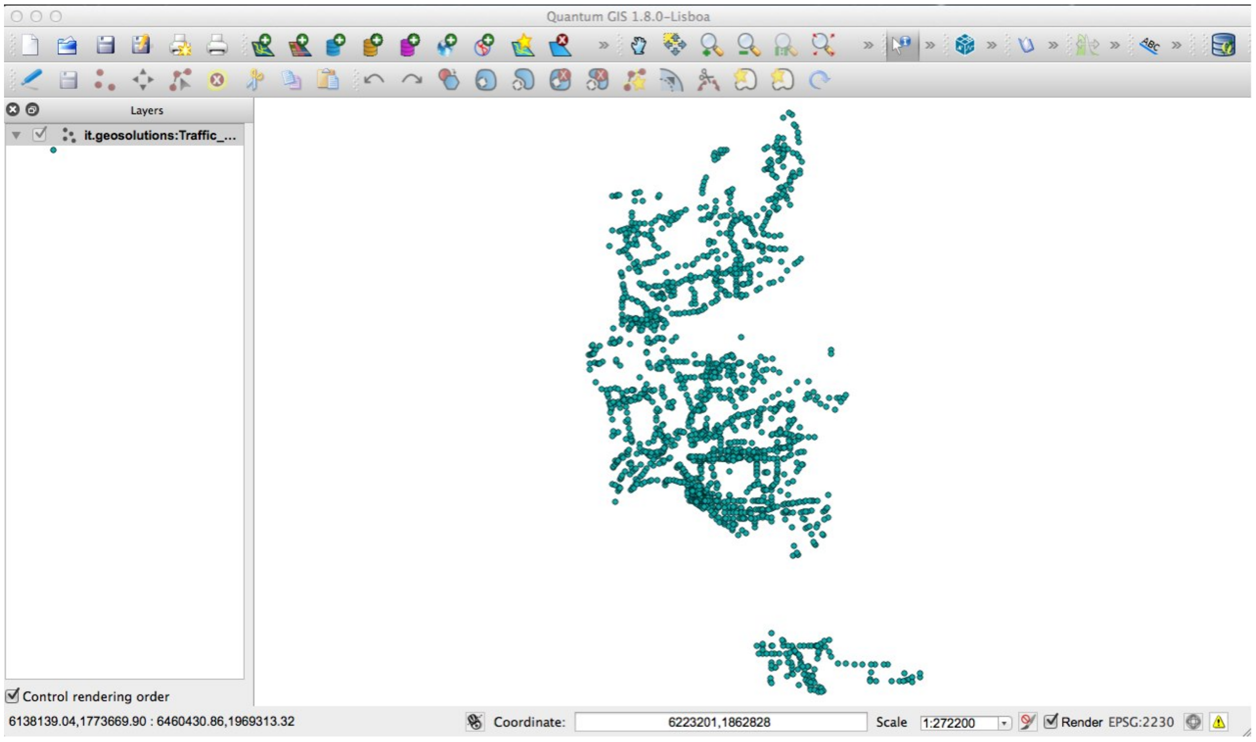


GeoNode Alpha Dev Server

Title	Name	Abstract	Cache Features	Filter
Unexplo...	it.geosoluti...	No abstract ...	✓	
Trauma_...	it.geosoluti...	No abstract ...	✓	
Trauma_...	it.geosoluti...	No abstract ...	✓	
Traffic_S...	it.geosoluti...	No abstract ...	✓	
Traffic_S...	it.geosoluti...	No abstract ...	✓	
Supervis...	it.geosoluti...	No abstract ...	✓	

Coordinate reference system

EPSG:2230



Alternatively, you can use the “View in Google Earth” option in the Layer Download menu to view the layer in Google Earth using the same methodology described above depending on the zoom level.

This will download a small KMZ to your desktop that contains a reference to the layers on the server and you can double click it to open it in Google Earth.

Note: The basic difference between these two options is that the first downloads *all* of the data to your desktop at once and as such, the downloaded file can be used offline while the second is simply a Network Link to the layer on the server. Choose whichever method is best for your own needs and purposes.

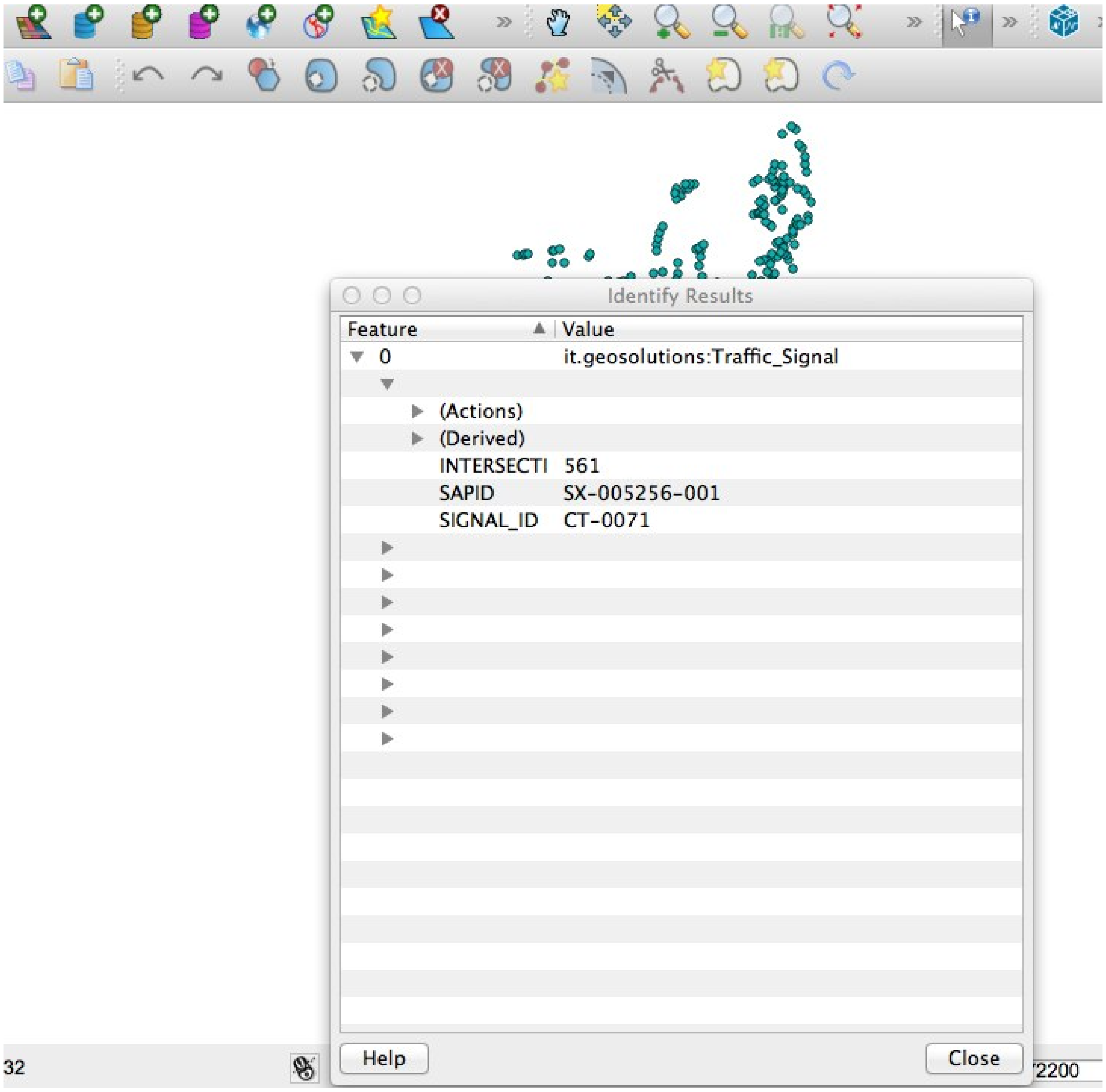
Once you have added your layers to the Places panel in Google Earth, you can move them from the Temporary Places section into My Places if you wish to use them after your current Google Earth session is complete. You can arrange them in folders and use Google Earth functionality to save your project to disk. Consult Google Earths documentation for more information about how to do this.

Spatial Processing with GeoNode

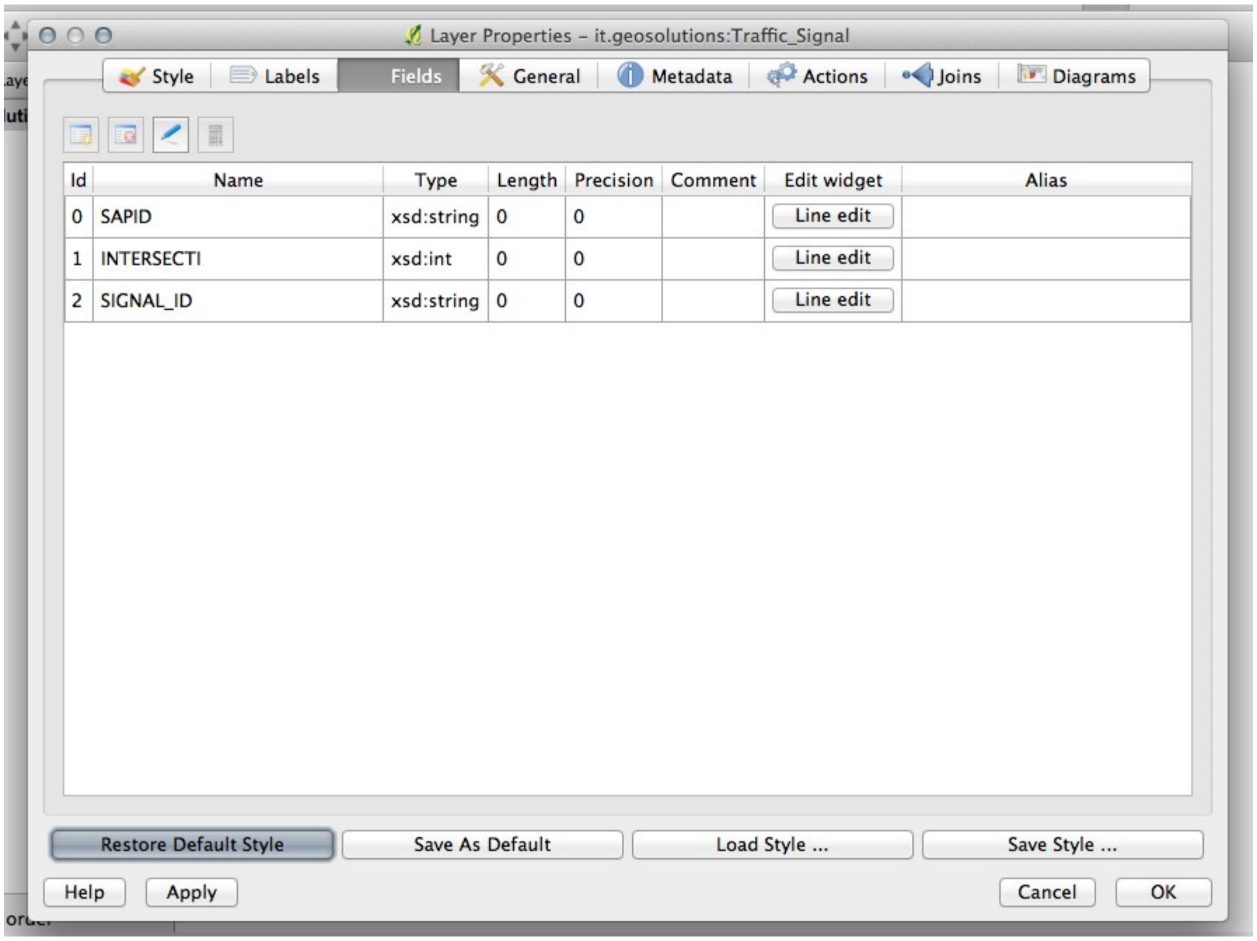
Spatial Processing with Sextante Once you have connected to a GeoNode service and its data is available from QGIS, analysis can be performed on it, just as if you were using local data.

SEXTANTE is the analysis and data processing framework of QGIS, and it can be used to run a large number of different analysis algorithms for both raster and vector data. Both WCS and WFS connections can be used to get data that can be processes through SEXTANTE.

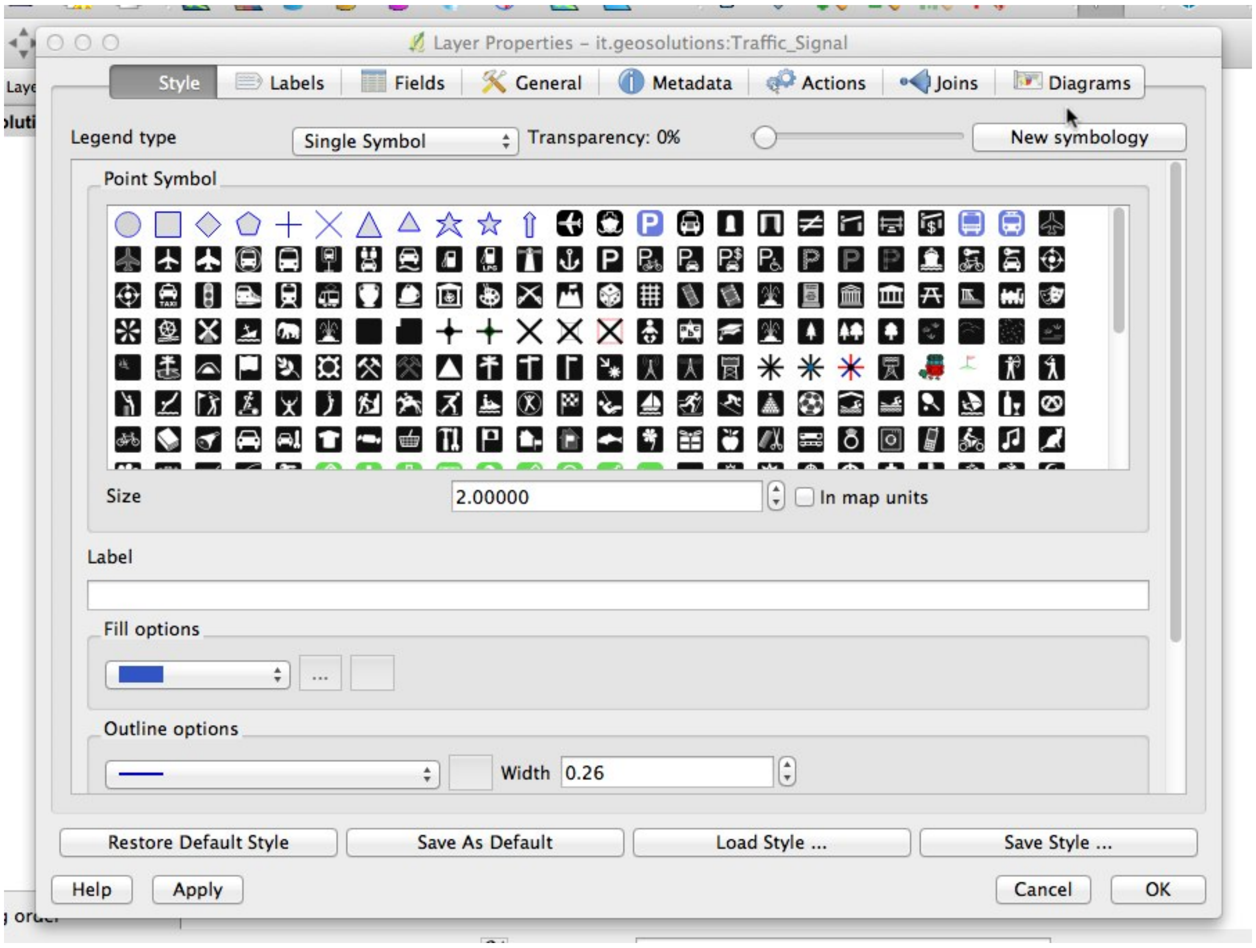
The SEXTANTE toolbox is the main component to call processing algorithms, and it contains a list all of available ones, organized by providers.



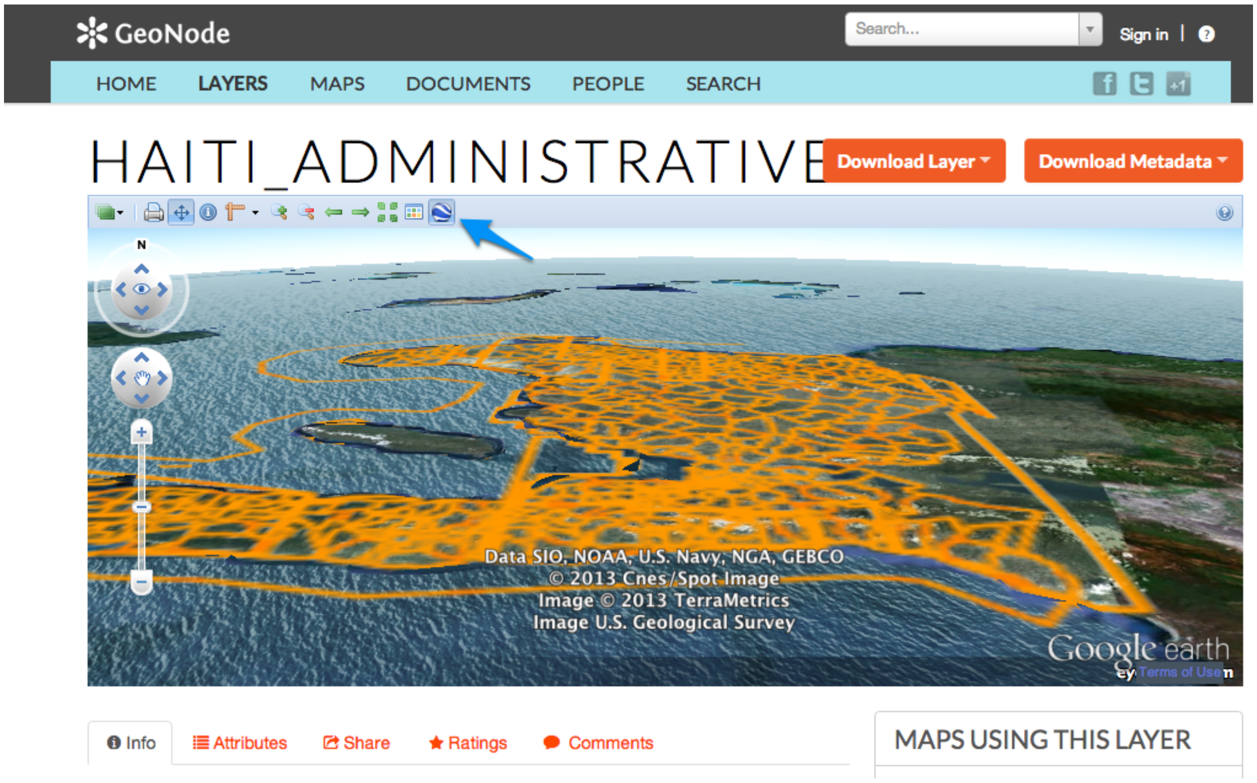
Feature	Value
0	it.geosolutions:Traffic_Signal
(Actions)	
(Derived)	
INTERSECTI	561
SAPID	SX-005256-001
SIGNAL_ID	CT-0071

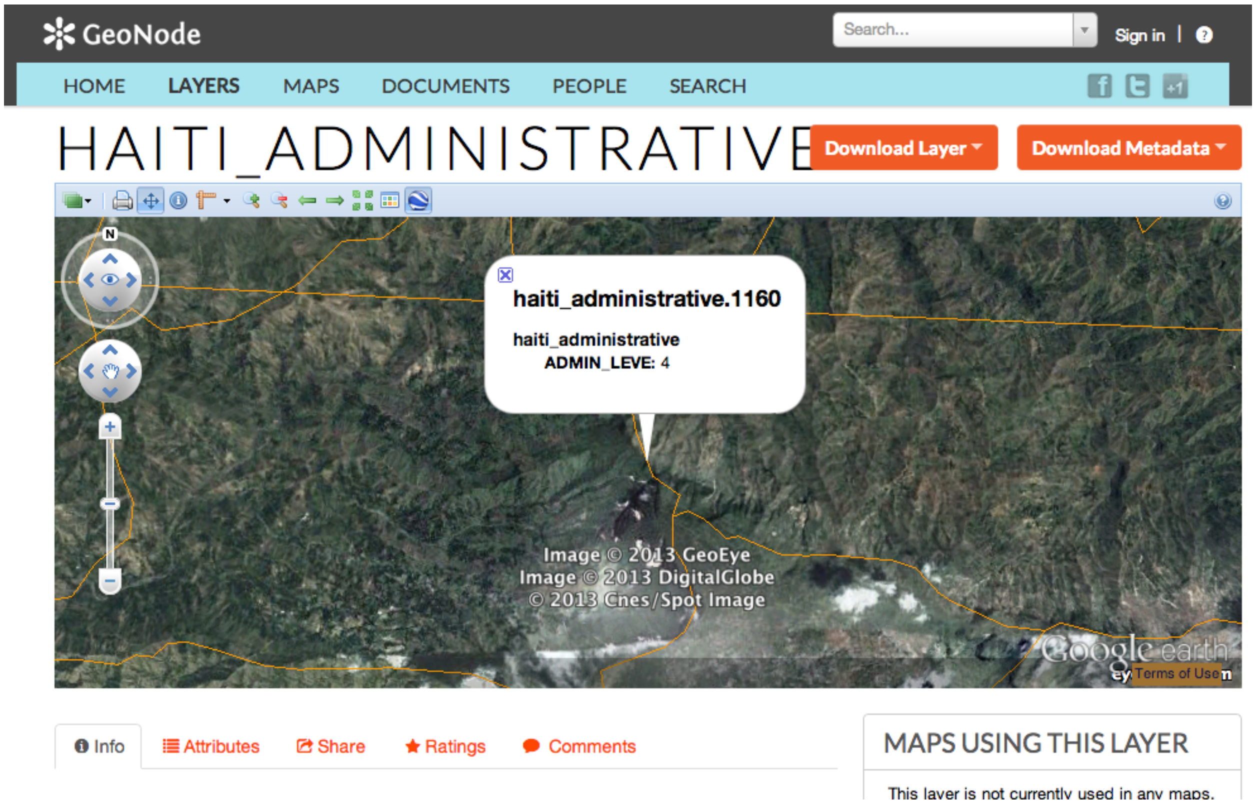


Id	Name	Type	Length	Precision	Comment	Edit widget	Alias
0	SAPID	xsd:string	0	0		Line edit	
1	INTERSECTI	xsd:int	0	0		Line edit	
2	SIGNAL_ID	xsd:string	0	0		Line edit	



Legend type: Single Symbol
Transparency: 0%
Size: 2.00000 ☐ In map units
Label
Fill options
Outline options
Width: 0.26

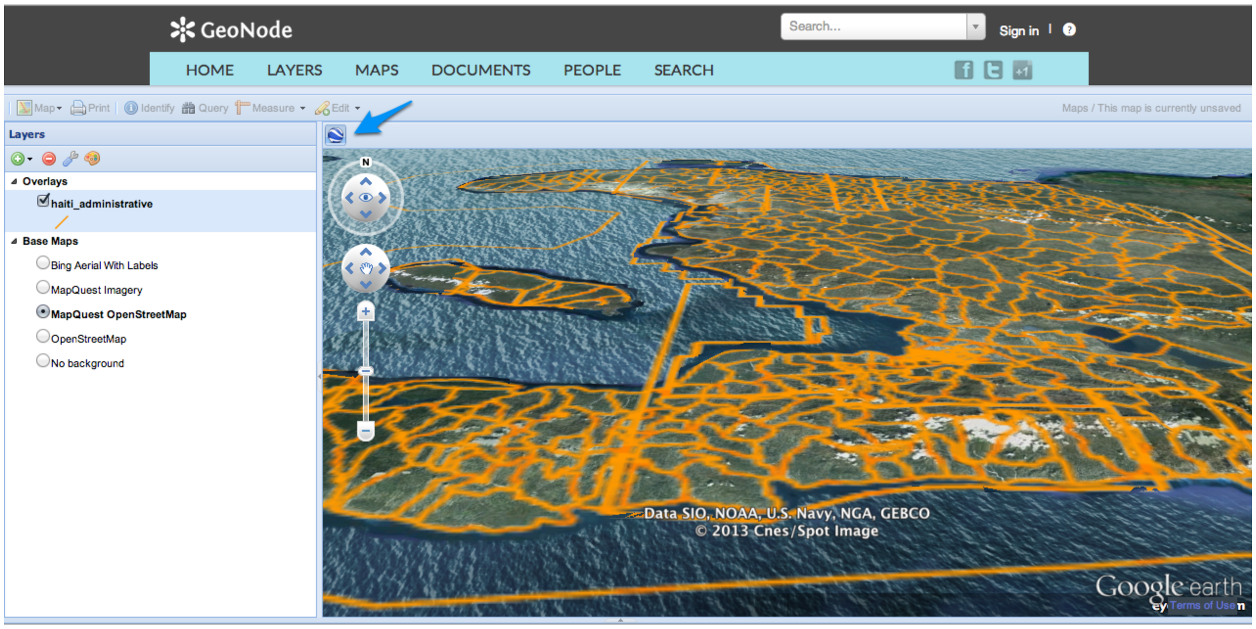


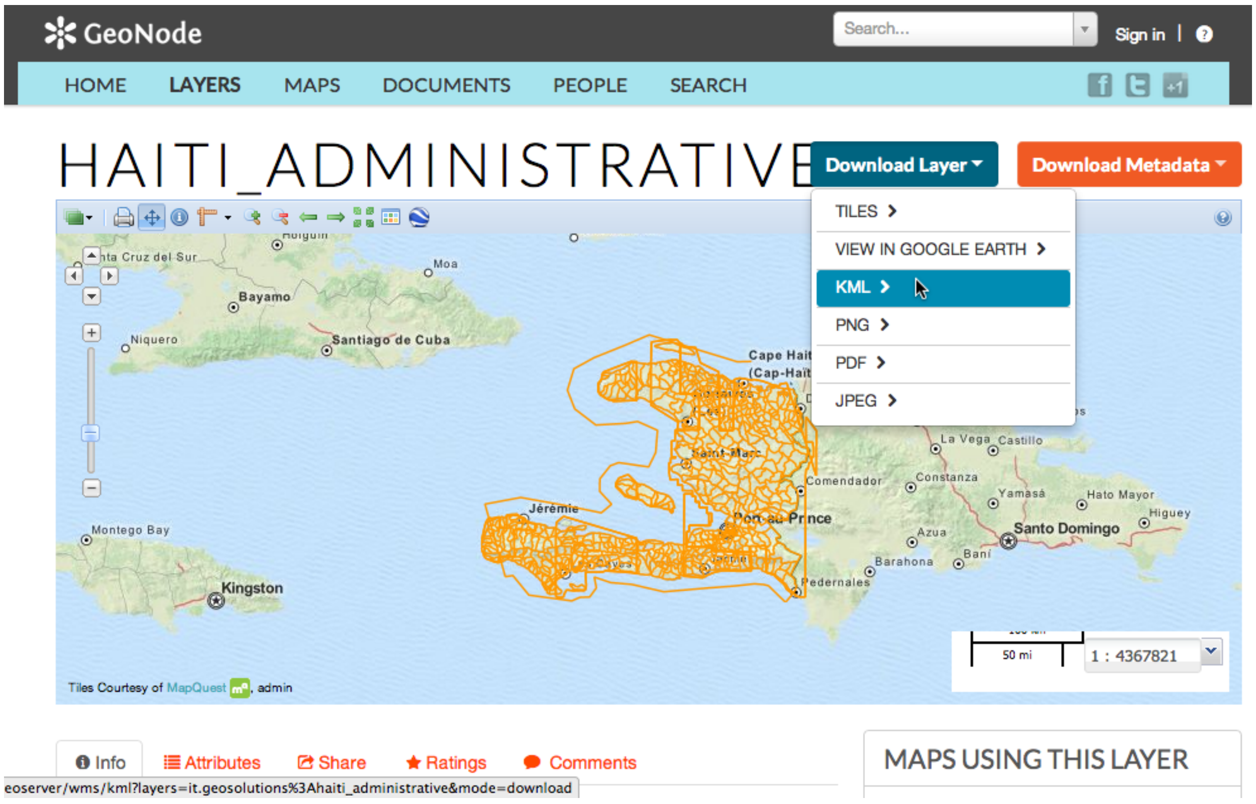


haiti_administrative.1160

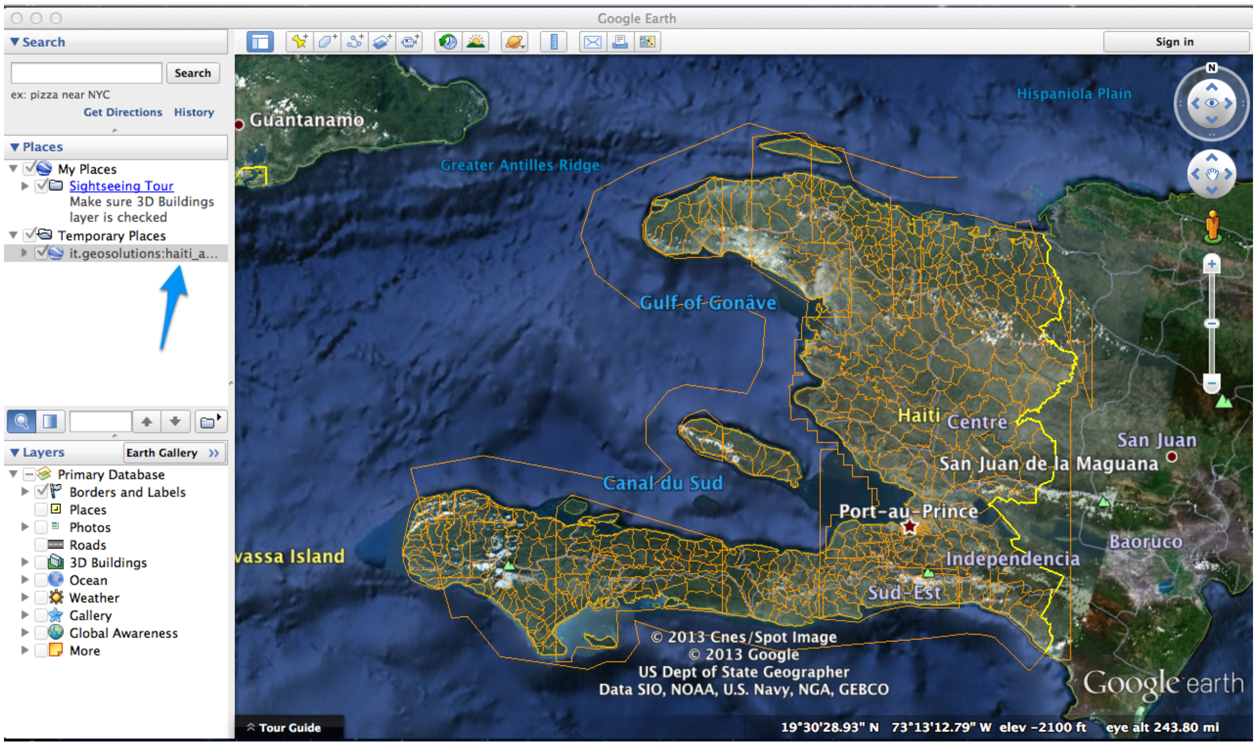
haiti_administrative
ADMIN_LEVE: 4

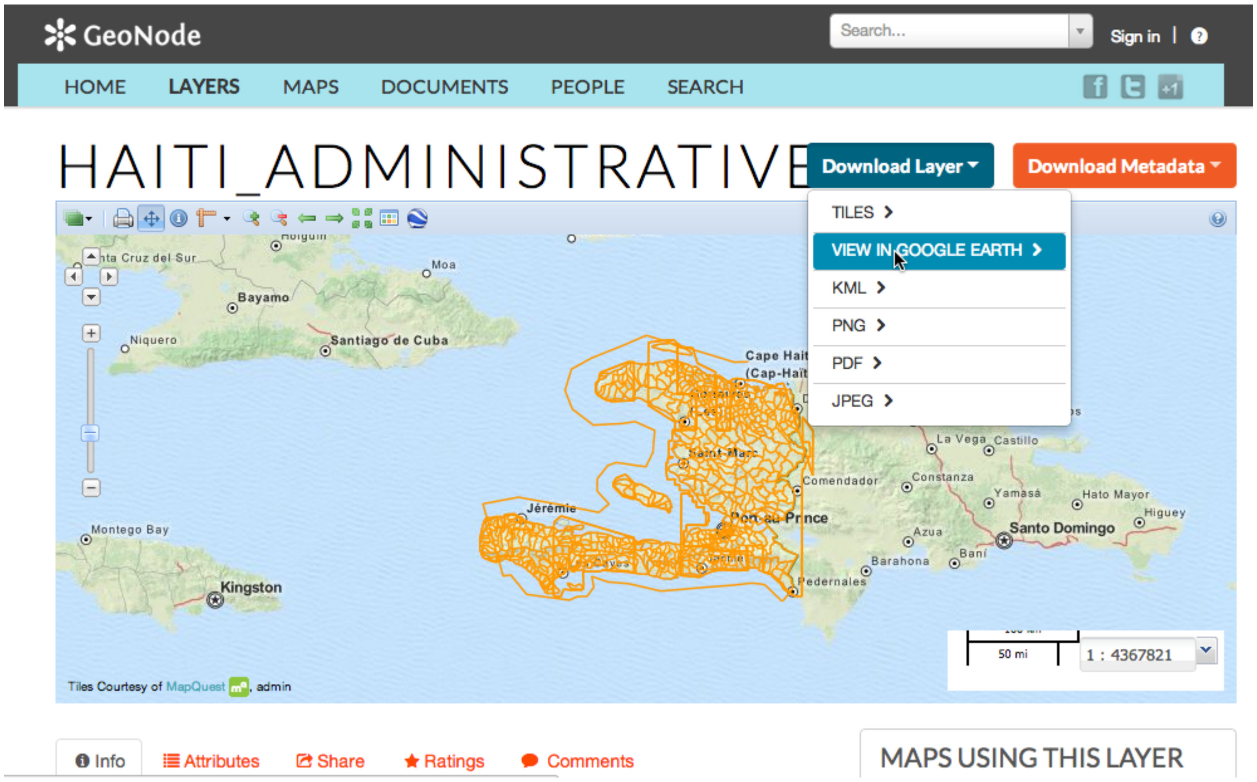
This layer is not currently used in any maps.

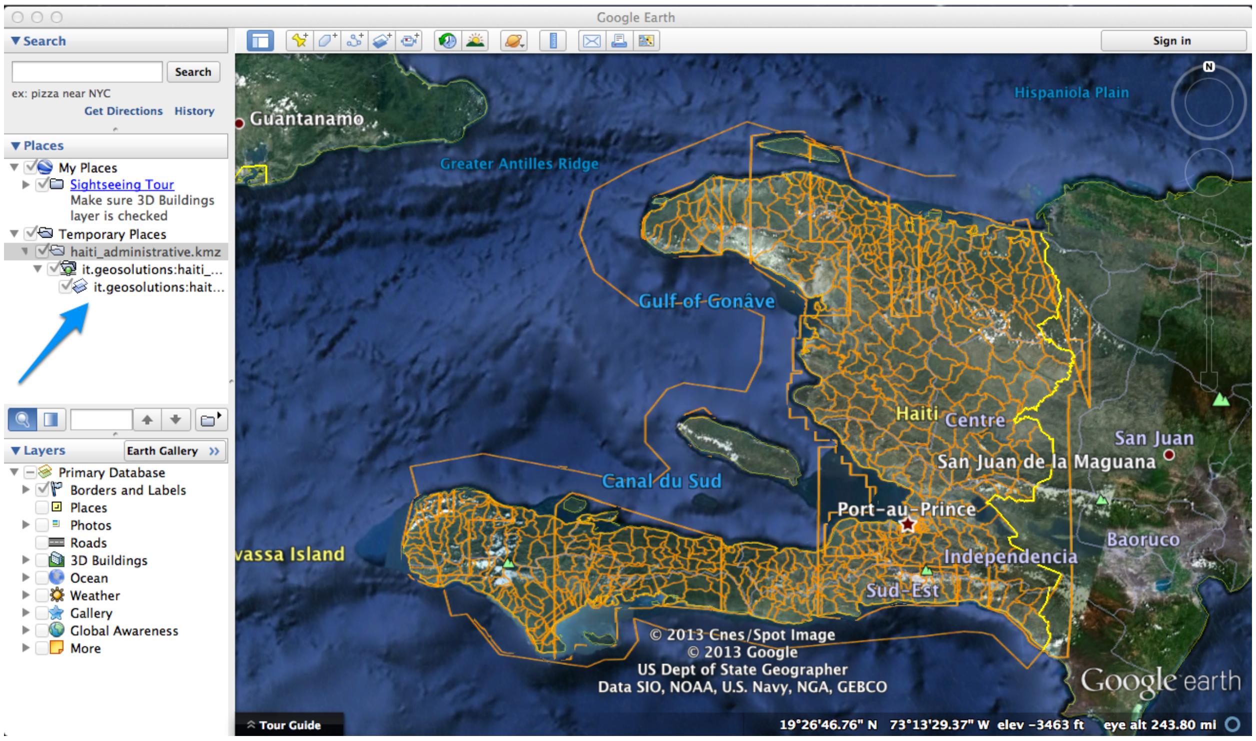


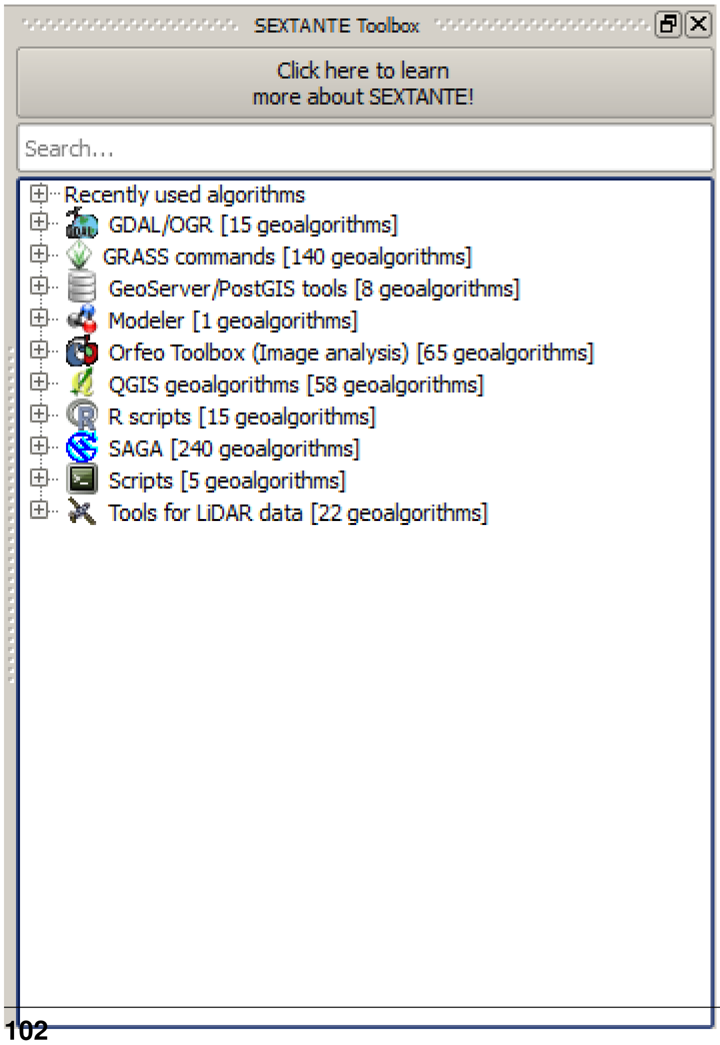


eoserver/wms/kml?layers=it.geosolutions%3Ahaiti_administrative&mode=download


Google Earth
Search
Search
ex: pizza near NYC
Get Directions History
Places
My Places
Sightseeing Tour
Make sure 3D Buildings layer is checked
Temporary Places
it.geosolutions:haiti_a...
Layers
Earth Gallery
Primary Database
Borders and Labels
Places
Photos
Roads
3D Buildings
Ocean
Weather
Gallery
Global Awareness
More
Sign in
Guantanamo
Hispaniola Plain
Greater Antilles Ridge
Gulf-of-Gonâve
Haiti
Centre
San Juan
San Juan de la Maguana
Canal du Sud
Port-au-Prince
vassa Island
Baoruco
Independencia
Sud-Est
© 2013 Cnes/Spot Image
© 2013 Google
US Dept of State Geographer
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Google earth
Tour Guide
19°30'28.93" N 73°13'12.79" W elev -2100 ft eye alt 243.80 mi


GeoNode
Search...
Sign in
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
HAITI_ADMINISTRATIVE
Download Layer
Download Metadata
TILES
VIEW IN GOOGLE EARTH
KML
PNG
PDF
JPEG
Holguin
Santa Cruz del Sur
Moa
Bayamo
Niquero
Santiago de Cuba
Cape Haitien (Cap-Haitien)
La Vega
Castillo
Constanza
Comendador
Yamasá
Hato Mayor
Higuey
Santo Domingo
Azua
Bani
Barahona
Pedernales
Jérémie
Port-au-Prince
Montego Bay
Kingston
50 mi
1 : 4367821
Tiles Courtesy of MapQuest, admin
Info
Attributes
Share
Ratings
Comments
MAPS USING THIS LAYER




SEXTANTE Toolbox

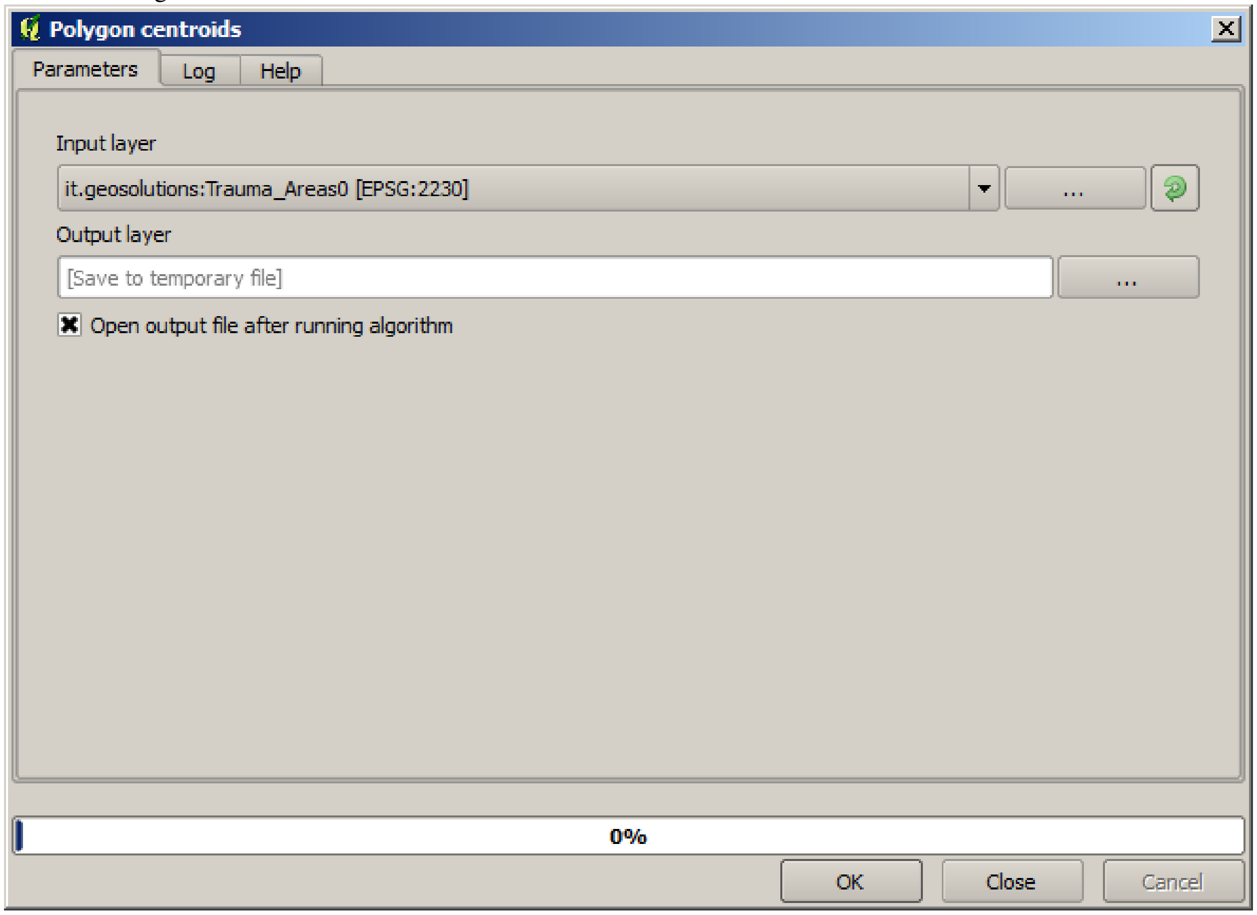
Click here to learn more about SEXTANTE!

Search...

- Recently used algorithms
- GDAL/OGR [15 geoalgorithms]
- GRASS commands [140 geoalgorithms]
- GeoServer/PostGIS tools [8 geoalgorithms]
- Modeler [1 geoalgorithms]
- Orfeo Toolbox (Image analysis) [65 geoalgorithms]
- QGIS geoalgorithms [58 geoalgorithms]
- R scripts [15 geoalgorithms]
- SAGA [240 geoalgorithms]
- Scripts [5 geoalgorithms]
- Tools for LiDAR data [22 geoalgorithms]

The *QGIS algorithms* group contains native python algorithms. Most of the remaining ones represent algorithms that depend on a third-party application. From the point of view of the user, however, there is no difference in the way these algorithms are executed, since they share a common UI and SEXTANTE takes care of the communication between the application and QGIS.

To run an algorithm, just click on its name in the toolbox, and a dialog will appear to define the inputs and outputs of the selected algorithm.



Input layer
it.geosolutions:Trauma_Areas0 [EPSG:2230]
Output layer
[Save to temporary file]
[✓] Open output file after running algorithm

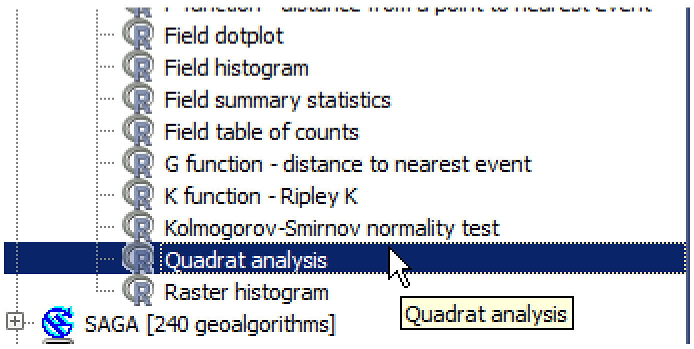
All dialogs used to entering the parameters needed for an algorithm are created on-the-fly, and they all share the same look and feel, no matter which application the algorithm relies on.

If the selected algorithm requires vector layers, all loaded vector layers will be available, including those from WFS connections. If it requires raster layers, all loaded raster layers will be available, including those from WCS connections.

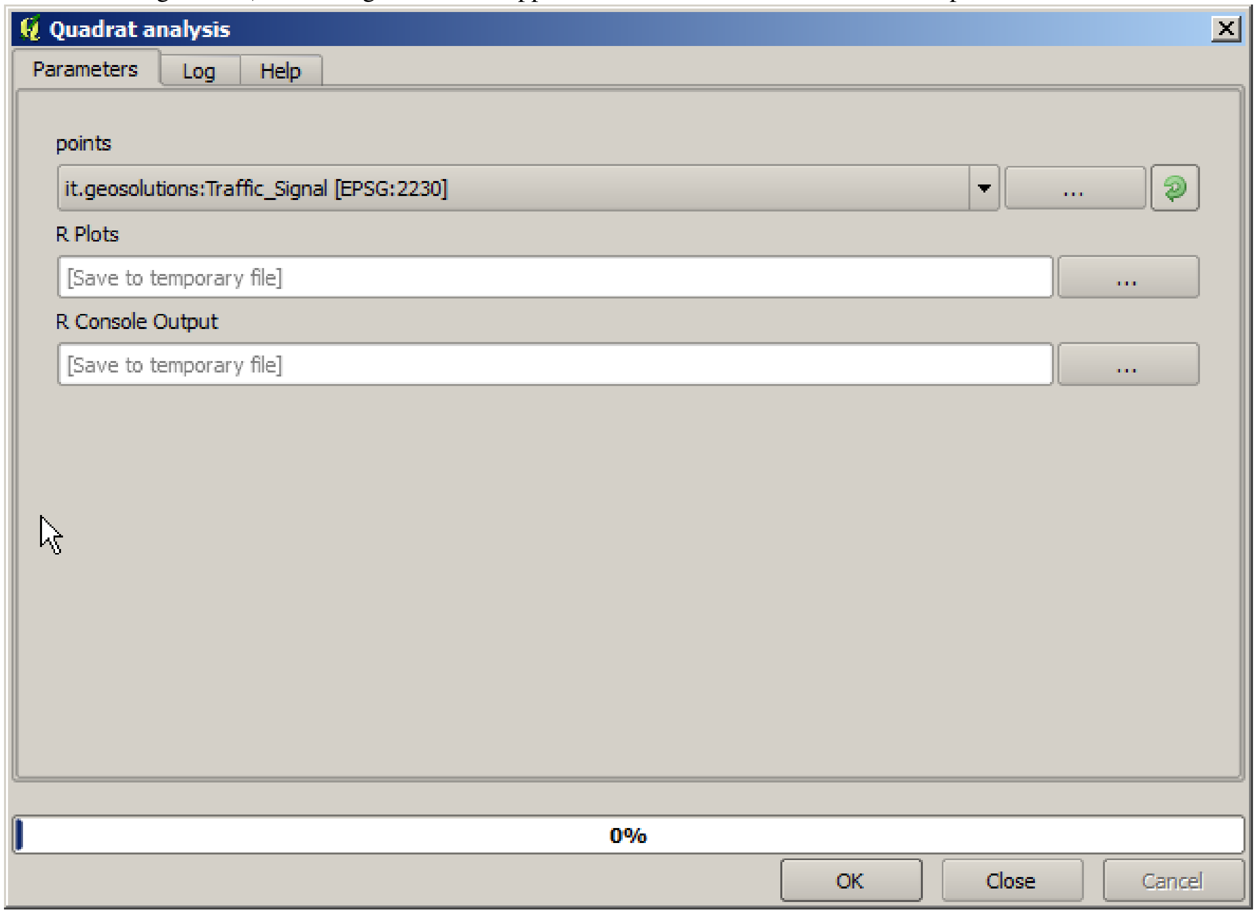
Click on *Ok* and the algorithm will be run and its outputs loaded in the QGIS canvas.

Algorithm depending on third-party application can be called in the same way. WFS and WMS layers will be available as well, even if the application does not support web services. SEXTANTE will take care of creating an intermediate file or selecting a compatible way of feeding the data into the application, performing all necessary import operations under the hood.

For instance, you can run algorithm that uses the R statistical computing software to perform a statistical analysis on the data loaded from a WFS connection.



The interface to define the input parameters is very similar to the one we saw in the previous example (and to those of all the other algorithms), and using an external application makes no difference from the point of view of the user.



points: it.geosolutions:Traffic_Signal [EPSG:2230]

R Plots: [Save to temporary file]

R Console Output: [Save to temporary file]

0%

SEXTANTE algorithms are aware of the state of layers used for input. That means that they can take into account whether there is a selection in a vector layer, even in the case of calling an external application. In the SEXTANTE configuration menu, in the *General* group, check the “Use selected features” option to enable this behavior.

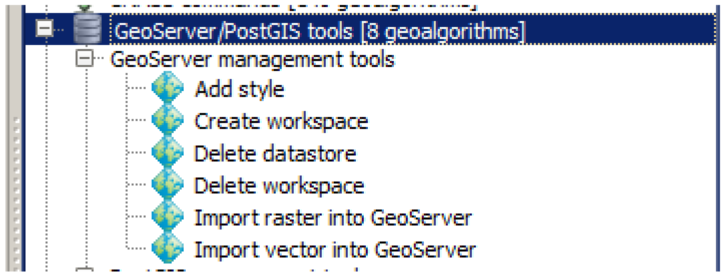
image:: use_selected.png

Now you can make a selection and call the R algorithm used before. It will just use those features that you selected.

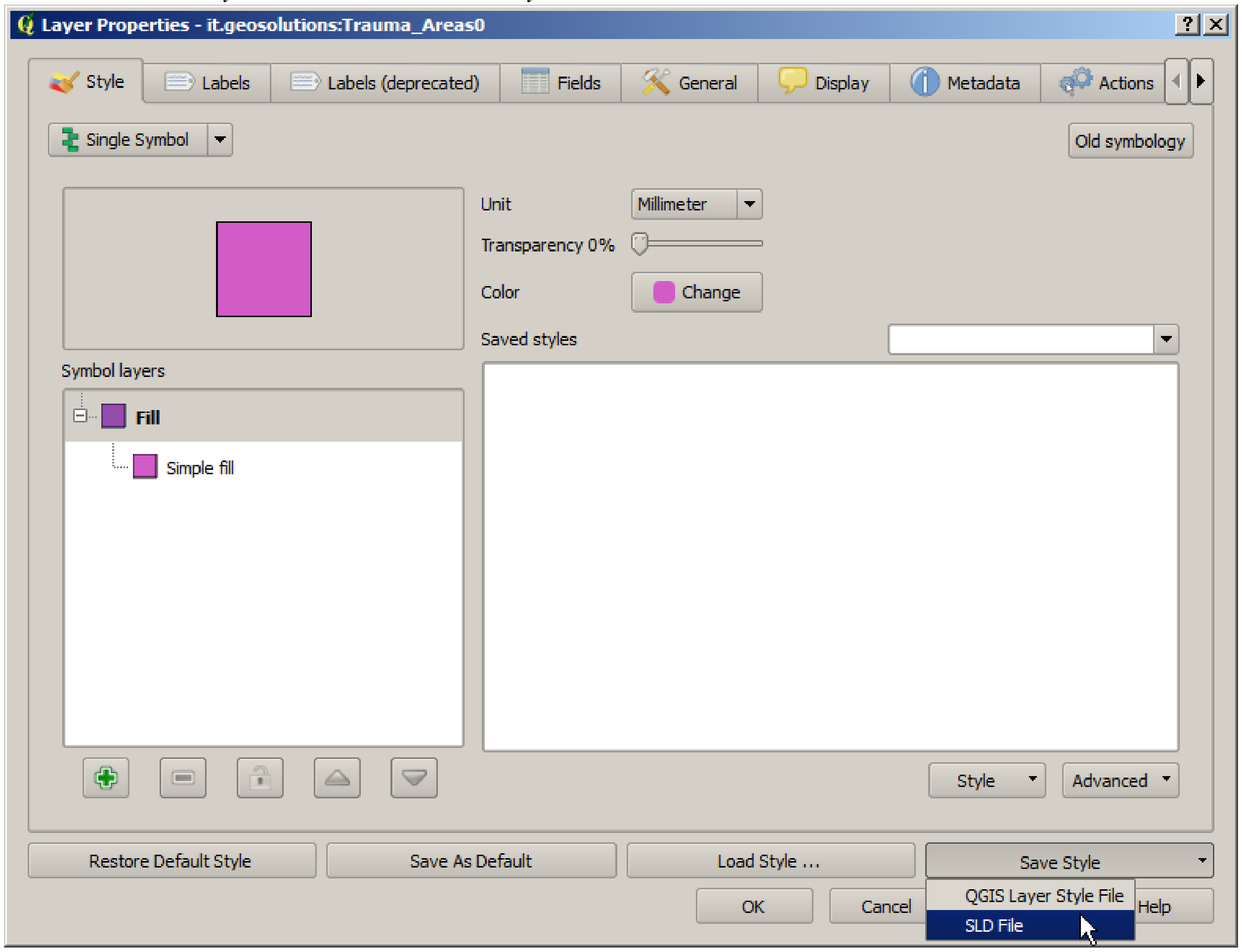
Notice that SEXTANTE provides the context for seamlessly integrating all the pieces and allowing to easily work with all of them together. In this case, GeoServer is providing the data, QGIS is providing the interface where we perform the selection, and R is providing the processing. SEXTANTE is just the mediator between these elements.

Using SEXTANTE we can interact with a GeoServer instance not just to consume its data, but to import into it, having a bidirectional connection.

In the *GeoServer/PostGIS tools* group, you will find some tools that you can use to create a new workspace, import a layer or import a style, among others.

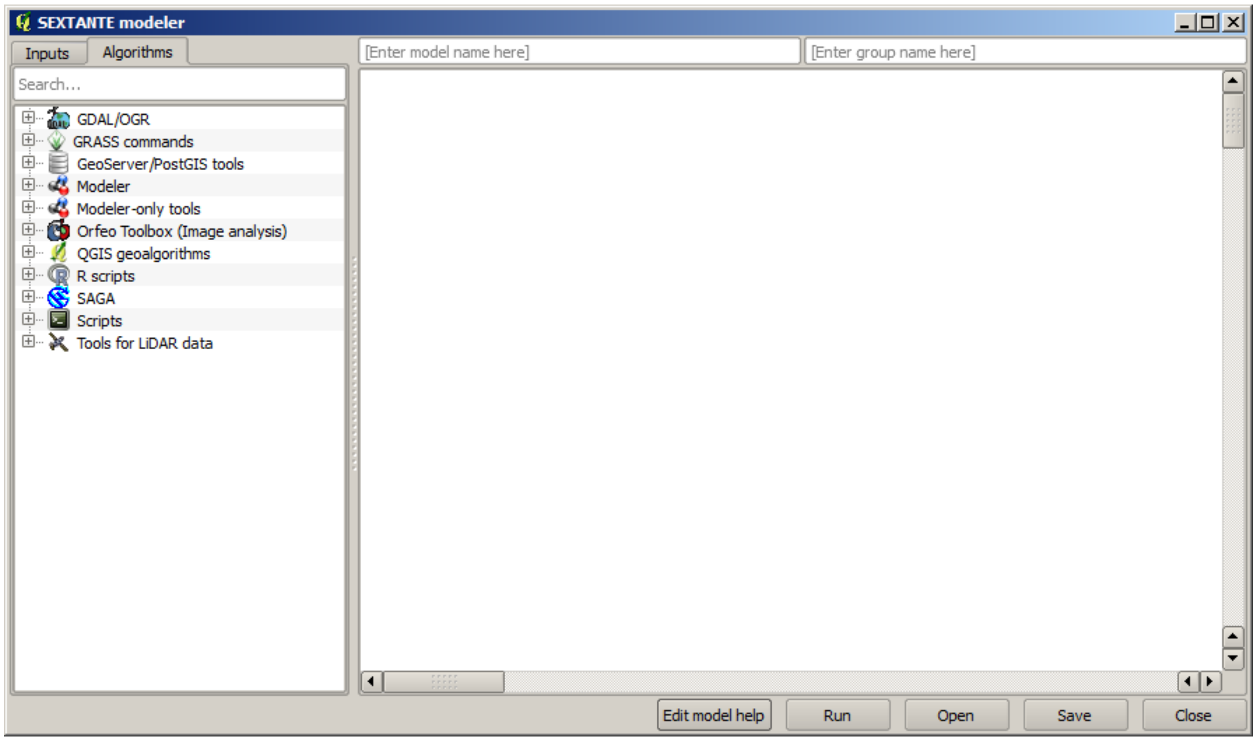

GeoServer/PostGIS tools [8 geoalgorithms]
GeoServer management tools
Add style
Create workspace
Delete datastore
Delete workspace
Import raster into GeoServer
Import vector into GeoServer

Styles are imported using an SLD file. Since QGIS supports exporting the style of a layer to SLD, you can create you styling using QGIS, then export it, and then use the corresponding SEXTANTE algorithm to add it to your GeoServer. To create and export a style, just right-click on a layer name in the TOC, select *Properties*, and then move to the *Style* tab. Clink on *Save Style...” and the select *SLD style*.



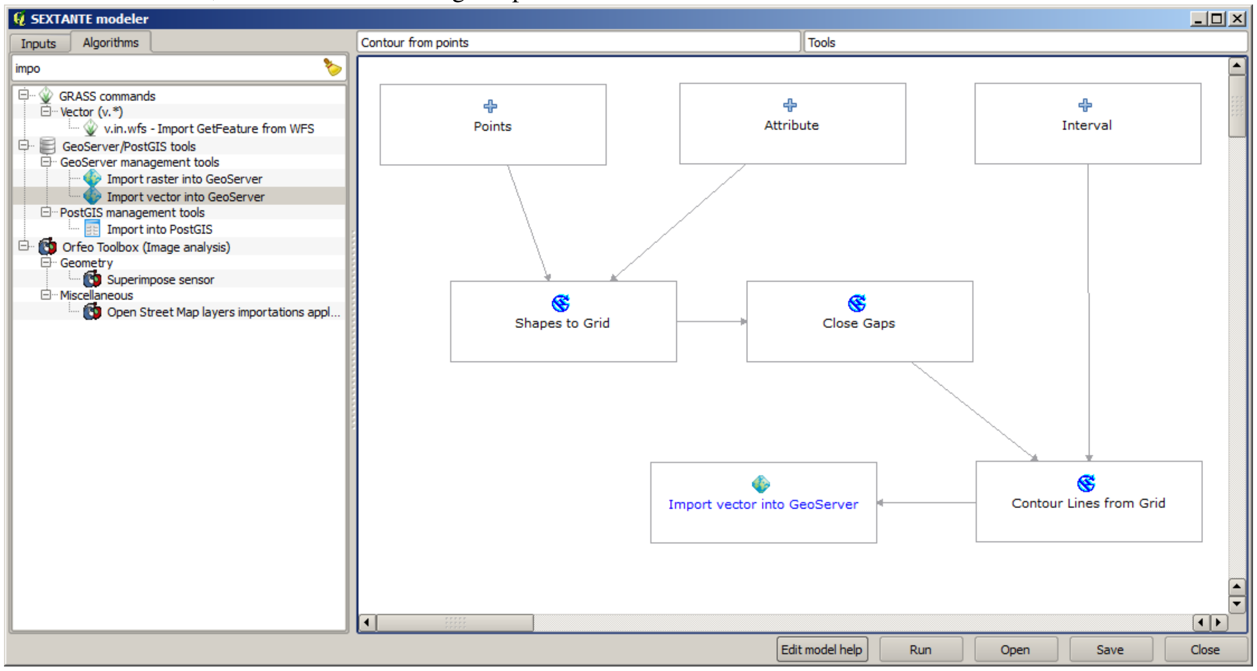
Calling algorithms from the toolbox is the most common way of processing data with SEXTANTE, but in some cases that might not be the best alternative, specially if we have a workflow that involves a large number of different processes. SEXTANTE includes a graphical modeler that allow to create complex workflows that can be later executed as a single process, thus simplifying the process.

The modeler interface is started from the SEXTANTE menu, using the “SEXTANTE modeler” menu.

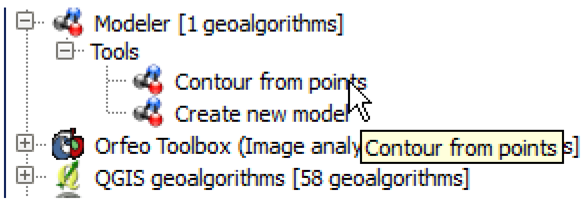


Creating a model involves adding a set of inputs and then the algorithms to use on then, defining the links between them.

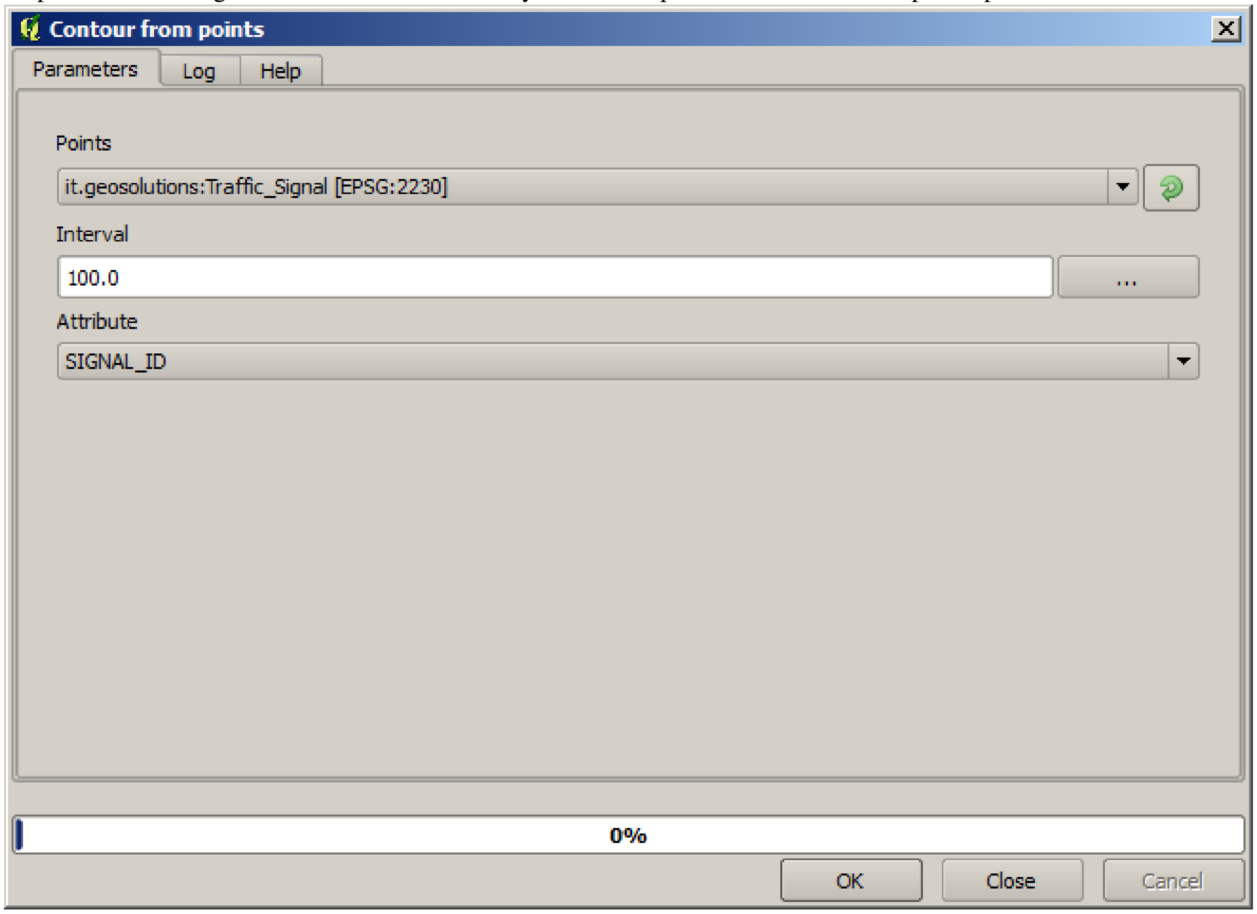
As an example, the model in the figure below takes a points layer, interpolates a raster layer from it and then extracts contour lines for that resulting raster layer, given an interval. The final layer is imported into a GeoServer instance. Used with a WFS service, this can be used to easily create an additional lines layer for a given points layer in a GeoServer instance, such as one containing temperature data for a set of sensors.



The model can be added to the toolbox and run as any of the built-in algorithms.


Modeler [1 geoalgorithms]
Tools
Contour from points
Create new model
Orfeo Toolbox (Image analy
Contour from points
s]
QGIS geoalgorithms [58 geoalgorithms]

Its parameters dialog is also created automatically from the inputs it takes and the outputs it produces.



Points
it.geosolutions:Traffic_Signal [EPSG:2230]

Interval
100.0

Attribute
SIGNAL_ID

SEXTANTE exposes its API through the QGIS python console, including not just all its available algorithms, but also several tools for easy handling both vector and raster data and creating new geoalgorithms. Users familiar with python will be able to create much more complex models with ease, and to automate tasks that involve processing.

algorithms are executed ussing the `runalg` method, passing the name of the algorithm and the parameters it requires. Importing a layer into geoserver can be done from the QGIS console using code like the one shown below:

```
import sextante
filename = "/home/gisdata/example.shp"
sextante.runalg("gspg:importvectorintogeoserver","http://localhost:8080/geoserver/rest","admin","geos
```

Algorithms in the `gspg` namespace include utilities for interacting with both GeoServer and PostGIS.

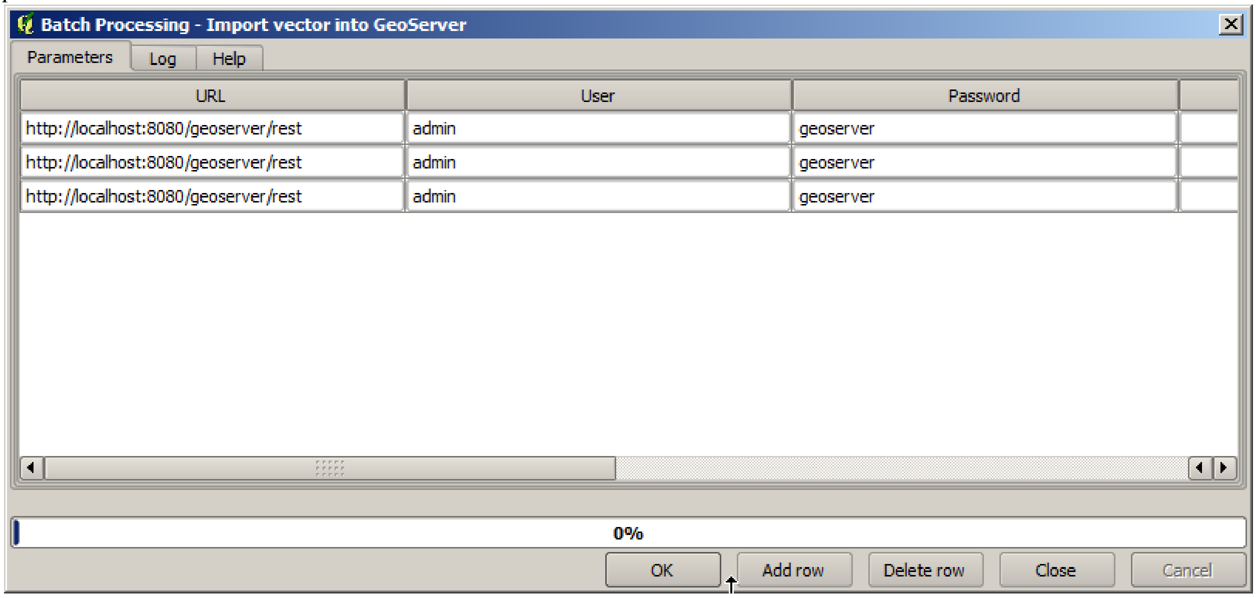
If the layer is loaded into QGIS, there is no need to enter the filename. The layer object can be obtained with the `getobject()` method and then passed to the algorithm call instead of the filename. For a layer named *mylayer*, that would be:

```
import sextante
layer = sextante.getobject("mylayer")
```

```
sextante.runalg("gspg:importvectorintogeoserver","http://localhost:8080/geoserver/rest","admin","geos
```

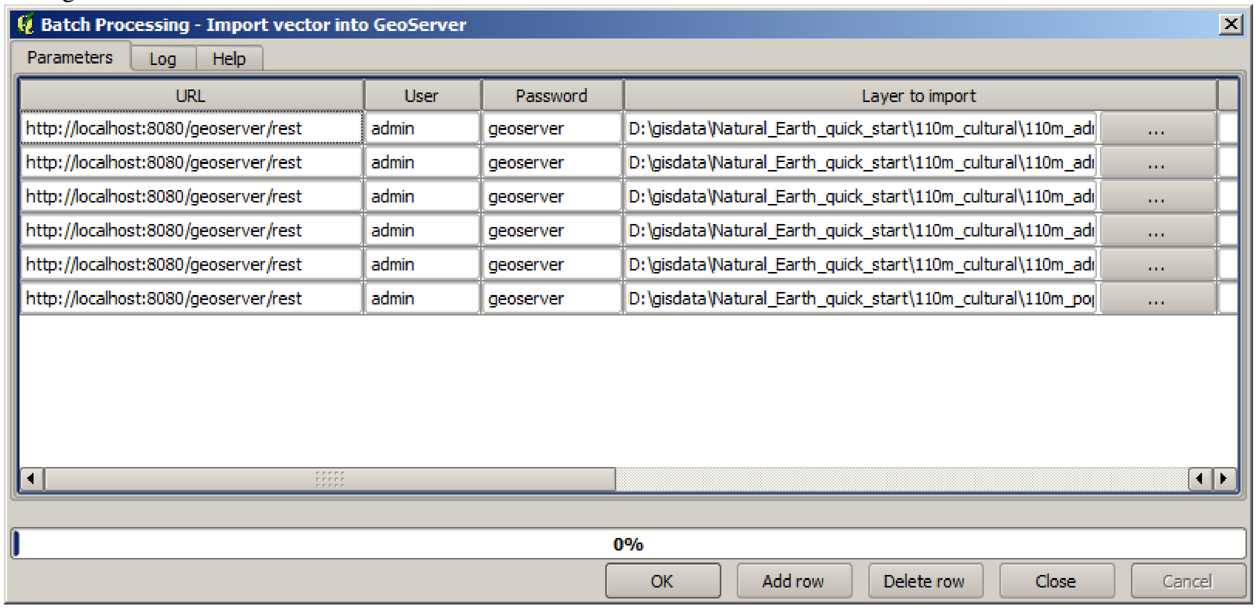
Finally, a last productivity tool is available in SEXTANTE: the batch processing interface. Repeated calls to a single algorithm are simplified by using the batch processing interface. This can be used, for instance, to perform a bulk import of layers into GeoServer, by setting the batch processing interface to call the *Import into GeoServer* algorithm as many times as layers are to be imported.

To open the batch processing interface, right-click on the name of an algorithm in the toolbox and select *Run as batch process*.



URL	User	Password
http://localhost:8080/geoserver/rest	admin	geoserver
http://localhost:8080/geoserver/rest	admin	geoserver
http://localhost:8080/geoserver/rest	admin	geoserver

Each row in the table (3 by default) represents a single execution, and more rows can be added manually, or will be automatically added when selecting a set of input layers, each of them to be used as input in a different execution of the algorithm.



URL	User	Password	Layer to import	
http://localhost:8080/geoserver/rest	admin	geoserver	D:\gisdata\Natural_Earth_quick_start\110m_cultural\110m_adı	...
http://localhost:8080/geoserver/rest	admin	geoserver	D:\gisdata\Natural_Earth_quick_start\110m_cultural\110m_adı	...
http://localhost:8080/geoserver/rest	admin	geoserver	D:\gisdata\Natural_Earth_quick_start\110m_cultural\110m_adı	...
http://localhost:8080/geoserver/rest	admin	geoserver	D:\gisdata\Natural_Earth_quick_start\110m_cultural\110m_adı	...
http://localhost:8080/geoserver/rest	admin	geoserver	D:\gisdata\Natural_Earth_quick_start\110m_cultural\110m_adı	...
http://localhost:8080/geoserver/rest	admin	geoserver	D:\gisdata\Natural_Earth_quick_start\110m_cultural\110m_poı	...

Models can also be run as a batch process. The model defined above, which computed a set of contour lines from a points layer and imported the result into GeoServer, can be called repeatedly using the same input layer and different intervals, to get contour layers of different level of detail, suitable to be rendered at different scales.

More detailed documentation about SEXTANTE can be found at a dedicated chapter in the current QGIS manual, at http://docs.qgis.org/html/en/docs/user_manual/sextante/index.html. For the most up-to-date version, check the corresponding entry at the QGIS documentation github repository, at https://github.com/qgis/QGIS-Documentation/tree/master/source/docs/user_manual/sextante

Administrators Workshop

Welcome to the GeoNode Administrators Workshop! This workshop will teach how to install and manage a deployment of the GeoNode software application.

Introduction

This tutorial will guide you through several different tasks of the GeoNode administration. From the installation and configuration to management, monitoring and restoring, the aim of this guide is to make the administrator fully aware of how GeoNode works and of the ways in which it can be manipulated.

The general suggestion is to try all of these guides on a clean Ubuntu 12.04 machine or virtual machine.

Preparing your environment

The environment suggested for the GeoNode installation is Ubuntu 12.04 LTS.

It can be either a real physical machine or a virtual machine.

If you intend to use virtual machine we suggest the use of Vagrant

Setup a Vagrant machine on Mac OsX We suggest to use Ubuntu 12.04 as guest operating system on the vagrant machine. Once the machine is started you can access it via `vagrant ssh`.

Install geonode via:

```
$ sudo apt-get update
$ sudo apt-get install python-software-properties
$ sudo add-apt-repository ppa:geonode/release
$ sudo apt-get update
$ sudo apt-get install geonode
```

Due to a character encoding bug between Vagrant and the OsX the database password and the secret_key are invalid so we need to correct them manually and change the database password.

Edit the local_settings file:

```
$ sudo nano /etc/geonode/local_settings.py
```

Edit the DATABASE_PASSWORD with another one and the SECRET_KEY with something of your choice.

Edit the geonode role's password in postgres using the same used in the local_settings:

```
$ sudo su postgres
$ psql -c "alter role geonode with password 'yourpassword';"
$ exit
```

Launch the command to change the SITEURL directive in local_setting.py to http://localhost:4567 and restart apache among other things.

> $ sudo geonode-updateip localhost:4567

Launch the Geonode command to create a new superuser account within Django for GeoNode

> $ geonode createsuperuser

Then exit the machine and shut it down with:

```
$ vagrant halt
```

Edit the Vagrantfile and activate the machine sharing with the host like:

```
config.vm.network :forwarded_port, guest: 80, host: 4567
```

Start the machine up. Now you should be able to reach and use geonode on at the address http://localhost:4567 on your host machine.

Setup a VirtualBox VM We suggest to use Ubuntu 12.04 as guest operating system on the VirtualBox virtual machine. To install this in the VM, please follow the quick_installation instructions.

Once this is completed and your Geonode is working within your VM, turn off the VM, go to the Network tab in VM settings and click the port forwarding button.

In there add a rule which forwards the address and port which Geonode runs on (this is by default http://127.0.0.1:8000) and forward to a port that your native system isn't using. Such port could be http://127.0.0.1:4567.

After this you should be able to access your Geonode running from a VirtualBox VM in your native environment's browser.

Setup a VMware Machine Currently in VMware Player, there is no facility to setup port forwarding, so Bridged Connection is neccessary.

We suggest to use Ubuntu 12.04 as guest operating system on the VMWare Machine. To install this in the VM, please follow the quick_installation instructions.

Once this is completed and your Geonode is working within your VM, go to the settings of the VM and enable *Bridged: Connected directly to the physical network*.

After this you should be able to access your Geonode running from a VMware VM in your native environment's browser.

Installing GeoNode

This section will guide you through the steps necessary to install GeoNode. Basically there are three different ways for you to install GeoNode. Depending on your purpose, you can choose between those. If you just want to test Geonode or just want a completely configured GeoNode, then try the quick_installation. If you intend to use Geonode with some already installed components, then use the custom_install guide. This installation will give you full control over the components and you will be able to configure the components the way you need them. If you instead want to do some development, use the *Install GeoNode for Development* guide.

quick_installation How to install GeoNode on Linux, OSX and Windows.

Install GeoNode for Development How to install GeoNode for development (for Developers) and prepare your development environment.

custom_install How to install GeoNode manually.

Other install links:

Installation of Postgresql-9.2 on Ubuntu 12.04

Installation The Installation of Postgresql-9.2 on Ubuntu 12.04 includes two main steps (Generally, this installation should work for each version of Postgresql!).

1. **Add repository to list**

   Create an empty file called *pgdg.list* placed in /etc/apt/sources.list.d/, using these commands

```
$ cd /etc/apt/sources.list.d/
$ sudo touch pgdg.list
```

   Now the following line has to be included the *pgdg.list* file.

```
deb http://apt.postgresql.org/pub/repos/apt/ codename-pgdg main
```

   Instead of *codename* write the name of your system. If you do not know this, type

```
$ lsb_release -c
```

   This will return you the name of your system.

   Warning: Make sure that you have changed *codename* to the name of your system before you copy the link in the next step!

   Now open the *pgdg.list* file using e.g.:

```
$ gksu gedit pdgd.list
```

   and copy deb *http://apt.postgresql.org/pub/repos/apt/ codename-pgdg main* into it.

2. **Import Repository key**

   Use the following command to do so:

```
$ wget --quiet -O - http://apt.postgresql.org/pub/repos/apt/ACCC4CF8.asc | sudo apt-key add -
```

   After that you have to update the package lists using

```
$ sudo apt-get update
```

   and then you can install postgresql-9.2 and pgadmin3 (if needed)

```
$ sudo apt-get install postgresql-9.2 pgadmin3
```

Configuration **General**

1. *Change password*

   First of all you have to set a new password for the superuser *postgres*. In order to do so, type the following command line into your terminal

```
$ sudo -u postgres psql postgres
```

   Now you are in *psql*, the command interface for postgresql, and in the database *postgres*. In your terminal it looks like this at the moment

```
$ postgres=#
```

   To change your password:

```
$ \password postgres
```

and type your new password when asked for it.

2. *Create a database* (for testing)

   If you want to create a db, posgresql has to know, which user you are. Therefore you have to type *-u username* in front of the command *createdb*. If you type the following, it means that you as the user *postgres* want to create a database wich is called *mydb*.

```
$ sudo -u postgres createdb mydb
```

For geonode

If you do not need that (from the beginning) you can follow this instruction. You can even add the *posgis extension* later following the steps from the section *Alternative*. Anyway it is recommended to do the common version.

1. *Create a new user (geonode)*

   In order to use postgresql in geonode, you have to create a new user *geonode*, which is the owner of a new database, also called *geonode*. So first create a new user by typing

```
$ sudo -u postgres createuser -P geonode
```

   This means, that you as the user *postgres* create a new user called *geonode*. *-P* indicates, that this user will have a password (*geonode*).

2. *Create new database*

   After creating the new user you can create the db *geonode*:

```
$ sudo -u postgres createdb -O geonode geonode
```

PostGIS 2.0.3 Installation on Ubuntu 12.04 The best is to visit http://postgis.net to get all the up to date installation details.

Warning: this document will eventually be out dated please refer to http://www.postgis.net for installation guide

1. **Install dependencies**

   Before you can install PostGis 2.0.3, some dependencies have to be installed first. You can do this by using the Linux command *apt-get*. It is recommended to install postgresql-9.2 before you install PostGis 2.0.3. For the installation guide of postgresql-9.2 see *Installation of Postgresql-9.2 on Ubuntu 12.04*.

   The first step is to type the following command into your terminal to install all dependencies.

```
$ sudo apt-get install build-essential postgresql-server-dev-9.2 libxml2-dev libproj-dev libjson
docbook-mathml libgdal1-dev
```

   Hint: libgdal1-dev is needed for raster support and is required if you want to build the postgresql extension!

2. **Build Geos 3.3.2 or higher**

   GEOS is used for the topology support and because Postgis 2.0 requires a GEOS version 3.3.2 or higher, you have to build this before you can install postgis itself. (genereally Ubuntu comes with an GEOS version lower than 3.3.2!) Download your favourite version of geos (has to be 3.3.2 or higher!) using the following command line:

```
$ wget http://download.osgeo.org/geos/geos-3.3.8.tar.bz2
```

Unpack it and go to the unpacked folder:

```
$ tar xvfj geos-3.3.8.tar.bz2
$ cd geos-3.3.8
```

Now you can install geos by using the following command lines (this process may take a while)

```
$ ./configure
$ make
$ sudo make install
$ cd ..
```

3. **Install postgis**

   The following steps are almost the same like instructed in *Step 2*. Download postgis 2.0.3, unpack it and go to unpacked folder.

```
$ wget http://download.osgeo.org/postgis/source/postgis-2.0.3.tar.gz
$ tar xfvz postgis-2.0.3.tar.gz
$ cd postgis-2.0.3
```

   Now postgis can be installed:

```
$ ./configure
$ make
$ sudo make install
$ sudo ldconfig
$ sudo make comments-install
```

   Hint: PostGIS 2.0.3 can be configured to disable topology or raster components using the configure flags *–without-raster* and/or *–without-topology*.

   The default is to build both. Note that the raster component is required for the extension installation method for postgresql!

Create the postgis extension for postgresql   Common way

The best way to install the postgis extension is by using templates. After downloading and installing postgresql and postgis you create a database called template_postigsxxx (xxx should be replaced by your version of postgis; in this case postgis 2.0.3 was used).

```
$ sudo -u postgres createdb template_postgis203
```

Before installing the extension you have to log in to the database

```
$ psql template_postgis203
```

and now you can create the extension

```
$ create extension postgis;
```

Warning: Do not forget the semicolon at the end! Otherwise this statement will have no effect!

Now you can easily create a new database wich automatically has the postgis extension as well, e.g create a new database called geonode like this

```
$ sudo -u postgres createdb -T template_postgis203 geonode
```

Alternative

If you already created a db called *geonode*, then you can use this alternative to create the extension for postigs.

By typing

```
$ sudo su postgres
```

you log in as the user *postgres*.

By using the following command line

```
$ psql geonode
```

you log in to the database *geonode*.

To install the posgis extension type

```
$ create extension postgis;
```

Now you should have successfully installed the postgis extension in your geonode database.

Configuring GeoNode for Production

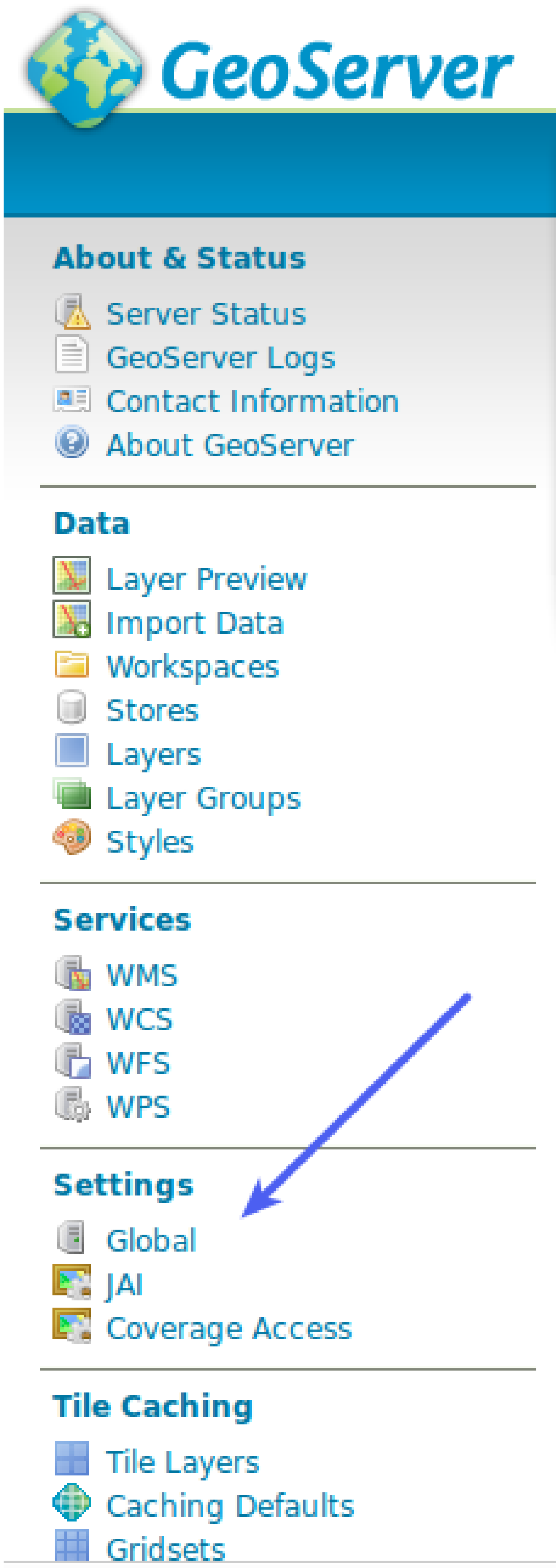
This page documents recommendations for configuring GeoNode in production environments. The steps mentioned in the first section are required to run GeoNode, the ones in the second section are either optional or ways to get more performance.

Note: This document makes numerous references to the **<host>** variable, please replace it with the IP Address of your GeoNode or the DNS entry.

For example: instead of `http://<host>/geoserver`, write down: `http://mygeonode.com/geoserver` or `http://127.0.0.1/geoserver`

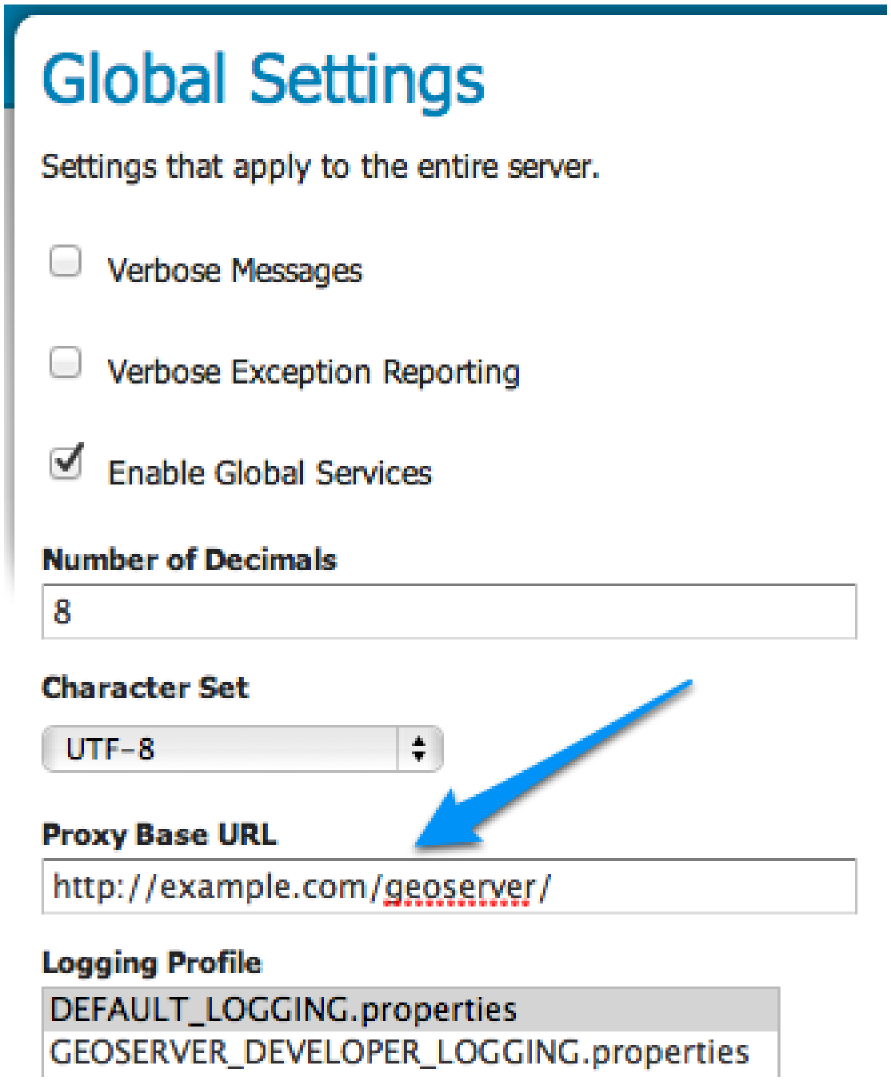
Set the correct GeoServer Proxy URL value Navigate to `http://localhost/geoserver`, log in and click on the `Global` link in the Settings section.

Note: The Geoserver default username is **admin** with **geoserver** as the password. Change this ASAP.


GeoServer
About & Status
Server Status
GeoServer Logs
Contact Information
About GeoServer
Data
Layer Preview
Import Data
Workspaces
Stores
Layers
Layer Groups
Styles
Services
WMS
WCS
WFS
WPS
Settings
Global
JAI
Coverage Access
Tile Caching
Tile Layers
Caching Defaults
Gridsets

Find the `Proxy Base URL` text field, put the complete address there:

```
http://<host>/geoserver/
```



Global Settings

Settings that apply to the entire server.

☐ Verbose Messages

☐ Verbose Exception Reporting

✓ Enable Global Services

Number of Decimals: 8

Character Set: UTF-8

Proxy Base URL: http://example.com/geoserver/

Logging Profile: DEFAULT_LOGGING.properties, GEOSERVER_DEVELOPER_LOGGING.properties

Configure the Printing Module This lives in the GeoServer Data dir `/usr/share/geoserver/data/printing/config.yaml`, add your server's IP address or domain name to the list of exceptions. Please refer to http://docs.geoserver.org/2.4.x/en/user/datadirectory/index.html for additional information on managing the geoserver data directory:

```
hosts:
  - !dnsMatch
    host: YOUR_IP_ADDRESS
    port: 80
```

Recommended Steps (optional)

Adding layers from Google, Bing and other providers Get an API key from Microsoft at http://bingmapsportal.com/ and place it in `local_settings.py`.:

```
BING_API_KEY="zxcxzcXAWdsdfkjsdfWWsdfjpowxcxz"
```

Copy the `MAP_BASELAYERS` dictionary from `settings.py` into `local_settings.py` and add the following snippet:

```
},{
"source": {
           "ptype":"gxp_bingsource",
           "apiKey": BING_API_KEY
          },
"group":"background",
"name":"Aerial",
"visibility": False,
"fixed": True,
```

Get an API key from Google at http://code.google.com/apis/maps/signup.html and place it in `local_settings.py`, for example:

```
GOOGLE_API_KEY="zxcxzcXAWdqwdQWWQEDzxcxz"
```

Copy the `MAP_BASELAYERS` dictionary from `settings.py` into `local_settings.py` (or edit the previously copied snippet) and add the following snippet:

```
},{
"source": {
     "ptype":"gxp_googlesource",
     "apiKey": GOOGLE_API_KEY
    },
"group":"background",
"name":"SATELLITE",
"visibility": False,
"fixed": True,
```

Sitemaps Configuration GeoNode can automatically generate a sitemap suitable for submission to search engines which can help them to index your GeoNode site more efficiently and effectively.

In order to generate the sitemap properly, the sites domain name must be set within the sites framework. This requires that an superuser login to the admin interface and navigate to the sites module and change example.com to the actual domain name (and port if applicable). The admin interface can be accessed at http://<host>/admin/sites/site/. Click on the `example.com` link, and change both the `Domain name` and `Display name` entries to match your system.

It is possible to 'inform' google of changes to your sitemap. This is accomplished using the ping_google management command. More information can be found here http://docs.djangoproject.com/en/dev/ref/contrib/sitemaps/#django.contrib.sitemaps.ping_google It is recommended to put this call into a cron (scheduled) job to update google periodically.

Configuring User Registration You can optionally configure GeoNode to allow new users to register through the web. New registrants will be sent an email inviting them to activate their account.

To allow new user registration:

1. Set up the email backend for Django (see Django documentation) and add the appropriate settings to `./src/GeoNodePy/geonode/local_settings.py`. For example:

```
    EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
    EMAIL_HOST = 'smtp.gmail.com'
    EMAIL_HOST_USER = 'foo@gmail.com'
    EMAIL_HOST_PASSWORD = 'bar'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
```

2. In the same settings file set:

```
    REGISTRATION_OPEN=True
```

3. With the Django application running, set the domain name of the service properly through the admin interface as specified above in the Sitemaps section. (This domain name is used in the account activation emails.).

5. Restart apache:

```
    $ sudo service apache2 restart
```

6. (Optional) Disable automatic approval of new users. Administrators would receive an email and need to manually approve new accounts. For this option to work, an email backed has to be defined in order to email the users with Staff status the notification to approve the new account:

```
    ACCOUNT_APPROVAL_REQUIRED = True
```

To register as a new user, click the ''Register" link in the GeoNode index header.

Additional Configuration Some other things that require tweaking:

- Web-accessible uploads directory for user profile photos

Robot Exclusion File GeoNode has several views that require considerable resources to properly respond - for example, the download links on layer detail pages require GeoServer to dynamically generate output in PDF, png, etc. format.

Crawlers for web search engines such as Google may cause problems by quickly following many such links in succession.

In order to request that "robots" do not make requests directly to GeoServer, you can ensure that requests to `/robots.txt` return a text file with the following content:

```
User-agent: *
Disallow: /geoserver/
```

This will only affect automated web agents; web browsers in particular are unaffected.

Memory Management At the time of the GeoNode 1.0 release, the GeoNode manual recommended at least 4GB RAM for servers running GeoNode.

While 4GB *physical* RAM is sufficient, it is recommended that machines be configured with at least 6GB total *virtual* memory.

For example, a machine with 4GB physical RAM and 2GB swap space should be able to run GeoNode, but if you would like to run without a swapfile then you should configure the machine with at least 6GB RAM.

On Linux and MacOSX hosts, you can check the available RAM with the following command:

```
$ free -m
             total        used       free     shared    buffers     cached
Mem:          6096        3863       2232          0          0          0
-/+ buffers/cache:        3863       2232
Swap:            0           0          0
```

The “total” column lists the available physical memory and swap space in megabytes; adding them together yields the amount of virtual memory available to the system.

In this example, there is no Swap space so that field is 0 and the available RAM on the system is 6096MB (6 GB).

Security Integration Optimization GeoServer delegates authentication and authorization to GeoNode. The default configuration uses an HTTP endpoint in GeoNode to discover the current user and the layers that are accessible. For production, it is advisable to use a database-level connection.

The SQL for the stored procedure is distributed with the GeoServer web application archive and can be found at `WEB-INF/classes/org/geonode/security/geonode_authorize_layer.sql` in the webapps directory. It can be loaded using the *psql* command by following these steps (if not using tomcat6 or Ubuntu, locate the webapps directory for your configuration):

```
$ cd /var/lib/tomcat6/webapps
$ sudo su - postgres
$ psql -d YOUR_DATABASE < geoserver/WEB-INF/classes/org/geonode/security/geonode_authorize_layer.sql
```

If a context configuration XML file does not already exist, create one for GeoServer. If using Tomcat6 on Ubuntu, this file resides at `/etc/tomcat6/Catalina/localhost/geoserver.xml`. If creating a new file, the following XML should be added (replace ALLCAPS with your specific values):

```
<Context path="/geoserver"
    antiResourceLocking="false" >
  <Parameter name="org.geonode.security.databaseSecurityClient.url"
    value="jdbc:postgresql://localhost:5432/DATABASE?user=USER&amp;password=PASSWORD"/>
</Context>
```

If the file exists already, just add the *Parameter* element from above.

To verify the settings change, look in the GeoServer logs for a line that notes: “using geonode database security client”. If any issues arise, check your connection configuration as specified in the context file above.

Configuring the Servlet Container GeoServer is the most resource intensive component of GeoNode.

There are some general notes on setting up GeoServer for production environments in the GeoServer manual .

However, the following are some GeoServer recommendations with GeoNode’s specific needs in mind.

The JRE used with GeoNode should be that distributed by Oracle.

Others such as OpenJDK may work but Oracle’s JRE is recommended for higher performance rendering.

Startup options should include the following:

```
-Xmx1024M -Xms1024M -XX:MaxPermSize=256M \
    -XX:CompileCommand=exclude,net/sf/saxon/event/ReceivingContentHandler.startEvent
```

These can be specified using the `CATALINA_OPTS` variable in Tomcat’s `bin/catalina.sh` file, or the `JETTY_OPTS` in Jetty’s `bin/jetty.sh` file.

While the JVM provides memory management for most operations in Java applications, the memory used for rendering (in GeoServer’s case, responding to WMS GetMap requests) is not managed this way, so it is allocated in addition to the memory permitted by the JVM options above.

If GeoServer receives many concurrent requests, it may increase the memory usage significantly, so it is recommended to constrain the number of concurrent requests at the servlet container (ie, Jetty or Tomcat).

For Tomcat, you can edit `conf/server.xml`. By default, this file contains an entry defining a ContextHandler:

```
<Connector port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    redirectPort="8443"/>
```

Add a `maxThreads` attribute to limit the number of threads (concurrent requests) to 50 (the default in Tomcat is 200):

```
<Connector port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    redirectPort="8443" maxThreads="50"/>
```

Note: This configuration is possible in Jetty as well but not yet documented in this manual.

Using the native-code implementation of JAI and JAI ImageIO speeds up GeoServer, thereby requiring less concurrency at the same level of throughput.

The GeoServer manual contains platform-specific instructions for configuring JAI and JAI ImageIO.

GeoServer Configuration There are a few controls to be set in the GeoServer configuration itself as well.

On the JAI Settings page

On the WMS Service page

Printing with the Mapfish Print Service The GeoNode map composer can "print" maps to PDF documents using the Mapfish print service. The recommended way to run this service is by using the printing extension to GeoServer (if you are using the pre-built GeoNode package, this extension is already installed for you). However, the print service includes restrictions on the servers that can provide map tiles for printed maps. These restrictions have a fairly strict default, so you may want to loosen these constraints.

Adding servers by hostname The Mapfish printing module is configured through a YAML configuration file, usually named `print.yaml`. If you are using the GeoServer plugin to run the printing module, this configuration file can be found at *GEOSERVER_DATA_DIR*`/printing/config.yaml`. The default configuration should contain an entry like so:

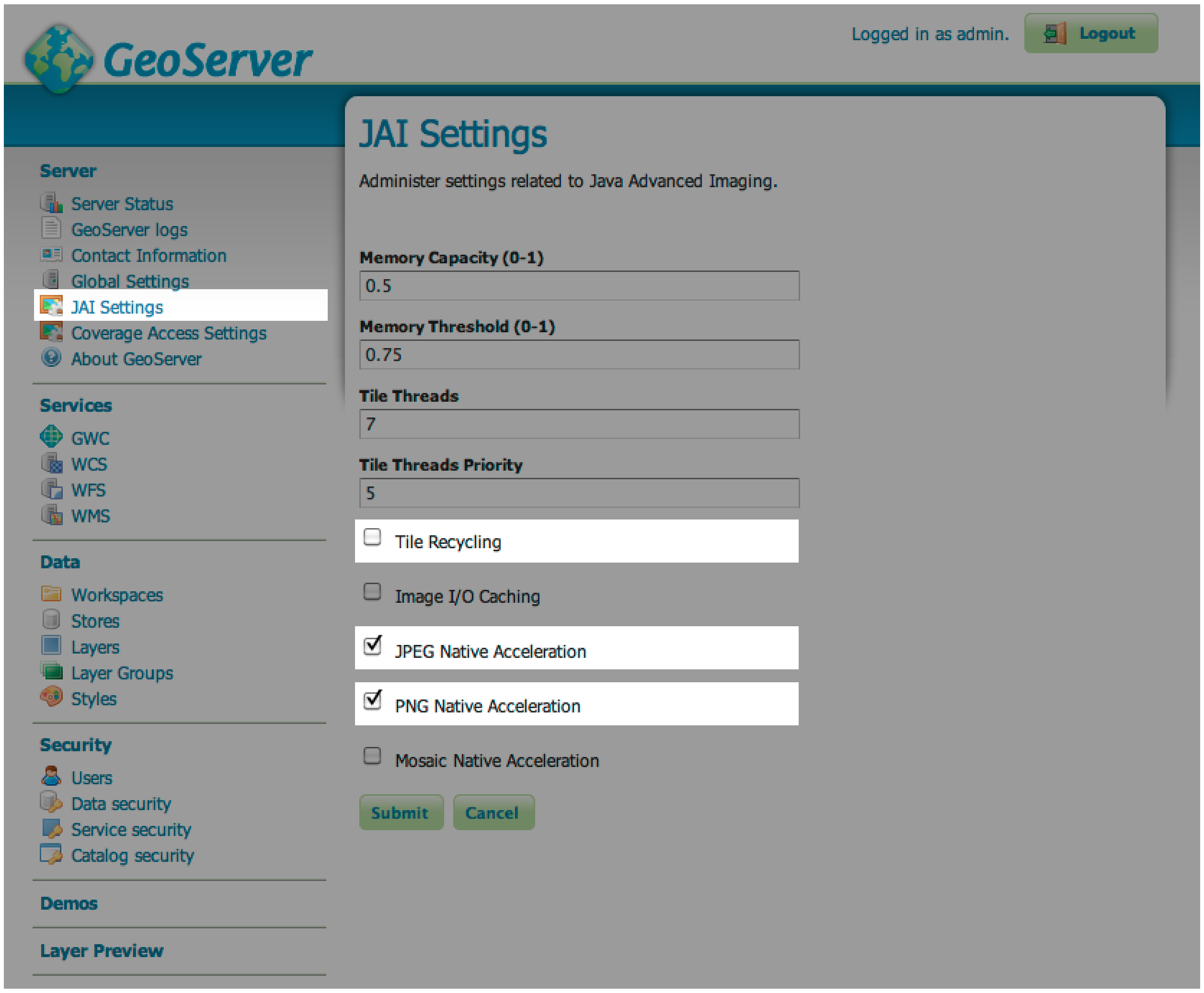
```
hosts:
  - !dnsMatch
    host: labs.metacarta.com
    port: 80
  - !dnsMatch
    host: terraservice.net
    port: 80
```

You can add host/port entries to this list to allow other servers.

See also:

The Mapfish documentation on configuring the print module.

The GeoServer documentation on configuring the print module.



JAI Settings

Administer settings related to Java Advanced Imaging.

Memory Capacity (0-1): 0.5

Memory Threshold (0-1): 0.75

Tile Threads: 7

Tile Threads Priority: 5

☐ Tile Recycling

☐ Image I/O Caching

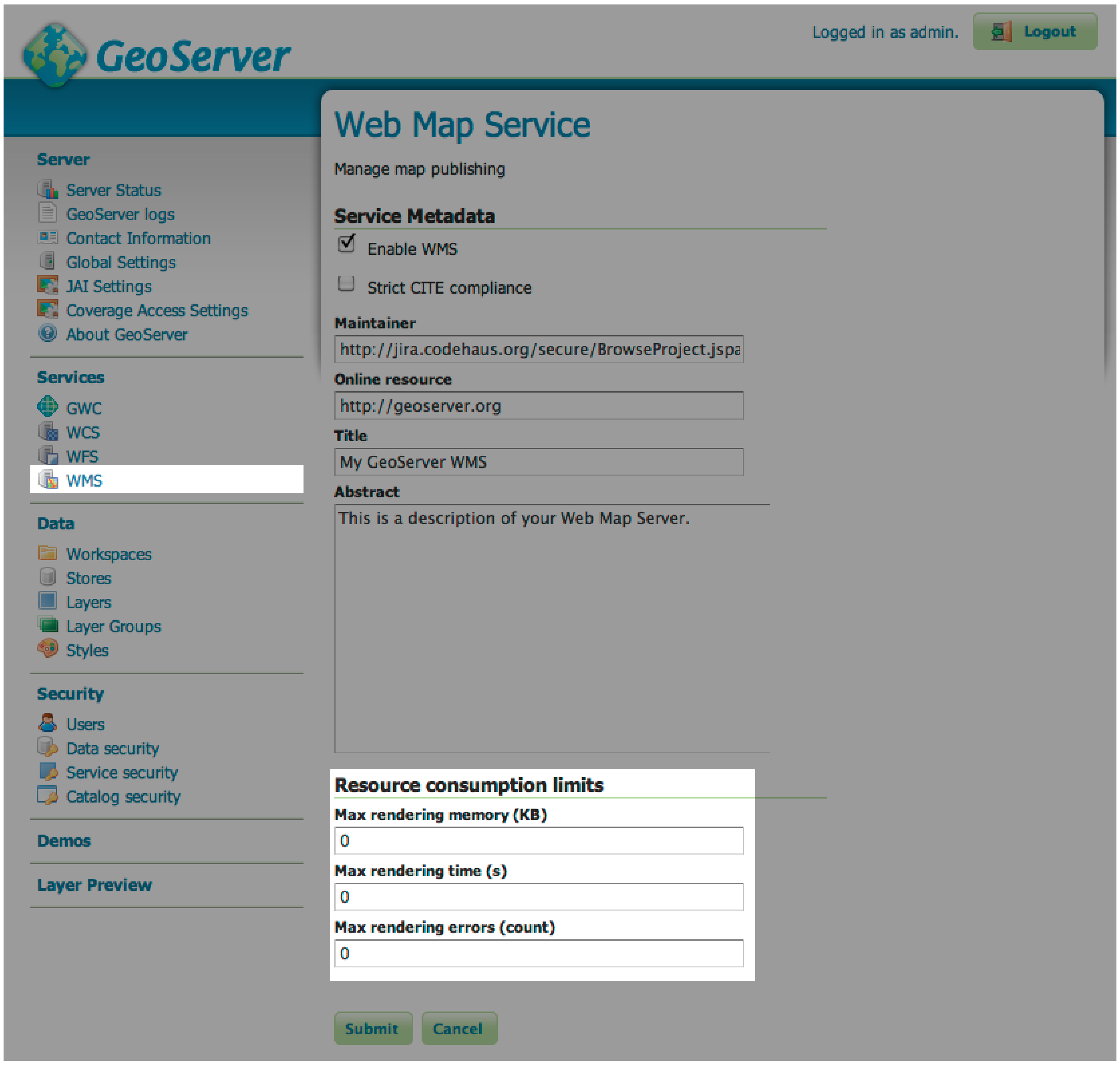
☑ JPEG Native Acceleration

☑ PNG Native Acceleration

☐ Mosaic Native Acceleration

Fig. 3.67: There are two considerations for the JAI settings.

- Enable JPEG and png Native Acceleration to speed up the performance of WMS requests
- Disable Tile Recycling as this optimization is less relevant on recent JVM implementations and has some overhead itself.



Web Map Service

Manage map publishing

Service Metadata

[✓] Enable WMS

[ ] Strict CITE compliance

Maintainer: http://jira.codehaus.org/secure/BrowseProject.jspa

Online resource: http://geoserver.org

Title: My GeoServer WMS

Abstract: This is a description of your Web Map Server.

Resource consumption limits

Max rendering memory (KB): 0

Max rendering time (s): 0

Max rendering errors (count): 0

Fig. 3.68: There is only one consideration for the Web Map Service page

- Don't set the "Resource Consumption Limits." This sounds a bit counter intuitive, but these limits are implemented in an inefficient way such that unless resource-intensive requests are common on your server it is more efficient to avoid the limits. A better implementation of this feature is available for GeoServer 2.1 and will be incorporated in GeoNode 1.1.

Managing a GeoNode Installation

Security and Permissions This tutorial will guide you through the steps that can be done in order to restrict the access on your data uploaded to geonode.

First of all it will be shown how a user can be created and what permissions he can have. Secondly we will take a closer look on to layers, maps and documents and the different opportunities you have in order to ban certain users from viewing or editing your data.

Users Your first step will be to create a user. There are three options to do so, depending on which kind of user you want to create you may choose a different option. We will start with creating a *superuser*, because this user is the most important. A superuser has all the permissions without explicitly assigning them.

The easiest way to create a superuser (in linux) is to open your terminal and type:

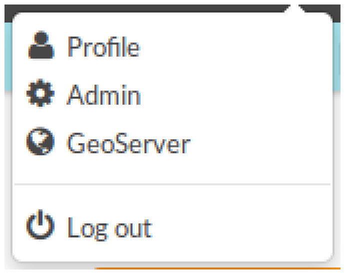
```
$ geonode createsuperuser
```

If you installed geonode by source you have to take the following commands:

```
$ source .venvs/geonode/bin activate
$ django-admin.py createsuperuser --settings=geonode.settings
```

You will be asked a username (in this tutorial we will call the superuser you now create *your_superuser*), an email address and a password.

Now you've created a superuser you should become familiar with the *Django Admin Interface*. As a superuser you are having access to this interface, where you can manage users, layers, permission and more. To learn more detailed about this interface check this LINK. For now it will be enough to just follow the steps. To attend the *Django Admin Interface*, go to your geonode website and *sign in* with *your_superuser*. Once you've logged in, the name of your user will appear on the top right. Click on it and the following menu will show up:



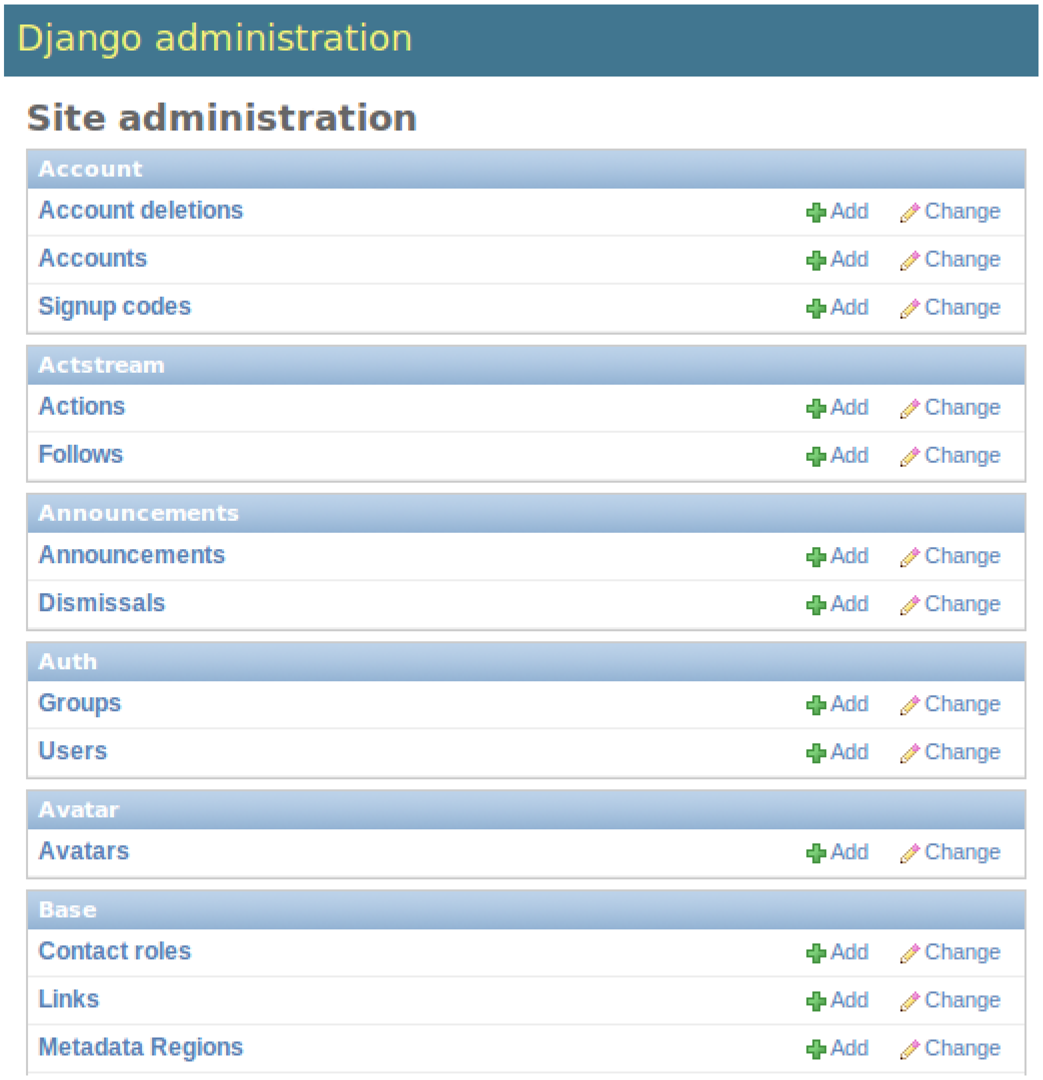
Clicking on *Admin* causes the interface to show up.

Go to *Auth* -> *Users* and you will see all the users that exist at the moment. In your case it will only be *your_superuser*. Click on it, and you will see a section on *Personal Info*, one on *Permissions* and one on *Important dates*. For the moment, the section on *Permissions* is the most important.

As you can see, there are three boxes that can be checked and unchecked. Because you've created a superuser, all three boxes are checked as default. If only the box *active* would have been checked, the user would not be a superuser and would not be able to access the *Django Admin Interface* (which is only available for users with the *staff* status). Therefore keep the following two things in mind:

- a superuser is able to access the *Django Admin Interface* and he has all permissions on the data uploaded to GeoNode.
- an ordinary user (created from the GeoNode interface) only has *active* permissions by default. The user will not have the ability to access the *Django Admin Interface* and certain permissions have to be added for him.

Until now we've only created superusers. So how do you create an ordinary user? You have two options:



Django administration

Site administration

Account		
Account deletions	Add	Change
Accounts	Add	Change
Signup codes	Add	Change

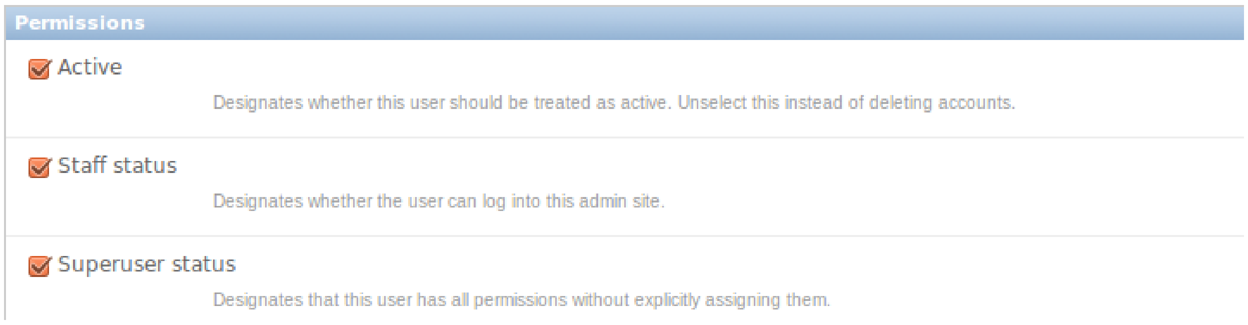
Actstream		
Actions	Add	Change
Follows	Add	Change

Announcements		
Announcements	Add	Change
Dismissals	Add	Change

Auth		
Groups	Add	Change
Users	Add	Change

Avatar		
Avatars	Add	Change

Base		
Contact roles	Add	Change
Links	Add	Change
Metadata Regions	Add	Change



Permissions

✓ Active

Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

✓ Staff status

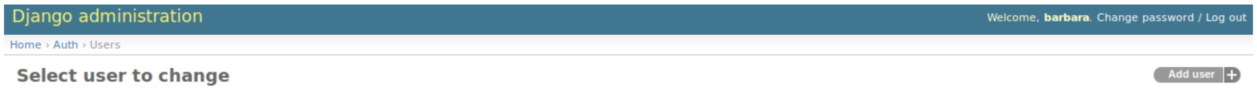
Designates whether the user can log into this admin site.

✓ Superuser status

Designates that this user has all permissions without explicitly assigning them.

1. Django Admin Interface

   First we will create a user via the *Django Admin Interface* because we've still got it open. Therefore go back to *Auth -> Users* and you should find a button on the right that says *Add user*.



Django administration

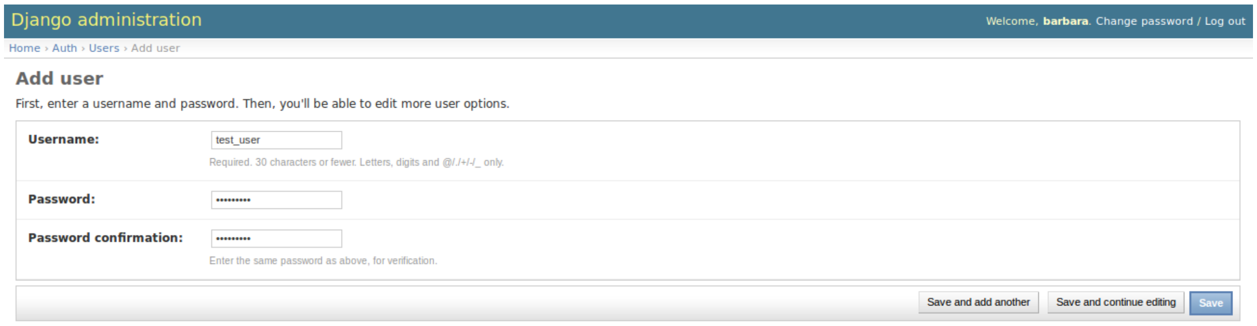
Welcome, **barbara**. Change password / Log out

Home › Auth › Users

Select user to change

Add user

   Click on it and a form to fill out will appear. Name the new user test_user, choose a password and click *save* at the right bottom of the site.



Django administration

Welcome, **barbara**. Change password / Log out

Home › Auth › Users › Add user

Add user

First, enter a username and password. Then, you'll be able to edit more user options.

Username:	test_user Required. 30 characters or fewer. Letters, digits and @/./+/-/_ only.
Password:	•••••••••
Password confirmation:	••••••••• Enter the same password as above, for verification.

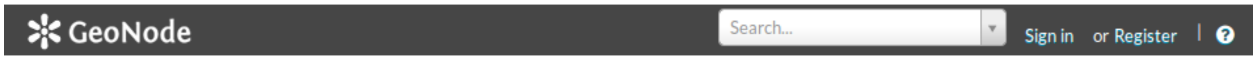
Save and add another | Save and continue editing | Save

   Now you should be directed to the site where you could change the permissions on the user *test_user*. As default only *active* is checked. If you want this user also to be able to attend this admin interface you could also check *staff status*. But for now we leave the settings as they are!

   To test whether the new user was successfully created, go back to the GeoNode web page and try to sign in.

2. GeoNode website

   To create an ordinary user you could also just use the GeoNode website. If you installed GeoNode using a release, you sho
      see a *Register* button on the top, beside the *Sign in* button (you might have to log out before).



   Hit the button and again a form will appear for you to fill out. This user will be named *geonode_user*

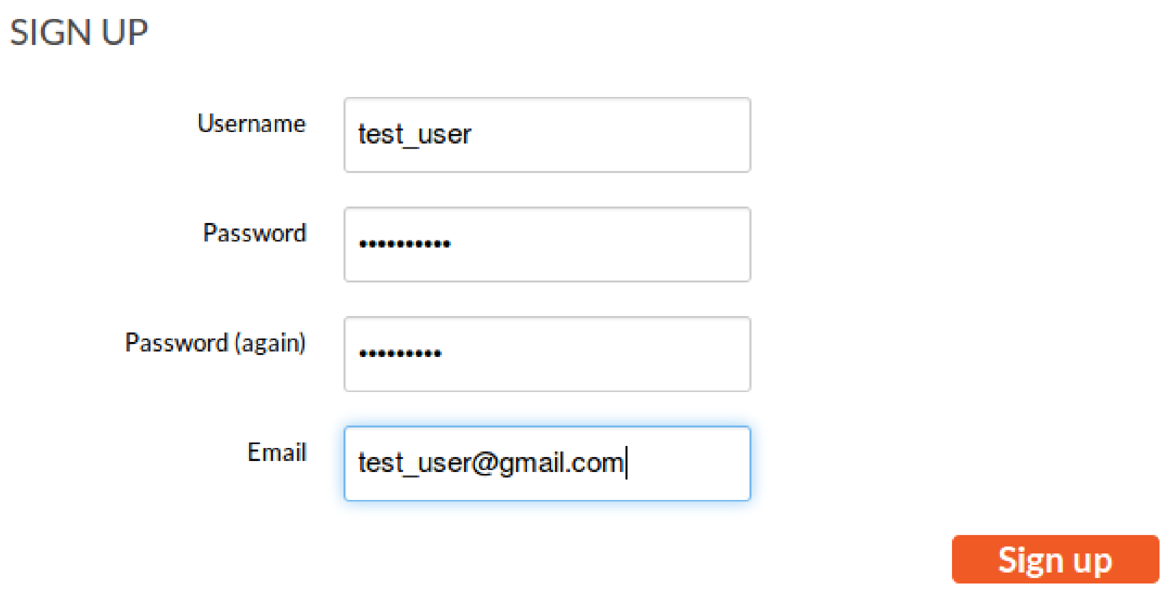
   By hitting *Sign up* the user will be signed up, as default only with the status *active*.

As mentioned before, this status can be changed as well. To do so, sign in with *your_superuser* again and attend the admin interface. Go again to *Auth -> Users*, where now three users should appear:

We now want to change the permission of the *geonode_user* so that he will be able to attend the admin interface as well. Click on to *geonode_user* and you will automatically be moved to the site where you can change the permissions. Check the box *staff status* and hit *save* to store the changes.

To sum it up, we have now created three users with different kind of permissions.

- **your_superuser**: This user is allowed to attend the admin interface and has all available permissions on layers, maps etc.
- **geonode_user**: This user is permitted to attend the admin interface, but permissions on layers, maps etc. have to be assigned.
- **test_user**: This user is not able to attend the admin interface, permissions on layers, maps etc. have also to be assigned.


SIGN UP

Username: test_user

Password: ••••••••••

Password (again): •••••••••

Email: test_user@gmail.com

You should know have an overview over the different kinds of users and how to create and edit them. You've also learned about the permissions a certain user has and how to change them using the *Django Admin Interface*.

Note: If you've installed GeoNode in developing mode, the *Register* button won't be seen from the beginning. To add this button to the website, you have to change the *REGISTRATION_OPEN = False* in the settings.py to *REGISTRATION_OPEN = True*. Then reload GeoNode and you should also be able to see the *Register* button.

Layers Now that we've already created some users, we will take a closer look on the security of layers, how you can protect your data not to be viewed or edited by unwanted users.

Hint: As already mentioned before it is important to know that a superuser does have unrestricted access to all your uploaded data. That means you cannot ban a superuser from viewing, downloading or editing a layer!

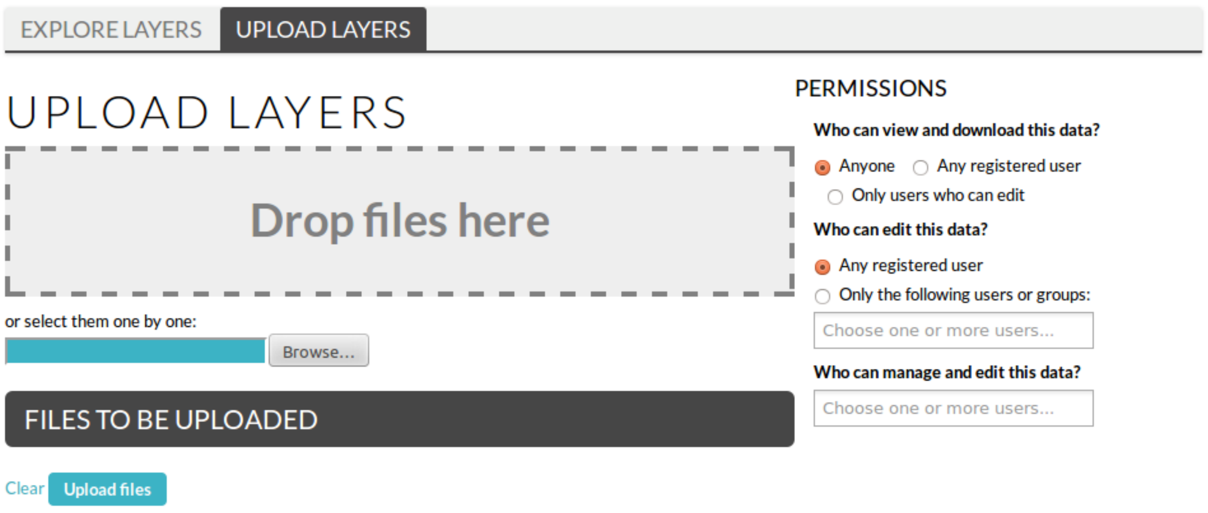
The permissions on a certain layer can already be set when uploading your files. When the upload form appears (*Layers -> Upload Layer*) you will see the permission section on the right side:

As it can be seen here, the access on your layer is split up into three groups:

- view and download data
- edit data
- manage and edit data

The difference between *manage and edit layer* and *edit layer* is that a user assigned to *edit layer* is not able to change the permissions on the layer whereas a user assigned to *manage and edit layer* can change the permissions. You can now choose whether you want your layer to be viewed and downloaded by

- anyone
- any registered user
- a certain user (or group)



PERMISSIONS

Who can view and download this data?

[✓ Anyone] Any registered user Only users who can edit

Who can edit this data?

[✓ Any registered user] Only the following users or groups:

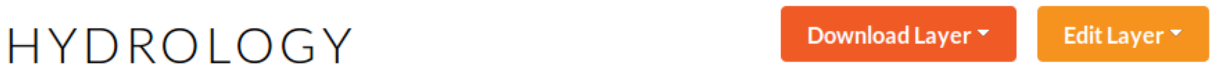
Choose one or more users...

Who can manage and edit this data?

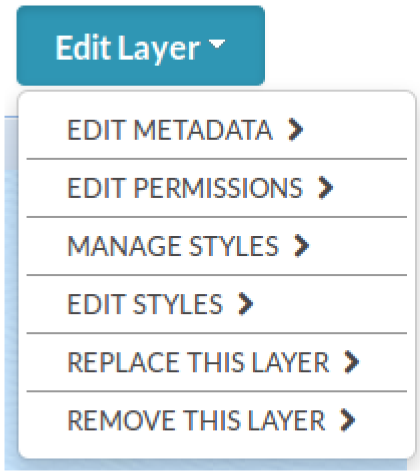
Choose one or more users...

We will now upload our **test layer** like shown HERE. If you want your layer only be viewed by certain users or a group, you have to choose *Only users who can edit* in the part *Who can view and download this data*. In the section *Who can edit this data* you write down the names of the users you want to have admission on this data. For this first layer we will choose the settings like shown in the following image:

If you now log out, your layer can still be seen, but the unregistered users won't be able to edit your layer. Now sign in as *geonode_user* and click on the **test layer**. Above the layer you can see this:



The *geonode_user* is able to edit the **test_layer**. But before going deeper into this, we have to first take a look on another case. As an administrator you might also upload your layers to geoserver and then make them available on GeoNode using *updatelayers*. Or you even add the layers via the terminal using *importlayers* (LINK TUTORIAL). To set the permissions on this layer, click on the **test layer** (you've uploaded via *updatelayers*) and you will see the same menu as shown in the image above. Click *Edit layer* and the menu will appear.



Choose *edit permissions* and a window with the permission settings will appear. This window can also be opened by scrolling down the website. On the right-hand side of the page you should be able to see a button like this.

Click on it and you will see the same window as before.

Now set the permissions of this layer using the following settings:

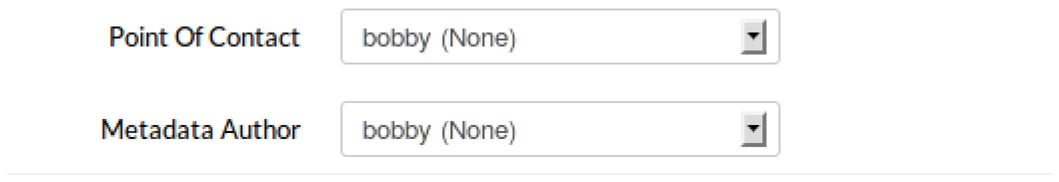
When you assign a user to be able to edit your data, this user is allowed to execute all of the following actions:

- edit metadata
- edit styles
- manage styles
- replace layer
- remove layer

So be aware that each user assigned to edit this layer can even remove it! In our case, only the user *test_user* and *your_superuser* do have the rights to do so. *Geonode_user* is neither able to view nor to download or edit this layer.

Now you are logged in as the user *test_user*. Below the **test_layer** you can see the following:

By clicking *Edit Layer* and *Edit Metadata* on top of the layer, you can change this information. The *test_user* is able to change all the metadata of this layer. We now want to change to *point of contact*, therefore scroll down until you see this:



Point Of Contact: bobby (None)

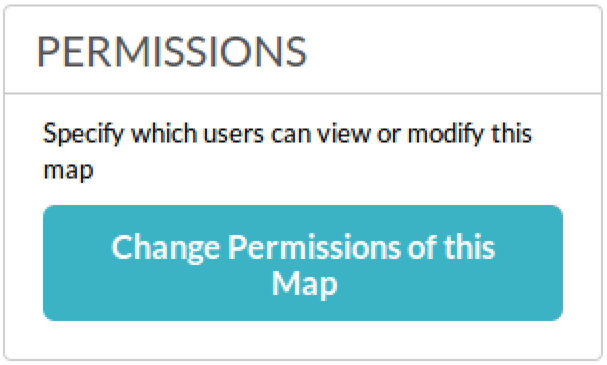
Metadata Author: bobby (None)

Change the *point of contact* from *_who_ever_created_this* to *test_user*. *Save* your changes and you will now be able to see the following:

> **Warning:** If you allow a user to view and download a layer, this user will also be able to edit the styles, even if he is not assigned to edit the layer! Keep this in mind!

To learn how you can edit metadata or change the styles go to this section LINK.

Maps The permission on maps are basically the same as on layers, just that there are fewer options on how to edit the map. Let's create a map (or already TUTORIAL?). Click on **test_map** and scroll down till you see this:

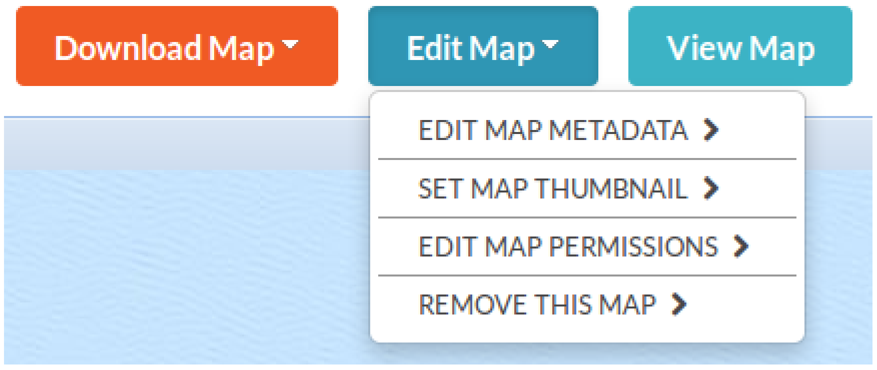


PERMISSIONS

Specify which users can view or modify this map

Here you can set the same permissions as known from the layer permissions! Set the permissions of this map as seen here:

Save your changes and then log out and log in as *test_user*. You should now be able to view the *test_map* and click on to *Edit map*.



As you may recognize, this user is not able to change the permissions on this map. If you log in as the user *geonode_user* you should be able to see the button *change map permissions* when you scroll down the page.

Documents All the same is also valid for your uploaded documents.

Monitoring a GeoNode

Backing up and Restoring GeoNode

In order to completely backup a GeoNode installation requires that each separate component of GeoNode be tackled separately. There are separate backup files for Geonode/Django and GeoServer.

Creating a backup

1. Make a backup of the GeoNode database (PostgreSQL needs to be running):

```
sudo -u postgres -i pg_dump -c -Fc geonode > geonodedb.backup
```

2. Stop all services:

```
sudo service apache2 stop
sudo /etc/init.d/tomcat7 stop
```

3. Backup Geonode config & GeoServer data directory:

```
tar -cvzf geonodeConfigBackup.tgz /etc/geonode
tar -cvzf geonodeDataBackup.tgz  /var/lib/geoserver/geonode-data/
```

4. In addition, any templates, design changes, and/or CSS files will also need to be captured.

5. Restart all services:

```
sudo service apache2 start
sudo /etc/init.d/tomcat7 start
```

Restoring a backup

1. First stop all services:

```
    sudo service apache2 stop
    sudo /etc/init.d/tomcat7 stop
```

2. Restore Geonode config & GeoServer data directory:

```
    sudo tar -C / -xvzf geonodeConfigBackup.tgz
    sudo tar -C / -xvzf geonodeDataBackup.tgz
```

3. In addition, any templates, design changes, and/or CSS files will need to be restored.
4. Make a backup of the GeoNode database (PostgreSQL needs to be running):

```
    sudo service postgresql start
    sudo -u postgres -i "psql -c 'drop database geonode;'"
    sudo -u postgres -i "psql -c 'create database geonode;'"
    sudo -u postgres -s "pg_restore -Fc -d geonode /path/to/geonodedb.backup"
```

5. Start all services:

```
    sudo service apache2 start
    sudo /etc/init.d/tomcat7 start
```

Notes This approach works between machines with the same version of Linux and Geonode, and may not work between different versions of Linux, PostgreSQL, and PostGIS.

Deploying GeoNode

Loading Data into a GeoNode

This module will walk you through the various options available to load data into your GeoNode from GeoServer, on the command-line or programatically. You can choose from among these techniques depending on what kind of data you have and how you have your geonode setup.

Using importlayers to import Data into GeoNode The geonode.layers app includes 2 management commands that you can use to load or configure data in your GeoNode. Both of these are invoked by using the manage.py script. This section will walk you through how to use the importlayers management command and the subsequent section will lead you through the process of using updatelayers.

The first thing to do is to use the –help option to the importlayers command to investigate the options to this management command. You can display this help by executing the following command:

```
$ geonode importlayers --help
```

This will produce output that looks like the following:

```
Usage: manage.py importlayers [options] path [path...]

Brings a data file or a directory full of data files into aGeoNode site.  Layers are added to the Dja

Options:
  -v VERBOSITY, --verbosity=VERBOSITY
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
      --settings=SETTINGS   The Python path to a settings module, e.g.
                            "myproject.settings.main". If this isn't provided, the
                            DJANGO_SETTINGS_MODULE environment variable will be
                            used.
      --pythonpath=PYTHONPATH
                            A directory to add to the Python path, e.g.
                            "/home/djangoprojects/myproject".
      --traceback           Print traceback on exception
      -u USER, --user=USER  Name of the user account which should own the imported
                            layers
      -i, --ignore-errors   Stop after any errors are encountered.
      -o, --overwrite       Overwrite existing layers if discovered (defaults
                            False)
      -k KEYWORDS, --keywords=KEYWORDS
                            The default keywords for the imported layer(s). Will
                            be the same for all imported layers if multiple
                            imports are done in one command
      --version             show program's version number and exit
      -h, --help            show this help message and exit
```

While the description of most of the options should be self explanatory, its worth reviewing some of the key options in a bit more detail.

- The -i option will force the command to stop when it first encounters an error. Without this option specified, the process will skip over errors that have layers and continue loading the other layers.
- The -o option specifies that layers with the same name as the base name will be loaded and overwrite the existing layer.
- The -u option specifies which will be the user that owns the imported layers. The same user will be the point of contact and the metadata author as well for that layer
- The -k option is used to add keywords for all of the layers imported.

The import layers management command is invoked by specifying options as described above and specifying the path to a single layer file or to a directory that contains multiple files. For purposes of this exercise, lets use the default set of testing layers that ship with geonode. You can replace this path with the directory to your own shapefiles:

```
$ geonode importlayers -v 3 /var/lib/geonode/lib/python2.7/site-packages/gisdata/data/good/vector/
```

This command will produce the following output to your terminal:

```
Verifying that GeoNode is running ...
Found 8 potential layers.
No handlers could be found for logger "pycsw"
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec
[created] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vec

Detailed report of failures:


Finished processing 8 layers in 30.0 seconds.

8 Created layers
```

```
0 Updated layers
0 Skipped layers
0 Failed layers
3.750000 seconds per layer
```

If you encounter errors while running this command, you can use the -v option to increase the verbosity of the output so you can debug the problem. The verbosity level can be set from 0-3 with 0 being the default. An example of what the output looks like when an error is encountered and the verbosity is set to 3 is shown below:

```
Verifying that GeoNode is running ...
Found 8 potential layers.
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect
[failed] Layer for '/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vect

Detailed report of failures:


/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/gisdata/data/good/vector/san_andres_y_prov
================
Traceback (most recent call last):
  File "/Users/jjohnson/projects/geonode/geonode/layers/utils.py", line 682, in upload
    keywords=keywords,
  File "/Users/jjohnson/projects/geonode/geonode/layers/utils.py", line 602, in file_upload
    keywords=keywords, title=title)
  File "/Users/jjohnson/projects/geonode/geonode/layers/utils.py", line 305, in save
    store = cat.get_store(name)
  File "/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/geoserver/catalog.py", line 176, i
    for ws in self.get_workspaces():
  File "/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/geoserver/catalog.py", line 489, i
    description = self.get_xml("%s/workspaces.xml" % self.service_url)
  File "/Users/jjohnson/.venvs/geonode/lib/python2.7/site-packages/geoserver/catalog.py", line 136, i
    response, content = self.http.request(rest_url)
  File "/Library/Python/2.7/site-packages/httplib2/__init__.py", line 1445, in request
    (response, content) = self._request(conn, authority, uri, request_uri, method, body, headers, red
  File "/Library/Python/2.7/site-packages/httplib2/__init__.py", line 1197, in _request
    (response, content) = self._conn_request(conn, request_uri, method, body, headers)
  File "/Library/Python/2.7/site-packages/httplib2/__init__.py", line 1133, in _conn_request
    conn.connect()
  File "/Library/Python/2.7/site-packages/httplib2/__init__.py", line 799, in connect
    raise socket.error, msg
error: [Errno 61] Connection refused
```

Note: This last section of output will be repeated for all layers, and only the first one is show above.

This error indicates that GeoNode was unable to connect to GeoServer to load the layers. To solve this, you should make sure GeoServer is running and re-run the command.

If you encounter errors with this command that you cannot solve, you should bring them up on the geonode users mailing list.

You should now have the knowledge necessary to import layers into your GeoNode project from a directory on the servers filesystem and can use this to load many layers into your GeoNode at once.

Note: If you do not use the -u command option, the ownership of the imported layers will be assigned to the primary superuser in your system. You can use GeoNodes Django Admin interface to modify this after the fact if you want them to be owned by another user.

GeoServer Data Configuration While it is possible to import layers directly from your servers filesystem into your GeoNode, you may have an existing GeoServer that already has data in it, or you may want to configure data from a GeoServer which is not directly supported by uploading data.

GeoServer supports a wide range of data formats and connections to database, and while many of them are not supported as GeoNode upload formats, if they can be configured in GeoServer, you can add them to your GeoNode by following the procedure described below.

GeoServer supports 3 types of data: Raster, Vector and Databases. For a list of the supported formats for each type of data, consult the following pages:

- http://docs.geoserver.org/latest/en/user/data/vector/index.html#data-vector
- http://docs.geoserver.org/latest/en/user/data/raster/index.html
- http://docs.geoserver.org/latest/en/user/data/database/index.html

Note: Some of these raster or vector formats or database types require that you install specific plugins in your GeoServer in order to use the. Please consult the GeoServer documentation for more information.

Lets walk through an example of configuring a new PostGIS database in GeoServer and then configuring those layers in your GeoNode.

First visit the GeoServer administration interface on your server. This is usually on port 8080 and is available at http://localhost:8080/geoserver/web/

You should login with the superuser credentials you setup when you first configured your GeoNode instance.

Once you are logged in to the GeoServer Admin interface, you should see the following.

Note: The number of stores, layers and workspaces may be different depending on what you already have configured in your GeoServer.

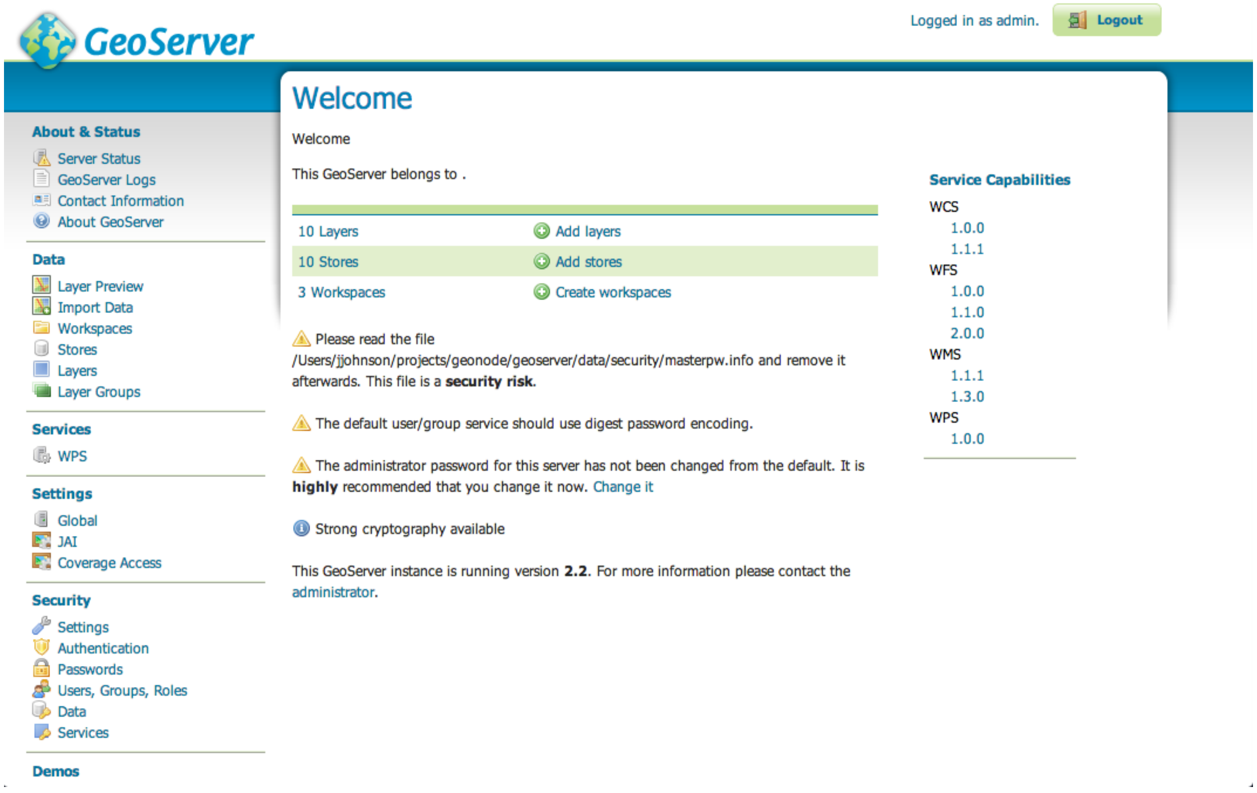
Next you want to select the "Stores" option in the left hand menu, and then the "Add new Store" option. The following screen will be displayed.

In this case, we want to select the PostGIS store type to create a connection to our existing database. On the next screen you will need to enter the parameters to connect to your PostGIS database (alter as necessary for your own database).

Note: If you are unsure about any of the settings, leave them as the default.

The next screen lets you configure the layers in your database. This will of course be different depending on the layers in your database.

Select the "Publish" button for one of the layers and the next screen will be displayed where you can enter metadata for this layer. Since we will be managing this metadata in GeoNode, we can leave these alone for now.



Logged in as admin. Logout

About & Status
- Server Status
- GeoServer Logs
- Contact Information
- About GeoServer

Data
- Layer Preview
- Import Data
- Workspaces
- Stores
- Layers
- Layer Groups

Services
- WPS

Settings
- Global
- JAI
- Coverage Access

Security
- Settings
- Authentication
- Passwords
- Users, Groups, Roles
- Data
- Services

Demos

Welcome

Welcome

This GeoServer belongs to .

10 Layers	Add layers
10 Stores	Add stores
3 Workspaces	Create workspaces

Please read the file /Users/jjohnson/projects/geonode/geoserver/data/security/masterpw.info and remove it afterwards. This file is a **security risk**.

The default user/group service should use digest password encoding.

The administrator password for this server has not been changed from the default. It is **highly** recommended that you change it now. Change it

Strong cryptography available

This GeoServer instance is running version **2.2**. For more information please contact the administrator.

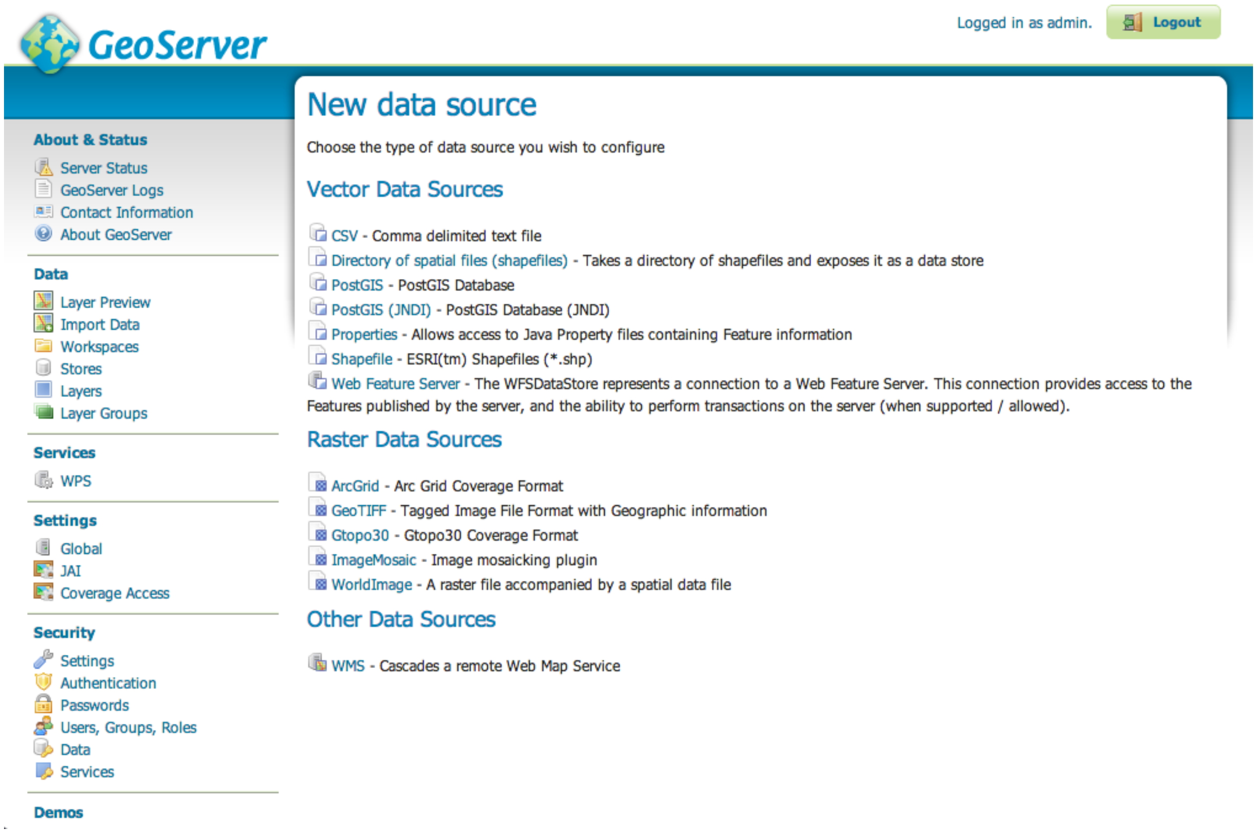
Service Capabilities

WCS
- 1.0.0
- 1.1.1

WFS
- 1.0.0
- 1.1.0
- 2.0.0

WMS
- 1.1.1
- 1.3.0

WPS
- 1.0.0



Logged in as admin. Logout

About & Status
- Server Status
- GeoServer Logs
- Contact Information
- About GeoServer

Data
- Layer Preview
- Import Data
- Workspaces
- Stores
- Layers
- Layer Groups

Services
- WPS

Settings
- Global
- JAI
- Coverage Access

Security
- Settings
- Authentication
- Passwords
- Users, Groups, Roles
- Data
- Services

Demos

New data source

Choose the type of data source you wish to configure

Vector Data Sources

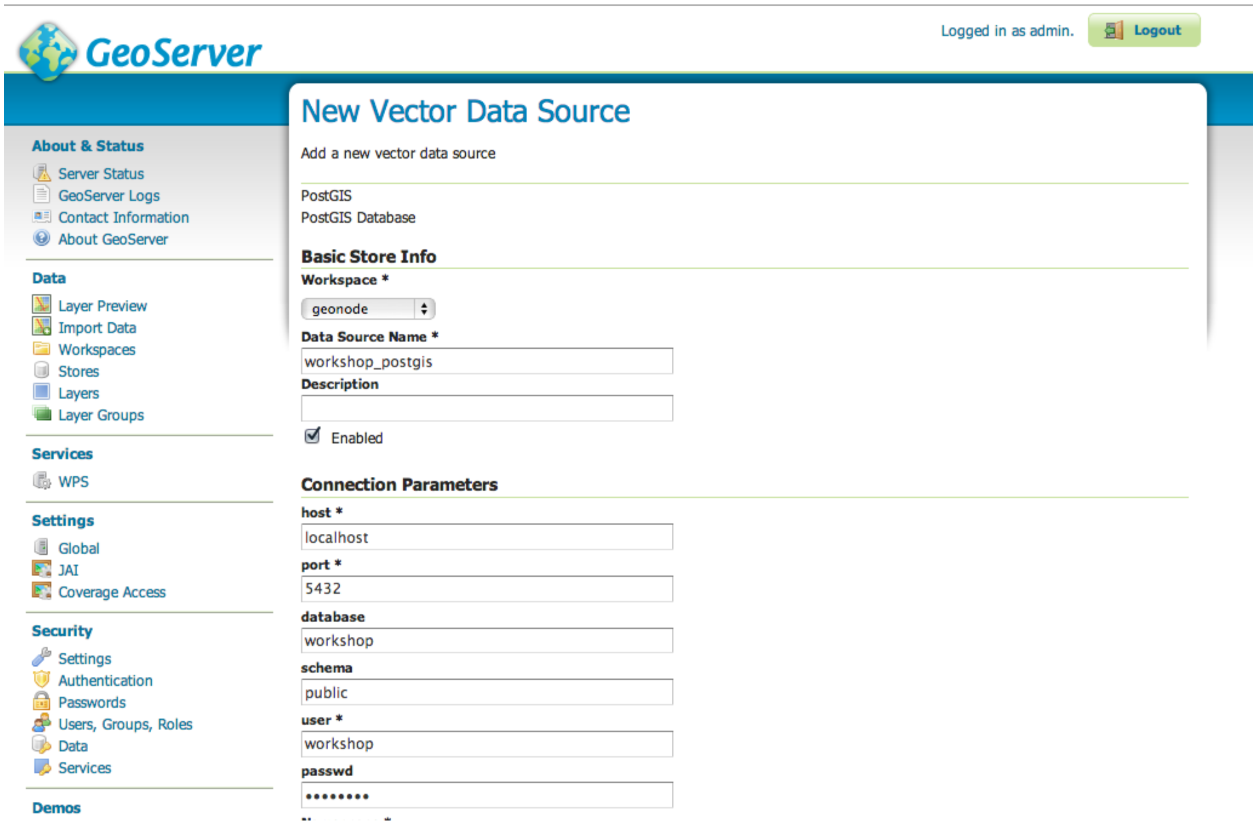
- CSV - Comma delimited text file
- Directory of spatial files (shapefiles) - Takes a directory of shapefiles and exposes it as a data store
- PostGIS - PostGIS Database
- PostGIS (JNDI) - PostGIS Database (JNDI)
- Properties - Allows access to Java Property files containing Feature information
- Shapefile - ESRI(tm) Shapefiles (*.shp)
- Web Feature Server - The WFSDataStore represents a connection to a Web Feature Server. This connection provides access to the Features published by the server, and the ability to perform transactions on the server (when supported / allowed).

Raster Data Sources

- ArcGrid - Arc Grid Coverage Format
- GeoTIFF - Tagged Image File Format with Geographic information
- Gtopo30 - Gtopo30 Coverage Format
- ImageMosaic - Image mosaicking plugin
- WorldImage - A raster file accompanied by a spatial data file

Other Data Sources

- WMS - Cascades a remote Web Map Service



New Vector Data Source

Add a new vector data source

PostGIS
PostGIS Database

Basic Store Info

Workspace * geonode

Data Source Name * workshop_postgis

Description

[✓] Enabled

Connection Parameters

host * localhost

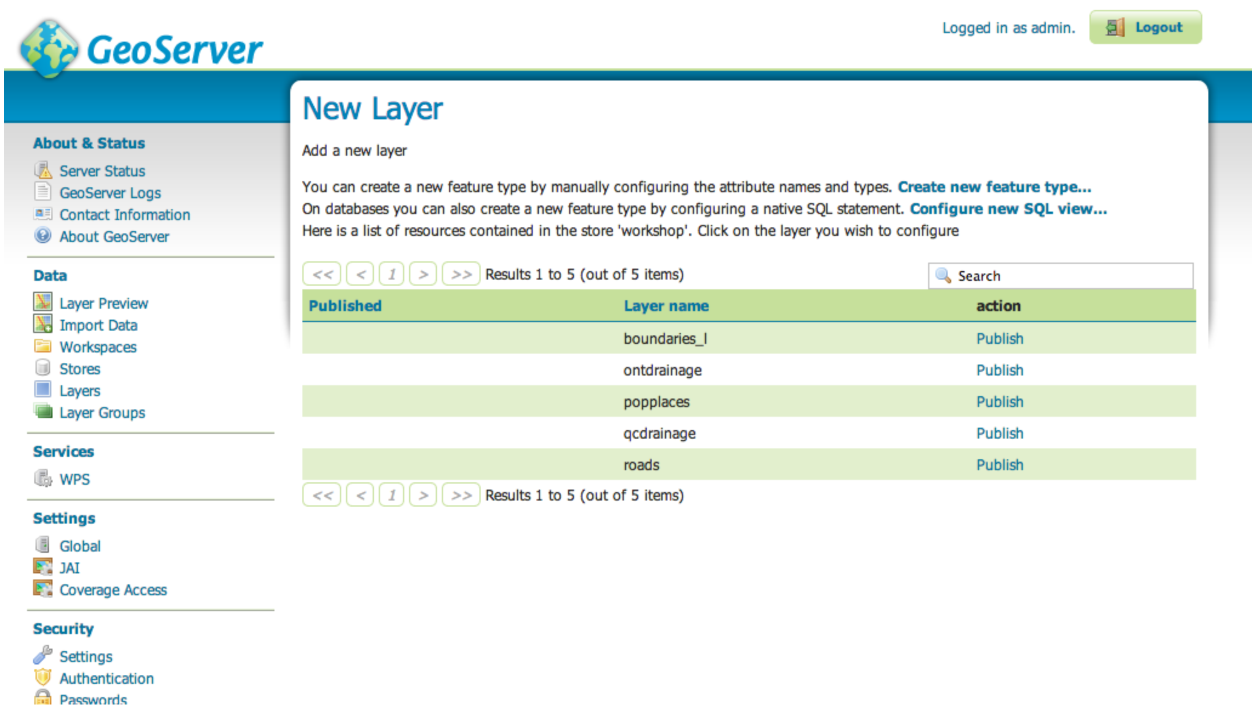
port * 5432

database workshop

schema public

user * workshop

passwd ••••••••



New Layer

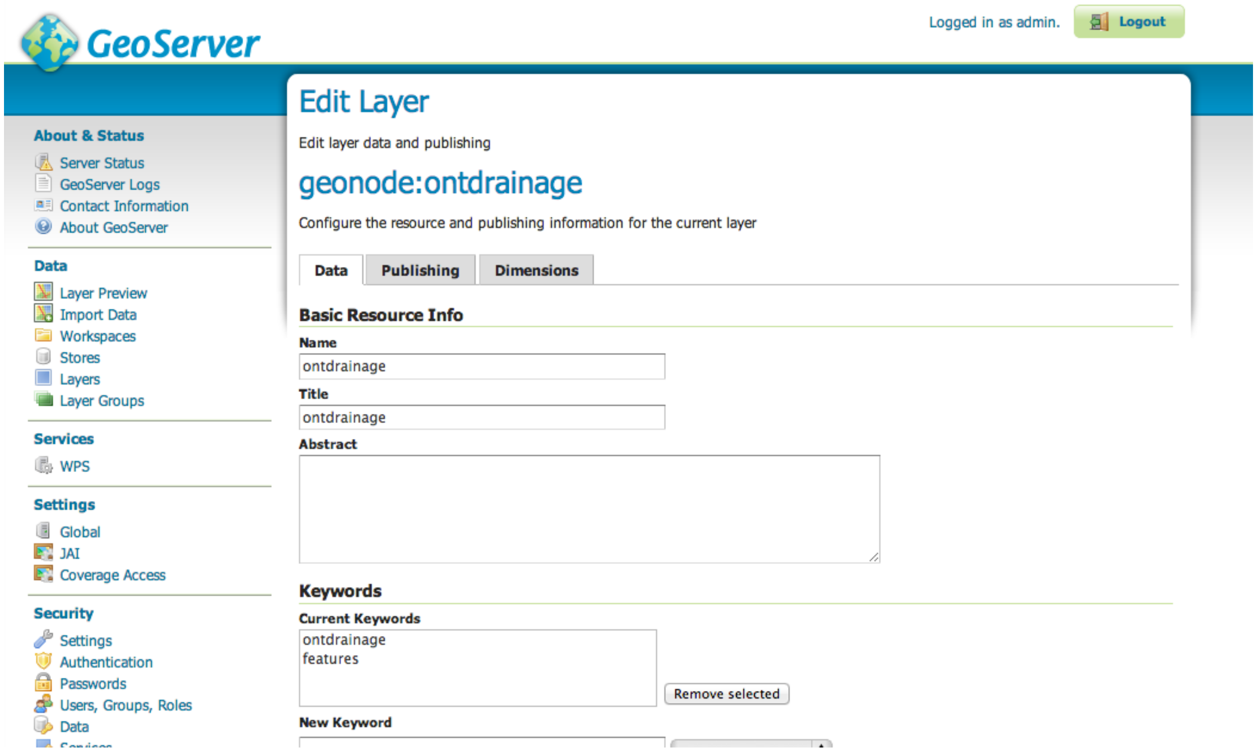
Add a new layer

You can create a new feature type by manually configuring the attribute names and types. **Create new feature type...**
On databases you can also create a new feature type by configuring a native SQL statement. **Configure new SQL view...**
Here is a list of resources contained in the store 'workshop'. Click on the layer you wish to configure

Results 1 to 5 (out of 5 items)

Published	Layer name	action
	boundaries_l	Publish
	ontdrainage	Publish
	popplaces	Publish
	qcdrainage	Publish
	roads	Publish

Results 1 to 5 (out of 5 items)



Edit Layer

Edit layer data and publishing

geonode:ontdrainage

Configure the resource and publishing information for the current layer

Basic Resource Info

Name: ontdrainage

Title: ontdrainage

Abstract:

Keywords

Current Keywords: ontdrainage, features

New Keyword

The things that *must* be specified are the Declared SRS and you must select the “Compute from Data” and “Compute from native bounds” links after the SRS is specified.

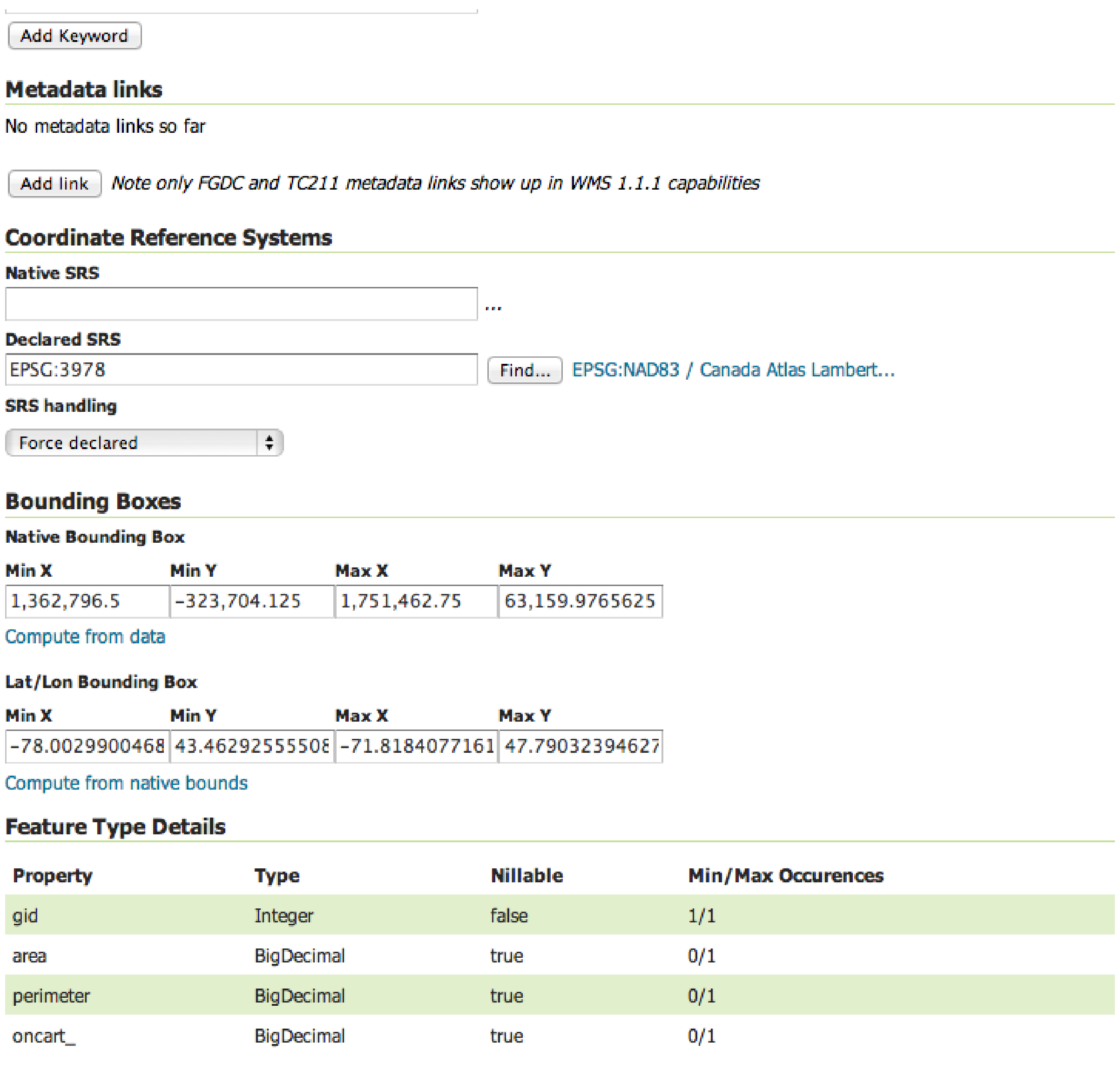
Click save and this layer will now be configured for use in your GeoServer.

The next step is to configure these layers in GeoNode. The updatelayers management command is used for this purpose. As with importlayers, its useful to look at the command line options for this command by passing the –help option:

```
$ python manage.py updatelayers --help
```

This help option displays the following:

```
Usage: manage.py updatelayers [options]

Update the GeoNode application with data from GeoServer

Options:
  -v VERBOSITY, --verbosity=VERBOSITY
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=verbose output, 3=very verbose output
  --settings=SETTINGS   The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
  --pythonpath=PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
  --traceback           Print traceback on exception
  -i, --ignore-errors   Stop after any errors are encountered.
  -u USER, --user=USER  Name of the user account which should own the imported
                        layers
```



Add Keyword

Metadata links

No metadata links so far

Add link *Note only FGDC and TC211 metadata links show up in WMS 1.1.1 capabilities*

Coordinate Reference Systems

Native SRS

...

Declared SRS

EPSG:3978 Find... EPSG:NAD83 / Canada Atlas Lambert...

SRS handling

Force declared

Bounding Boxes

Native Bounding Box

Min X	Min Y	Max X	Max Y
1,362,796.5	-323,704.125	1,751,462.75	63,159.9765625

Compute from data

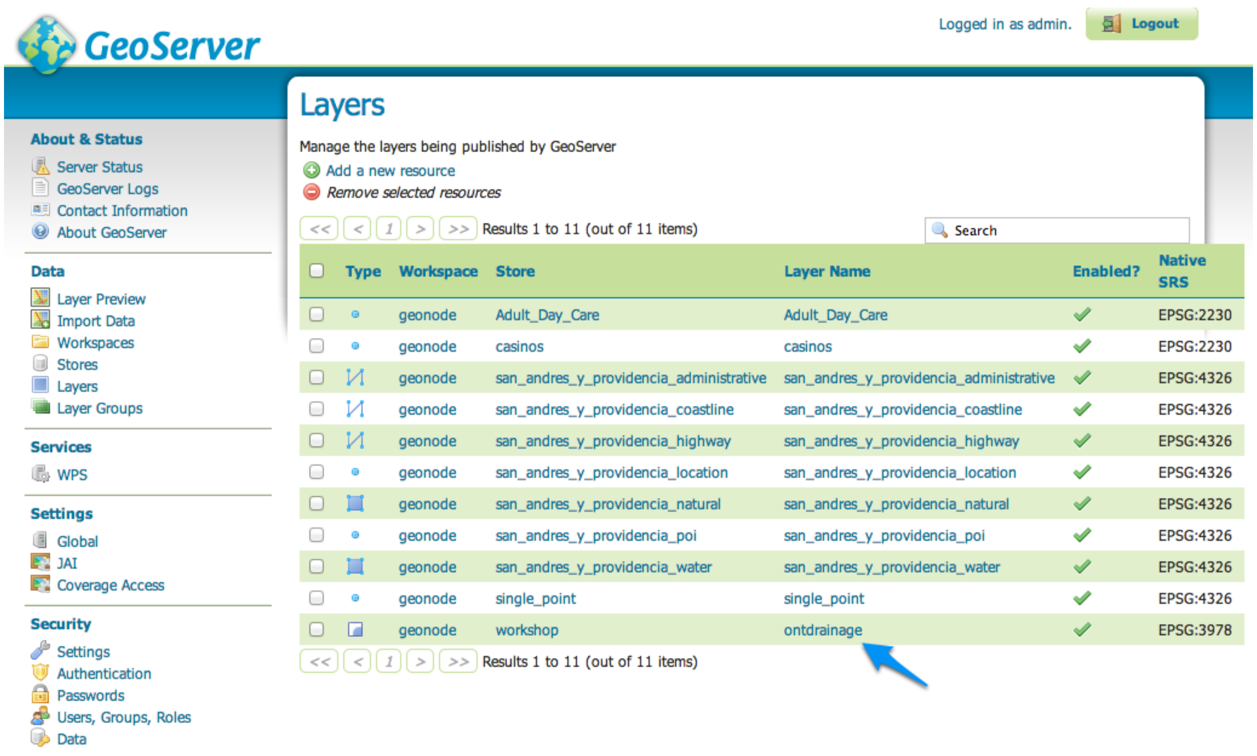
Lat/Lon Bounding Box

Min X	Min Y	Max X	Max Y
-78.0029900468	43.46292555508	-71.8184077161	47.79032394627

Compute from native bounds

Feature Type Details

Property	Type	Nillable	Min/Max Occurences
gid	Integer	false	1/1
area	BigDecimal	true	0/1
perimeter	BigDecimal	true	0/1
oncart_	BigDecimal	true	0/1



Layers

Manage the layers being published by GeoServer

Add a new resource

Remove selected resources

<< < 1 > >> Results 1 to 11 (out of 11 items)

Type	Workspace	Store	Layer Name	Enabled?	Native SRS
●	geonode	Adult_Day_Care	Adult_Day_Care	✓	EPSG:2230
●	geonode	casinos	casinos	✓	EPSG:2230
	geonode	san_andres_y_providencia_administrative	san_andres_y_providencia_administrative	✓	EPSG:4326
	geonode	san_andres_y_providencia_coastline	san_andres_y_providencia_coastline	✓	EPSG:4326
	geonode	san_andres_y_providencia_highway	san_andres_y_providencia_highway	✓	EPSG:4326
●	geonode	san_andres_y_providencia_location	san_andres_y_providencia_location	✓	EPSG:4326
■	geonode	san_andres_y_providencia_natural	san_andres_y_providencia_natural	✓	EPSG:4326
●	geonode	san_andres_y_providencia_poi	san_andres_y_providencia_poi	✓	EPSG:4326
■	geonode	san_andres_y_providencia_water	san_andres_y_providencia_water	✓	EPSG:4326
●	geonode	single_point	single_point	✓	EPSG:4326
	geonode	workshop	ontdrainage	✓	EPSG:3978

<< < 1 > >> Results 1 to 11 (out of 11 items)

```
  -w WORKSPACE, --workspace=WORKSPACE
                        Only update data on specified workspace
  --version             show program's version number and exit
  -h, --help            show this help message and exit
```

For this sample, we can use the default options. So enter the following command to configure the layers from our GeoServer into our GeoNode:

```
$ python manage.py updatelayers
```

The output will look something like the following:

```
[created] Layer Adult_Day_Care (1/11)
[created] Layer casinos (2/11)
[updated] Layer san_andres_y_providencia_administrative (3/11)
[updated] Layer san_andres_y_providencia_coastline (4/11)
[updated] Layer san_andres_y_providencia_highway (5/11)
[updated] Layer san_andres_y_providencia_location (6/11)
[updated] Layer san_andres_y_providencia_natural (7/11)
[updated] Layer san_andres_y_providencia_poi (8/11)
[updated] Layer san_andres_y_providencia_water (9/11)
[updated] Layer single_point (10/11)
[created] Layer ontdrainage (11/11)


Finished processing 11 layers in 45.0 seconds.

3 Created layers
8 Updated layers
0 Failed layers
4.090909 seconds per layer
```

Note: This example picked up 2 additional layers that were already in our GeoServer, but were not already in our GeoNode.

For layers that already exist in your GeoNode, they will be updated and the configuration synchronized between GeoServer and GeoNode.

You can now view and use these layers in your GeoNode.

Using GDAL and OGR to convert your Data for use in GeoNode GeoNode supports uploading data in shapefiles, GeoTiff, csv and kml formats (for the last two formats only if you are using the geonode.importer backend in the UPLOAD variable in settings.py). If your data is in other formats, you will need to convert it into one of these formats for use in GeoNode. This section will walk you through the steps necessary to convert your data into formats suitable for uploading into GeoNode.

You will need to make sure that you have the gdal library installed on your system. On Ubuntu you can install this package with the following command:

```
$ sudo apt-get install gdal-bin
```

OGR (Vector Data) OGR is used to manipulate vector data. In this example, we will use MapInfo .tab files and convert them to shapefiles with the ogr2ogr command. We will use sample MapInfo files from the website linked below.

http://services.land.vic.gov.au/landchannel/content/help?name=sampledata

You can download the Admin;(Postcode) layer by issuing the following command:

```
$ wget http://services.land.vic.gov.au/sampledata/mif/admin_postcode_vm.zip
```

You will need to unzip this dataset by issuing the following command:

```
$ unzip admin_postcode_vm.zip
```

This will leave you with the following files in the directory where you executed the above commands:

```
|-- ANZVI0803003025.htm
|-- DSE_Data_Access_Licence.pdf
|-- VMADMIN.POSTCODE_POLYGON.xml
|-- admin_postcode_vm.zip
--- vicgrid94
    --- mif
        --- lga_polygon
            --- macedon\ ranges
                |-- EXTRACT_POLYGON.mid
                |-- EXTRACT_POLYGON.mif
                --- VMADMIN
                    |-- POSTCODE_POLYGON.mid
                    --- POSTCODE_POLYGON.mif
```

First, lets inspect this file set using the following command:

```
$ ogrinfo -so vicgrid94/mif/lga_polygon/macedon\ ranges/VMADMIN/POSTCODE_POLYGON.mid POSTCODE_POLYGON
```

The output will look like the following:

```
Had to open data source read-only.
INFO: Open of `vicgrid94/mif/lga_polygon/macedon ranges/VMADMIN/POSTCODE_POLYGON.mid'
    using driver `MapInfo File' successful.

Layer name: POSTCODE_POLYGON
Geometry: 3D Unknown (any)
Feature Count: 26
Extent: (2413931.249367, 2400162.366186) - (2508952.174431, 2512183.046927)
Layer SRS WKT:
PROJCS["unnamed",
    GEOGCS["unnamed",
        DATUM["GDA94",
            SPHEROID["GRS 80",6378137,298.257222101],
            TOWGS84[0,0,0,-0,-0,-0,0]],
        PRIMEM["Greenwich",0],
        UNIT["degree",0.0174532925199433]],
    PROJECTION["Lambert_Conformal_Conic_2SP"],
    PARAMETER["standard_parallel_1",-36],
    PARAMETER["standard_parallel_2",-38],
    PARAMETER["latitude_of_origin",-37],
    PARAMETER["central_meridian",145],
    PARAMETER["false_easting",2500000],
    PARAMETER["false_northing",2500000],
    UNIT["Meter",1]]
PFI: String (10.0)
POSTCODE: String (4.0)
FEATURE_TYPE: String (6.0)
FEATURE_QUALITY_ID: String (20.0)
PFI_CREATED: Date (10.0)
UFI: Real (12.0)
UFI_CREATED: Date (10.0)
UFI_OLD: Real (12.0)
```

This gives you information about the number of features, the extent, the projection and the attributes of this layer.

Next, lets go ahead and convert this layer into a shapefile by issuing the following command:

```
$ ogr2ogr -t_srs EPSG:4326 postcode_polygon.shp vicgrid94/mif/lga_polygon/macedon\ ranges/VMADMIN/POS
```

Note that we have also reprojected the layer to the WGS84 spatial reference system with the -t_srs ogr2ogr option.

The output of this command will look like the following:

```
Warning 6: Normalized/laundered field name: 'FEATURE_TYPE' to 'FEATURE_TY'
Warning 6: Normalized/laundered field name: 'FEATURE_QUALITY_ID' to 'FEATURE_QU'
Warning 6: Normalized/laundered field name: 'PFI_CREATED' to 'PFI_CREATE'
Warning 6: Normalized/laundered field name: 'UFI_CREATED' to 'UFI_CREATE'
```

This output indicates that some of the field names were truncated to fit into the constraint that attributes in shapefiles are only 10 characters long.

You will now have a set of files that make up the postcode_polygon.shp shapefile set. We can inspect them by issuing the following command:

```
$ ogrinfo -so postcode_polygon.shp postcode_polygon
```

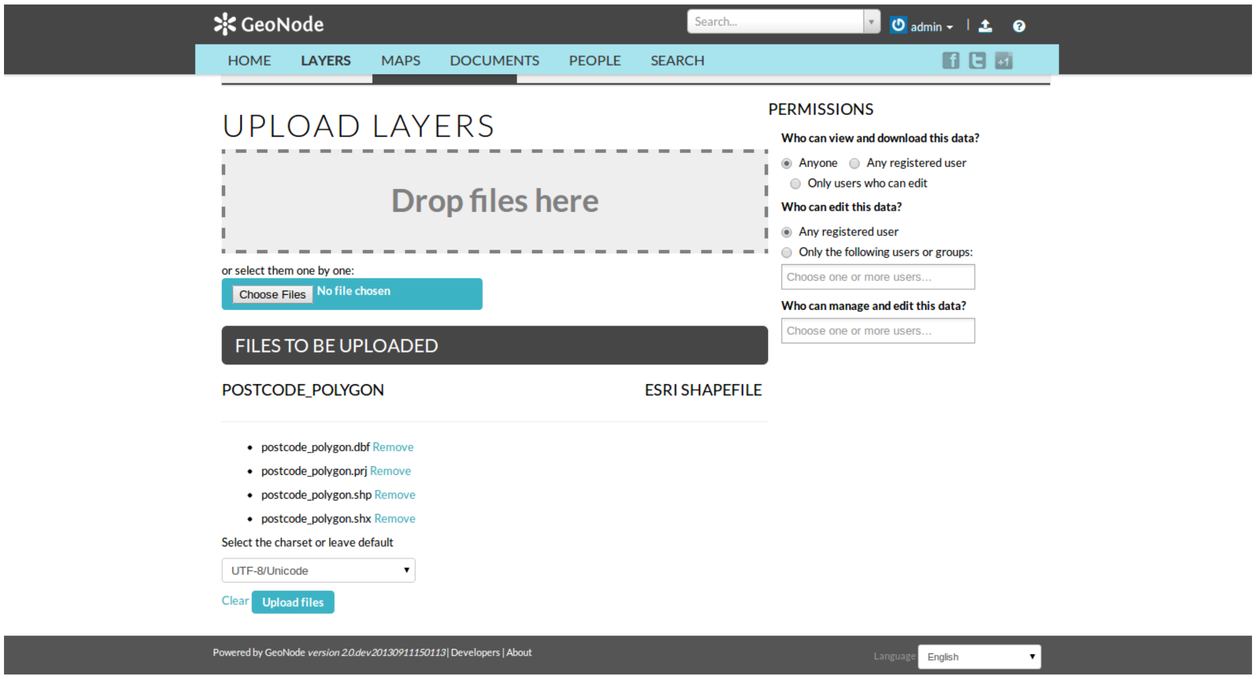
The output will look similar to the output we saw above when we inspected the MapInfo file we converted from:

```
INFO: Open of `postcode_polygon.shp'
      using driver `ESRI Shapefile' successful.
```

```
Layer name: postcode_polygon
Geometry: Polygon
Feature Count: 26
Extent: (144.030296, -37.898156) - (145.101137, -36.888878)
Layer SRS WKT:
GEOGCS["GCS_WGS_1984",
    DATUM["WGS_1984",
        SPHEROID["WGS_84",6378137,298.257223563]],
    PRIMEM["Greenwich",0],
    UNIT["Degree",0.017453292519943295]]
PFI: String (10.0)
POSTCODE: String (4.0)
FEATURE_TY: String (6.0)
FEATURE_QU: String (20.0)
PFI_CREATE: Date (10.0)
UFI: Real (12.0)
UFI_CREATE: Date (10.0)
UFI_OLD: Real (12.0)
```

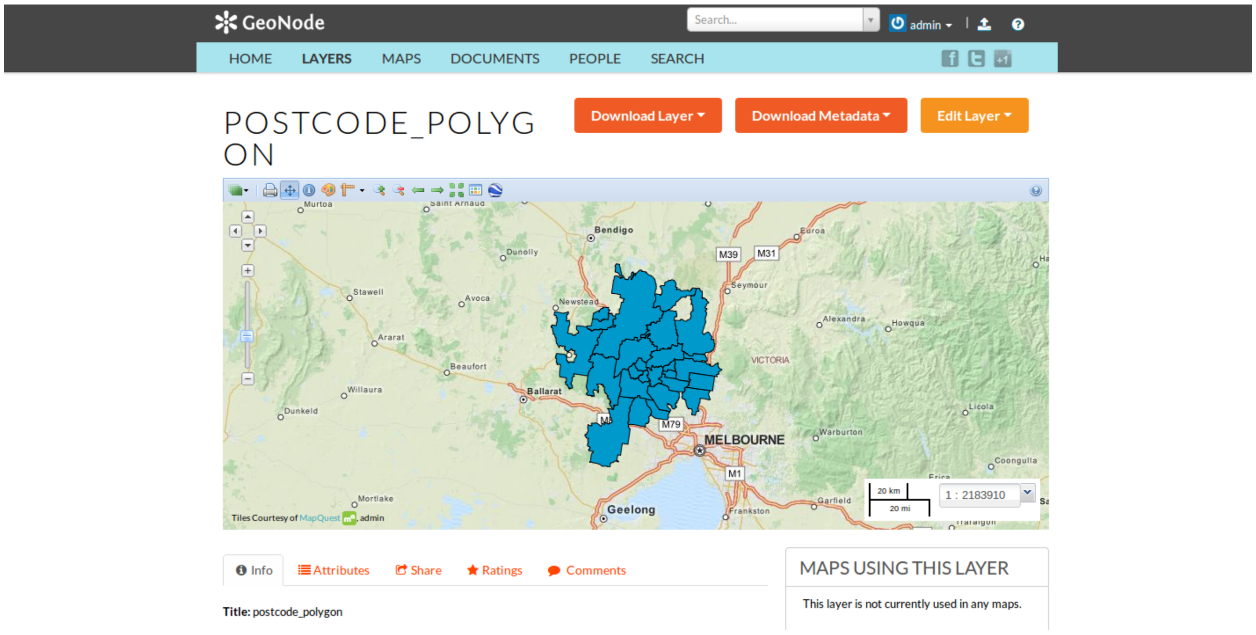
These files can now be loaded into your GeoNode instance via the normal uploader.

Visit the upload page in your GeoNode, drag and drop the files that composes the shapefile that you have generated using the GDAL ogr2ogr command (postcode_polygon.dbf, postcode_polygon.prj, postcode_polygon.shp, postcode_polygon.shx). Give the permissions as needed and then click the "Upload files" button.


GeoNode
Search...
admin
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
UPLOAD LAYERS
Drop files here
or select them one by one:
Choose Files
No file chosen
FILES TO BE UPLOADED
POSTCODE_POLYGON
ESRI SHAPEFILE
postcode_polygon.dbf Remove
postcode_polygon.prj Remove
postcode_polygon.shp Remove
postcode_polygon.shx Remove
Select the charset or leave default
UTF-8/Unicode
Clear
Upload files
PERMISSIONS
Who can view and download this data?
Anyone
Any registered user
Only users who can edit
Who can edit this data?
Any registered user
Only the following users or groups:
Choose one or more users...
Who can manage and edit this data?
Choose one or more users...
Powered by GeoNode version 2.0.dev20130911150113 | Developers | About
Language
English

As soon as the import process completes, you will have the possibility to go straight to the layer info page ("Layer Info" button), or to edit the metadata for that layer ("Edit Metadata" button), or to manage the styles for that layer ("Manage Styles").

GDAL (Raster Data) Now that we have seen how to convert vector layers into shapefiles using ogr2ogr, we will walk through the steps necessary to perform the same operation with Raster layers. For this example, we will work with Arc/Info Binary and ASCII Grid data and convert it into GeoTiff format for use in GeoNode.



Title: postcode_polygon

This layer is not currently used in any maps.

First, you need to download the sample data to work with it. You can do this by executing the following command:

```
$ wget http://dev.opengeo.org/~jjohnson/sample_asc.tgz
```

You will need to uncompress this file by executing this command:

```
$ tar -xvzf sample_asc.tgz
```

You will be left with the following files on your filesystem:

```
|-- batemans_ele
|   |-- dblbnd.adf
|   |-- hdr.adf
|   |-- metadata.xml
|   |-- prj.adf
|   |-- sta.adf
|   |-- w001001.adf
|   |-- w001001x.adf
|-- batemans_elevation.asc
```

The file batemans_elevation.asc is an Arc/Info ASCII Grid file and the files in the batemans_ele directory are an Arc/Info Binary Grid file.

You can use the gdalinfo command to inspect both of these files by executing the following command:

```
$ gdalinfo batemans_elevation.asc
```

The output should look like the following:

```
Driver: AAIGrid/Arc/Info ASCII Grid
Files: batemans_elevation.asc
Size is 155, 142
Coordinate System is `'
Origin = (239681.000000000000000,6050551.000000000000000)
Pixel Size = (100.000000000000000,-100.000000000000000)
Corner Coordinates:
Upper Left  (  239681.000, 6050551.000)
```

```
Lower Left  (  239681.000, 6036351.000)
Upper Right (  255181.000, 6050551.000)
Lower Right (  255181.000, 6036351.000)
Center      (  247431.000, 6043451.000)
Band 1 Block=155x1 Type=Float32, ColorInterp=Undefined
    NoData Value=-9999
```

You can then inspect the batemans_ele files by executing the following command:

```
$ gdalinfo batemans_ele
```

And this should be the corresponding output:

```
Driver: AIG/Arc/Info Binary Grid
Files: batemans_ele
    batemans_ele/dblbnd.adf
    batemans_ele/hdr.adf
    batemans_ele/metadata.xml
    batemans_ele/prj.adf
    batemans_ele/sta.adf
    batemans_ele/w001001.adf
    batemans_ele/w001001x.adf
Size is 155, 142
Coordinate System is:
PROJCS["unnamed",
    GEOGCS["GDA94",
        DATUM["Geocentric_Datum_of_Australia_1994",
            SPHEROID["GRS 1980",6378137,298.257222101,
                AUTHORITY["EPSG","7019"]],
            TOWGS84[0,0,0,0,0,0,0],
            AUTHORITY["EPSG","6283"]],
        PRIMEM["Greenwich",0,
            AUTHORITY["EPSG","8901"]],
        UNIT["degree",0.0174532925199433,
            AUTHORITY["EPSG","9122"]],
        AUTHORITY["EPSG","4283"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",153],
    PARAMETER["scale_factor",0.9996],
    PARAMETER["false_easting",500000],
    PARAMETER["false_northing",10000000],
    UNIT["METERS",1]]
Origin = (239681.000000000000000,6050551.000000000000000)
Pixel Size = (100.000000000000000,-100.000000000000000)
Corner Coordinates:
Upper Left  (  239681.000, 6050551.000) (150d 7'28.35"E, 35d39'16.56"S)
Lower Left  (  239681.000, 6036351.000) (150d 7'11.78"E, 35d46'56.89"S)
Upper Right (  255181.000, 6050551.000) (150d17'44.07"E, 35d39'30.83"S)
Lower Right (  255181.000, 6036351.000) (150d17'28.49"E, 35d47'11.23"S)
Center      (  247431.000, 6043451.000) (150d12'28.17"E, 35d43'13.99"S)
Band 1 Block=256x4 Type=Float32, ColorInterp=Undefined
    Min=-62.102 Max=142.917
NoData Value=-3.4028234663852886e+38
```

You will notice that the batemans_elevation.asc file does *not* contain projection information while the batemans_ele file does. Because of this, lets use the batemans_ele files for this exercise and convert them to a GeoTiff for use in GeoNode. We will also reproject this file into WGS84 in the process. This can be accomplished with the following command.

$ gdalwarp -t_srs EPSG:4326 batemans_ele batemans_ele.tif

The output will show you the progress of the conversion and when it is complete, you will be left with a batemans_ele.tif file that you can upload to your GeoNode.

You can inspect this file with the gdalinfo command:

```
$ gdalinfo batemans_ele.tif
```

Which will produce the following output:

```
Driver: GTiff/GeoTIFF
Files: batemans_ele.tif
Size is 174, 130
Coordinate System is:
GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563,
            AUTHORITY["EPSG","7030"]],
        AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0],
    UNIT["degree",0.0174532925199433],
    AUTHORITY["EPSG","4326"]]
Origin = (150.119938943722502,-35.654598806259330)
Pixel Size = (0.001011114155919,-0.001011114155919)
Metadata:
    AREA_OR_POINT=Area
Image Structure Metadata:
    INTERLEAVE=BAND
Corner Coordinates:
Upper Left  ( 150.1199389, -35.6545988) (150d 7'11.78"E, 35d39'16.56"S)
Lower Left  ( 150.1199389, -35.7860436) (150d 7'11.78"E, 35d47' 9.76"S)
Upper Right ( 150.2958728, -35.6545988) (150d17'45.14"E, 35d39'16.56"S)
Lower Right ( 150.2958728, -35.7860436) (150d17'45.14"E, 35d47' 9.76"S)
Center      ( 150.2079059, -35.7203212) (150d12'28.46"E, 35d43'13.16"S)
Band 1 Block=174x11 Type=Float32, ColorInterp=Gray
```

You can then follow the same steps we used above to upload the GeoTiff file we created into the GeoNode, and you will see your layer displayed in the Layer Info page.

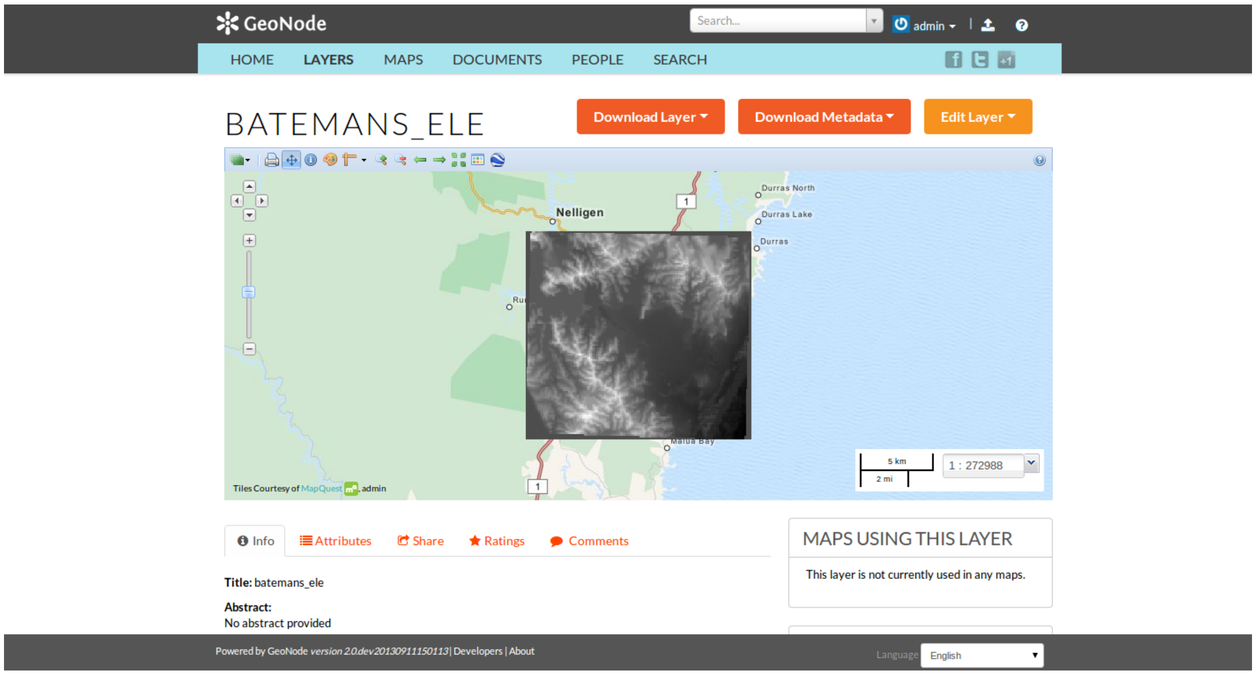
Now that you have seen how to convert layers with both OGR and GDAL, you can use these techniques to work with your own data and get it prepared for inclusion in your own GeoNode.

Loading OSM Data into GeoNode In this section, we will walk through the steps necessary to load OSM data into your GeoNode project. As discussed in previous sections, your GeoNode already uses OSM tiles from MapQuest and the main OSM servers as some of the available base layers. This session is specifically about extracting actual data from OSM and converting it for use in your project and potentially for Geoprocessing tasks.

The first step in this process is to get the data from OSM. We will be using the OSM Overpass API since it lets us do more complex queries than the OSM API itself. You should refer to the OSM Overpass API documentation to learn about all of its features. It is an extremely powerful API that lets you extract data from OSM using a very sophisticated API.

- http://wiki.openstreetmap.org/wiki/Overpass_API
- http://wiki.openstreetmap.org/wiki/Overpass_API/Language_Guide

In this example, we will be extracting building footprint data around Port au Prince in Haiti. To do this we will use an interactive tool that makes it easy construct a Query against the Overpass API. Point your browser at http://overpass-turbo.eu/ and use the search tools to zoom into Port Au Prince and Cite Soleil specifically.



Title: batemans_ele

Abstract:
No abstract provided

This layer is not currently used in any maps.

Powered by GeoNode version 2.0.dev20130911150113 | Developers | About

You will need to cut and paste the query specified below to get all of the appropriate data under the bbox:

```
<osm-script>
  <union into="_">
    <bbox-query {{bbox}}/>
    <recurse into="x" type="node-relation"/>
    <query type="way">
      <bbox-query {{bbox}}/>
      <has-kv k="building" v="yes"></has-kv>
    </query>
    <recurse into="x" type="way-node"/>
    <recurse type="way-relation"/>
  </union>
  <print mode="meta"/>
</osm-script>
```

This should look like the following.

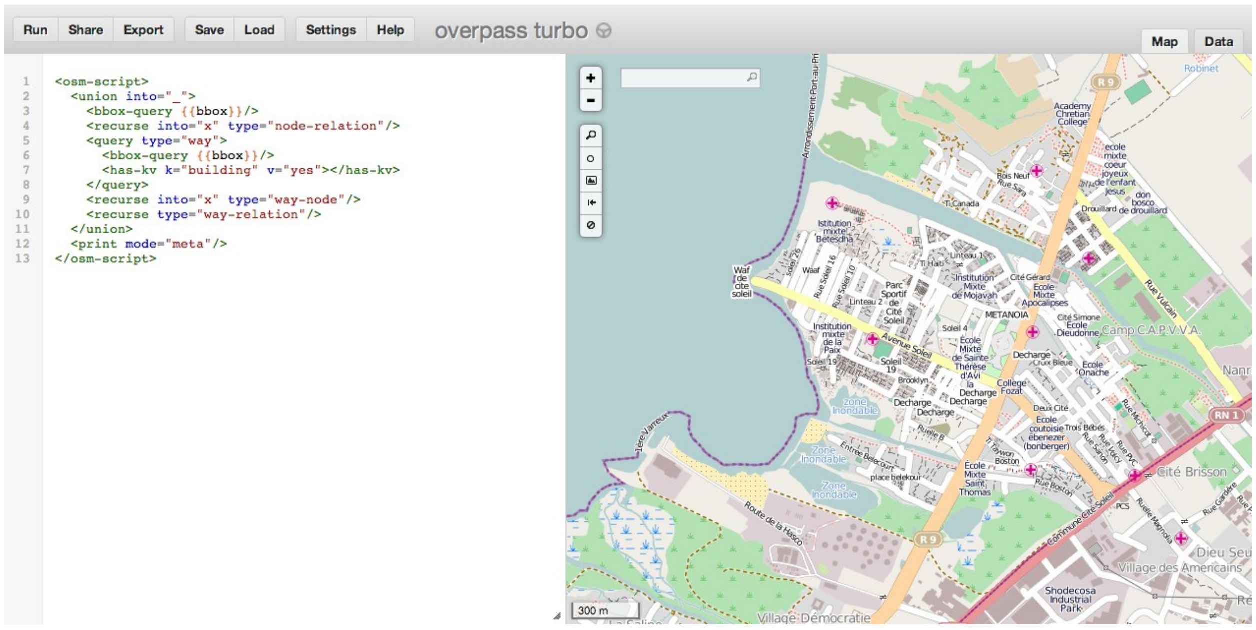
When you have the bbox and query set correctly, click the "Export" button on the menu to bring up the export menu, and then click the API interpreter link to download the OSM data base on the query you have specified.

This will download a file named 'export.osm' on your file system. You will probably want to rename it something else more specific. You can do that by issuing the following command in the directory where it was downloaded:

```
$ mv export.osm cite_soleil_buildings.osm
```

Note: You can also rename the file in your Operating Systems File management tool (Windows Explorer, Finder etc).

Exporting OSM data to shapefile using QGIS Now that we have osm data on our filesystem, we will need to convert it into a format suitable for uploading into your GeoNode. There are many ways to accomplish this, but for purposes of this example, we will use an OSM QGIS plugin that makes if fairly easy. Please consult the wiki page that



```
<osm-script>
  <union into="_">
    <bbox-query {{bbox}}/>
    <recurse into="x" type="node-relation"/>
    <query type="way">
      <bbox-query {{bbox}}/>
      <has-kv k="building" v="yes"></has-kv>
    </query>
    <recurse into="x" type="way-node"/>
    <recurse type="way-relation"/>
  </union>
  <print mode="meta"/>
</osm-script>
```

explains how to install this plugin and make sure it is installed in your QGIS instance. Once its installed, you can use the Web Menu to load your file.

This will bring up a dialog box that you can use to find and convert the osm file we downloaded.

When the process has completed, you will see your layers in the Layer Tree in QGIS.

Since we are only interested in the polygons, we can turn the other 2 layers off in the Layer Tree.

The next step is to use QGIS to convert this layer into a Shapefile so we can upload it into GeoNode. To do this, select the layer in the Layer tree, right click and then select the Save As option.

This will bring up the Save Vector Layer as Dialog.

Specify where on disk you want your file saved, and hit Save then OK.

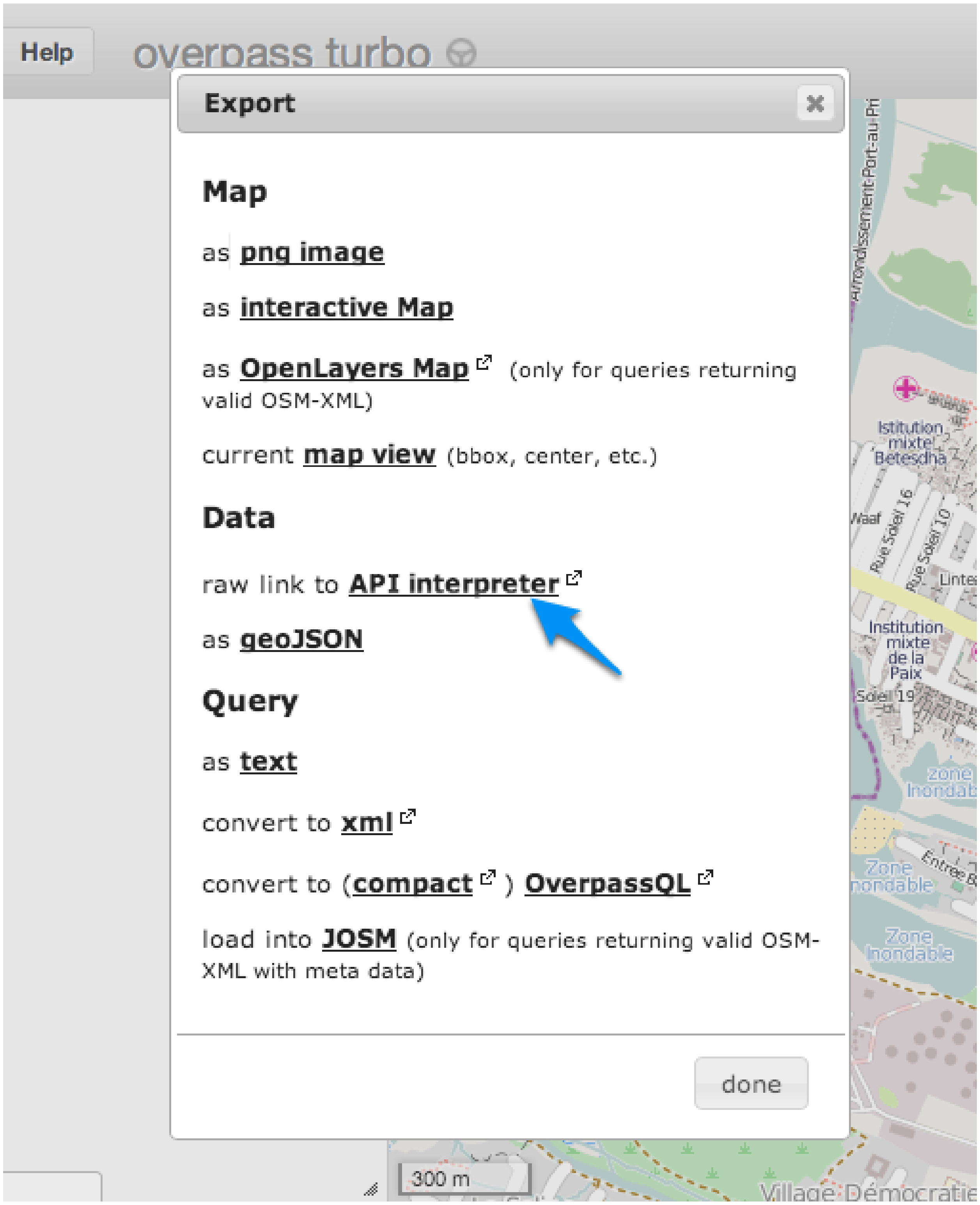
You now have a shapefile of the data you extracted from OSM that you can use to load into GeoNode. Use the GeoNode Layer Upload form to load the Shapefile parts into your GeoNode, and optionally edit the metadata and then you can view your layer in the Layer Info page in your geonode.

Note: You may want to switch to an imagery layer in order to more easily see the buildings on the OSM background.

Exporting OSM data to shapefile using GDAL An alternative way to export the .osm file to a shapefile is to use ogr2ogr combined with the GDAL osm driver, available from GDAL version 1.10.

As a first step, inspect how the GDAL osm driver sees the .osm file using the ogrinfo command:

```
$ ogrinfo cite_soleil_buildings.osm
Had to open data source read-only.
INFO: Open of `cite_soleil_buildings.osm'
      using driver `OSM' successful.
1: points (Point)
2: lines (Line String)
3: multilinestrings (Multi Line String)
4: multipolygons (Multi Polygon)
5: other_relations (Geometry Collection)
```



Export

Map

as png image

as interactive Map

as OpenLayers Map (only for queries returning valid OSM-XML)

current map view (bbox, center, etc.)

Data

raw link to API interpreter

as geoJSON

Query

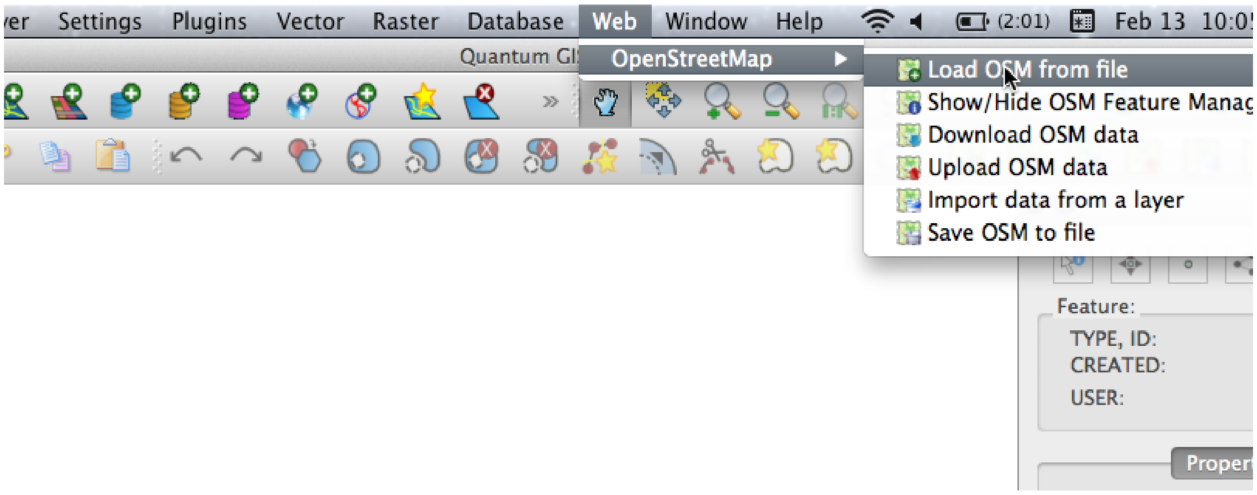
as text

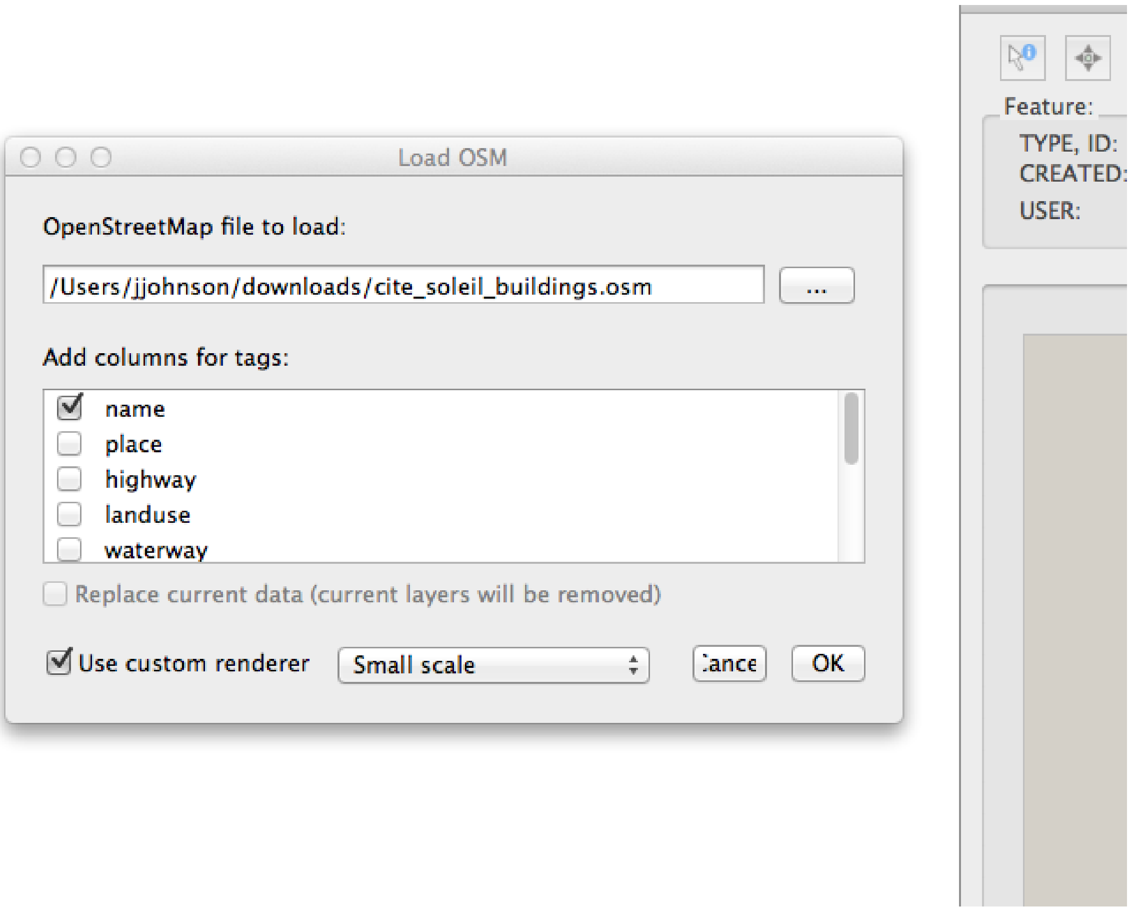
convert to xml

convert to (compact) OverpassQL

load into JOSM (only for queries returning valid OSM-XML with meta data)

done




Load OSM

OpenStreetMap file to load:

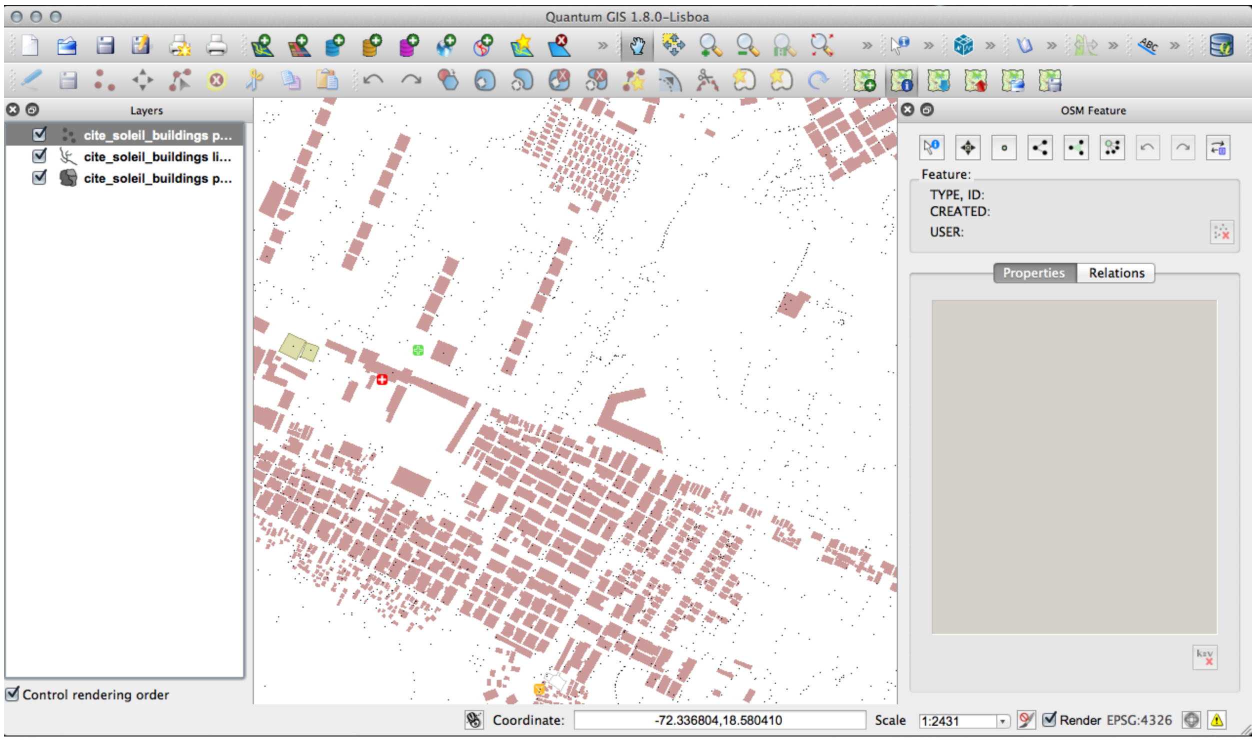
/Users/jjohnson/downloads/cite_soleil_buildings.osm

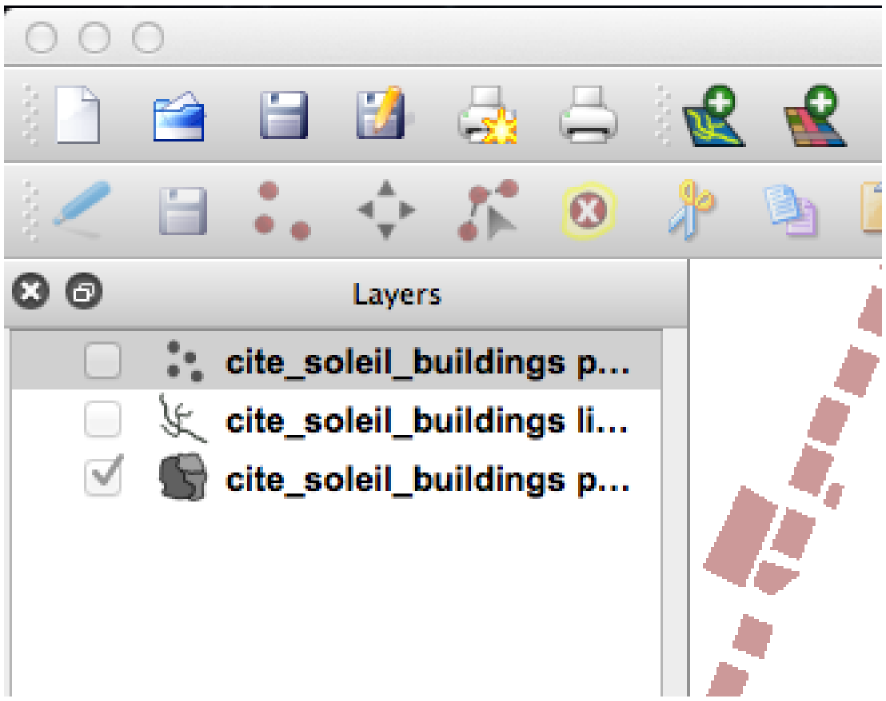
Add columns for tags:

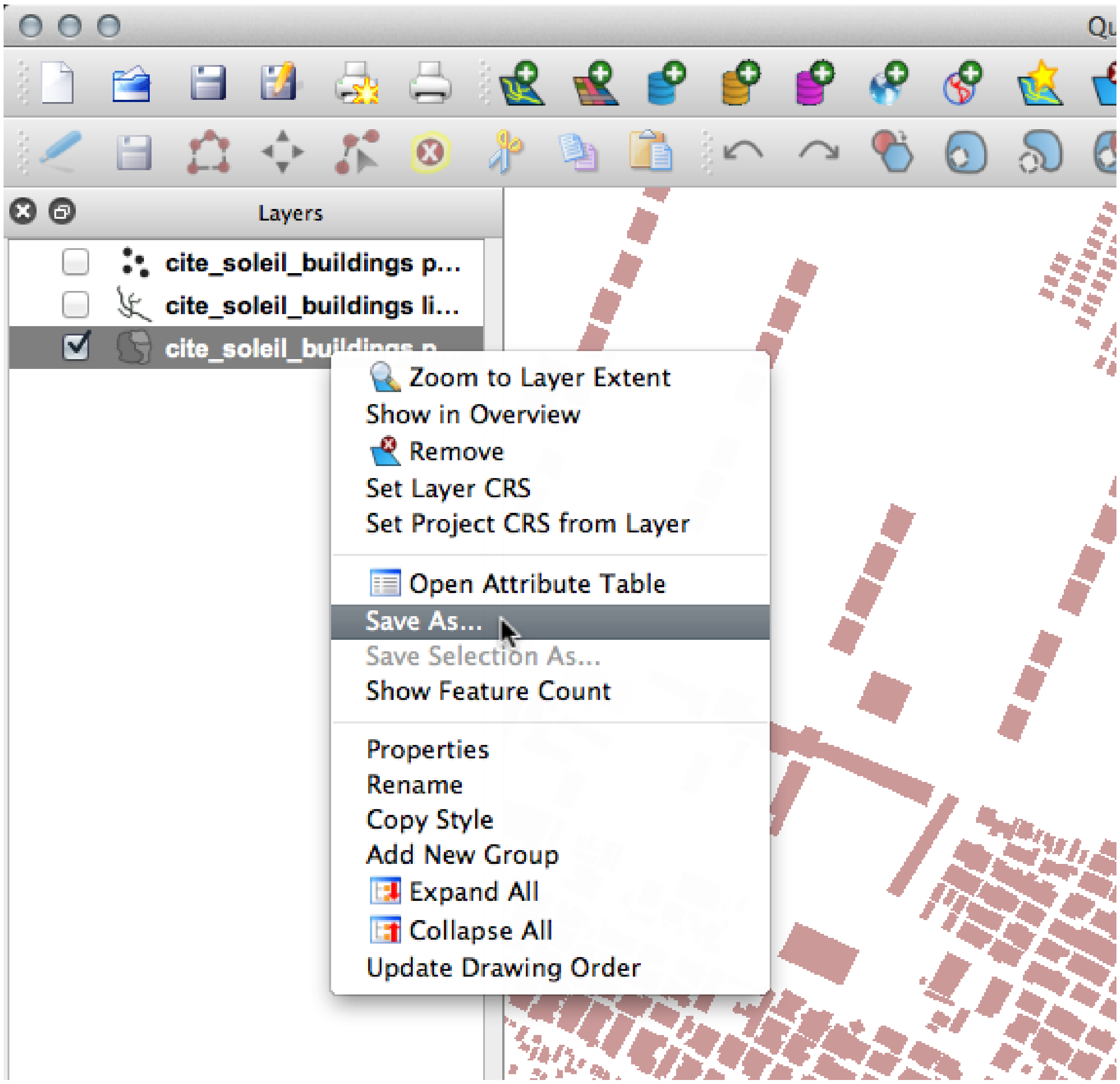
[✓] name
[ ] place
[ ] highway
[ ] landuse
[ ] waterway

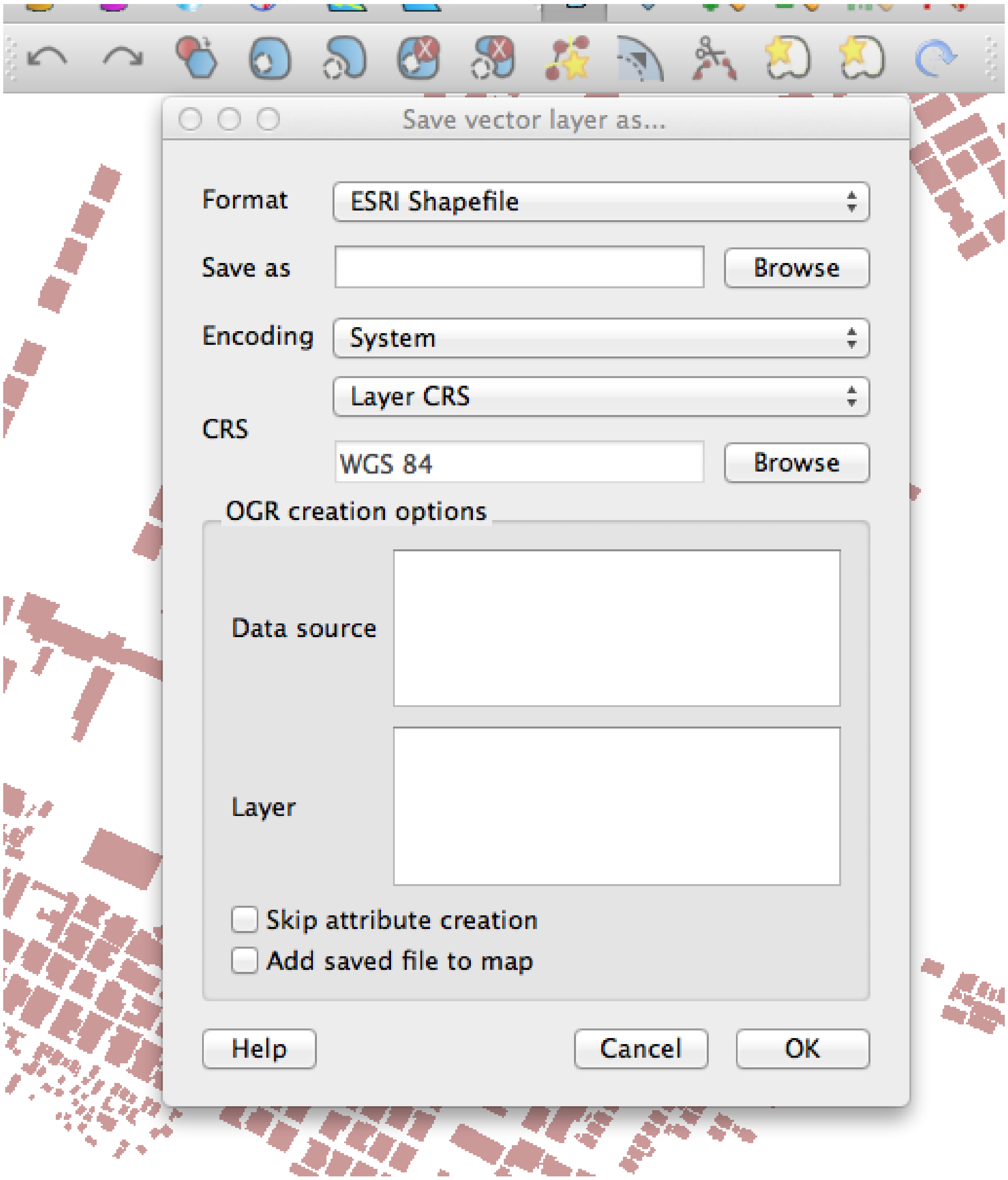
[ ] Replace current data (current layers will be removed)

[✓] Use custom renderer: Small scale


Quantum GIS 1.8.0-Lisboa
Layers
cite_soleil_buildings p...
cite_soleil_buildings li...
cite_soleil_buildings p...
Control rendering order
OSM Feature
Feature:
TYPE, ID:
CREATED:
USER:
Properties
Relations
Coordinate:
-72.336804,18.580410
Scale
1:2431
Render
EPSG:4326


Layers
cite_soleil_buildings p...
cite_soleil_buildings li...
cite_soleil_buildings p...


Layers
cite_soleil_buildings p...
cite_soleil_buildings li...
cite_soleil_buildings p
Zoom to Layer Extent
Show in Overview
Remove
Set Layer CRS
Set Project CRS from Layer
Open Attribute Table
Save As...
Save Selection As...
Show Feature Count
Properties
Rename
Copy Style
Add New Group
Expand All
Collapse All
Update Drawing Order



Format: ESRI Shapefile

Save as:

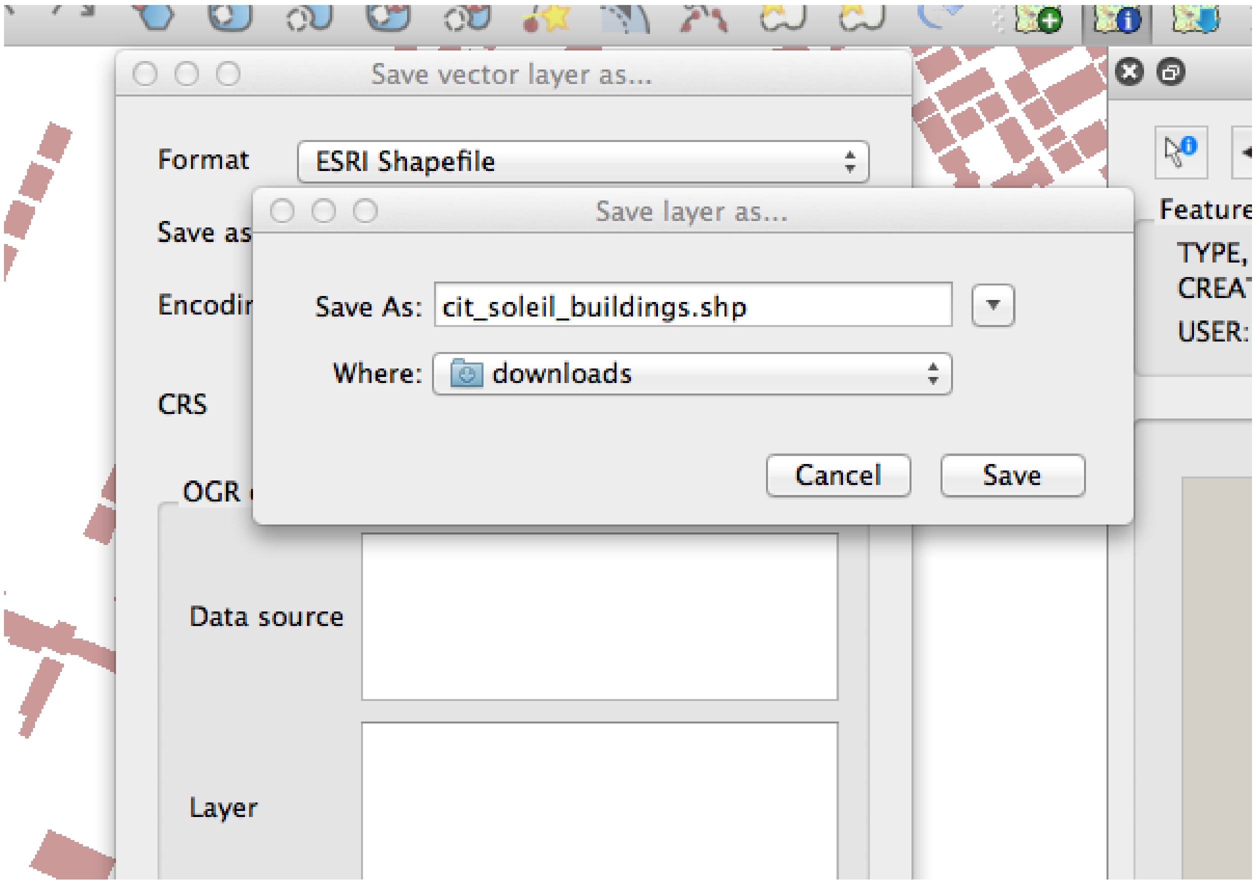
Encoding: System

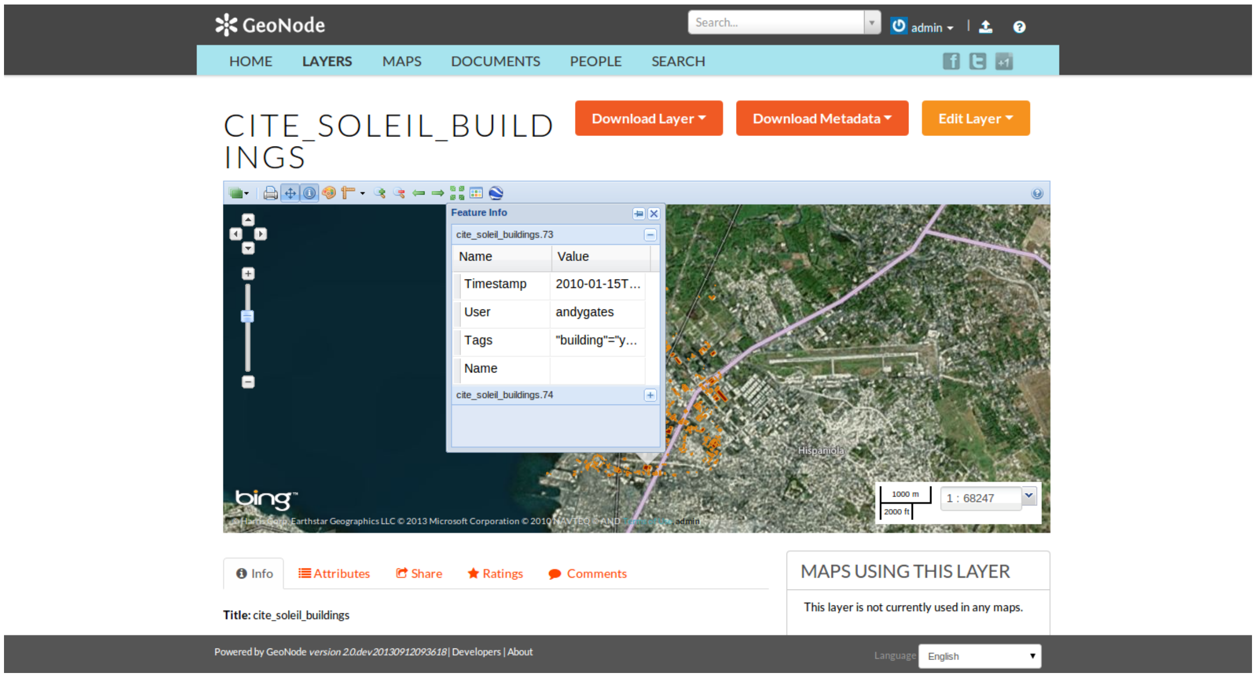
CRS: Layer CRS

WGS 84

Skip attribute creation [ ]

Add saved file to map [ ]


Save vector layer as...
Format
ESRI Shapefile
Save as
Encodin
CRS
OGR
Data source
Layer
Save layer as...
Save As:
cit_soleil_buildings.shp
Where:
downloads
Cancel
Save
Feature
TYPE,
CREA
USER:


GeoNode
Search...
admin
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
CITE_SOLEIL_BUILDINGS
Download Layer
Download Metadata
Edit Layer
Feature Info
cite_soleil_buildings.73
Name
Value
Timestamp
2010-01-15T...
User
andygates
Tags
"building"="y...
Name
cite_soleil_buildings.74
1 : 68247
Info
Attributes
Share
Ratings
Comments
MAPS USING THIS LAYER
This layer is not currently used in any maps.
Title: cite_soleil_buildings
Powered by GeoNode version 2.0.dev20130912093618 | Developers | About
Language
English

ogrinfo has detected 5 different geometric layers inside the osm data source. As we are just interested in the buildings, you will just export to a new shapefile the multipolygons layer using the GDAL ogr2ogr command utility:

```
$ ogr2ogr cite_soleil_buildings cite_soleil_buildings.osm multipolygons
```

Now you can upload the shapefile to GeoNode using the GeoNode Upload form in the same manner as you did in the previous section.

Usage of the GeoNode's Django Administration Panel

GeoNode has an administration panel based on the Django admin which can be used to do some database operations. Although most of the operations can and should be done through the normal GeoNode interface, the admin panel provides a quick overview and management tool over the database.

It should be highlighted that the sections not covered in this guide are meant to be managed through GeoNode.

Accessing the admin panel Only the staff users (including the superusers) can access the admin interface.

Note: User's staff membership can be set by the admin panel itself, see how in the *Manage users and groups through the admin panel* section.

The link to access the admin interface can be found by clicking in the upper right corner on the user name, see figure

Manage users and groups through the admin panel The admin section called Auth has the two links to access the Users and Groups sections, see figure

Users

Adding a user By clicking on the "add" link on the right of the Users link is possible to add a new users to the GeoNode site. A simple form asking for username and password will be presented, see figure

Upon clicking "save" a new form will be presented asking for some personal information and the rights the user should have.

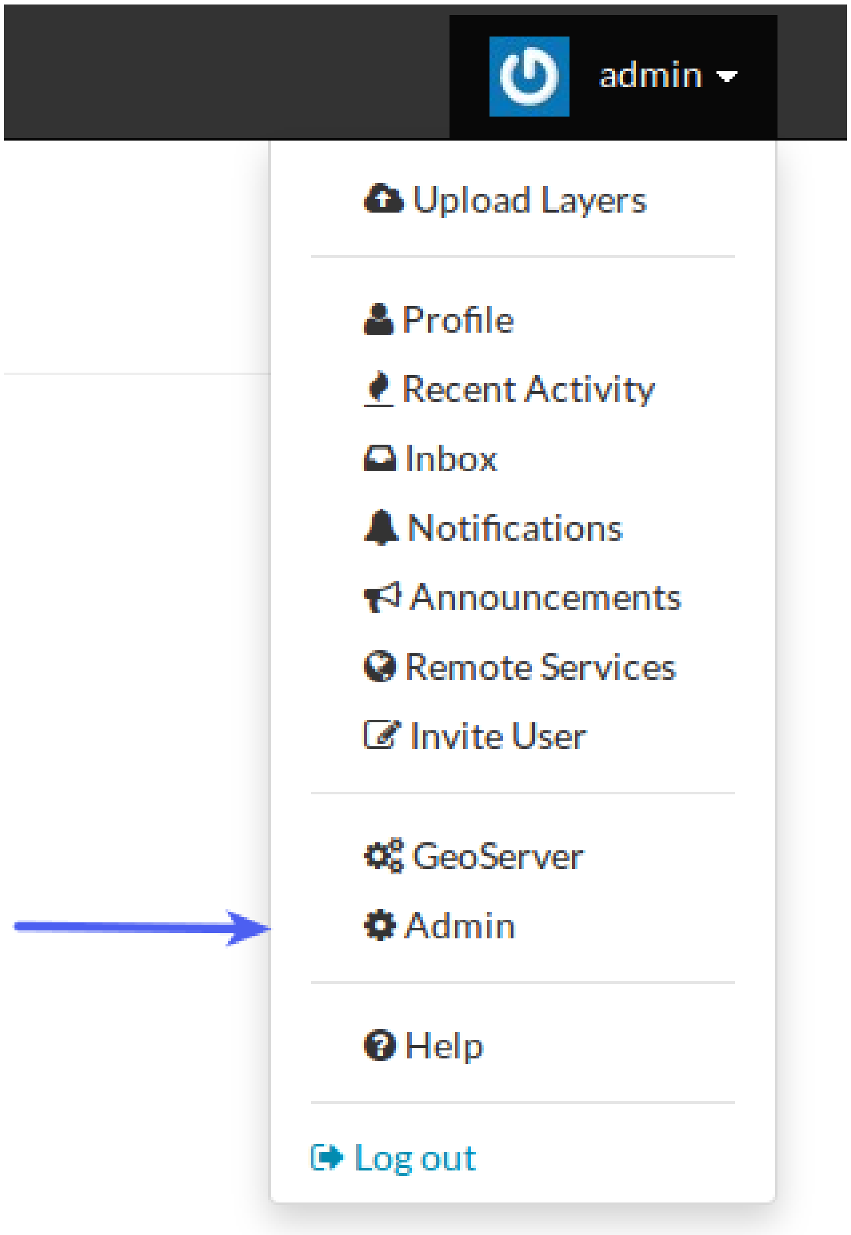
For a normal, not privileged user is enough to just fill the personal information and then confirm with "save".

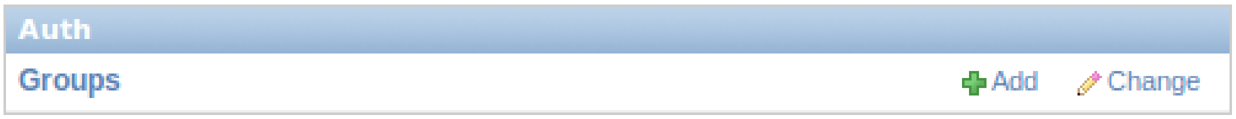
If the user has to access the admin panel or be a superuser it's enough just to tick the "staff" and "superuser" checkboxes.

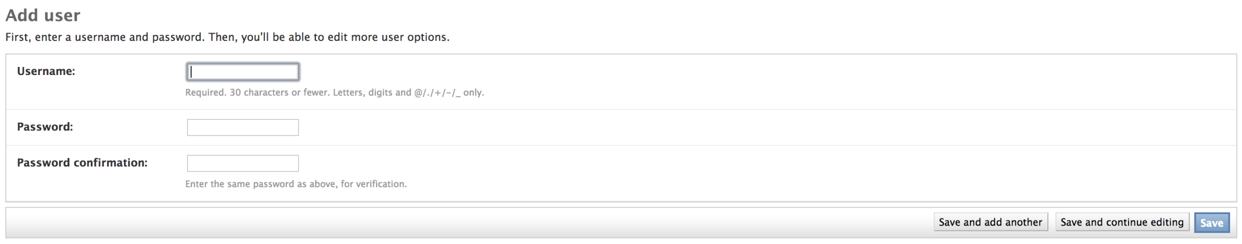
Changing a user To modify an existing user click on "Users" then on a username in the list. The same form will be presented.

Groups Although the "Groups" permissions system is not implemented yet in GeoNode is possible to create new groups with set of permissions which will be inherited by all the group members.

The creation and management of a Group is done in a very similar way that the user one.

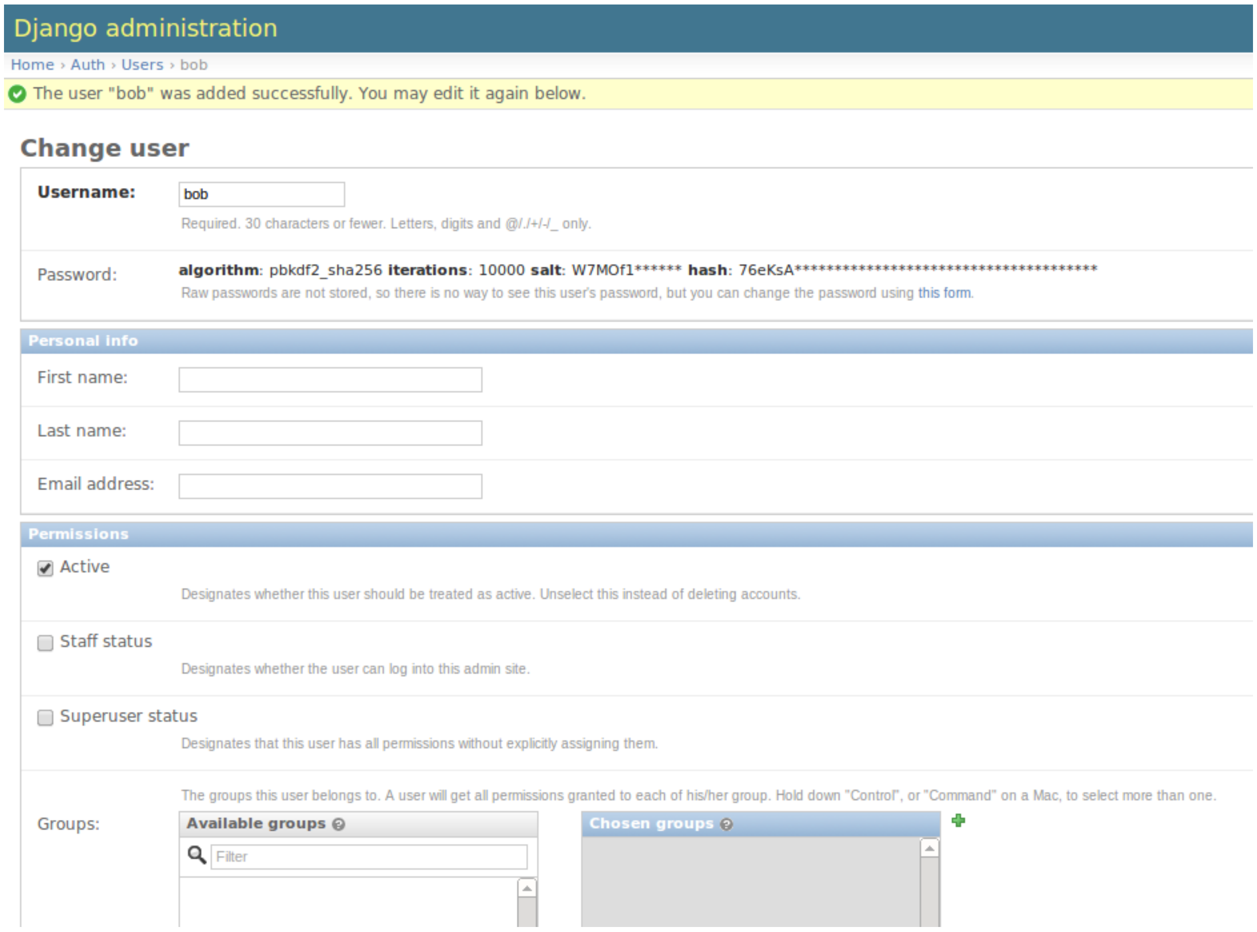





Add user

First, enter a username and password. Then, you'll be able to edit more user options.

Username:	Required. 30 characters or fewer. Letters, digits and @/./+/-/_ only.
Password:	
Password confirmation:	Enter the same password as above, for verification.



Django administration

Home › Auth › Users › bob

The user "bob" was added successfully. You may edit it again below.

Change user

Username:	bob Required. 30 characters or fewer. Letters, digits and @/./+/-/_ only.
Password:	**algorithm**: pbkdf2_sha256 **iterations**: 10000 **salt**: W7MOf1****** **hash**: 76eKsA************************************** Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

Personal info

First name:	
Last name:	
Email address:	

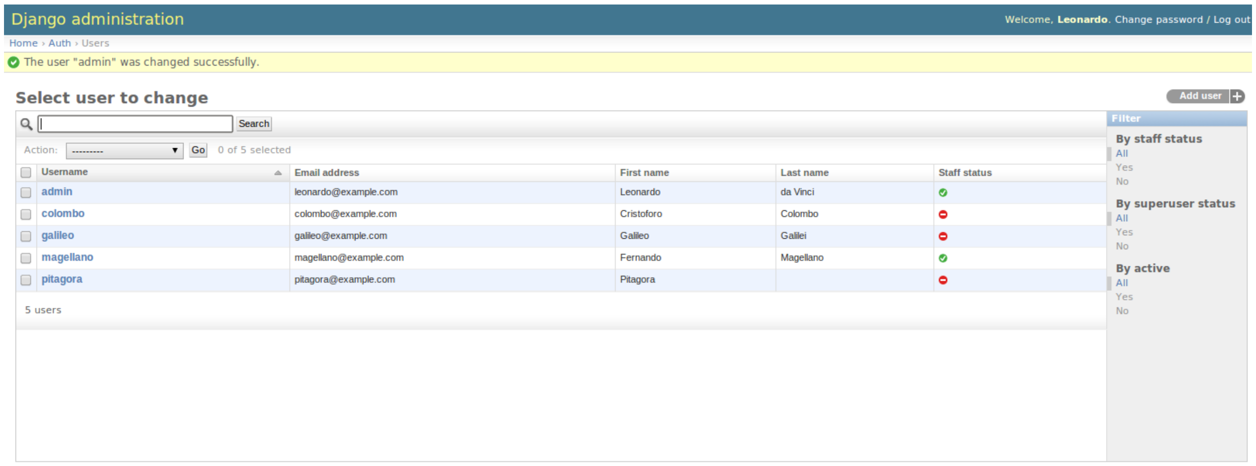
Permissions

[✓] Active
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

[ ] Staff status
Designates whether the user can log into this admin site.

[ ] Superuser status
Designates that this user has all permissions without explicitly assigning them.

Groups: The groups this user belongs to. A user will get all permissions granted to each of his/her group. Hold down "Control", or "Command" on a Mac, to select more than one.



Django administration

Welcome, **Leonardo**. Change password / Log out

Home › Auth › Users

The user "admin" was changed successfully.

Select user to change

0 of 5 selected

Username	Email address	First name	Last name	Staff status
admin	leonardo@example.com	Leonardo	da Vinci	✓
colombo	colombo@example.com	Cristoforo	Colombo	⛔
galileo	galileo@example.com	Galileo	Galilei	⛔
magellano	magellano@example.com	Fernando	Magellano	✓
pitagora	pitagora@example.com	Pitagora		⛔

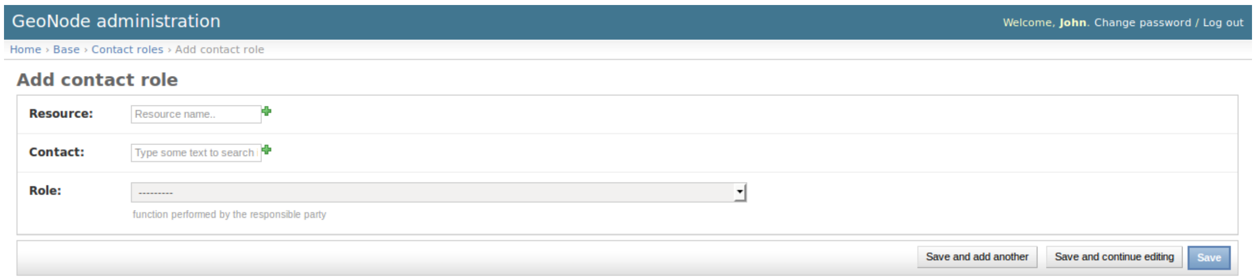
5 users

Manage profiles using the admin panel So far GeoNode implements two distinct roles, that can be assigned to resources such as layers, maps or documents:

- party who authored the resource
- party who can be contacted for acquiring knowledge about or acquisition of the resource

This two profiles can be set in the GeoNode interface by accessing the metadata page and setting the "Point of Contact" and "Metadata Author" fields respectively.

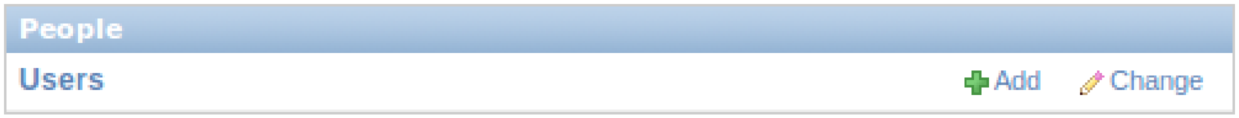
Is possible for an administrator to add new roles if needed, by clicking on the "Add Role" button in the "People" -> "Roles" section:


GeoNode administration
Welcome, John. Change password / Log out
Home › Base › Contact roles › Add contact role
Save and add another
Save and continue editing
Save

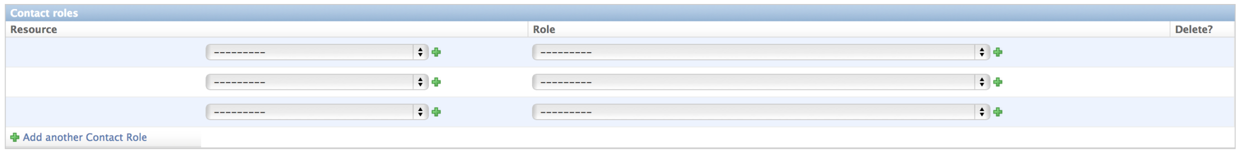
Add contact role

Resource:	Resource name..
Contact:	Type some text to search
Role:	--------- function performed by the responsible party

Clicking on the "People" -> "Profiles" section (see figure) will open a web for with some personal information plus a section called "Contact roles".



Is important that this last section is not modified here unless the administrator is very confident in that operation.



Contact roles

Resource	Role	Delete?
---------	---------	
---------	---------	
---------	---------	

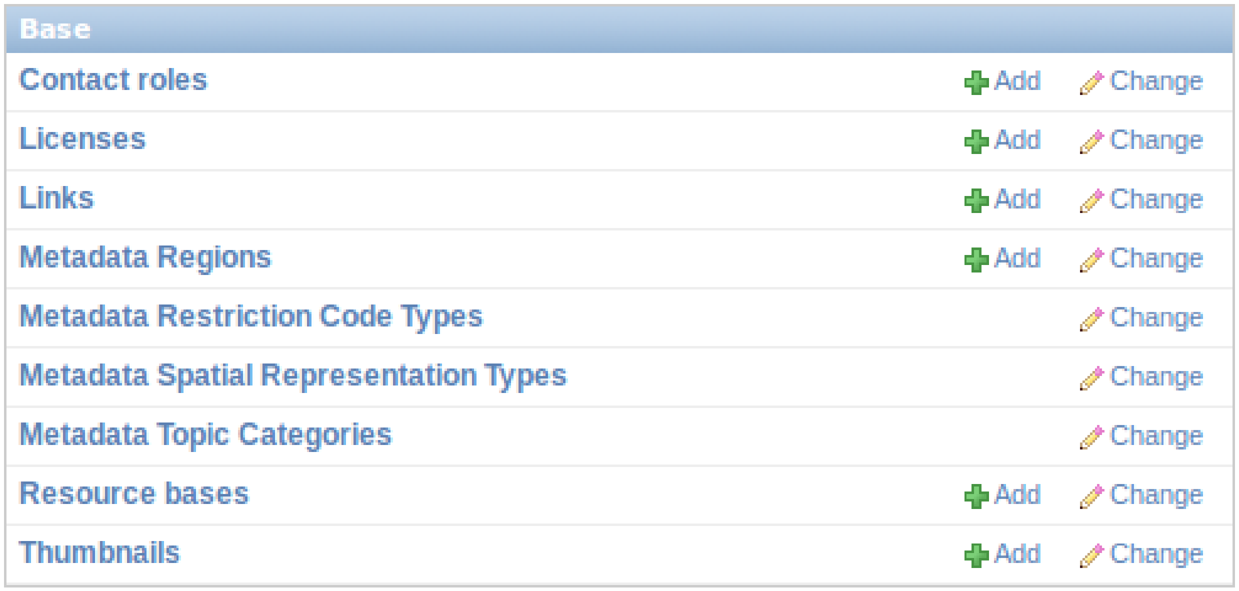
Manage the metadata categories using the admin panel In the "Base" section of the admin panel there are the links to manage the metadata categories used in GeoNode

The metadata categories are:

- Regions
- Restriction Code Types
- Spatial Representation Types
- Topic Categories

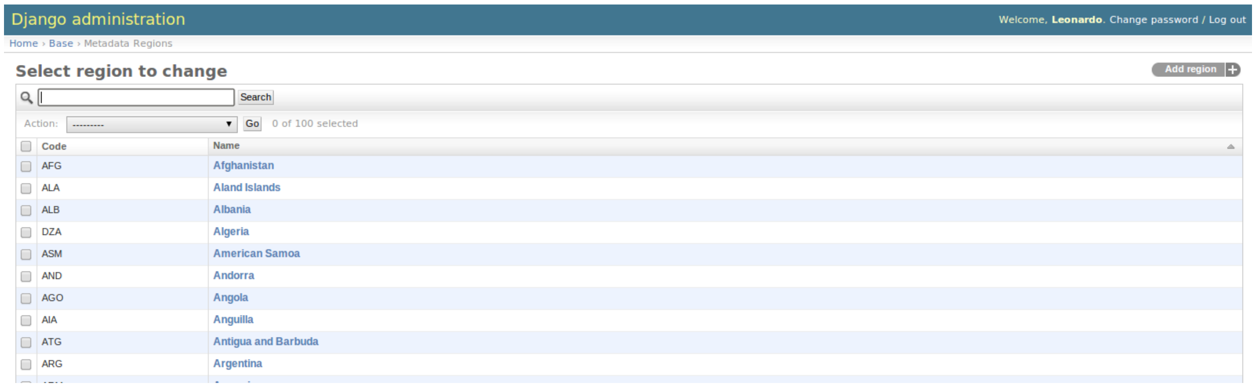
The other links available should not be used.

Regions The Regions can be updated, deleted and added on needs. Just after a GeoNode fresh installation the regions contain all of the world countries, identified by their ISO code.



Base

Contact roles	Add	Change
Licenses	Add	Change
Links	Add	Change
Metadata Regions	Add	Change
Metadata Restriction Code Types		Change
Metadata Spatial Representation Types		Change
Metadata Topic Categories		Change
Resource bases	Add	Change
Thumbnails	Add	Change



Django administration
Welcome, Leonardo. Change password / Log out
Home › Base › Metadata Regions
Add region
Search
Action: ---------
Go

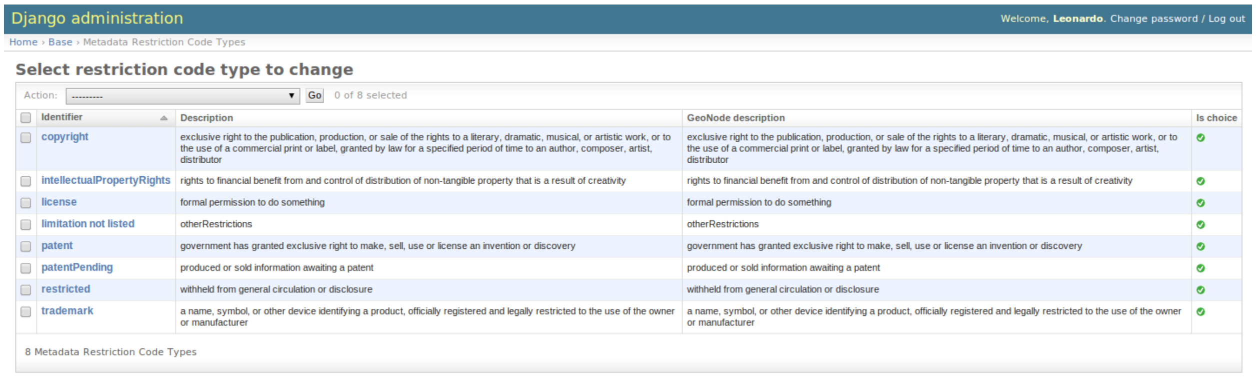
Select region to change

0 of 100 selected

Code	Name
AFG	Afghanistan
ALA	Aland Islands
ALB	Albania
DZA	Algeria
ASM	American Samoa
AND	Andorra
AGO	Angola
AIA	Anguilla
ATG	Antigua and Barbuda
ARG	Argentina

Restriction Code Types Being GeoNode strictly tied to the standards, the restrictions cannot be added/deleted or modified in their identifier. This behavior is necessary to keep the consistency in case of federation with the CSW catalogues.

The Restrictions GeoNode description field can in any case be modified is some kind of customisation is necessary, since it's just the string that will appear on the layer metadata page. If some of the restrictions are not needed within the GeoNode instance, it is possible to hide them by unchecking the "Is choice" field.



Django administration

Welcome, **Leonardo**. Change password / Log out

Home › Base › Metadata Restriction Code Types

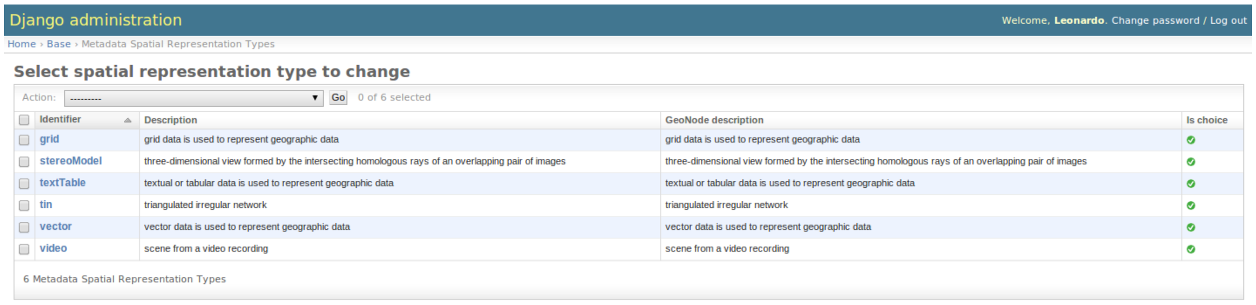
Select restriction code type to change

Action: --------- Go 0 of 8 selected

Identifier	Description	GeoNode description	Is choice
copyright	exclusive right to the publication, production, or sale of the rights to a literary, dramatic, musical, or artistic work, or to the use of a commercial print or label, granted by law for a specified period of time to an author, composer, artist, distributor	exclusive right to the publication, production, or sale of the rights to a literary, dramatic, musical, or artistic work, or to the use of a commercial print or label, granted by law for a specified period of time to an author, composer, artist, distributor	✓
intellectualPropertyRights	rights to financial benefit from and control of distribution of non-tangible property that is a result of creativity	rights to financial benefit from and control of distribution of non-tangible property that is a result of creativity	✓
license	formal permission to do something	formal permission to do something	✓
limitation not listed	otherRestrictions	otherRestrictions	✓
patent	government has granted exclusive right to make, sell, use or license an invention or discovery	government has granted exclusive right to make, sell, use or license an invention or discovery	✓
patentPending	produced or sold information awaiting a patent	produced or sold information awaiting a patent	✓
restricted	withheld from general circulation or disclosure	withheld from general circulation or disclosure	✓
trademark	a name, symbol, or other device identifying a product, officially registered and legally restricted to the use of the owner or manufacturer	a name, symbol, or other device identifying a product, officially registered and legally restricted to the use of the owner or manufacturer	✓

8 Metadata Restriction Code Types

Spatial Representation Types For this section the same concepts of the Restriction Code Types applies.



Django administration

Welcome, **Leonardo**. Change password / Log out

Home › Base › Metadata Spatial Representation Types

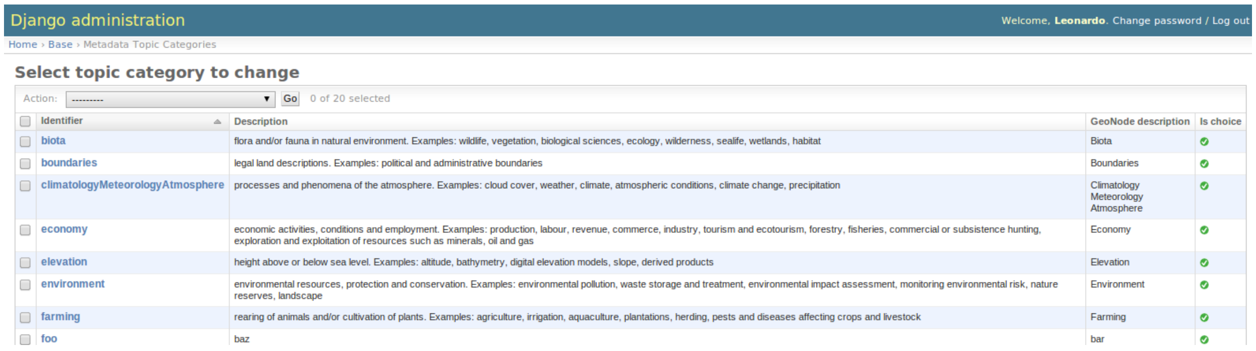
Select spatial representation type to change

Action: --------- Go 0 of 6 selected

Identifier	Description	GeoNode description	Is choice
grid	grid data is used to represent geographic data	grid data is used to represent geographic data	✓
stereoModel	three-dimensional view formed by the intersecting homologous rays of an overlapping pair of images	three-dimensional view formed by the intersecting homologous rays of an overlapping pair of images	✓
textTable	textual or tabular data is used to represent geographic data	textual or tabular data is used to represent geographic data	✓
tin	triangulated irregular network	triangulated irregular network	✓
vector	vector data is used to represent geographic data	vector data is used to represent geographic data	✓
video	scene from a video recording	scene from a video recording	✓

6 Metadata Spatial Representation Types

Topic Categories Also for the Topic Categories the only part editable is the GeoNode description. Being standard is assumed that every possible data type will fall under these category identifiers. If some of the categories are not needed within the GeoNode instance, it is possible to hide them by unchecking the "Is choice" field.



Django administration

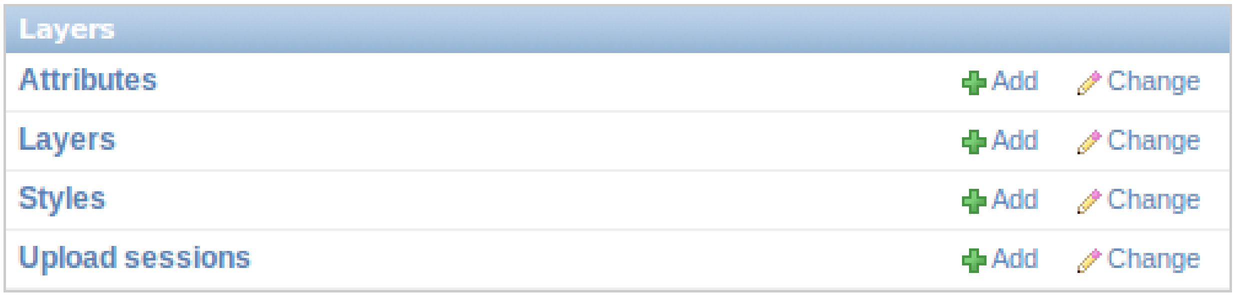
Welcome, **Leonardo**. Change password / Log out

Home › Base › Metadata Topic Categories

Select topic category to change

Action: --------- Go 0 of 20 selected

Identifier	Description	GeoNode description	Is choice
biota	flora and/or fauna in natural environment. Examples: wildlife, vegetation, biological sciences, ecology, wilderness, sealife, wetlands, habitat	Biota	✓
boundaries	legal land descriptions. Examples: political and administrative boundaries	Boundaries	✓
climatologyMeteorologyAtmosphere	processes and phenomena of the atmosphere. Examples: cloud cover, weather, climate, atmospheric conditions, climate change, precipitation	Climatology Meteorology Atmosphere	✓
economy	economic activities, conditions and employment. Examples: production, labour, revenue, commerce, industry, tourism and ecotourism, forestry, fisheries, commercial or subsistence hunting, exploration and exploitation of resources such as minerals, oil and gas	Economy	✓
elevation	height above or below sea level. Examples: altitude, bathymetry, digital elevation models, slope, derived products	Elevation	✓
environment	environmental resources, protection and conservation. Examples: environmental pollution, waste storage and treatment, environmental impact assessment, monitoring environmental risk, nature reserves, landscape	Environment	✓
farming	rearing of animals and/or cultivation of plants. Examples: agriculture, irrigation, aquaculture, plantations, herding, pests and diseases affecting crops and livestock	Farming	✓
foo	baz	bar	✓



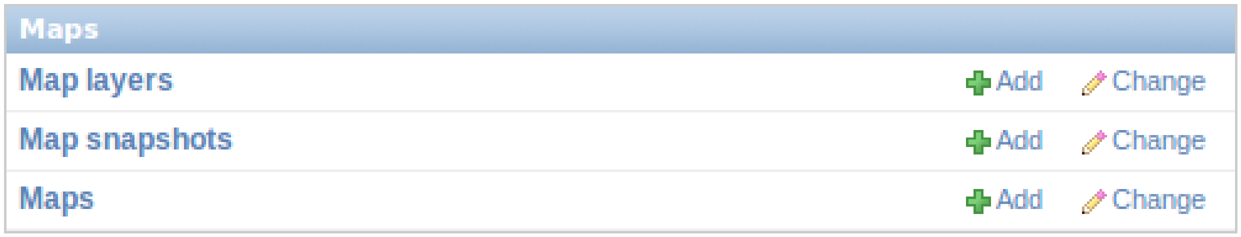
Layers		
Attributes	Add	Change
Layers	Add	Change
Styles	Add	Change
Upload sessions	Add	Change

Manage layers using the admin panel Some of the layers information can be edited directly through the admin interface although the best place is in the layer -> metadata page in GeoNode.

Is not recommended to modify the Attributes neither the Styles.

Clicking on the Layers link will present a list of layers. By selecting one of them is possible to modify some information like the metadata, the keywords etc. It's strongly recommended to limit the edits to the metadata and similar information.

Manage the maps using the admin panel Currently the maps admin panel allows more metadata options that the GeoNode maps metadata page. Thus is a good place where to add some more detailed information.

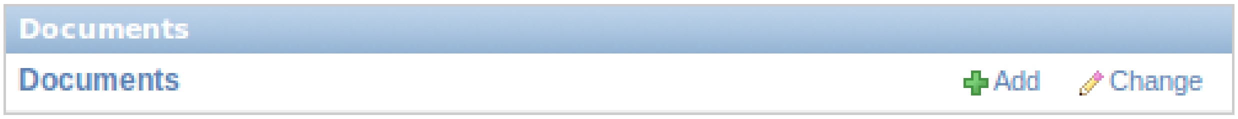


Maps		
Map layers	Add	Change
Map snapshots	Add	Change
Maps	Add	Change

The "Map Layers" section should not be used.

By clicking on a map in the maps list the metadata web form will be presented. Is possible to add or modify the information here. As for the layers, the more specific entries like the layers stack or the map coordinates should not be modified.

Manage the documents using the admin panel As for the layers, most of the information related to the documents can and should be modified using the GeoNode's document metadata page.



Documents		
Documents	Add	Change

Through the document detail page is possible to edit the metadata information. The fields related to the bounding box or the file attached should not be edited directly.

Django external applications

GeoNode is installed with some external django applications, suchs as:

- django-notification, to manage notification system
- django-announcements, to manage announcements
- agon-ratings, to manage content rating
- django-user-messages, to manage private message system
- django-avatar, to manage avatars
- dialogos, for managing the comment system

django-notification The notification system is enabled by default in INSTALLED_APPS. The GeoNode administrator should adjust some settings such as:

- DEFAULT_FROM_EMAIL, a Django general setting, will be the email used to send notifications
- NOTIFICATION_QUEUE_ALL, is set to False by default, therefore all the notifications are sent immediately. It is recommended for production to set this variable to True, and send the queued messages by using the emit_notices command using a scheduled task

Notification must rely on a notification backend. For testing purpose you may set the EMAIL_BACKEND variable to output emails to console window:

```
EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
```

It is possible to customize the notification messages by overriding the text files in the templates/notification directory.

For more information please refer to the django-notification official documentation: http://django-notification.readthedocs.org/en/latest/

django-announcements TODO...

agon-ratings TODO...

django-user-messages TODO...

django-avatar TODO...

dialogos TODO...

Management Commands for GeoNode

GeoNode comes with administrative commands to help with day to day tasks.

Below is the list of the ones that come from the GeoNode application, the full list can be obtained by doing:

```
geonode help
```

importlayers Imports a file or folder with geospatial files to GeoNode.

It supports data in Shapefile and GeoTiff format. It also picks up the styles if a `.sld` file is present.

Usage:

```
geonode importlayers <data_dir>
```

Additional options:

```
--user=USER           Name of the user account which should own the imported
                       layers
--keywords=KEYWORDS   The default keywords for the imported layer(s). Will
                       be the same for all imported layers if multiple
                       imports are done in one command
```

updatelayers Update the GeoNode application with data from GeoServer.

This is useful to add data in formats that are not supported in GeoNode by default, and for example to link it it to ArcSDE datastores. The updatelayers command provides several options that can be used to control how layer information is read from GeoServer and updated in GeoNode. Refer to 'Additional Options'.

Usage:

```
geonode updatelayers
```

Additional options:

```
--ignore-errors       Do not stop if errors are encountered.

--skip-unadvertised   Skip processing any layers that are marked 'advertised=False' in GeoServer

--skip-geonode-registered   Just processing GeoServer layers still not registered in GeoNode.
                      Useful if you are importing layers through GeoServer and not through GeoNode
                      and you don't want to run updatelayers on every GeoServer layer but just on the

--remove-deleted      Remove layers from GeoNode that have been deleted (or marked unavailable) from
                       Note: this option can be combined with --workspace and --store to only check
                       for deleted layers in a particular workspace or store.
                       It can also be combined with --skip-unadvertised, in which case any layers mar
                       'advertised=False' will be removed from GeoNode
                       Also, --remove-deleted does not consider the --filter parameter in determining
                       GeoServer layers to be deleted.  When --filter is combined with --remove-delet
                       filter will be applied to layers to update, but layers that do not match the
                       not be marked for deletion from GeoNode.

-u
--user                 Name of the GeoNode user account that should own the imported layers.

-f
--filter               Only update layers from GeoServer that match the given filter prefix.

-s
--store                Only update layers for the given GeoServer store name.

-w
--workspace            Only update layers for the given GeoServer workspace name.
```

emit_notices Sends email notices, for example after layers have been uploaded.

It should be configured as a cronjob, running for example every 5 minutes, otherwise the notices will be created (available in the system) but not sent. It requires the email gateway to be set correctly.

Usage:

```
geonode emit_notices
```

fixsitename Uses SITENAME and SITEURL to set the values of the default site object.

This information is used in the page titles and when sending emails from GeoNode, for example, new registrations.

Usage:

```
geonode fixsitename
```

Configuring Alternate CSW Backends

pycsw is the default CSW server implementation provided with GeoNode. This section will explain how to configure GeoNode to operate against alternate CSW server implementations.

Supported CSW server implementations GeoNode additionally supports the following CSW server implementations:

- GeoNetwork opensource
- deegree

Since GeoNode communicates with alternate CSW configurations via HTTP, the CSW server can be installed and deployed independent of GeoNode if desired.

Installing the CSW

GeoNetwork opensource Installation

- Deploy GeoNetwork opensource by downloading geonetwork.war (see http://geonetwork-opensource.org/downloads.html) and deploying into your servlet container
- Follow the instructions at http://geonetwork-opensource.org/manuals/2.6.4/eng/users/quickstartguide/installing/index.html to complete the installation
- test the server with a GetCapabilities request (http://localhost:8080/geonetwork/srv/en/csw?service=CSW&version=2.0.2&request

See http://geonetwork-opensource.org/docs.html for further documentation.

deegree Installation

- Deploy deegree by downloading the deegree3 cswDemo .war (see http://wiki.deegree.org/deegreeWiki/DownloadPage) and deploying into your servlet container
- Create a PostGIS-enabled PostgreSQL database
- Follow the instructions at http://wiki.deegree.org/deegreeWiki/deegree3/CatalogueService#Run_your_own_installation to complete the installation
- test the server with a GetCapabilities request (http://localhost:8080/deegree-csw-demo-3.0.4/services?service=CSW&version=2.0.2&request=GetCapabilities)

See http://wiki.deegree.org/deegreeWiki/deegree3/CatalogueService for further documentation.

Customizing GeoNode CSW configuration At this point, the CSW alternate backend is ready for GeoNode integration. GeoNode's CSW configuration (in `geonode/settings.py`) must be updated to point to the correct CSW. The example below exemplifies GeoNetwork as an alternate CSW backend:

```
# CSW settings
CATALOGUE = {
    'default': {
        # The underlying CSW implementation
        # default is pycsw in local mode (tied directly to GeoNode Django DB)
        #'ENGINE': 'geonode.catalogue.backends.pycsw_local',
        # pycsw in non-local mode
        #'ENGINE': 'geonode.catalogue.backends.pycsw',
        # GeoNetwork opensource
        'ENGINE': 'geonode.catalogue.backends.geonetwork',
        # deegree and others
        #'ENGINE': 'geonode.catalogue.backends.generic',

        # The FULLY QUALIFIED base url to the CSW instance for this GeoNode
        #'URL': '%scatalogue/csw' % SITEURL,
        'URL': 'http://localhost:8080/geonetwork/srv/en/csw',
        #'URL': 'http://localhost:8080/deegree-csw-demo-3.0.4/services',

        # login credentials (for GeoNetwork)
        'USER': 'admin',
        'PASSWORD': 'admin',
    }
}
```

Customize the look and feel

Warning: These instructions are only valid if you've installed GeoNode systemwide using apt-get!!

Note: If you've installed GeoNode in developing mode and now want to theme your GeoNode use the instructions from *Customized GeoNode Projects* (Developers Workshop).

You might want to change the look of GeoNode, editing the colors and the logo of the website and adjust the templates for your needs. To do so, you first have to set up your own geonode project from a template. If you've successfully done this, you can go further and start theming your geonode project.

Setup steps

Warning: These instructions are only valid if you've installed GeoNode systemwide using apt-get!!

If you are working remotely, you should first connect to the machine that has your GeoNode installation. You will need to perform the following steps in a directory where you intend to keep your newly created project.

```
$ django-admin startproject my_geonode --template=https://github.com/GeoNode/geonode-project/archive/
$ sudo pip install -e my_geonode
```

Note: You should NOT use the name *geonode* for your project as it will conflict with your default geonode package name.

These commands create a new template based on the geonode example project.

Rename the local_settings.py.sample to local_settings.py and edit it's content by setting the SITEURL and SITENAME. This file will be your main settings file for your project. It inherits all the settings from the original one plus you can override the ones that you need.

Note: In order for the edits to the local_settings.py file to take effect, you have to restart apache.

Edit the file /etc/apache2/sites-available/geonode and change the following directive from:

```
WSGIScriptAlias / /var/www/geonode/wsgi/geonode.wsgi
```

to:

```
WSGIScriptAlias / /path/to/my_geonode/my_geonode/wsgi.py
```

Then restart apache

```
$ sudo service apache2 restart
```

Now you can edit the templates in my_geonode/templates, the css and images to match your needs like shown in <Theming your Geonode>_. #Link!

Note: After going through the theming guide you'll have to return to this site to execute one more command in order to finish the theming!

When you've done the changes, run the following command in the *my_geonode* folder:

```
$ python manage.py collectstatic
```

And now you should see all the changes you've made to your GeoNode.

Source code revision control It is recommended that you immediately put your new project under source code revision control. The GeoNode development team uses Git and GitHub and recommends that you do the same. If you do not already have a GitHub account, you can easily set one up. A full review of Git and distributed source code revision control systems is beyond the scope of this tutorial, but you may find the Git Book useful if you are not already familiar with these concepts.

1. Create a new repository in GitHub. You should use the GitHub user interface to create a new repository for your new project.
2. Initialize your own repository in the my_geonode folder:

```
    $ git init
```

3. Add the remote repository reference to your local git configuration:

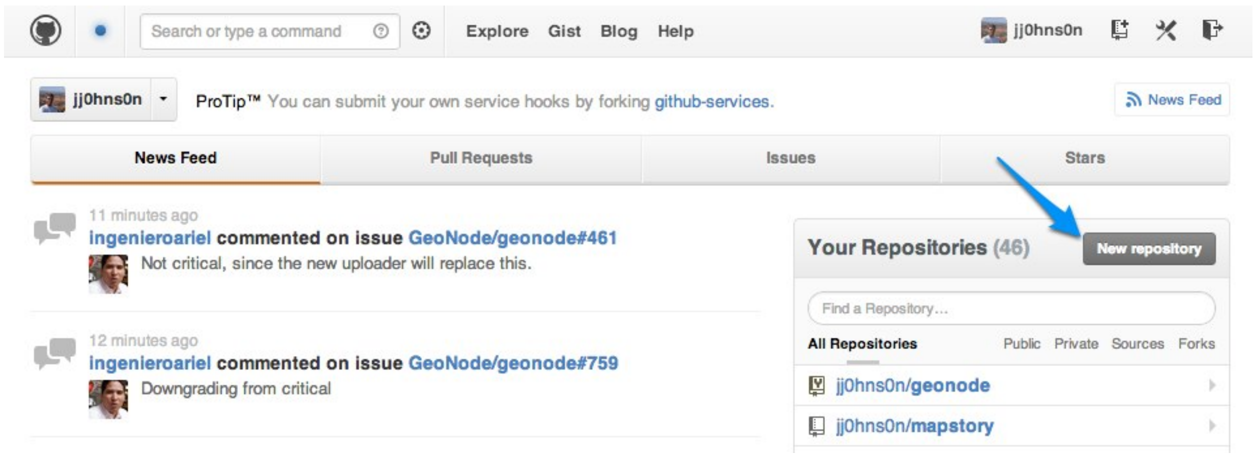
```
    $ git remote add
```

4. Add your project files to the repository:

```
    $ git add .
```

5. Commit your changes:

```
    $ git commit -am "Initial commit"
```



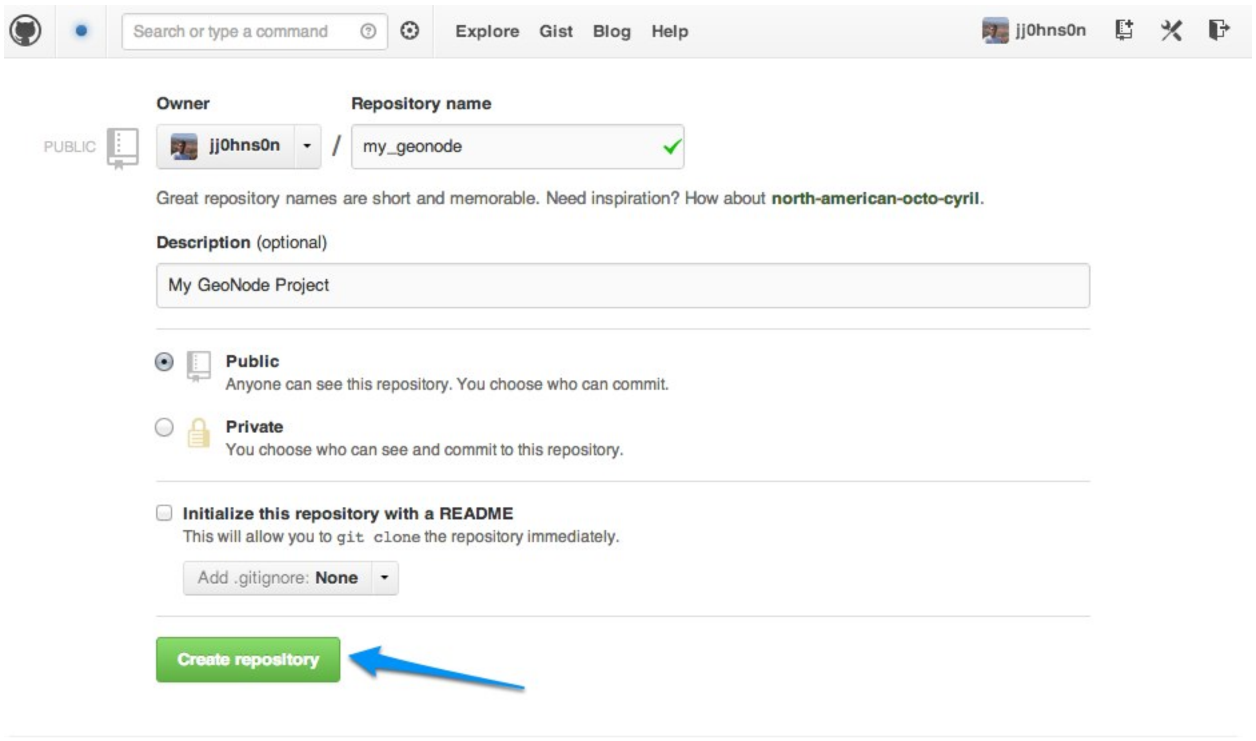
jj0hns0n ProTip™ You can submit your own service hooks by forking github-services.

11 minutes ago
ingenieroariel commented on issue GeoNode/geonode#461
Not critical, since the new uploader will replace this.

12 minutes ago
ingenieroariel commented on issue GeoNode/geonode#759
Downgrading from critical

Your Repositories (46)
jj0hns0n/geonode
jj0hns0n/mapstory

Fig. 3.69: *Creating a new GitHub Repository From GitHub's Homepage*



Owner: jj0hns0n / Repository name: my_geonode

Great repository names are short and memorable. Need inspiration? How about north-american-octo-cyril.

Description (optional): My GeoNode Project

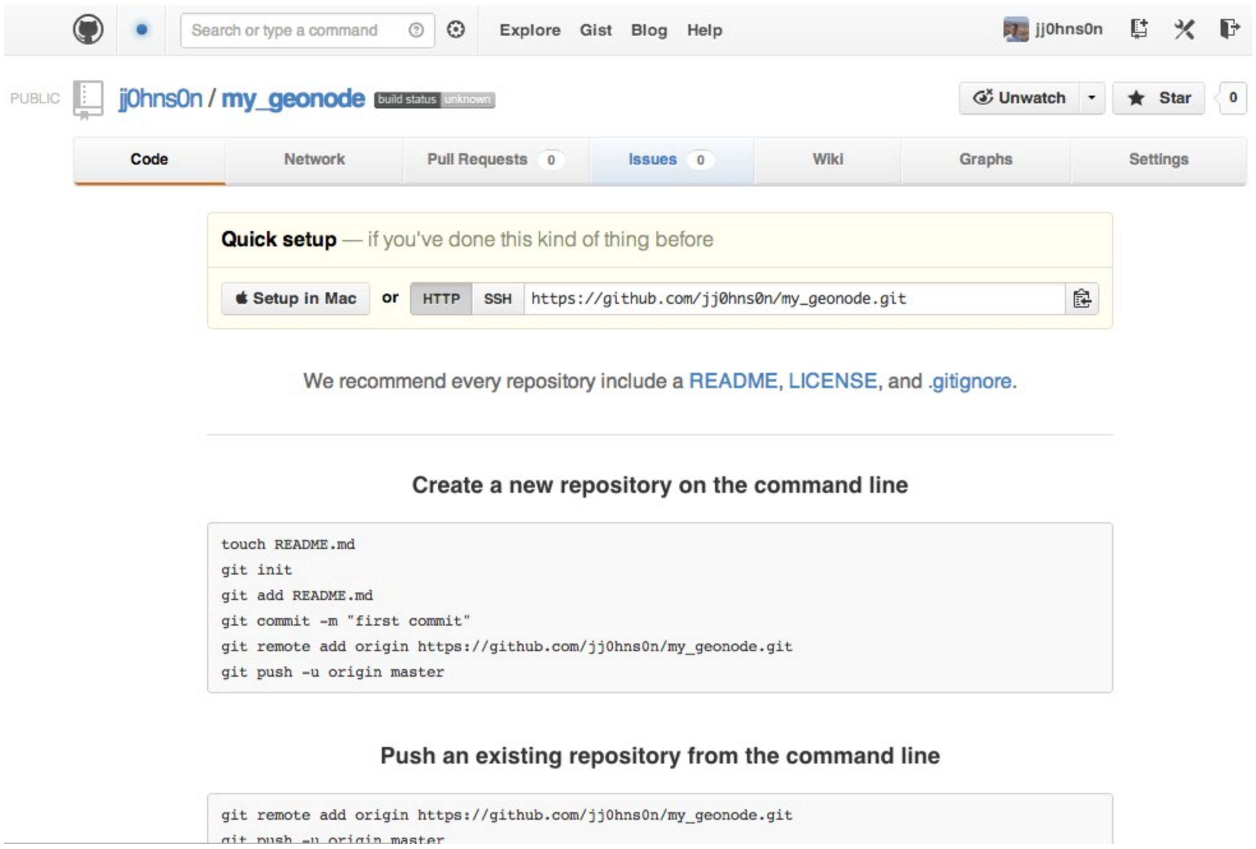
[✓ Public] Anyone can see this repository. You choose who can commit.

Private — You choose who can see and commit to this repository.

☐ Initialize this repository with a README
This will allow you to `git clone` the repository immediately.

Add .gitignore: None

Fig. 3.70: *Specifying new GitHub Repository Parameters*



Quick setup — if you've done this kind of thing before

https://github.com/jj0hns0n/my_geonode.git

We recommend every repository include a README, LICENSE, and .gitignore.

Create a new repository on the command line

```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/jj0hns0n/my_geonode.git
git push -u origin master
```

Push an existing repository from the command line

```
git remote add origin https://github.com/jj0hns0n/my_geonode.git
git push -u origin master
```

Fig. 3.71: *Your new Empty GitHub Repository*

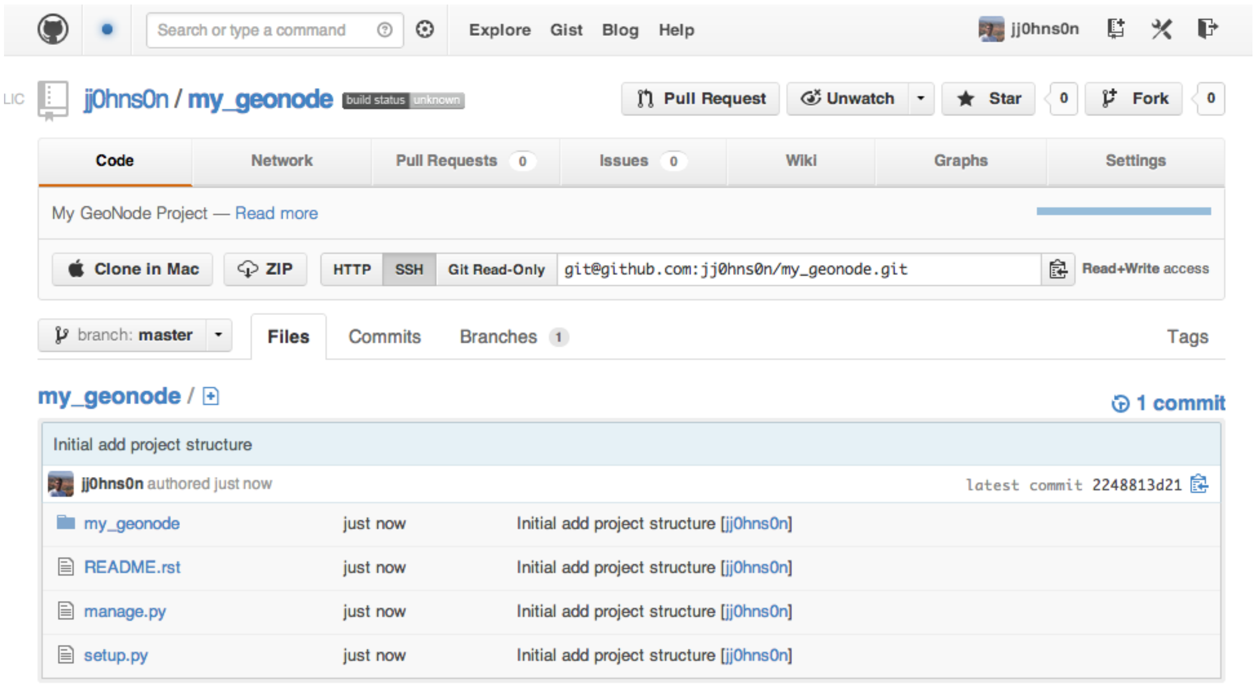
6. Push to the remote repository:

```
$ git push origin master
```

Project structure Your GeoNode project will now be structured as depicted below:

```
|-- README.rst
|-- manage.py
|-- my_geonode
|   |-- __init__.py
|   |-- settings.py
|   |-- local_settings.py
|   |-- static
|   |   |-- README
|   |   |-- css
|   |   |   |-- site_base.css
|   |   |-- img
|   |   |   |-- README
|   |   |-- js
|   |       |-- README
|   |-- templates
|   |   |-- site_base.html
|   |   |-- site_index.html
|   |-- urls.py
|   |-- wsgi.py
|-- setup.py
```

You can also view your project on GitHub.



My GeoNode Project — Read more

git@github.com:jj0hns0n/my_geonode.git

my_geonode /

Initial add project structure

jj0hns0n authored just now latest commit 2248813d21

my_geonode	just now	Initial add project structure [jj0hns0n]
README.rst	just now	Initial add project structure [jj0hns0n]
manage.py	just now	Initial add project structure [jj0hns0n]
setup.py	just now	Initial add project structure [jj0hns0n]

Fig. 3.72: *Viewing your project on GitHub*

Each of the key files in your project are described below.

manage.py `manage.py` is the main entry point for managing your project during development. It allows running all the management commands from each app in your project. When run with no arguments, it will list all of the management commands.

settings.py `settings.py` is the primary settings file for your project. It imports the settings from the system geonode and adds the local paths. It is quite common to put all sensible defaults here and keep deployment specific configuration in the `local_settings.py` file. All of the possible settings values and their meanings are detailed in the Django documentation.

A common paradigm for handing 'local settings' (and in other areas where some python module may not be available) is:

try: from local_settings import *

except: pass

This is not required and there are many other solutions to handling varying deployment configuration requirements.

urls.py `urls.py` is where your application specific URL routes go. Additionally, any *overrides* can be placed here, too.

wsgi.py This is a generated file to make deploying your project to a WSGI server easier. Unless there is very specific configuration you need, `wsgi.py` can be left alone.

setup.py There are several packaging options in python but a common approach is to place your project metadata (version, author, etc.) and dependencies in `setup.py`.

This is a large topic and not necessary to understand while getting started with GeoNode development but will be important for larger projects and to make development easier for other developers.

More: http://docs.python.org/2/distutils/setupscript.html

static The `static` directory will contain your fixed resources: css, html, images, etc. Everything in this directory will be copied to the final media directory (along with the *static* resources from other apps in your project).

templates All of your projects templates go in the `templates` directory. While no organization is required for your project specific templates, when overriding or replacing a template from another app, the path must be the same as the template to be replaced.

Staying in sync with mainline GeoNode One of the primary reasons to set up your own GeoNode project using this method is so that you can stay in sync with the mainline GeoNode as the core development team makes new releases. Your own project should not be adversely affected by these changes, but you will receive bug fixes and other improvements by staying in sync.

Upgrade GeoNode:

```
$ apt-get update
$ apt-get install geonode
```

Verify that your new project works with the upgraded GeoNode:

```
$ python manage.py runserver
```

Navigate to http://localhost:8000.

Theming your GeoNode project There are a range of options available to you if you want to change the default look and feel of your GeoNode project. Since GeoNode's style is based on Bootstrap you will be able to make use of all that Bootstrap has to offer in terms of theme customization. You should consult Bootstrap's documentation as your primary guide once you are familiar with how GeoNode implements Bootstrap and how you can override GeoNode's theme and templates in your own project.

Logos and graphics GeoNode intentionally does not include a large number of graphics files in its interface. This keeps page loading time to a minimum and makes for a more responsive interface. That said, you are free to customize your GeoNode's interface by simply changing the default logo, or by adding your own images and graphics to deliver a GeoNode experience the way you envision int.

Your GeoNode project has a directory already set up for storing your own images at `<my_geonode>/static/img`. You should place any image files that you intend to use for your project in this directory.

Let's walk through an example of the steps necessary to change the default logo.

1. Change into the `img` directory:

```
$ cd <my_geonode>/static/img
```

2. If you haven't already, obtain your logo image. The URL below is just an example, so you will need to change this URL to match the location of your file or copy it to this location:

```
$ wget http://www2.sta.uwi.edu/~anikov/UWI-logo.JPG
$ cd ../../..
```

3. Override the CSS that displays the logo by editing `<my_geonode>/static/css/site_base.css` with your favorite editor and adding the following lines, making sure to update the width, height, and URL to match the specifications of your image.

```
.nav-logo {
    width: 373px;
    height: 79px;
    background: url(../img/UWI-logo.JPG) no-repeat;
}
```

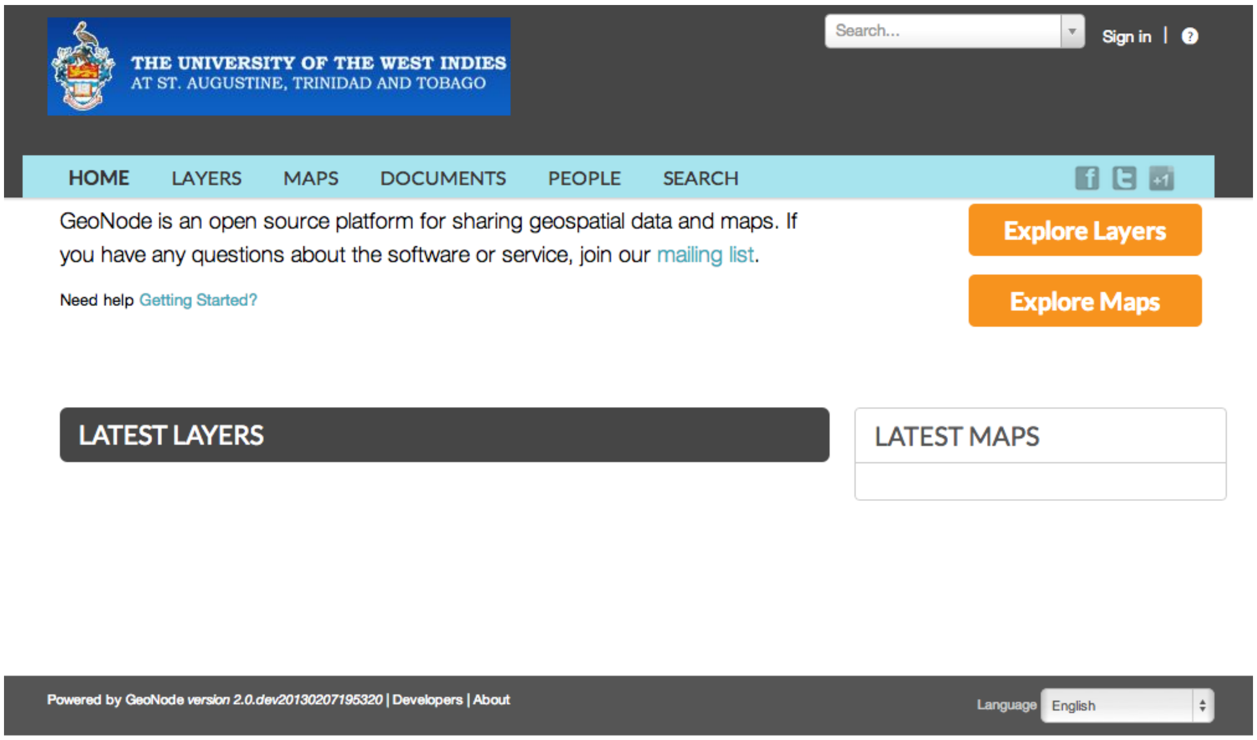
4. Restart your GeoNode project and look at the page in your browser:

```
$ python manage.py runserver
```

Visit your site at http://localhost:8000/ or the remote URL for your site.

You can see that the header has been expanded to fit your graphic. In the following sections you will learn how to customize this header to make it look and function the way you want.

Note: You should commit these changes to your repository as you progress through this section, and get in the habit of committing early and often so that you and others can track your project on GitHub. Making many atomic commits and staying in sync with a remote repository makes it easier to collaborate with others on your project.



GeoNode is an open source platform for sharing geospatial data and maps. If you have any questions about the software or service, join our mailing list.

Need help Getting Started?

Powered by GeoNode version 2.0.dev20130207195320 | Developers | About

Fig. 3.73: *Custom logo*

Cascading Style Sheets In the last section you already learned how to override GeoNode's default CSS rules to include your own logo. You are able to customize any aspect of GeoNode's appearance this way. In the last screenshot, you saw that the main area in the homepage is covered up by the expanded header.

First, we'll walk through the steps necessary to displace it downward so it is no longer hidden, then change the background color of the header to match the color in our logo graphic.

1. Reopen `<my_geonode>/static/css/site_base.css` in your editor and add the following rule after the one added in the previous step:

```
.content-wrap {
    margin: 75px 75px;
}
```

2. Add a rule to change the background color of the header to match the logo graphic we used:

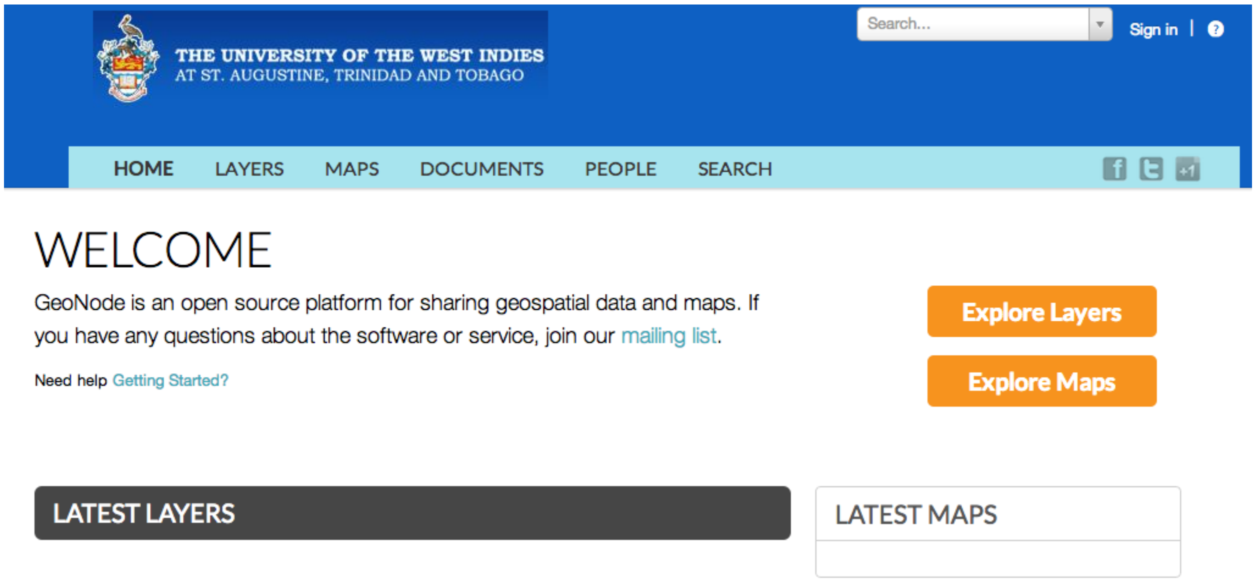
```
.navbar .navbar-inner {
    background: #0e60c3;
}
```

3. Your project CSS file should now look like this:

```
.nav-logo {
    width: 373px;
    height: 79px;
    background: url(../img/UWI-logo.JPG) no-repeat;
}

.content-wrap {
    margin: 75px 75px;
    }

    .navbar .navbar-inner {
        background: #0e60c3;
    }
```

4. Restart the development server and reload the page:

```
    $ python manage.py runserver
```



WELCOME

GeoNode is an open source platform for sharing geospatial data and maps. If you have any questions about the software or service, join our mailing list.

Need help Getting Started?

Fig. 3.74: *CSS overrides*

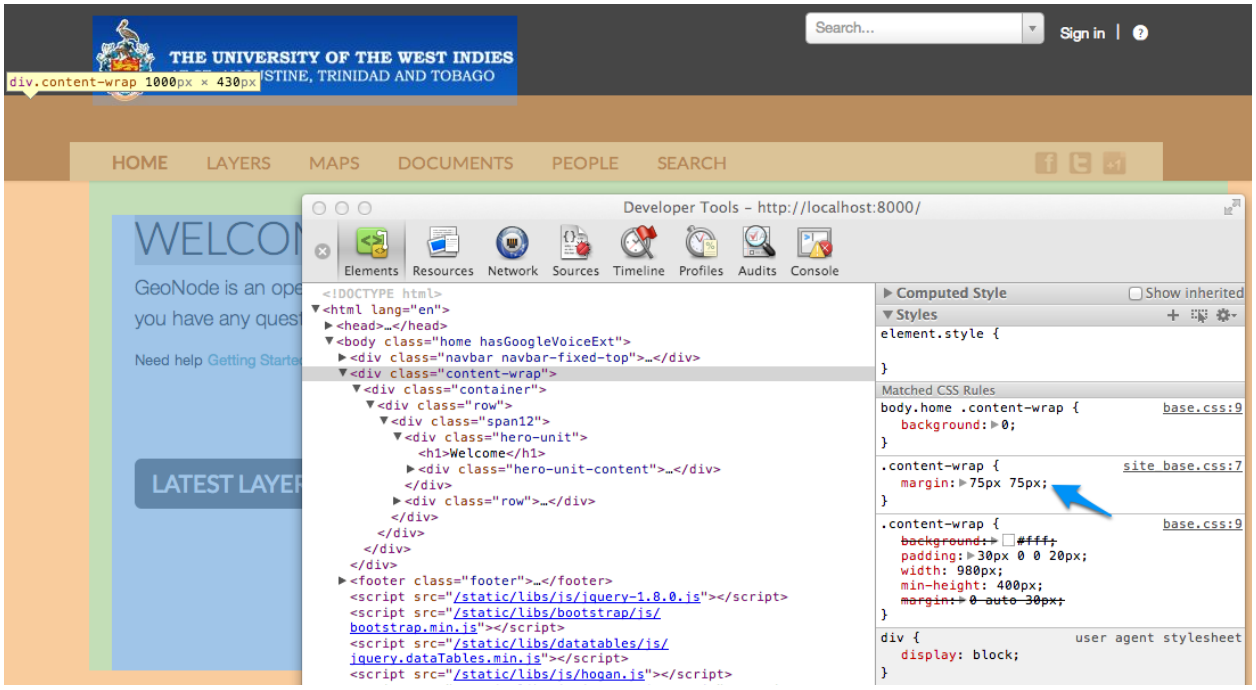
Note: You can continue adding rules to this file to override the styles that are in the GeoNode base CSS file which is built from base.less. You may find it helpful to use your browser's development tools to inspect elements of your site that you want to override to determine which rules are already applied. See the screenshot below. Another section of this workshop covers this topic in much more detail.

Templates and static pages Now that we have changed the default logo and adjusted our main content area to fit the expanded header, the next step is to update the content of the homepage itself. Your GeoNode project includes two basic templates that you will use to change the content of your pages.

The file `site_base.html` (in `<my_geonode>/templates/`) is the basic template that all other templates inherit from and you will use it to update things like the header, navbar, site-wide announcement, footer, and also to include your own JavaScript or other static content included in every page in your site. It's worth taking a look at GeoNode's base file on GitHub. You have several blocks available to you to for overriding, but since we will be revisiting this file in future sections of this workshop, let's just look at it for now and leave it unmodified.

Open `<my_geonode>/templates/site_base.html` in your editor:

```
{% extends "base.html" %}
{% block extra_head %}
    <link href="{{ STATIC_URL }}css/site_base.css" rel="stylesheet"/>
{% endblock %}
```



```
<!DOCTYPE html>
▼<html lang="en">
  ▶<head>…</head>
  ▼<body class="home hasGoogleVoiceExt">
    ▶<div class="navbar navbar-fixed-top">…</div>
    ▼<div class="content-wrap">
      ▼<div class="container">
        ▼<div class="row">
          ▼<div class="span12">
            ▼<div class="hero-unit">
                <h1>Welcome</h1>
              ▶<div class="hero-unit-content">…</div>
              </div>
            ▶<div class="row">…</div>
            </div>
          </div>
        </div>
      </div>
    ▶<footer class="footer">…</footer>
      <script src="/static/libs/js/jquery-1.8.0.js"></script>
      <script src="/static/libs/bootstrap/js/
      bootstrap.min.js"></script>
      <script src="/static/libs/datatables/js/
      jquery.dataTables.min.js"></script>
      <script src="/static/libs/js/hogan.js"></script>
```

```
element.style {

}
body.home .content-wrap {                 base.css:9
   background:▶0;
}
.content-wrap {                      site_base.css:7
   margin:▶75px 75px;
}
.content-wrap {                           base.css:9
   ~~background:▶ #fff;~~
   padding:▶30px 0 0 20px;
   width: 980px;
   min-height: 400px;
   ~~margin:▶0 auto 30px;~~
}
div {                         user agent stylesheet
   display: block;
}
```

Fig. 3.75: *Screenshot of using Chrome's debugger to inspect the CSS overrides*

You will see that it extends from `base.html`, which is the GeoNode template referenced above and it currently only overrides the `extra_head` block to include our project's `site_base.css` which we modified in the previous section. You can see on line 14 of the GeoNode base.html template that this block is included in an empty state and is set up specifically for you to include extra CSS files as your project is already set up to do.

Now that we have looked at `site_base.html`, let's actually override a different template.

The file `site_index.html` is the template used to define your GeoNode project's homepage. It extends GeoNode's default `index.html` template and gives you the option to override specific areas of the homepage like the hero area, but also allows you leave area like the "Latest Layers" and "Maps" and the "Contribute" section as they are. You are of course free to override these sections if you choose and this section shows you the steps necessary to do that below.

1. Open `<my_geonode>/templates/site_index.html` in your editor.
2. Edit the `<h1>` element on line 13 to say something other than "Welcome":

```
<h1>{% trans "UWI GeoNode" %}</h1>
```

3. Edit the introductory paragraph to include something specific about your GeoNode project:

```
<p>
    {% blocktrans %}
    UWI's GeoNode is setup for students and faculty to collaboratively
    create and share maps for their class projects. It is maintained by the
    UWI Geographical Society.
    {% endblocktrans %}
</p>
```

4. Change the *Getting Started* link to point to another website:

```
<span>
    For more information about the UWI Geographical society,
    <a href="http://uwigsmona.weebly.com/">visit our website</a>
```

```
</span>
```

5. Add a graphic to the hero area above the paragraph replaced in step 3:

```
<img src = 'http://uwigsmona.weebly.com/uploads/1/3/2/4/13241997/1345164334.png'>
```

6. Your edited `site_index.html` file should now look like this:

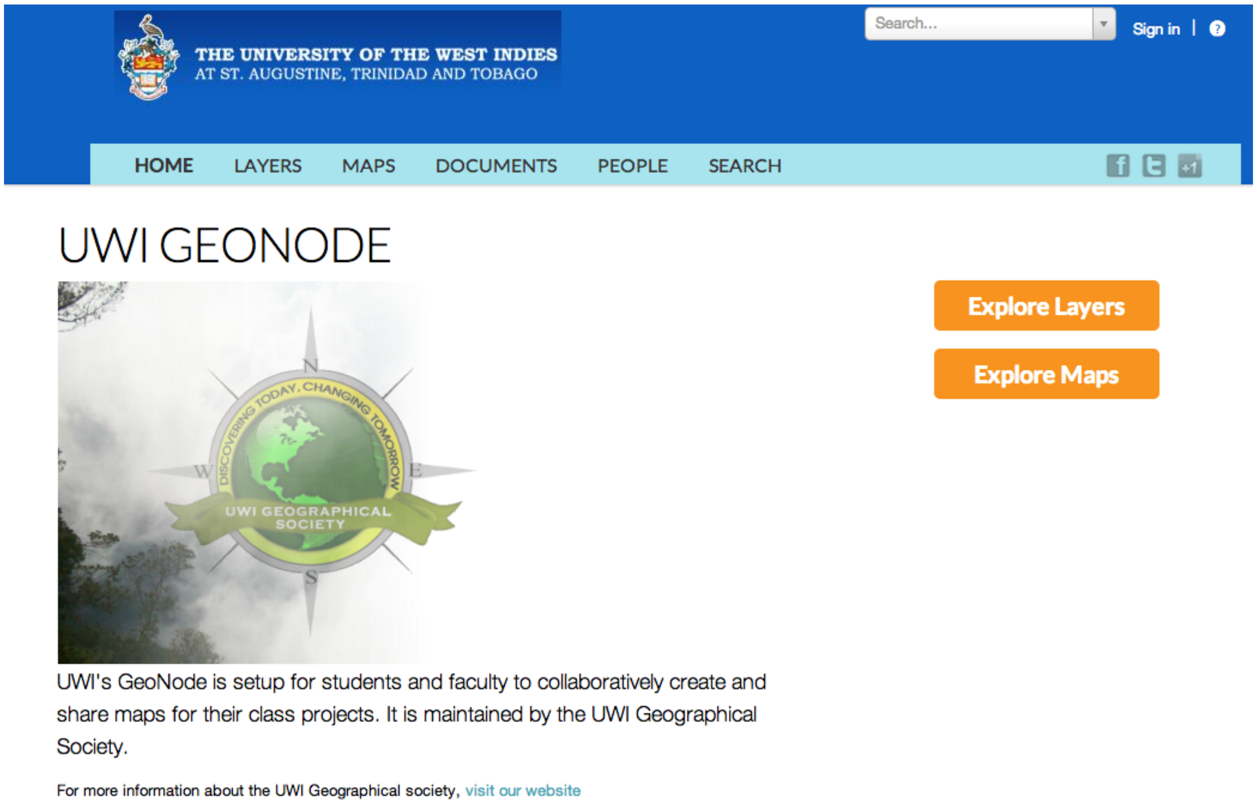
```
{% extends 'index.html' %}
{% load i18n %}
{% load maps_tags %}
{% load layers_tags %}
{% load pagination_tags %}
{% load staticfiles %}
{% load url from future %}
{% comment %}
This is where you can override the hero area block. You can simply modify the content below or r
{% endcomment %}
{% block hero %}
    <div class="hero-unit">
        <h1>{% trans "UWI GeoNode" %}</h1>
        <div class="hero-unit-content">
        <div class="intro">
            <img src = 'http://uwigsmona.weebly.com/uploads/1/3/2/4/13241997/1345164334.png'>
        <p>
            {% blocktrans %}
            UWI's GeoNode is setup for students and faculty to collaboratively
            create and share maps for their class projects. It is maintained by the
            UWI Geographical Society.
            {% endblocktrans %}
        </p>
        <span>
            For more information about the UWI Geographical society,
            <a href="http://uwigsmona.weebly.com/">visit our website</a>
        </span>
    </div>
    <div class="btns">
        <a class="btn btn-large" href="{% url "layer_browse" %}">
        {% trans "Explore Layers" %}
        </a>
        <a class="btn btn-large" href="{% url "maps_browse" %}">
        {% trans "Explore Maps" %}
        </a>
    </div>
</div>
{% endblock %}
```

7. Restart your GeoNode project and view the changes in your browser at http://localhost:8000/ or the remote URL for your site:

```
$ python manage.py runserver
```

From here you can continue to customize your `site_index.html` template to suit your needs. This workshop will also cover how you can add new pages to your GeoNode project site.

Other theming options You are able to change any specific piece of your GeoNode project's style by adding CSS rules to `site_base.css`, but since GeoNode is based on Bootstrap, there are many pre-defined themes that you


THE UNIVERSITY OF THE WEST INDIES
AT ST. AUGUSTINE, TRINIDAD AND TOBAGO
Search...
Sign in
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
Explore Layers
Explore Maps
DISCOVERING TODAY. CHANGING TOMORROW
UWI GEOGRAPHICAL SOCIETY

UWI GEONODE

UWI's GeoNode is setup for students and faculty to collaboratively create and share maps for their class projects. It is maintained by the UWI Geographical Society.

For more information about the UWI Geographical society, visit our website

can simply drop into your project to get a whole new look. This is very similar to WordPress themes and is a powerful and easy way to change the look of your site without much effort.

Bootswatch Bootswatch is a site where you can download ready-to-use themes for your GeoNode project site. The following steps will show you how to use a theme from Bootswatch in your own GeoNode site.

1. Visit http://bootswatch.com and select a theme (we will use Amelia for this example). Select the *download bootstrap.css option* in the menu:
2. Put this file in `<my_geonode>/static/css`.
3. Update the `site_base.html` template to include this file. It should now look like this:

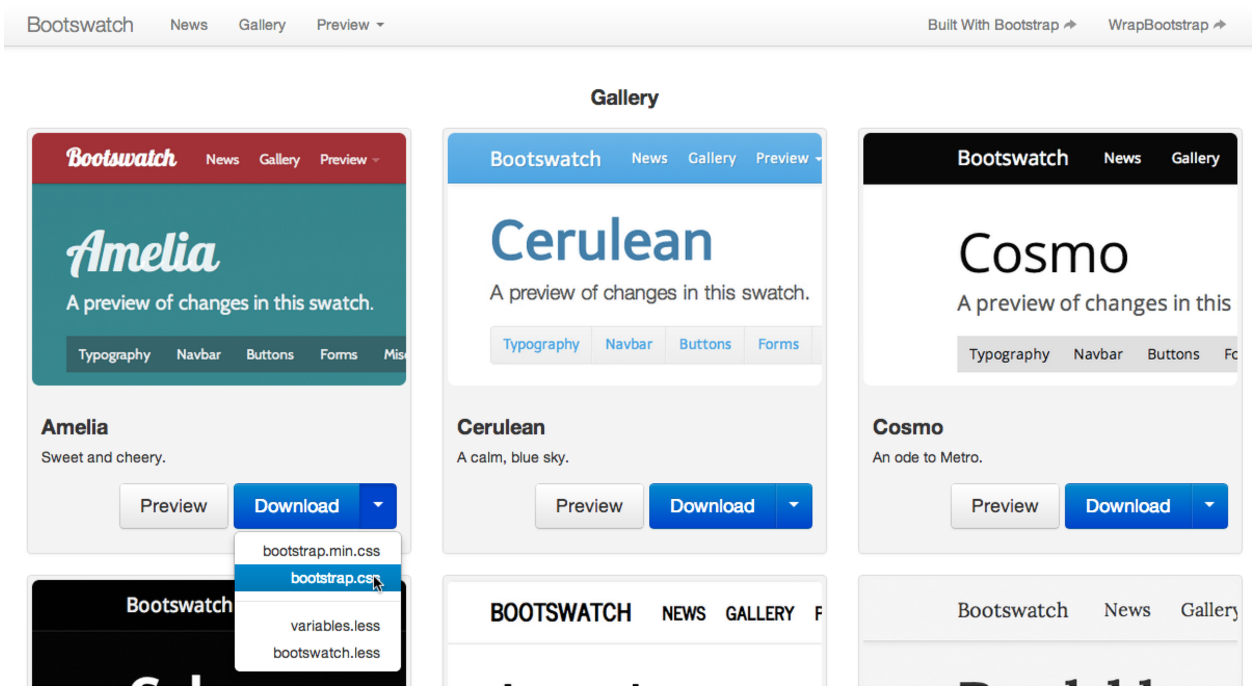
```
{% extends "base.html" %}
{% block extra_head %}
    <link href="{{ STATIC_URL }}css/site_base.css" rel="stylesheet"/>
    <link href="{{ STATIC_URL }}css/bootstrap.css" rel="stylesheet"/>
{% endblock %}
```

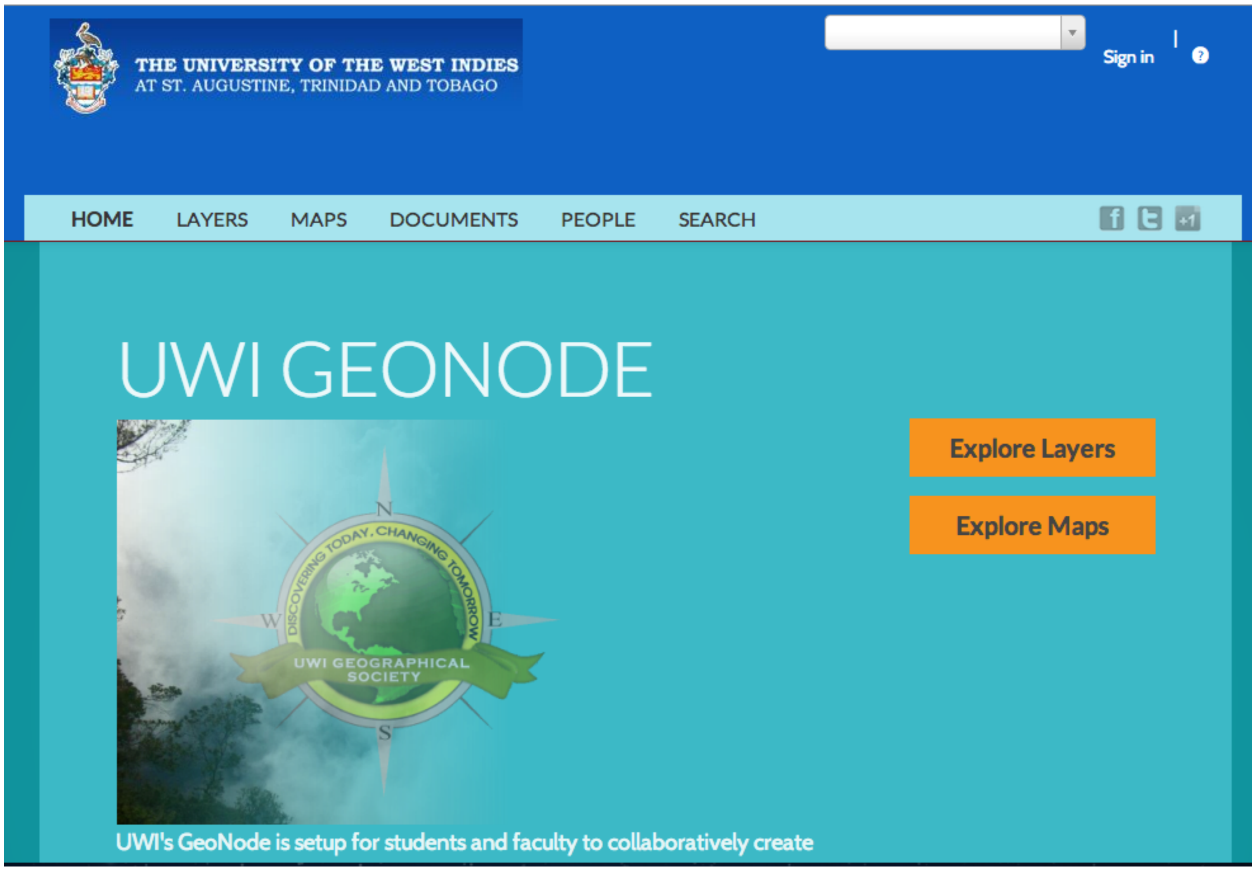
4. Restart the development server and visit your site:

Your GeoNode project site is now using the Amelia theme in addition to the changes you have made.

Setup steps Setup your own geonode project

Theming your GeoNode project Theme your geonode project


Bootswatch
News
Gallery
Preview
Built With Bootstrap
WrapBootstrap
Gallery
Amelia
A preview of changes in this swatch.
Amelia
Sweet and cheery.
Preview
Download
bootstrap.min.css
bootstrap.css
variables.less
bootswatch.less
Cerulean
A preview of changes in this swatch.
Cerulean
A calm, blue sky.
Preview
Download
Cosmo
A preview of changes in this
Cosmo
An ode to Metro.
Preview
Download


THE UNIVERSITY OF THE WEST INDIES
AT ST. AUGUSTINE, TRINIDAD AND TOBAGO
Sign in
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
UWI GEONODE
Explore Layers
Explore Maps
UWI GEOGRAPHICAL SOCIETY
UWI's GeoNode is setup for students and faculty to collaboratively create

Debugging GeoNode Installations

There are several mechanisms to debug GeoNode installations, the most common ones are discussed in the following sections.

Viewing the logs There are many kinds of logs in GeoNode, most of them are located in `/var/log/geonode/` and will be explained below in order of relevance:

- **GeoNode main log**: This is the output of the Django application generated by Apache, it may contain detailed information about uploads and high level problems.

  The default location is `/var/log/geonode/apache.log` or `/var/log/apache2/error.log`.

  It is set to a very low level (not very much information is logged) by default, but it's output can be increased by setting the logging level to `DEBUG` in `/etc/geonode/local_settings.py`.

- **GeoServer log**: It contains most of the information related to problems with data, rendering and styling errors.

  This one can be accessed at `GEOSERVER_DATA_DIR/logs/geoserver.log`, which is usually `/var/lib/geoserver/geonode-data/logs/geoserver.log`'.

  It may also be symlinked in `/var/log/geonode/geoserver.log`.

- **Tomcat logs**: Tomcat logs could indicate problems loading GeoServer.

  They can be found at `/var/lib/tomcat6/logs/catalina.out`.

- **PostgreSQL logs**: PostgreSQL is accessed by GeoServer and Django, therefore information about errors which are very hard to debug may be found by looking at PostgreSQL's logs.

  They are located at `/var/log/postgresql/postgresql-8.4-main.log`.

Enabling DEBUG mode Django can be set to return nicely formatted exceptions which are useful for debugging instead of generic `500 errors`.

This is enabled by setting `DEBUG=True` in `/etc/geonode/local_settings.py`.

After enabling DEBUG, the Apache server has to be restarted for the changes to be picked up. In Ubuntu:

```
sudo service apache2 restart
```

Other tips and tricks

Modifying GeoServer's output strategy Up to version 1.1, GeoNode used by default the `SPEED` output strategy of GeoServer, this meant that proper error messages were being sacrificed for performance. Unfortunately, this caused many errors to be masked as XML parsing errors when layers were not properly configured.

It is recommended to verify the output strategy is set to `PARTIAL_BUFFER2` (or a safer one, e.g. `FILE`) with a high value for the buffer size. More information about the different strategies and the performance vs correctness trade off is available at GeoServer's web.xml file.

The typical location of the file that needs to be modified is `/var/lib/tomcat6/webapps/geoserver/WEB-INF/web.xml` as shown below:

```
<context-param>
  <param-name>serviceStrategy</param-name>
  <param-value>PARTIAL_BUFFER2</param-value>
</context-param>
```

Add the Django debug toolbar The django debug toolbar offers a lot of information on about how the page you are seeing is created and used. From the database hits to the views involved. It is a configurable set of panels that display various debug information about the current request/response and when clicked, display more details about the panel's content.

To install it:

```
$ pip install django-debug-toolbar
```

Then edit your settings and add the following to the MIDDLEWARE_CLASSES:

```
MIDDLEWARE_CLASSES = (
    # ...
    'debug_toolbar.middleware.DebugToolbarMiddleware',
    # ...
)
```

Add the following to your INSTALLED_APPS:

```
INSTALLED_APPS = (
  # ...
  'debug_toolbar',
)
```

Add also the following settings:

```
INTERNAL_IPS = ('127.0.0.1',)

DEBUG_TOOLBAR_CONFIG = {
    'INTERCEPT_REDIRECTS': False,
}
```

For more set up and customize the panels read the official docs here:

http://django-debug-toolbar.readthedocs.org/en/latest/

Changing the Default Language

GeoNode's default language is English, but GeoNode users can change the interface language with the pulldown menu at the top-right of most GeoNode pages. Once a user selects a language GeoNode remembers that language for subsequent pages.

GeoNode Configuration As root edit the geonode config file `/etc/geonode/local_settings.py` and change `LANGUAGE_CODE` to the desired default language. Note a list of language codes can be found in the global django config file `/var/lib/geonode/lib/python2.6/site-packages/django/conf/global_settings.py`. For example, to make French the default language use:

```
LANGUAGE_CODE = 'fr'
```

Unfortunately Django overrides this setting, giving the language setting of a user's browser priority. For example, if LANGUAGE_CODE is set to French, but the user has configured their operating system for Spanish they may see the Spanish version when they first visit GeoNode.

Additional Steps If this is not the desired behaviour, and all users should initially see the default LANGUAGE_CODE, regardless of their browser's settings, do the following steps to ensure Django ignores the browser language settings. (Users can always use the pulldown language menu to change the language at any time.)

As create a new directory within GeoNode's site packages:

```
sudo mkdir /var/lib/geonode/lib/python2.6/site-packages/setmydefaultlanguage
```

As root create and edit a new file /var/lib/geonode/lib/python2.6/site-packages/setmydefaultlanguage/__i and add the following lines:

```
class ForceDefaultLanguageMiddleware(object):
    """
    Ignore Accept-Language HTTP headers

    This will force the I18N machinery to always choose settings.LANGUAGE_CODE
    as the default initial language, unless another one is set via sessions or cookies

    Should be installed *before* any middleware that checks request.META['HTTP_ACCEPT_LANGUAGE'],
    namely django.middleware.locale.LocaleMiddleware
    """
    def process_request(self, request):
        if request.META.has_key('HTTP_ACCEPT_LANGUAGE'):
            del request.META['HTTP_ACCEPT_LANGUAGE']
```

At the end of the GeoNode configuration file /etc/geonode/local_settings.py add the following lines to ensure the above class is executed:

```
MIDDLEWARE_CLASSES = (
    'django.middleware.common.CommonMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'setmydefaultlanguage.ForceDefaultLanguageMiddleware',
    'django.middleware.locale.LocaleMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
)
```

Restart Finally restart Apache with:

```
sudo /etc/init.d/apache2 restart
```

Please refer to Translating GeoNode for information on editing GeoNode pages in different languages and create new GeoNode Translations.

GeoSites: GeoNode Multi-Tenancy

GeoSites is an approach to support multiple websites with GeoNode. Each GeoSite can have different templates, apps, and data permissions but share a single database (useful for sharing users and data layers), GeoServer, and CSW. Some of this functionality is currently available with the dev branch of GeoNode, although a fully working system will require extending the GeoNode security app to support the Django Sites framework.

A key component in managing multiple sites is keeping data organized and using a structured series of settings files so that common settings can be shared and only site specific settings are separated out.

One Website to Rule Them All The GeoSites approach uses a 'master' website in order to simplify data management. This master site serves as a single website from which all data can be managed. Additionally, if desired, any or all of the django apps installed in the other sites can be added to the master site to provide a single admin interface that gives full access to all apps. The master website also serves as the single URL which GeoServer authenticates to.

GeoServer and PostGIS The single GeoServer instance is the main method in which new data is added to the websites. To keep data organized, workspaces should be used:

- geonode: This is the default workspace, and should not contain anything at first since it will be the workspace data is uploaded to by users.
- common: This workspace should contain data that is common to all websites.
- website: One workspace should be created for each website that contains data that is specific to that workspace.
- other: Other workspaces may be created for specific themed data that may be shared across multiple, but not all, sites.

Since in GeoServer a datastore (PostGIS database) belongs to a single workspace and the database may contain data for multiple sites, schemas should be used to keep the data separated out and should mirror the workspaces above (with the public schema being for common data). One PostGIS datastore will then be created for each workspace/schema.

Settings Files and Web Server The majority of settings are common to all GeoNode sites and these should be separated out into a settings_master.py file. Apache (or other web server) will have a virtual site set-up for each website (including the master site, which will be the default), with each one pointing to a separate .wsgi file. Each .wsgi file will point to a settings file unique to that site. The site settings file will be limited in scope, only containing the differences between the sites including:

- SITE_ID: Each one is unique, the master site should have a SITE_ID of 1.
- SITENAME
- SITEURL
- ROOT_URLCONF: This may be optional. The main site url.conf can be configured to automatically import the urls.py of all SITE_APPS, so a different ROOT_URLCONF is only needed if there are further differences.
- SITE_APPS: Containing the site specific apps
- App settings: Any further settings required for the above sites
- Other site specific settings, such as REGISTRATION_OPEN

At the end of the site specific settings file it will include the global settings file:

```
PROJECT_ROOT = os.path.dirname(mastersite.__file__)
execfile(os.path.join(PROJECT_ROOT,'settings_global.py')
```

Where PROJECT_ROOT is the location of the entire project containing all the websites and the global settings file. The global settings file is read in via 'exec' rather than import so that it can then modify and use variables from the site specific settings file.

The main difference between the settings_global.py file and the default GeoNode settings file is the inclusion of a SITE_ROOT directory. Therefore, three directories will be defined:

- SITE_ROOT: The directory where the site specific settings and files are located (templates, static)
- PROJECT_ROOT: The top-level directory of all the GeoSites which should include the global settings file as well as template and static files
- GEONODE_ROOT: The GeoNode directory.

The TEMPLATE_DIRS, and STATICFILES_DIRS will then include all three directories as shown:

```
TEMPLATE_DIRS = (
    os.path.join(SITE_ROOT, 'templates/'),
    os.path.join(PROJECT_ROOT,'templates/'),  # files common to all sites
    os.path.join(GEONODE_ROOT, 'templates/')
)

STATICFILES_DIRS = (
    os.path.join(SITE_ROOT, 'static/'),
    os.path.join(PROJECT_ROOT, 'static/'),
    os.path.join(GEONODE_ROOT, 'static/')
)
```

At the end of the settings_global.py the following variables will be set based on site specific settings:

```
STATIC_URL = os.path.join(SITEURL,'static/')
GEONODE_CLIENT_LOCATION = os.path.join(STATIC_URL,'geonode/')
GEOSERVER_BASE_URL = SITEURL + 'geoserver/'
if SITE_APPS:
    INSTALLED_APPS += SITE_APPS
```

..note the settings files also require differences between development and production servers which are not currently discussed here.

Templates and Static Files As mentioned above for each website there will be three directories used for template and static files. The first template file found will be the one used so templates in the SITE_ROOT/templates directory will override those in PROJECT_ROOT/templates, which will override those in GEONODE_ROOT/templates.

Static files work differently because (at least on a production server) they are collected and stored in a single location. Because of this care must be taken to avoid clobbering of files between sites, so each site directory should contain all static files in a subdirectory with the name of the site (e.g., static/masshealth/logo.png)

The location of the proper static directory can then be found in the templates syntax such as:

```
{{ STATIC_URL }}{{ SITENAME|lower }}/logo.png
```

Site Permissions The main aspect of GeoSites that is currently not implemented is to be able to set permissions by site, so that data can be viewable from one site but not another. This will require extending the GeoNode security app to utilize the Django sites framework, so that every object can be set to: No Sites: If no site is set the object should still be accessible by the master site to ensure that data is never orphaned. All Sites: All sites have access to data One of more individual Sites: Set specific sites.

To help in management it is further suggested that a management command be added (or extend updatelayers) which will batch set the permissions for data. The site specific settings file can specifiy one or more GeoServer workspaces. The management command will then set all data within those workspaces to be accessible from that site. (note: this command will be used in a loop, cycling through all sites).

Running GeoNode under SSL

Enabling SSL will encrypt traffic between your GeoNode server and client browsers. This approach involves re-configuring Apache to serve on port 443, instead of port 80. Other approaches exist and should be added to this document.

Generate SSL Key & Certificate The first step is to generate a DES key.:

```
# for CommonName use GeoNode domain name or ip address as specified in GeoNode's SITEURL
openssl genrsa -des3 -out server.key 1024
openssl req -new -key server.key -out server.csr

# generate new server.key without challenge password, or Apache will ask for password at startup
mv server.key server.key.tmp
openssl rsa -in server.key.tmp -out server.key

# generate certificate
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

Copy the key and certificate to the standard locations:

```
sudo cp server.crt /etc/ssl/certs/geonode.crt
sudo cp server.key /etc/ssl/private/geonode.key
```

Next add the certificate to the cacerts file for python and java:

```
sudo -s "cat server.crt >> /var/lib/geonode/lib/python2.6/site-packages/httplib2/cacerts.txt"
sudo keytool -import -alias geonodessl -keystore /etc/ssl/certs/java/cacerts -file server.crt
```

Note keytool will ask for a password and the standard password for the java cacerts file is `changeit`.

Apache Configuration Enable the ssl module in Apache with the command:

```
sudo a2enmod ssl
```

Next as root edit the Apache geonode config file `/etc/apache2/sites-available/geonode`. At the beginning of the file replace:

```
<VirtualHost *:80>
```

with:

```
<IfModule mod_ssl.c>
<VirtualHost _default_:443>
```

At the bottom of the file, replace:

```
</VirtualHost>
```

with:

```
    SSLEngine on
    SSLCertificateFile    /etc/ssl/certs/geonode.crt
    SSLCertificateKeyFile /etc/ssl/private/geonode.key
    BrowserMatch "MSIE [2-6]" \
        nokeepalive ssl-unclean-shutdown \
        downgrade-1.0 force-response-1.0
    # MSIE 7 and newer should be able to use keepalive
    BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
</VirtualHost>
</IfModule>

<VirtualHost  *:80>
    Redirect permanent / https://192.168.10.10/
</VirtualHost>
```

This tells Apache where to fine the key and certificate. There are also some additional lines to handle MSIE, taken from Apache's default-ssl file.

Tomcat Configuration As root edit the Tomcat server config file `/var/lib/tomcat6/conf/server.xml`, and replace:

```
<Connector port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    URIEncoding="UTF-8"
    redirectPort="8443"
/>
```

with:

```
<Connector port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000"
    URIEncoding="UTF-8"
    scheme="https"
    proxyName="<yourServersIPorDomainName>"
    proxyPort="443"
/>
```

This tells Tomcat that it is running behind an https proxy. If this is omitted Tomcat will try to redirect to http.

GeoNode Configuration As root edit the geonode config file `/etc/geonode/local_settings.py` and change the `SITEURL` protocol to https:

```
SITEURL = 'https://<ipaddressOrDomainName>/'
```

GeoServer Configuration As root edit the file `/var/lib/tomcat6/webapps/geoserver/WEB-INF/web.xml` and ensure the `GEONODE_BASE_URL` is specified as follows:

```
<context-param>
    <param-name>GEONODE_BASE_URL</param-name>
    <param-value>https://localhost/</param-value>
</context-param>
```

Also update `proxyBaseUrl` in the Geoserver global settings file `/var/lib/geoserver/geonode-data/global.xml`:

```
<proxyBaseUrl>https://192.168.10.10/geoserver/</proxyBaseUrl>
```

Restart Finally restart Apache and Tomcat with:

```
sudo /etc/init.d/apache2 restart
sudo /etc/init.d/tomcat6 restart
```

This information was complied from a number of sources. The main links are listed below. Please contact the GeoNode list with any updates or corrections.

- http://confluence.atlassian.com/display/JIRA/Connecting+to+SSL+services
- http://confluence.atlassian.com/display/JIRA/Integrating+JIRA+with+Apache+using+SSL
- http://www.akadia.com/services/ssh_test_certificate.html
- https://help.ubuntu.com/10.04/serverguide/C/httpd.html
- https://help.ubuntu.com/10.04/serverguide/C/certificates-and-security.html

Introduction Learn about GeoNode administration

Preparing your environment Learn about preparing an environment for installation of GeoNode

Installing GeoNode Learn how to install GeoNode

migrate Learn how to migrate an existing GeoNode 1.2 instance to GeoNode 2.0

Configuring GeoNode for Production Learn how to configure GeoNode for production use.

Managing a GeoNode Installation Learn how to manage a GeoNode installation

Monitoring a GeoNode Learn how to monitor usage and other activity in a GeoNode Installation

Backing up and Restoring GeoNode Learn how to backup and restore GeoNode configuration and datas

Deploying GeoNode Review a sample GeoNode project deployment and learn how to create your own

Loading Data into a GeoNode Learn how to load data into a GeoNode with GeoServer, on the command line or programatically with scripts.

Usage of the GeoNode's Django Administration Panel Manage users and data through the administration panel

Django external applications Manage external Django applications

Management Commands for GeoNode Review GeoNode's management commands

Debugging GeoNode Installations Learn how to debug and troubleshoot a GeoNode Installation

Configuring Alternate CSW Backends Learn how to configure an alternate CS-W backend for your GeoNode

Customize the look and feel Learn how to customize the look and feel of your GeoNode installation

Changing the Default Language Learn how to change the default language of your GeoNode installation

GeoSites: GeoNode Multi-Tenancy Learn how to configure multiple GeoNode sites on the same server

Running GeoNode under SSL Learn how to run GeoNode with Secure Sockets Layer (SSL)

Developers Workshop

Welcome to the Developers Workshop! This workshop will teach how to develop with and for the GeoNode software application.

Introduction to GeoNode development

This module will introduce you to the components that GeoNode is built with, the standards that it supports and the services it provides based on those standards, and an overview its architecture.

GeoNode is a web based GIS tool, and as such, in order to do development on GeoNode itself or to integrate it into your own application, you should be familiar with basic web development concepts as well as with general GIS concepts.

A set of reference links on these topics is included at the end of this module.

Standards GeoNode is based on a set of Open Geospatial Consortium (OGC) standards. These standards enable GeoNode installations to be interoperable with a wide variety of tools that support these OGC standards and enable federation with other OGC compliant services and infrastructure. Reference links about these standards are also included at the end of this module.

GeoNode is also based on Web Standards ...

Open Geospatial Consortium (OGC) Standards

Web Map Service (WMS) The Web Map Service (WMS) specification defines an interface for requesting rendered map images across the web. It is used within GeoNode to display maps in the pages of the site and in the GeoExplorer application to display rendered layers based on default or custom styles.

Web Feature Service (WFS) The Web Feature Service (WFS) specification defines an interface for reading and writing geographic features across the web. It is used within GeoNode to enable downloading of vector layers in various formats and within GeoExplorer to enable editing of Vector Layers that are stored in a GeoNode.

Web Coverage Service (WCS) The Web Coverage Service (WCS) specification defines an interface for reading and writing geospatial raster data as "coverages" across the web. It is used within GeoNode to enable downloading of raster layers in various formats.

Catalogue Service for Web (CSW) The Catalogue Service for Web (CSW) specification defines an interface for exposing a catalogue of geospatial metadata across the web. It is used within GeoNode to enable any application to search GeoNode's catalogue or to provide federated search that includes a set of GeoNode layers within another application.

Tile Mapping Service (TMS/WMTS) The Tile Mapping Service (TMS) specification defines and interface for retrieving rendered map tiles over the web. It is used within geonode to enable serving of a cache of rendered layers to be included in GeoNode's web pages or within the GeoExplorer mapping application. Its purpose is to improve performance on the client vs asking the WMS for rendered images directly.

Web Standards

HTML

CSS

REST

Exercises

Components and Services Hint, if `bash-completion` is installed, try <TAB><TAB> to get completions.

1. **start/stop services**

```
$ sudo service apache2
$ sudo service apache2 reload
$ sudo service tomcat7
$ sudo service postgresql
```

2. **basic psql interactions**

```
$ sudo su - postgres
$ psql
=> help                 # get help
=> \?                   # psql specific commands
=> \l                   # list databases
=> \c geonode           # switch database
```

```
        => \ds                  # list tables
        => \dS layers_layer     # describe table
```

OGC Standards

WMS

1. Use the layer preview functionality in GeoServer to bring up a web map.
2. Copy a the URL for the image in the map.
3. Alter URL parameters for the request.
4. Use *curl* to get the capabilities document

```
    $ curl 'http://localhost/geoserver/wms?request=getcapabilities'
```

More: http://docs.geoserver.org/stable/en/user/services/wms/index.html

WFS

1. Describe a feature type using curl (replace ws:name with your layer)

```
    $ curl 'http://localhost/geoserver/wfs?request=describefeaturetype&name=ws:name
```

More: http://docs.geoserver.org/stable/en/user/services/wfs/reference.html

Development References

Basic Web based GIS Concepts and Background

- OGC Services
  - http://www.opengeospatial.org/
  - http://en.wikipedia.org/wiki/Open_Geospatial_Consortium
- Web Application Architecture
  - http://en.wikipedia.org/wiki/Web_application
  - http://www.w3.org/2001/tag/2010/05/WebApps.html
  - http://www.amazon.com/Web-Application-Architecture-Principles-Protocols/dp/047051860X
- AJAX and REST
  - http://en.wikipedia.org/wiki/Ajax_(programming)
  - http://en.wikipedia.org/wiki/Representational_state_transfer
- OpenGeo Suite
  - http://workshops.opengeo.org/suiteintro/
  - http://suite.opengeo.org/opengeo-docs/
- GeoServer Administration
  - http://suite.opengeo.org/opengeo-docs/geoserver/
  - https://docs.google.com/a/opengeo.org/presentation/d/15fvUDYg0TO6WGFQlMLM2J1qiTVBYpfjCp0aQBDT0GrM/edit#

  - http://suite.opengeo.org/docs/sysadmin/index.html#sysadmin
- PostgreSQL and PostGIS Administration - http://workshops.opengeo.org/postgis-intro/ - http://workshops.opengeo.org/postgis-spatialdbtips/

Core development tools and libraries

- python
  - http://docs.python.org/2/tutorial/
  - http://www.learnpython.org/
  - http://learnpythonthehardway.org/book/
- django
  - https://docs.djangoproject.com/en/dev/intro/tutorial01/
  - https://code.djangoproject.com/wiki/Tutorials
- javascript
  - http://www.crockford.com/javascript/inheritance.html
  - http://geoext.org/tutorials/quickstart.html
- jquery
  - http://www.w3schools.com/jquery/default.asp
  - http://docs.jquery.com/Tutorials:Getting_Started_with_jQuery
  - http://www.jquery-tutorial.net/
- bootstrap
  - http://twitter.github.io/bootstrap/
  - http://www.w3resource.com/twitter-bootstrap/tutorial.php
- geotools/geoscript/geoserver
  - http://docs.geotools.org/stable/tutorials/feature/csv2shp.html
  - http://geoscript.org/tutorials/index.html
  - http://docs.geotools.org/stable/tutorials/
  - https://github.com/dwins/gsconfig.py/blob/master/README.rst
- geopython
  - http://pycsw.org/docs/documentation.html
  - http://geopython.github.io/OWSLib/
  - https://github.com/toblerity/shapely
  - https://github.com/sgillies/Fiona
  - http://pypi.python.org/pypi/pyproj
- gdal/ogr
  - http://www.gdal.org/gdal_utilities.html
  - http://www.gdal.org/ogr_utilities.html

Development Prerequsites and Core Modules

This module will introduce you to the

GeoNode's Development Prerequisites

Basic Shell Tools

ssh and sudo ssh and sudo are very basic terminal skills which you will need to deploy, maintain and develop with GeoNode. If you are not already familiar with their usage, you should review the basic descriptions below and follow the external links to learn more about how to use them effectively as part of your development workflow.

ssh is the network protocol used to connect to a remote server where you run your GeoNode instance whether on your own network or on the cloud. You will need to know how to use an the ssh command from the terminal on your unix machine or how to use a ssh client like putty or winscp on windows. You may need to use pki certificates to connect to your remove server, and should be familiar with the steps and options necessary to connect this way. More information about ssh can be found in the links below.

- http://winscp.net/eng/docs/ssh

sudo is the command used to execute a terminal command as the superuser when you are logged in with a normal user. You will to use sudo in order to start, stop and restart key services on your GeoNode instance. If you are not able to grant yourself these privileges on the machine you are using for your GeoNode instance, you may need to consult with your network administrator to arrange for your user to be granted sudo permissions. More information about sudo can be found in the links below.

- http://en.wikipedia.org/wiki/Sudo

bash *Bash* is the most common unix shell which will usually be the default on servers where you will be deploying your GeoNode instance. You should be familiar with the most common bash commands in order to be able to deploy, maintain and modify a geonode instance. More information about Bash and common bash commands can be found in the links below.

- http://en.wikipedia.org/wiki/Bash_(Unix_shell)

apt *apt* is the packaging tool that is used to install GeoNode on ubuntu and other debian based systems. You will need to be familiar with adding Personal Package Archives to your list of install sources, and will need to be familiar with basic apt commands. More information about apt can be found in the links below.

- http://en.wikipedia.org/wiki/Advanced_Packaging_Tool

Python Development Tools The GeoNode development process relies on several widely used python development tools in order to make things easier for developers and other users of the systems that GeoNode developers work on or where GeoNodes are deployed. They are considered best practices for modern python development, and you should become familiar with these basic tools and be comfortable using them on your own projects and systems.

virtualenv *virtualenv* is a tool used to create isolated python development environments such that the the versions of project dependencies are sandboxed from the system-wide python packages. This eliminates the commonly encountered problem of different projects on the same system using different versions of the same library. You should be familiar with how to create and activate virtual environments for the projects you work on. More information about virtualenv can be found in the links below.

- http://pypi.python.org/pypi/virtualenv

- http://www.virtualenv.org/en/latest/

virtualenvwrapper is a wrapper around the virtualenv package that makes it easier to create and switch between virtual environments as you do development. Using it will make your life much easier, so its recommended that you install and configure it and use its commands as part of your virtualenv workflow. More info about virtualenvwrapper can be found in the links below.

- http://www.doughellmann.com/projects/virtualenvwrapper/

pip *pip* is a tool for installing and managing python packages. Specifically it is used to install and upgrade packages found in the Python Pacakge Index. GeoNode uses pip to install itself, and to manage all of the python dependencies that are needed as part of a GeoNode instance. As you learn to add new modules to your geonode, you will need to become familiar with the use of pip and about basic python packaging usage. More information about pip can be found in the links below.

- http://www.pip-installer.org/en/latest/
- http://pypi.python.org/pypi/pip
- http://en.wikipedia.org/wiki/Pip_(Python)

miscellaneous *ipython* is a set of tools to make your python development and debugging experience easier. The primary tool you want to use is an interactive shell that adds introspection, integrated help and command completion and more. While not strictly required to do GeoNode development, learning how to use ipython will make your development more productive and pleasant. More information about ipython can be found in the links below.

- http://ipython.org/
- http://pypi.python.org/pypi/ipython
- https://github.com/ipython/ipython
- http://en.wikipedia.org/wiki/IPython

pdb is a standard python module that is used to interactively debug your python code. It supports setting conditional breakpoints so you can step through the code line by line and inspect your variables and perform arbitrary execution of statements. Learning how to effectively use pdb will make the process of debugging your application code significantly easier. More information about pdb can be found in the links below.

- http://docs.python.org/2/library/pdb.html

Django GeoNode is built on top of the *Django web framework*, and as such, you will need to become generally familiar with Django itself in order to become a productive GeoNode developer. Django has excellent documentation, and you should familiarize yourself with Django by following the Django workshop and reading through its documentation as required.

Model Template View Django is based on the Model Template View paradigm (more commonly called Model View Controller). Models are used to define objects that you use in your application and Django's ORM is used to map these models to a database. Views are used to implement the business logic of your application and provide objects and other context for the templates. Templates are used to render the context from views into a page for display to the user. You should become familiar with this common paradigm used in most modern web frameworks, and how it is specifically implemented and used in Django. The Django tutorial itself is a great place to start. More information about MTV in Django can be found in the links below.

- http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
- http://www.codinghorror.com/blog/2008/05/understanding-model-view-controller.html

- https://docs.djangoproject.com/en/1.4/

HTTP Request Response Django and all other web frameworks are based on the HTTP Request Response cycle. Requests come in to the server from remote clients which are primarily web browsers, but also can be api clients, and the server returns with a Response. You should be familiar with these very basic HTTP principles and become familiar with the way that Django implements them. More information about HTTP, Requests and Responses and Djangos implementation in the links below.

- http://devhub.fm/http-requestresponse-basics/
- http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- https://docs.djangoproject.com/en/dev/ref/request-response/

Management Commands Django projects have access to a set of management commands that are used to manage your project. Django itself provides a set of these commands, and django apps (including GeoNode) can provide their own. Management commands are used to do things like synchronize your models with your database, load data from fixtures or back up your database with fixtures, start the development server, initiate the debugger and many other things. GeoNode provides management commands for synchronizing with a GeoServer or updating the layers already in your GeoNode. You should become familiar with the basic management commands that come with Django, and specifically with the commands that are part of GeoNode. The GeoNode specific commands are covered in section. More information about management commands can be found in the links below.

- https://docs.djangoproject.com/en/dev/ref/django-admin/

Django Admin Interface Django provides a build-in management console that administrators and developers can use to look at the data in the database that is part of the installed applications. Administrators can use this console to perform many common administration tasks that are a necessary part of running a GeoNode instance, and as a developer, you will use this interface during your development process to inspect the database and the data stored in your models. More information about the django admin interface can be found in the links below.

- https://docs.djangoproject.com/en/dev/ref/contrib/admin/

Template Tags Django templates make use of a set of tags to inject, filte and format content into a rendered HTML page. Django itself includes a set of built-in template tags and filters that you will use in your own templates, and GeoNode provides a geonode specific set of tags that are used in the GeoNode templates. You should become familiar with the built-in tag set and with GeoNode's specific tags as you work on developing your own templates or extending from GeoNode's. More information about Django template tags can be found in the links below.

- https://docs.djangoproject.com/en/dev/ref/templates/builtins/

GeoNode's Core Modules GeoNode is made up of a set of core Django pluggable modules (known as apps in Django) that provide the functionality of the application. Together they make up the key components of a GeoNode site. While your own use case and implementation may not require that you work directly on these modles, it is important that you become familiar with their layout, structure and the functionality that they provide. You may need to import these apps into your own apps, and as such, becoming familiar with them is an important step in becoming a proficient GeoNode developer.

geonode.layers geonode.layers is the most key GeoNode module. It is used to represent layers of data stored in a GeoNode's paired GeoServer. The layer model class inherits fields from the ResourceBase class which provides all of the fields necessary for the metadata catalogue, and adds fields that map the object to its corresponding layer in GeoServer. When your users upload a layer via the user interface, the layer is imported to GeoServer and a record is added to GeoNode's database to represent that GeoServer layer within GeoNode itself.

The Layer model class provides a set of helper methods that are used to perform operations on a Layer object, and also to return things such as the list of Download or Metadata links for that layer. Additional classes are used to model the layers Attributes, Styles, Contacts and Links. The Django signals framework is used to invoke specific functions to synchronize with GeoServer before and after the layer is saved.

The views in the layers app are used to perform functions such as uploading, replacing, removing or changing the points of contact for a layer, and views are also used to update layer styles, download layers in bulk or change a layers permissions.

The forms module in the layer app is used to drive the user interface forms necessary for performing the business logic that the views provide.

The Layers app also includes a set of templates that are paired with views and used to drive the user interface. A small set of layer template tags is also used to help drive the layer explore and search pages.

Some helper modules such as geonode.layers.metadata and geonode.layers.ows are used by the layer views to perform specific functions and help keep the main views module more concise and legible.

Additionally, the GeoNode specific management commands are a part of the geonode.layers app.

You should spend some time to review the layers app through GitHubs code browsing interface.

https://github.com/GeoNode/geonode/tree/master/geonode/layers

geonode.maps The geonode.maps app is used to group together GeoNodes multi layer map functionality. The Map and MapLayer objects are used to model and implement maps created with the GeoExplorer application. The Map object also extends from the ResourceBase class which provides the ability to manage a full set of metadata fields for a Map.

The views in the maps app perform many of the same functions as the views in the layers app such as adding, changing, replacing or removing a map and also provide the endpoints for returning the map configuration from the database that is used to initialize the GeoExplorer app.

The maps app also includes a set of forms, customization of the Django admin, some utility functions and a set of templates and template tags.

You can familiarize yourself with the maps app on GitHub.

https://github.com/GeoNode/geonode/tree/master/geonode/layers

geonode.security The geonode.security app is used to provide object level permissions within the GeoNode Django application. It is a custom Django authentication backend and is used to assign Generic, User and Group Permissions to Layers, Maps and other objects in the GeoNode system. Generic permissions are used to enable public anonymous or authenticated viewing and/or editing of your data layers and maps, and User and Group specific permissions are used to allow specific users or groups to access and edit your layers.

geonode.search The geonode.search module provides the search API that is used to drive the GeoNode search pages. It is configured to index layers, maps, documents and profiles, but is extensible to allow you to use it to index your own model classes. This module is currently based on the Django ORM and as such has a limited set of search features, but the GeoNode development team is actively working on making it possible to use this module with more feature-rich search engines.

geonode.catalogue The geonode.catalogue app provides a key set of metadata catalogue functions within GeoNode itself. GeoNode is configured to use an integrated version of the pycsw library to perform these functions, but can also be configured to use any OGC compliant CS-W implementation such as GeoNetwork or Deegree. The metadata app allows users to import and/or edit metadata for their layers, maps and documents, and it provides an OGC compliant search interface for use in federating with other systems.

geonode.geoserver The geonode.geoserver module is used to interact with GeoServer from within GeoNode's python code. It relies heavily on the gsconfig library which addresses GeoServer's REST configuration API. Additionally, the geonode.geoserver.uploader module is used to interact with GeoServers Importer API for uploading and configuring layers.

geonode.people The geonode.people module is used to model and store information about both GeoNode users and people outside of the system who are listed as Points of Contact for particular layers. It is the foundational module for GeoNode's social features. It provides a set of forms for users to edit and manage their own profiles as well as to view and interact with the profiles of other users.

geoexplorer GeoNode's core GIS client functions are performed by GeoExplorer. The GeoExplorer app is in turn based on GeoExt, OpenLayers and ExtJS. It provides functionality for constructing maps, styling layers and connecting to remote services. GeoExplorer is the reference implementation of the OpenGeo Suite SDK which is based on GXP. GeoNode treats GeoExplorer as an external module that is used out of the box in GeoNode, but it is possible for you to create your own Suite SDK app and integrate it with GeoNode.

Static Site The front end of GeoNode is composed of a set of core templates, specific templates for each module, cascading style sheets to style those pages and a set of javascript modules that provide the interactive functionality in the site.

Templates GeoNode includes a basic set of core templates that use Django's template inheritance system to provide a modular system for constructing the web pages in GeoNode's interface. These core templates drive the overall page layout and things like the home page. You will start the process of customizing your GeoNode instance by overriding these templates, so you should familiarize yourself with their tructure and how they inherit from each other to drive the pages.

Additionally, most of the apps described above have their own set of templates that are used to drive the pages for each module. You may also want to override these templates for your own purposes and as such should familiarize yourself with a few of the key ones.

CSS GeoNode's css is based on Twitter's Bootstrap Library which uses the lessc dynamic stylesheet language. GeoNode extends from the basic Boostrap style and you are able to create your own bootstrap based style to customize the look and feel of your own GeoNode instance. Sites like bootswatch.com also provide ready made styles that you can simply drop in to your project to change the style.

Javascript The interactive functionality in GeoNode pages is provided by the jQuery javascript framework and a set of jQuery plugins. The core set of GeoNode javascript modules closely aligns with the apps described above, and there are also a few pieces of functionality provided as javascript modules that are used through out all of the apps. You are able to add your own jQuery code and/or plugins to perform interactive functionality in your own application.

Exercises

Shell and Utilities

1. *ssh* into your virtual machine or other instance
2. *sudo* to modify the *sshd_config* settings to verify disabling of dns resolution (UseDNS=no)
3. install a command line helper

```
$ sudo apt-get install bash-completion
```

1. exercise command completion

```
$ apt-get install <TAB><TAB>
```

1. activate/deactivate the *virtualenv* on your instance

```
$ source /var/lib/geonode/bin/activate
$ deactivate
```

1. set the *DJANGO_SETTINGS_MODULE* env variable

```
$ export DJANGO_SETTINGS_MODULE=geonode.settings
```

1. install the *httpie* utility via pip

```
$ pip install httpie
$ http http://localhost/geoserver/rest
$ http -a admin http://localhost/geoserver/rest
<type in password - geoserver>
```

Python

1. launch *ipython* and experiment

```
> x = "some text"
> x.<TAB><TAB>
> x.split.__doc__
> ?
```

1. execute a script with *ipython* and open the REPL

```
$ echo "twos = [ x*2 for x in range(5)]" > test.py
$ ipython -i test.py
> twos
```

Install GeoNode for Development

In order to install Geonode 2.0 in developing mode on Ubuntu 12.04 the following steps are required:

For Windows: (_install_win_devmode)

1. install build tools and libraries
2. install dependencies (Python and Java) and supporting tools
3. add PPA repository
4. set up a virtual environment (virtualenv)
5. clone geonode from github and install it in the virtual environment
6. run paver to get install geoserver and start the development servers

Note: The following steps have to be executed in your terminal. The steps have to be done as a root user, therefore don´t forget to type sudo in front!

1. retrieve latest apt-get list

```
$ sudo apt-get update
```

2. Install build tools and libraries

```
$ sudo apt-get install -y build-essential libxml2-dev libxslt1-dev libpq-dev zlib1g-dev
```

3. Install dependencies

   Python native dependencies

```
$ sudo apt-get install -y python-dev python-imaging python-lxml python-pyproj python-shapely pyt
```

   Install Python Virtual Environment

```
$ sudo pip install virtualenvwrapper
```

   Java dependencies

```
$ sudo apt-get install -y --force-yes openjdk-6-jdk ant maven2 --no-install-recommends
```

   supporting tools

```
$ sudo apt-get install -y git gettext
```

4. Node and tools required for static development

   This is required for static development

```
$ sudo add-apt-repository -y ppa:chris-lea/node.js
$ sudo apt-get update
$ sudo apt-get install -y nodejs
$ sudo npm install -y -g bower
$ sudo npm install -y -g grunt-cli
```

5. Set up a virtual environment

   Here is where Geonode will later be running.

   Add the virtualenvwrapper to your new environement

```
    $ export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python
    $ export WORKON_HOME=~/.venvs
    $ source /usr/local/bin/virtualenvwrapper.sh
    $ export PIP_DOWNLOAD_CACHE=$HOME/.pip-downloads
```

   set up the local virtual environment for Geonode

```
    $ mkvirtualenv geonode
    $ workon geonode
```

   This creates a new directory where you want your project to be and creates a new virtualenvironment

6. Get the code

   To download the latest geonode version from github, the command *clone* is used

```
$ git clone https://github.com/GeoNode/geonode.git
```

7. Install GeoNode in the new active local virtualenv

```
$ pip install -e geonode --use-mirrors
$ cd geonode
```

If the install fails because of an error related to pyproj not being verified (happens on pip 1.5), use the following:

```
$ pip install -e geonode --use-mirrors --allow-external pyproj --allow-unverified pyproj
```

8. Compile and Start the server

   The last step is to compile GeoServer and setup

```
$ paver setup
```

9. Now we can start our geonode instance

```
$ paver start
```

   Visit the geonode site by typing http://localhost:8000 into your browser window.

   If the start fails because of an import error related to osgeo, then please consult the GDAL for Development Guide.

10. To stop the server

    type hold **Ctrl c** on your keyboard to stop the server

    now type:

```
$ paver stop    # to stop all django, geoserver services
```

11. Next create a superuser for your django geonode

    Create a superuser so you can log on to your local geonode installation at http://localhost:8000

```
$ django-admin.py createsuperuser --settings=geonode.settings
```

Start working on Geonode the next day after install

With every restart of your machine, you have to restart geonode as well. That means, you will not be able to open http://localhost:8000 directly after starting your machine new. In order to be able to use geonode now, you have to activate your virtualenvironment and to start the development servers.

Note: *username* is the name of your machine and personal folder!

1. Activate virtualenv

   To activate your virtualenv you just need to type

```
$ workon geonode
```

   or

```
$ source /home/username/.venvs/geonode/bin/activate
```

   Note: Be careful with the path, it might not be the same for you!

2. Start the server

```
$ cd geonode
$ paver start_geoserver
$ paver start_django
```

Now you are able to access http://localhost:8000 again.

Note: Remember that you have to do these steps each time you restart your machine!!

Hint: Now you've followed these installation instructions, geonode is running in development mode. This also means that you are using all the default settings of geonode. If you want to change them, e.g use Tomcat instead of Jetty, or Postgresql instead of sqlite3, you may follow the steps from the section **Configure Manually** in custom_install.

Customized GeoNode Projects

This module will teach you about how to set up and customize your own GeoNode-based project by changing the theme, adding additional modules, and integrating with other systems. When complete, you should understand how Downstream GeoNode projects work, and how to set up a project of your own.

Introduction to GeoNode Projects GeoNode enables you to set up a complete site simply by installing the packages and adding your data. If you want to create your own project based on GeoNode, there are a several options available that enable you to customize the look and feel of your GeoNode site. You can add additional modules that are necessary for your own use case and to integrate your GeoNode project with other external sites and services.

This module assumes that you have installed a GeoNode site with the Ubuntu Packages and that you have a working GeoNode based on that setup. If you want to follow this same methodology on a different platform, you can follow this module and adapt as necessary for your environment.

Overview GeoNode is an out-of-the-box, full-featured Spatial Data Infrastructure solution, but many GeoNode implementations require either customization of the default site or the use of additional modules, whether they be third-party Django Pluggables or modules developed by a GeoNode implementer.

There are quite a few existing Downstream GeoNode projects some of which follow the methodology described in this module. You should familiarize yourself with these projects and how and why they extend GeoNode. You should also carefully think about what customization and additional modules you need for your own GeoNode-based project and research the options that are available to you. The Django Packages site is a great place to start looking for existing modules that may meet your needs.

Existing downstream GeoNode projects

- Harvard Worldmap
- MapStory
- Risiko/SAFE

Django template projects GeoNode follows the Django template projects paradigm introduced in Django 1.4. At a minimum, a Django project consists of a `settings.py` file and a `urls.py` file; Django apps are used to add specific pieces of functionality. The GeoNode development team has created a template project which contains these required files with all the GeoNode configuration you need to get up and running with your own GeoNode project. If you would like learn more about Django projects and apps, you should consult the Django Documentation

Setting up your GeoNode project

Warning: These instructions are only valid if you've installed GeoNode in development mode!!

Note: If you've installed GeoNode using apt-get and want to customize your GeoNode then go to the *Customize the look and feel* of the Administrator Workshop.

This section will walk you through the steps necessary to set up your own GeoNode project. It assumes that you have installed GeoNode in developing mode.

Setup steps If you are working remotely, you should first connect to the machine that has your GeoNode installation. You will need to perform the following steps in a directory where you intend to keep your newly created project.

1. Activate GeoNode's Virtual Environment

```
$ workon geonode
```

2. Create your GeoNode project from the template

```
$ django-admin.py startproject my_geonode --template=https://github.com/GeoNode/geonode-project/
$ cd my_geonode
```

3. Update your local_settings.py. You will need to check the local_settings.py that is included with the template project and be sure that it reflects your own local environment. You should pay particular attention to the Database settings especially if you intend to reuse the database that was set up with your base GeoNode installation.
4. Run the test server

```
$ python manage.py runserver
```

5. Visit your new GeoNode site at http://localhost:8000.

Source code revision control It is recommended that you immediately put your new project under source code revision control. The GeoNode development team uses Git and GitHub and recommends that you do the same. If you do not already have a GitHub account, you can easily set one up. A full review of Git and distributed source code revision control systems is beyond the scope of this tutorial, but you may find the Git Book useful if you are not already familiar with these concepts.

1. Create a new repository in GitHub. You should use the GitHub user interface to create a new repository for your new project.
2. Initialize your own repository:

```
$ git init
```

3. Add the remote repository reference to your local git configuration:

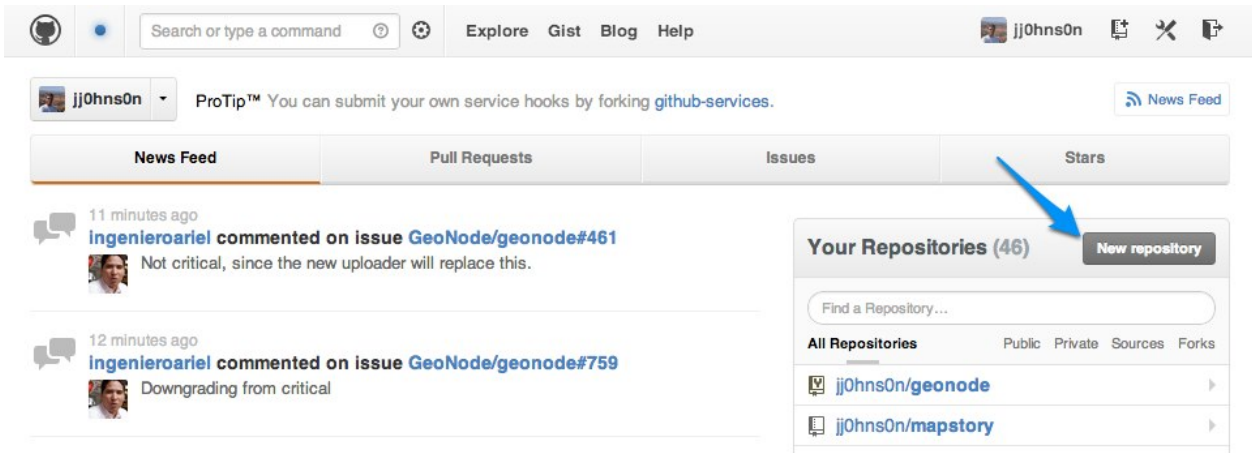
```
$ git remote add
```

4. Add your project files to the repository:

```
$ git add .
```

5. Commit your changes:

```
$ git commit -am "Initial commit"
```

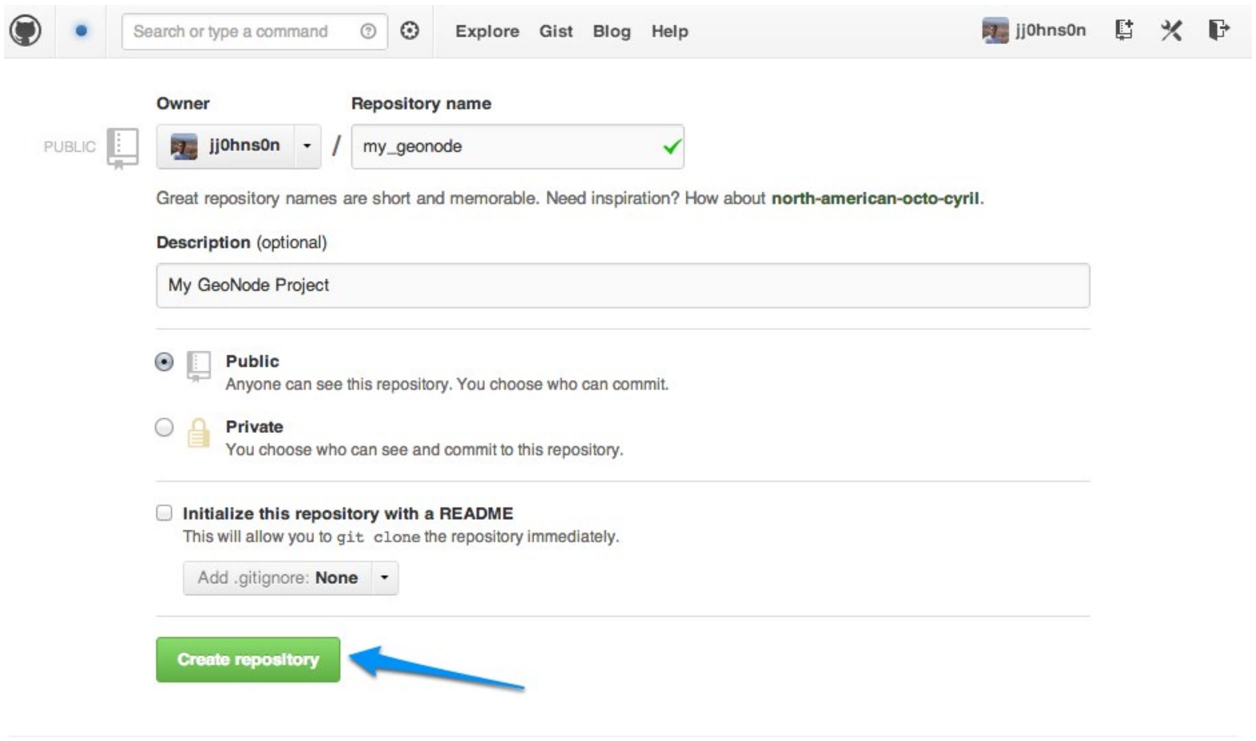


ProTip™ You can submit your own service hooks by forking github-services.

11 minutes ago
ingenieroariel commented on issue GeoNode/geonode#461
Not critical, since the new uploader will replace this.

12 minutes ago
ingenieroariel commented on issue GeoNode/geonode#759
Downgrading from critical

Fig. 3.76: *Creating a new GitHub Repository From GitHub's Homepage*



Owner: jj0hns0n / Repository name: my_geonode

Great repository names are short and memorable. Need inspiration? How about north-american-octo-cyril.

Description (optional): My GeoNode Project

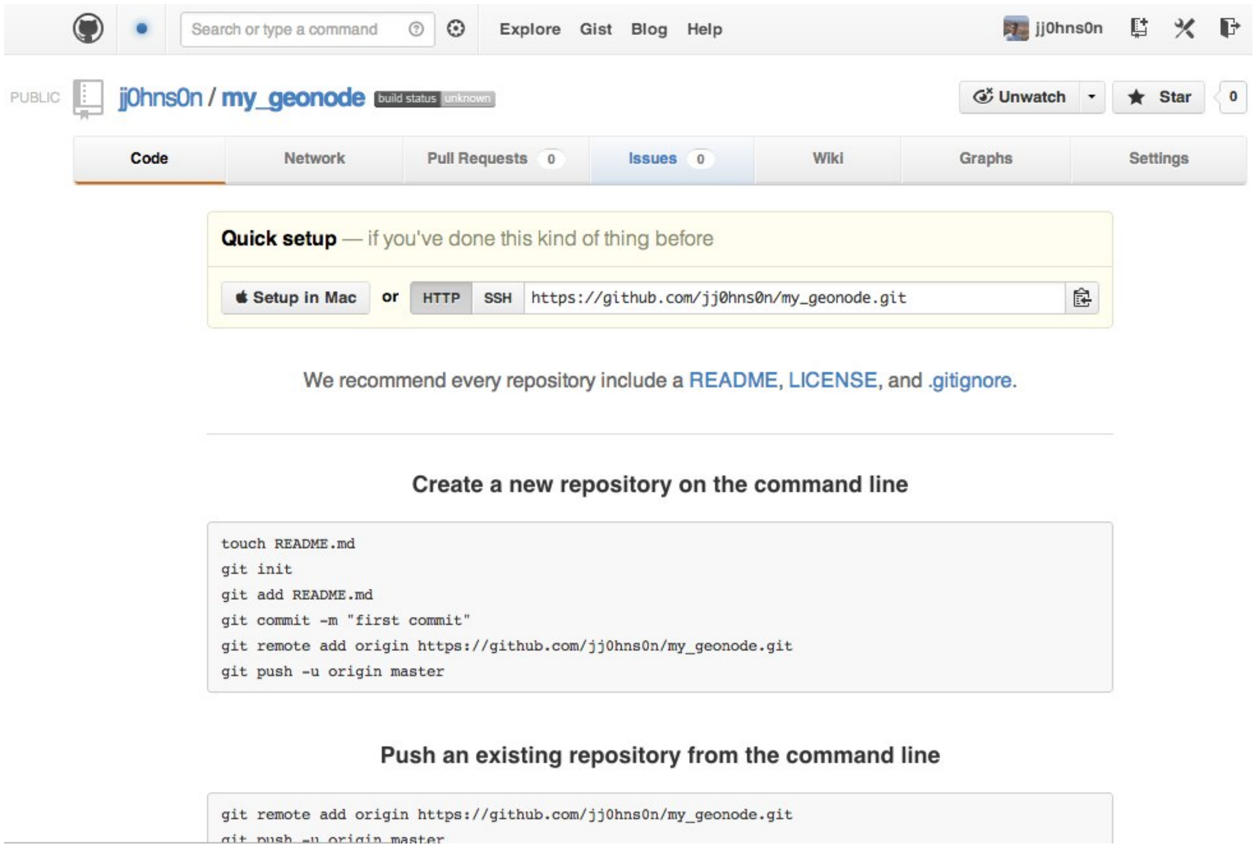
[✓ Public] Anyone can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

☐ Initialize this repository with a README
This will allow you to `git clone` the repository immediately.

Add .gitignore: None

Fig. 3.77: *Specifying new GitHub Repository Parameters*



Quick setup — if you've done this kind of thing before

https://github.com/jj0hns0n/my_geonode.git

We recommend every repository include a README, LICENSE, and .gitignore.

Create a new repository on the command line

```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/jj0hns0n/my_geonode.git
git push -u origin master
```

Push an existing repository from the command line

```
git remote add origin https://github.com/jj0hns0n/my_geonode.git
git push -u origin master
```

Fig. 3.78: *Your new Empty GitHub Repository*

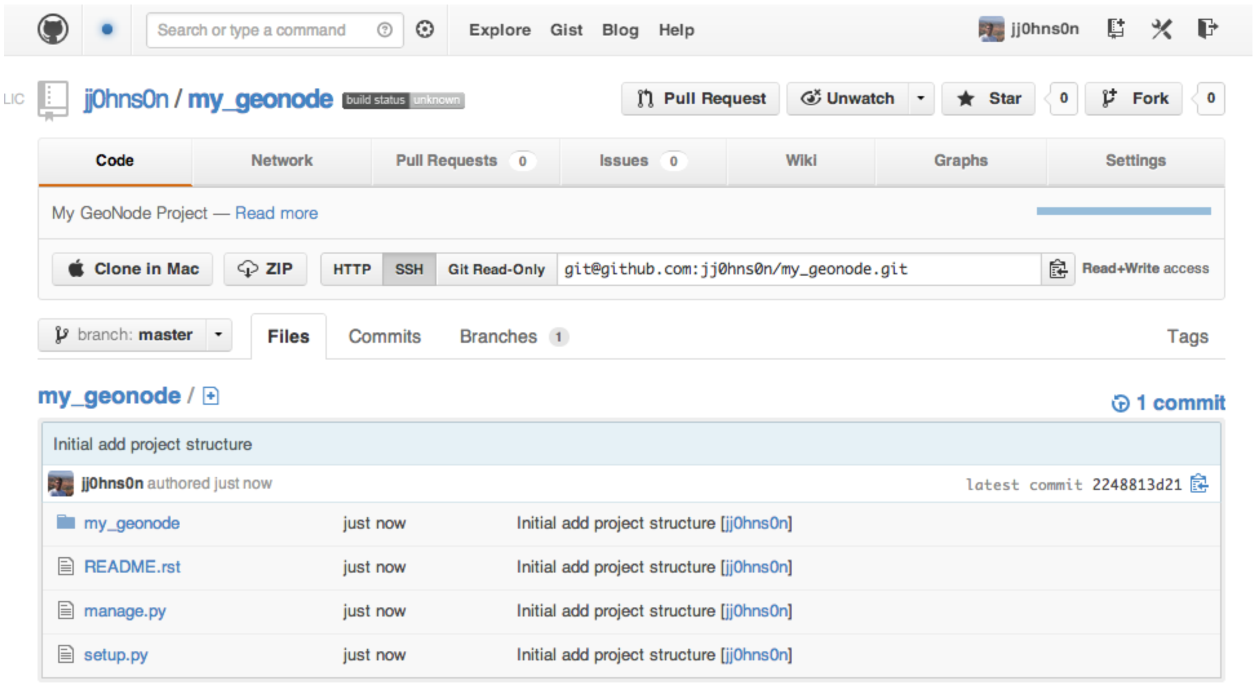
6. Push to the remote repository:

```
$ git push origin master
```

Project structure Your GeoNode project will now be structured as depicted below:

```
|-- README.rst
|-- manage.py
|-- my_geonode
|   |-- __init__.py
|   |-- settings.py
|   |-- static
|   |   |-- README
|   |   |-- css
|   |   |   |-- site_base.css
|   |   |-- img
|   |   |   |-- README
|   |   |-- js
|   |       |-- README
|   |-- templates
|   |   |-- site_base.html
|   |   |-- site_index.html
|   |-- urls.py
|   |-- wsgi.py
|-- setup.py
```

You can also view your project on GitHub.



My GeoNode Project — Read more

git@github.com:jj0hns0n/my_geonode.git

my_geonode /

Initial add project structure

jj0hns0n authored just now — latest commit 2248813d21

my_geonode	just now	Initial add project structure [jj0hns0n]
README.rst	just now	Initial add project structure [jj0hns0n]
manage.py	just now	Initial add project structure [jj0hns0n]
setup.py	just now	Initial add project structure [jj0hns0n]

Fig. 3.79: *Viewing your project on GitHub*

Each of the key files in your project are described below.

manage.py `manage.py` is the main entry point for managing your project during development. It allows running all the management commands from each app in your project. When run with no arguments, it will list all of the management commands.

settings.py `settings.py` is the primary settings file for your project. It is quite common to put all sensible defaults here and keep deployment-specific configuration in the `local_settings.py` file. All of the possible settings values and their meanings are detailed in the Django documentation.

A common paradigm for handing 'local settings' (and in other areas where some Python module may not be available) is:

```
try:
    from local_settings import *
except:
    pass
```

This is not required and there are many other solutions to handling varying deployment configuration requirements.

Note: If you define something in `local_settings.py`, take note of any dependent configuration. For example, `settings.MAP_BASELAYERS` makes use of `settings.OGC_SERVER`. If you redefine `OGC_SERVER` in `local_settings.py`, `MAP_BASELAYERS` may need to be updated as well.

urls.py `urls.py` is where your application specific URL routes go. Additionally, any *overrides* can be placed here, too.

wsgi.py This is a generated file to make deploying your project to a WSGI server easier. Unless there is very specific configuration you need, `wsgi.py` can be left alone.

setup.py There are several packaging options in python but a common approach is to place your project metadata (version, author, etc.) and dependencies in `setup.py`.

This is a large topic and not necessary to understand while getting started with GeoNode development but will be important for larger projects and to make development easier for other developers.

More: http://docs.python.org/2/distutils/setupscript.html

static The `static` directory will contain your fixed resources: css, html, images, etc. Everything in this directory will be copied to the final media directory (along with the *static* resources from other apps in your project).

templates All of your projects templates go in the `templates` directory. While no organization is required for your project specific templates, when overriding or replacing a template from another app, the path must be the same as the template to be replaced.

Staying in sync with mainline GeoNode If you want to stay in sync with the mainline GeoNode you have to execute the following commands. Your own project should not be adversely affected by these changes, but you will receive bug fixes and other improvements by staying in sync.

```
$ git fetch upstream master
$ git merge
```

Theming your GeoNode project There are a range of options available to you if you want to change the default look and feel of your GeoNode project. Since GeoNode's style is based on Bootstrap you will be able to make use of all that Bootstrap has to offer in terms of theme customization. You should consult Bootstrap's documentation as your primary guide once you are familiar with how GeoNode implements Bootstrap and how you can override GeoNode's theme and templates in your own project.

Logos and graphics GeoNode intentionally does not include a large number of graphics files in its interface. This keeps page loading time to a minimum and makes for a more responsive interface. That said, you are free to customize your GeoNode's interface by simply changing the default logo, or by adding your own images and graphics to deliver a GeoNode experience the way you envision int.

Your GeoNode project has a directory already set up for storing your own images at `<my_geonode>/static/img`. You should place any image files that you intend to use for your project in this directory.

Let's walk through an example of the steps necessary to change the default logo.

1. Change into the `img` directory:

```
$ cd <my_geonode>/static/img
```

2. If you haven't already, obtain your logo image. The URL below is just an example, so you will need to change this URL to match the location of your file or copy it to this location:

```
$ wget http://www2.sta.uwi.edu/~anikov/UWI-logo.JPG
$ cd ../../..
```

3. Override the CSS that displays the logo by editing `<my_geonode>/static/css/site_base.css` with your favorite editor and adding the following lines, making sure to update the width, height, and URL to match the specifications of your image.

```
.nav-logo {
    width: 373px;
    height: 79px;
    background: url(../img/UWI-logo.JPG) no-repeat;
}
```

4. Restart your GeoNode project and look at the page in your browser:

```
$ python manage.py runserver
```

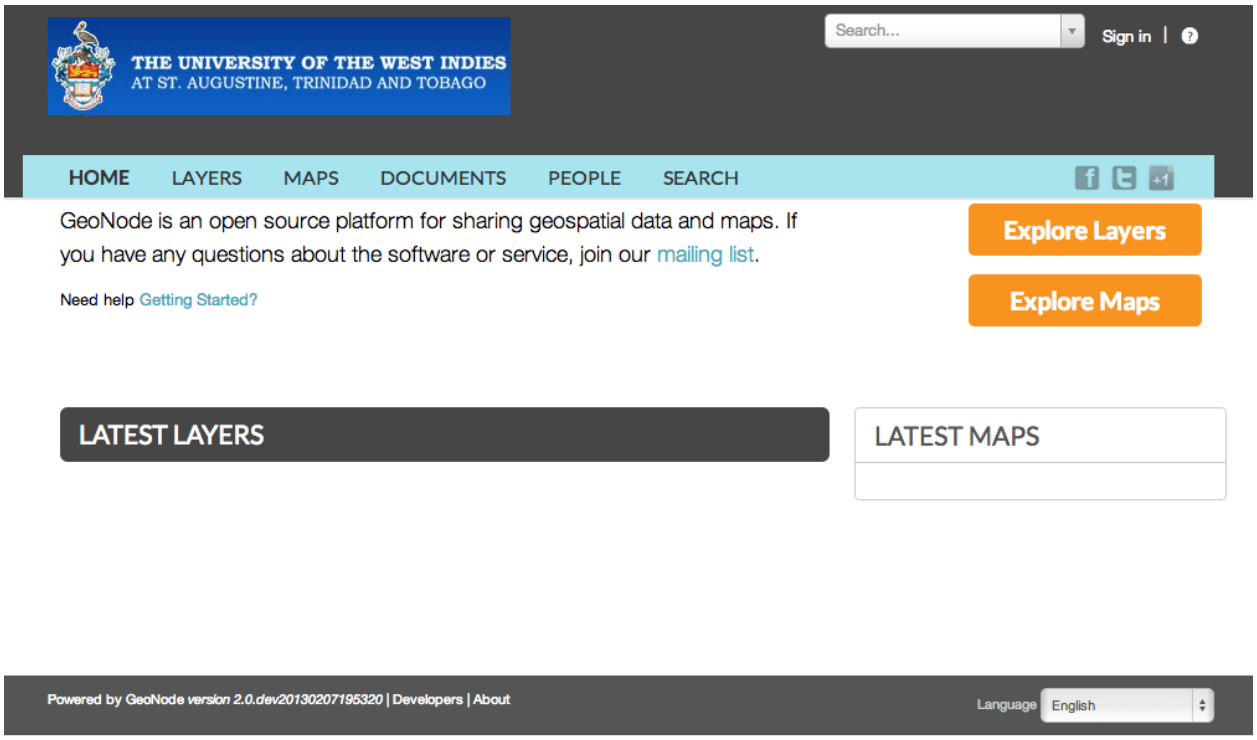
Visit your site at http://localhost:8000/ or the remote URL for your site.

You can see that the header has been expanded to fit your graphic. In the following sections you will learn how to customize this header to make it look and function the way you want.

Note: You should commit these changes to your repository as you progress through this section, and get in the habit of committing early and often so that you and others can track your project on GitHub. Making many atomic commits and staying in sync with a remote repository makes it easier to collaborate with others on your project.

Cascading Style Sheets In the last section you already learned how to override GeoNode's default CSS rules to include your own logo. You are able to customize any aspect of GeoNode's appearance this way. In the last screenshot, you saw that the main area in the homepage is covered up by the expanded header.

First, we'll walk through the steps necessary to displace it downward so it is no longer hidden, then change the background color of the header to match the color in our logo graphic.



GeoNode is an open source platform for sharing geospatial data and maps. If you have any questions about the software or service, join our mailing list.

Need help Getting Started?

Powered by GeoNode version 2.0.dev20130207195320 | Developers | About

Fig. 3.80: *Custom logo*

1. Reopen `<my_geonode>/static/css/site_base.css` in your editor and add the following rule after the one added in the previous step:

```
.content-wrap {
    margin: 75px 75px;
}
```

2. Add a rule to change the background color of the header to match the logo graphic we used:

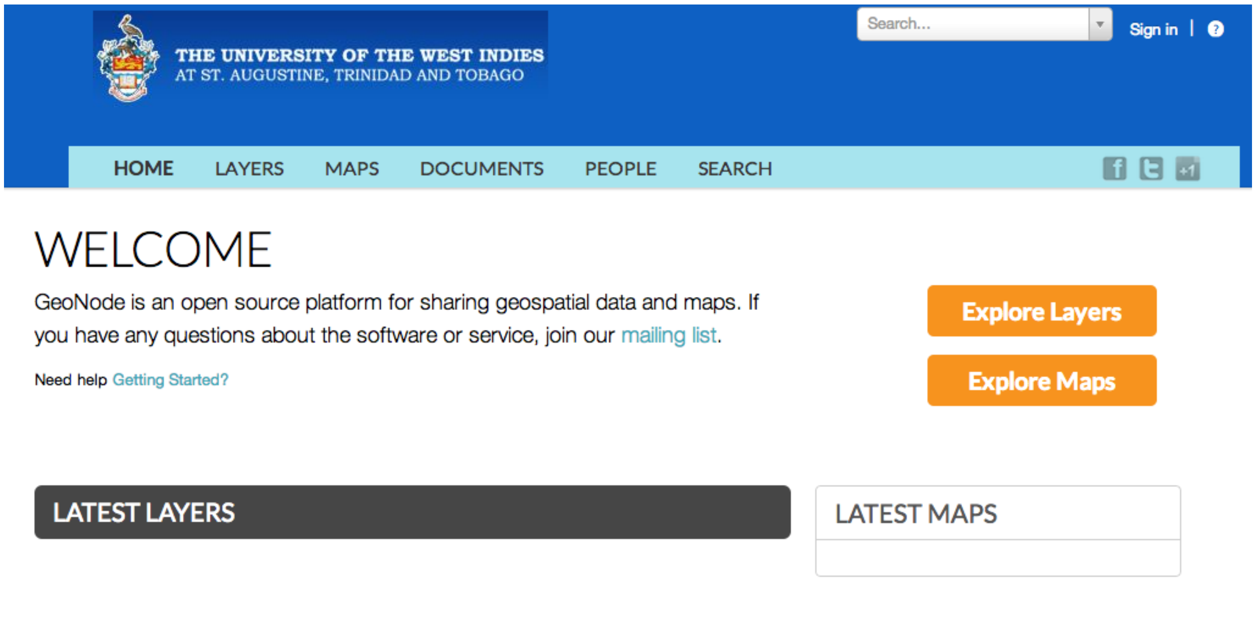
```
.navbar .navbar-inner {
    background: #0e60c3;
}
```

3. Your project CSS file should now look like this:

```
.nav-logo {
    width: 373px;
    height: 79px;
    background: url(../img/UWI-logo.JPG) no-repeat;
}

.content-wrap {
    margin: 75px 75px;
}

.navbar .navbar-inner {
    background: #0e60c3;
}
```

4. Restart the development server and reload the page:

```
$ python manage.py runserver
```



GeoNode is an open source platform for sharing geospatial data and maps. If you have any questions about the software or service, join our mailing list.

Need help Getting Started?

Fig. 3.81: *CSS overrides*

Note: You can continue adding rules to this file to override the styles that are in the GeoNode base CSS file which is built from base.less. You may find it helpful to use your browser's development tools to inspect elements of your site that you want to override to determine which rules are already applied. See the screenshot below. Another section of this workshop covers this topic in much more detail.

Templates and static pages Now that we have changed the default logo and adjusted our main content area to fit the expanded header, the next step is to update the content of the homepage itself. Your GeoNode project includes two basic templates that you will use to change the content of your pages.

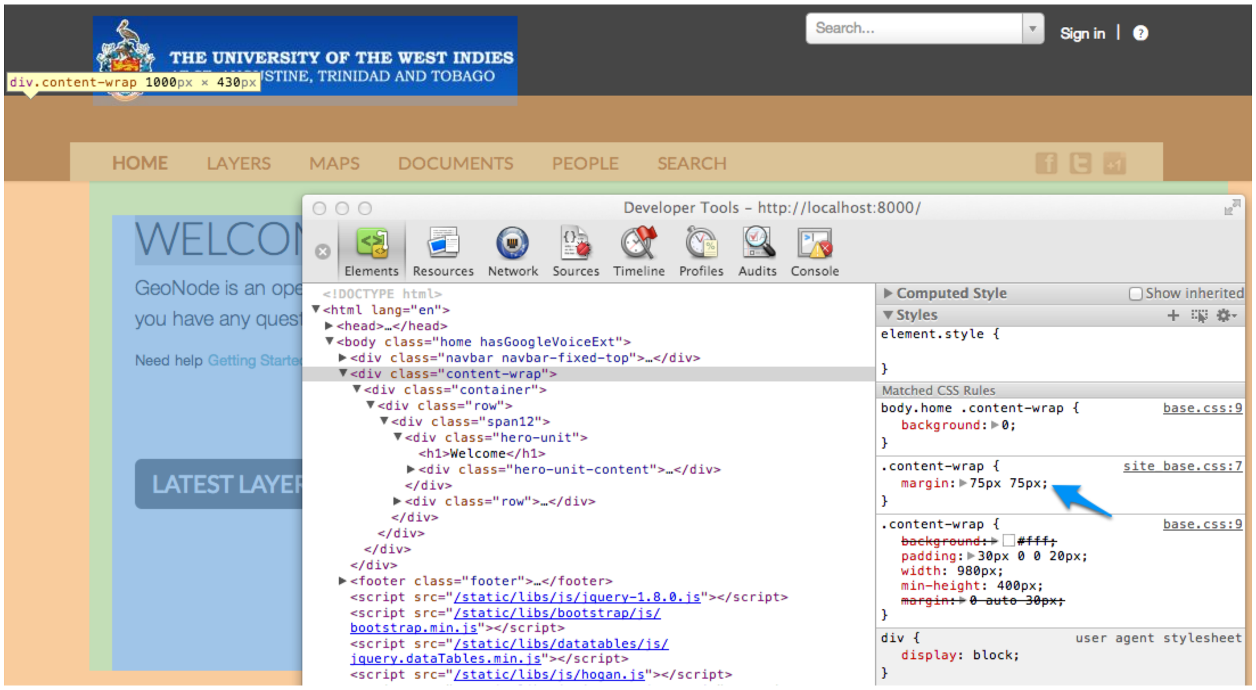
The file `site_base.html` (in `<my_geonode>/templates/`) is the basic template that all other templates inherit from and you will use it to update things like the header, navbar, site-wide announcement, footer, and also to include your own JavaScript or other static content included in every page in your site. It's worth taking a look at GeoNode's base file on GitHub. You have several blocks available to you to for overriding, but since we will be revisiting this file in future sections of this workshop, let's just look at it for now and leave it unmodified.

Open `<my_geonode>/templates/site_base.html` in your editor:

```
{% extends "base.html" %}
{% block extra_head %}
    <link href="{{ STATIC_URL }}css/site_base.css" rel="stylesheet"/>
{% endblock %}
```

You will see that it extends from `base.html`, which is the GeoNode template referenced above and it currently only overrides the `extra_head` block to include our project's `site_base.css` which we modified in the previous section. You can see on line 14 of the GeoNode base.html template that this block is included in an empty state and is set up specifically for you to include extra CSS files as your project is already set up to do.

Now that we have looked at `site_base.html`, let's actually override a different template.



```
<!DOCTYPE html>
▼<html lang="en">
  ▶<head>…</head>
  ▼<body class="home hasGoogleVoiceExt">
    ▶<div class="navbar navbar-fixed-top">…</div>
    ▼<div class="content-wrap">
      ▼<div class="container">
        ▼<div class="row">
          ▼<div class="span12">
            ▼<div class="hero-unit">
               <h1>Welcome</h1>
              ▶<div class="hero-unit-content">…</div>
             </div>
            ▶<div class="row">…</div>
           </div>
         </div>
       </div>
     </div>
    ▶<footer class="footer">…</footer>
     <script src="/static/libs/js/jquery-1.8.0.js"></script>
     <script src="/static/libs/bootstrap/js/
     bootstrap.min.js"></script>
     <script src="/static/libs/datatables/js/
     jquery.dataTables.min.js"></script>
     <script src="/static/libs/js/hogan.js"></script>
```

```
element.style {

}
body.home .content-wrap {                base.css:9
   background:▶0;
}
.content-wrap {                     site_base.css:7
   margin:▶75px 75px;
}
.content-wrap {                          base.css:9
   ~~background:▶ #fff;~~
   padding:▶30px 0 0 20px;
   width: 980px;
   min-height: 400px;
   ~~margin:▶0 auto 30px;~~
}
div {                        user agent stylesheet
   display: block;
}
```

Fig. 3.82: *Screenshot of using Chrome's debugger to inspect the CSS overrides*

The file `site_index.html` is the template used to define your GeoNode project's homepage. It extends GeoNode's default `index.html` template and gives you the option to override specific areas of the homepage like the hero area, but also allows you leave area like the "Latest Layers" and "Maps" and the "Contribute" section as they are. You are of course free to override these sections if you choose and this section shows you the steps necessary to do that below.

1. Open `<my_geonode>/templates/site_index.html` in your editor.
2. Edit the `<h1>` element on line 13 to say something other than "Welcome":

```
<h1>{% trans "UWI GeoNode" %}</h1>
```

3. Edit the introductory paragraph to include something specific about your GeoNode project:

```
<p>
    {% blocktrans %}
    UWI's GeoNode is setup for students and faculty to collaboratively
    create and share maps for their class projects. It is maintained by the
    UWI Geographical Society.
    {% endblocktrans %}
</p>
```

4. Change the *Getting Started* link to point to another website:

```
<span>
    For more information about the UWI Geographical society,
    <a href="http://uwigsmona.weebly.com/">visit our website</a>
</span>
```

5. Add a graphic to the hero area above the paragraph replaced in step 3:

```
<img src = 'http://uwigsmona.weebly.com/uploads/1/3/2/4/13241997/1345164334.png'>
```

6. Your edited `site_index.html` file should now look like this:

```
{% extends 'index.html' %}
{% load i18n %}
{% load maps_tags %}
{% load layers_tags %}
{% load pagination_tags %}
{% load staticfiles %}
{% load url from future %}
{% comment %}
This is where you can override the hero area block. You can simply modify the content below or r
{% endcomment %}
{% block hero %}
    <div class="hero-unit">
        <h1>{% trans "UWI GeoNode" %}</h1>
        <div class="hero-unit-content">
        <div class="intro">
            <img src = 'http://uwigsmona.weebly.com/uploads/1/3/2/4/13241997/1345164334.png'>
        <p>
            {% blocktrans %}
            UWI's GeoNode is setup for students and faculty to collaboratively
            create and share maps for their class projects. It is maintained by the
            UWI Geographical Society.
            {% endblocktrans %}
        </p>
        <span>
            For more information about the UWI Geographical society,
            <a href="http://uwigsmona.weebly.com/">visit our website</a>
        </span>
    </div>
    <div class="btns">
        <a class="btn btn-large" href="{% url "layer_browse" %}">
        {% trans "Explore Layers" %}
        </a>
        <a class="btn btn-large" href="{% url "maps_browse" %}">
        {% trans "Explore Maps" %}
        </a>
    </div>
</div>
{% endblock %}
```

7. Restart your GeoNode project and view the changes in your browser at http://localhost:8000/ or the remote URL for your site:

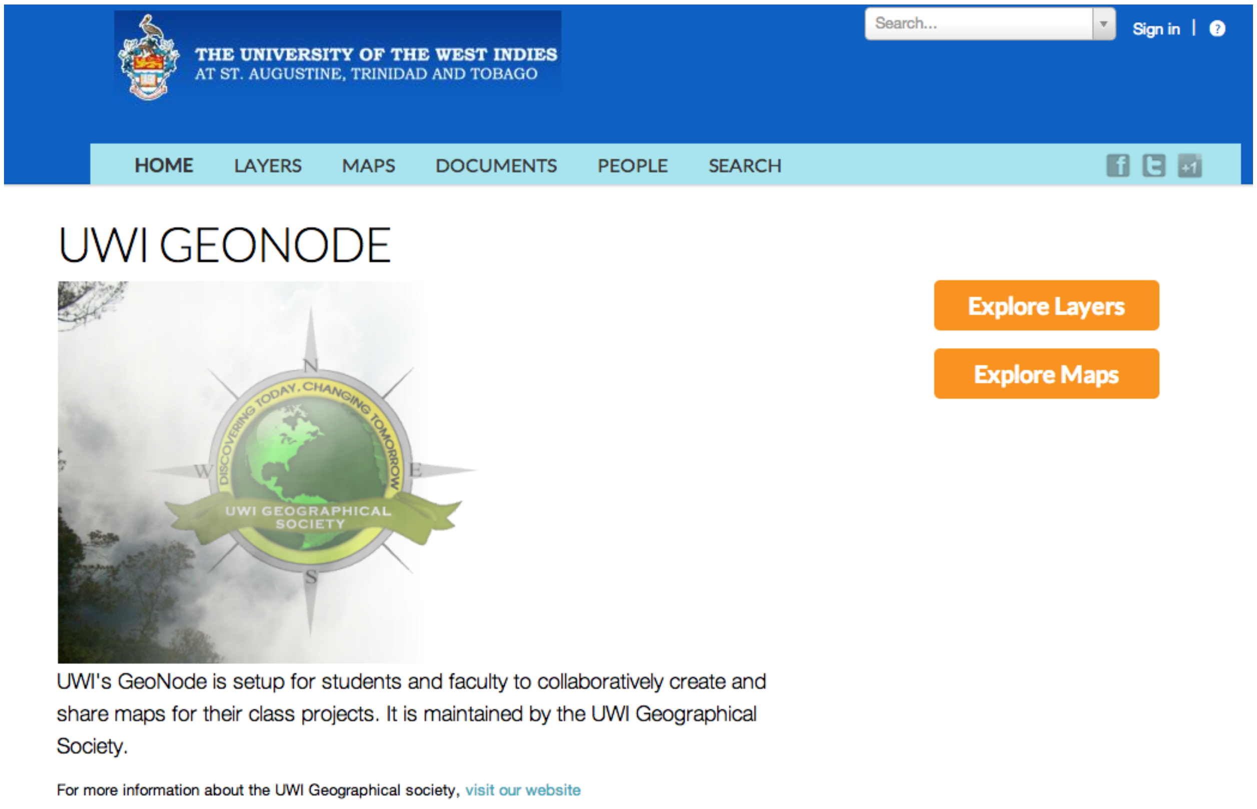
```
$ python manage.py runserver
```

From here you can continue to customize your `site_index.html` template to suit your needs. This workshop will also cover how you can add new pages to your GeoNode project site.

Other theming options You are able to change any specific piece of your GeoNode project's style by adding CSS rules to `site_base.css`, but since GeoNode is based on Bootstrap, there are many pre-defined themes that you can simply drop into your project to get a whole new look. This is very similar to WordPress themes and is a powerful and easy way to change the look of your site without much effort.

Bootswatch Bootswatch is a site where you can download ready-to-use themes for your GeoNode project site. The following steps will show you how to use a theme from Bootswatch in your own GeoNode site.

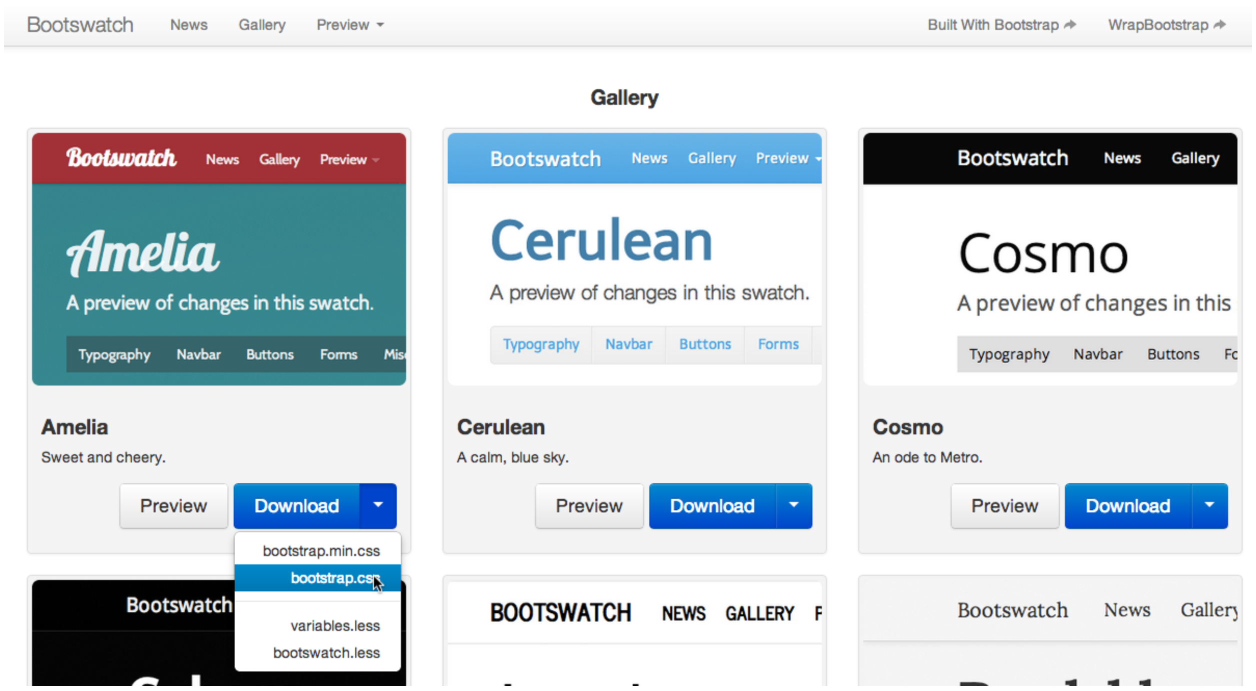
1. Visit http://bootswatch.com and select a theme (we will use Amelia for this example). Select the *download bootstrap.css option* in the menu:



UWI GEONODE

UWI's GeoNode is setup for students and faculty to collaboratively create and share maps for their class projects. It is maintained by the UWI Geographical Society.

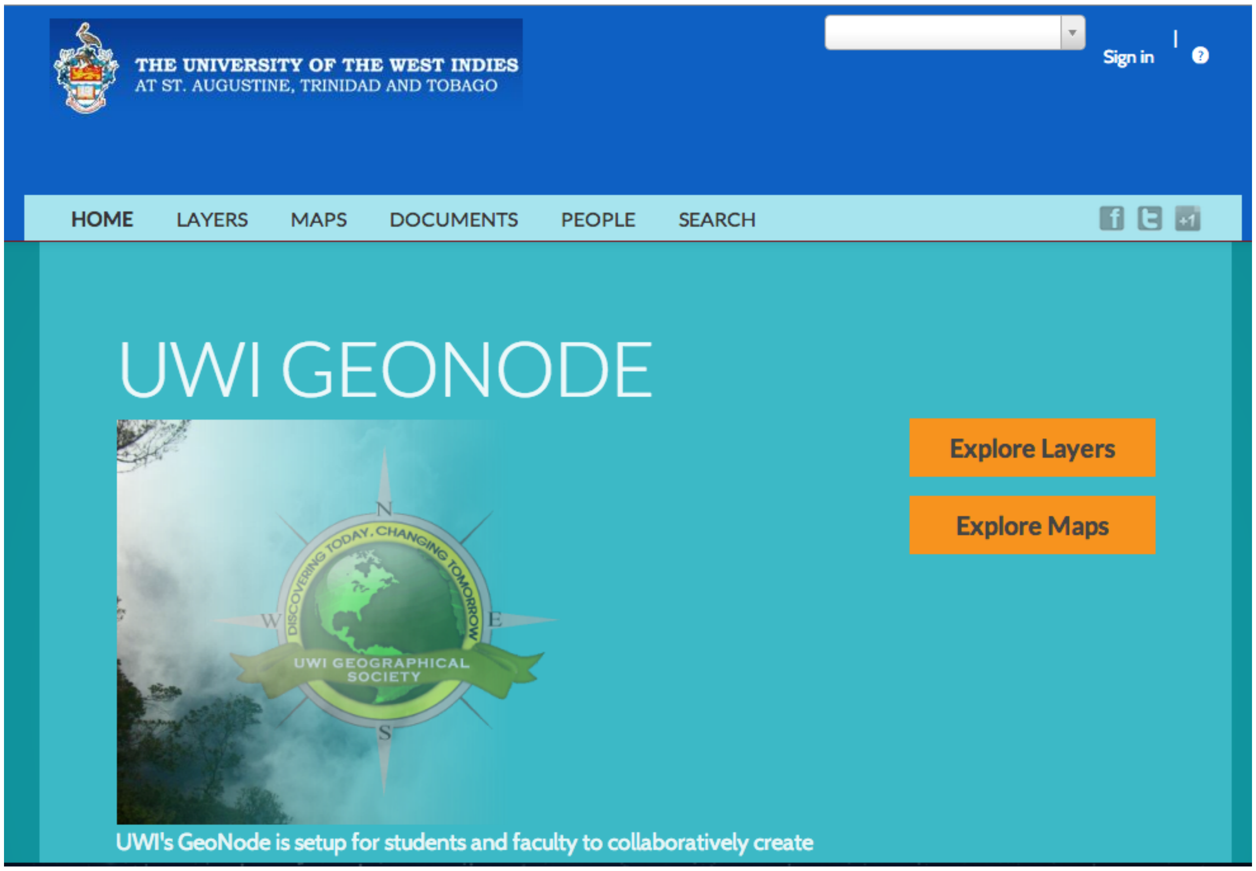
For more information about the UWI Geographical society, visit our website



2. Put this file in `<my_geonode>/static/css`.
3. Update the `site_base.html` template to include this file. It should now look like this:

```
{% extends "base.html" %}
{% block extra_head %}
    <link href="{{ STATIC_URL }}css/site_base.css" rel="stylesheet"/>
    <link href="{{ STATIC_URL }}css/bootstrap.css" rel="stylesheet"/>
{% endblock %}
```

4. Restart the development server and visit your site:


THE UNIVERSITY OF THE WEST INDIES
AT ST. AUGUSTINE, TRINIDAD AND TOBAGO
Sign in
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
UWI GEONODE
Explore Layers
Explore Maps
DISCOVERING TODAY, CHANGING TOMORROW
UWI GEOGRAPHICAL SOCIETY
UWI's GeoNode is setup for students and faculty to collaboratively create

Your GeoNode project site is now using the Amelia theme in addition to the changes you have made.

Adding additional Django apps to your GeoNode Project Since GeoNode is based on Django, your GeoNode project can be augmented and enhanced by adding additional third-party pluggable Django apps or by writing an app of your own.

This section of the workshop will introduce you to the Django pluggable app ecosystem, and walk you through the process of writing your own app and adding a blog app to your project.

Django pluggable apps The Django app ecosystem provides a large number of apps that can be added to your project. Many are mature and used in many existing projects and sites, while others are under active early-stage development. Websites such as Django Packages provide an interface for discovering and comparing all the apps that can plugged in to your Django project. You will find that some can be used with very little effort on your part, and some will take more effort to integrate.

Adding your own Django app Let's walk through the an example of the steps necessary to create a very basic Django polling app to and add it to your GeoNode project. This section is an abridged version of the Django tutorial itself and it is strongly recommended that you go through this external tutorial along with this section as it provides much more background material and a signficantly higher level of detail. You should become familiar with all of the information in the Django tutorial as it is critical to your success as a GeoNode project developer.

Throughout this section, we will walk through the creation of a basic poll application. It will consist of two parts:

- A public site that lets people view polls and vote in them.
- An admin site that lets you add, change, and delete polls.

1. Create app structure

   Since we have already created our GeoNode project from a template project, we will start by creating our app structure and then adding models:

```
    $ python manage.py startapp polls
```

   That'll create a directory polls, which is laid out like this:

```
polls/
    __init__.py
    models.py
    tests.py
    views.py
```

   This directory structure will house the poll application.

2. Add models

   The next step in writing a database web app in Django is to define your models—essentially, your database layout with additional metadata.

   In our simple poll app, we'll create two models: polls and choices. A poll has a question and a publication date. A choice has two fields: the text of the choice and a vote tally. Each choice is associated with a poll.

   These concepts are represented by simple Python classes.

   Edit the `polls/models.py` file so it looks like this:

```
from django.db import models

class Poll(models.Model):
    question = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
    def __unicode__(self):
        return self.question

class Choice(models.Model):
    poll = models.ForeignKey(Poll)
    choice = models.CharField(max_length=200)
    votes = models.IntegerField()
    def __unicode__(self):
        return self.choice
```

   That small bit of model code gives Django a lot of information. With it, Django is able to:

   - Create a database schema (CREATE TABLE statements) for this app.
   - Create a Python database-access API for accessing Poll and Choice objects.

   But first we need to tell our project that the polls app is installed.

Edit the `<my_geonode>/settings.py` file, and update the INSTALLED_APPS setting to include the string “polls”. So it will look like this:

```
INSTALLED_APPS = (

    # Apps bundled with Django
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.admin',
    'django.contrib.sitemaps',
    'django.contrib.staticfiles',
    'django.contrib.messages',
    'django.contrib.humanize',

    #Third party apps

    # <snip>

    # GeoNode internal apps
    'geonode.maps',
    'geonode.upload',
    'geonode.layers',
    'geonode.people',
    'geonode.proxy',
    'geonode.security',
    'geonode.search',
    'geonode.catalogue',
    'geonode.documents',

    # My GeoNode apps
    'polls',
)
```

Now Django knows to include the polls app. Let’s run another command:

```
$ python manage.py syncdb
```

The `syncdb` command runs the SQL from `sqlall` on your database for all apps in INSTALLED_APPS that don’t already exist in your database. This creates all the tables, initial data, and indexes for any apps you’ve added to your project since the last time you ran `syncdb`. `syncdb` can be called as often as you like, and it will only ever create the tables that don’t exist.

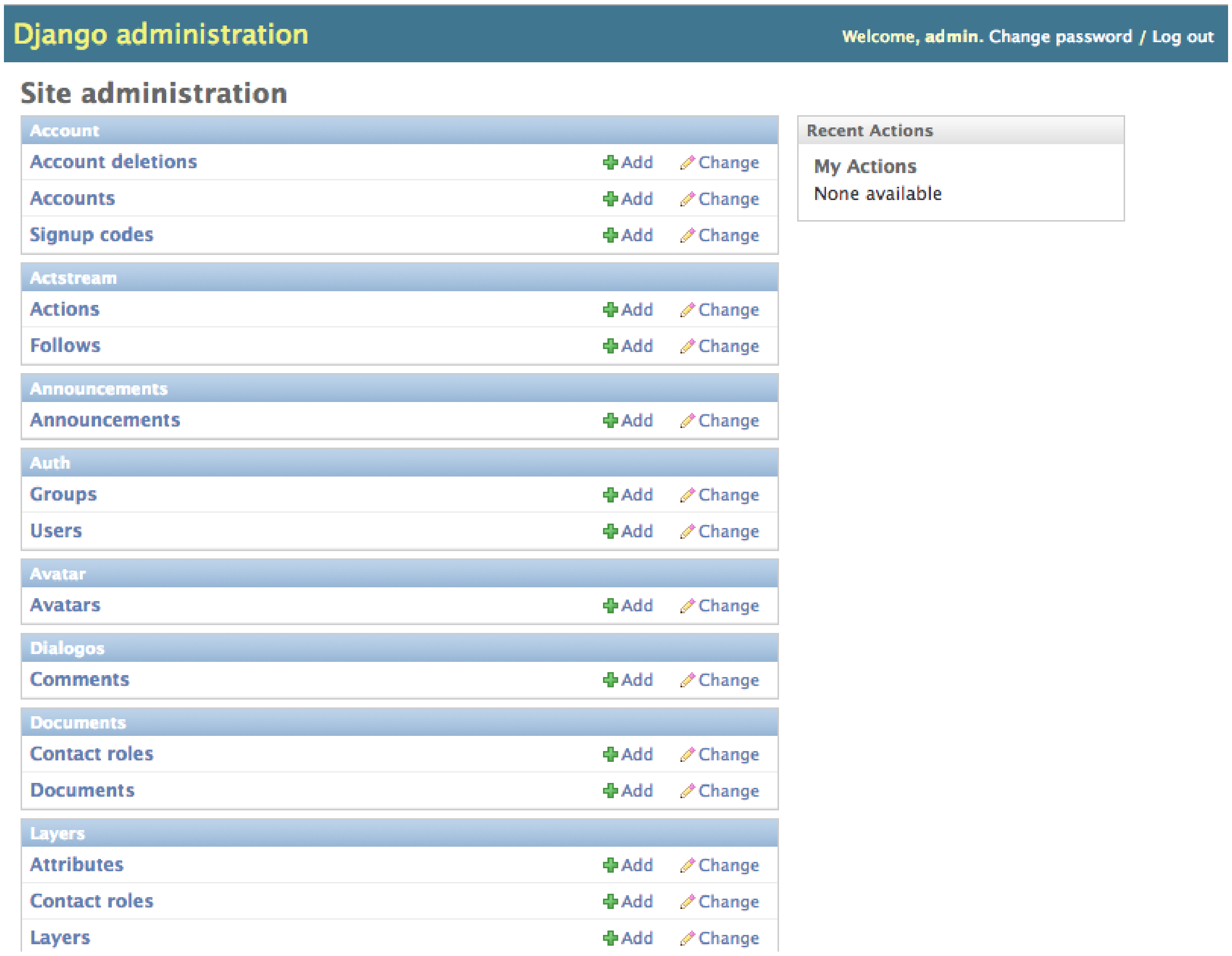
GeoNode uses south for migrations ...

3. Add Django Admin Configuration

   Next, let’s add the Django admin configuration for our polls app so that we can use the Django Admin to manage the records in our database. Create and edit a new file called `polls/admin.py` and make it look like the this:

```
from polls.models import Poll
from django.contrib import admin

admin.site.register(Poll)
```

Run the development server and explore the polls app in the Django Admin by pointing your browser to http://localhost:8000/admin/ and logging in with the credentials you specified when you first ran `syncdb`.



Django administration

Welcome, admin. Change password / Log out

Site administration

Account		
Account deletions	Add	Change
Accounts	Add	Change
Signup codes	Add	Change
Actstream		
Actions	Add	Change
Follows	Add	Change
Announcements		
Announcements	Add	Change
Auth		
Groups	Add	Change
Users	Add	Change
Avatar		
Avatars	Add	Change
Dialogos		
Comments	Add	Change
Documents		
Contact roles	Add	Change
Documents	Add	Change
Layers		
Attributes	Add	Change
Contact roles	Add	Change
Layers	Add	Change

Recent Actions

My Actions

None available

You can see all of the other apps that are installed as part of your GeoNode project, but we are specifically interested in the Polls app for now.

Next we will add a new poll via automatically generated admin form.

You can enter any sort of question you want for initial testing and select today and now for the publication date.

4. Configure Choice model

   The next step is to configure the Choice model in the admin, but we will configure the choices to be editable in-line with the Poll objects they are attached to. Edit the same `polls/admin.py` so it now looks like the following:

```
from polls.models import Poll, Choice
from django.contrib import admin

class ChoiceInline(admin.StackedInline):
    model = Choice
    extra = 3

class PollAdmin(admin.ModelAdmin):
    fieldsets = [
        (None,               {'fields': ['question']}),
        ('Date information', {'fields': ['pub_date'], 'classes': ['collapse']}),
    ]
    inlines = [ChoiceInline]

admin.site.register(Poll, PollAdmin)
```

   This tells Django that Choice objects are edited on the Poll admin page, and by default, provide enough fields for 3 choices.

5. Add/edit poll

   You can now return to the Poll admin and either add a new poll or edit the one you already created and see that you can now specify the poll choices inline with the poll itself.

6. Create views

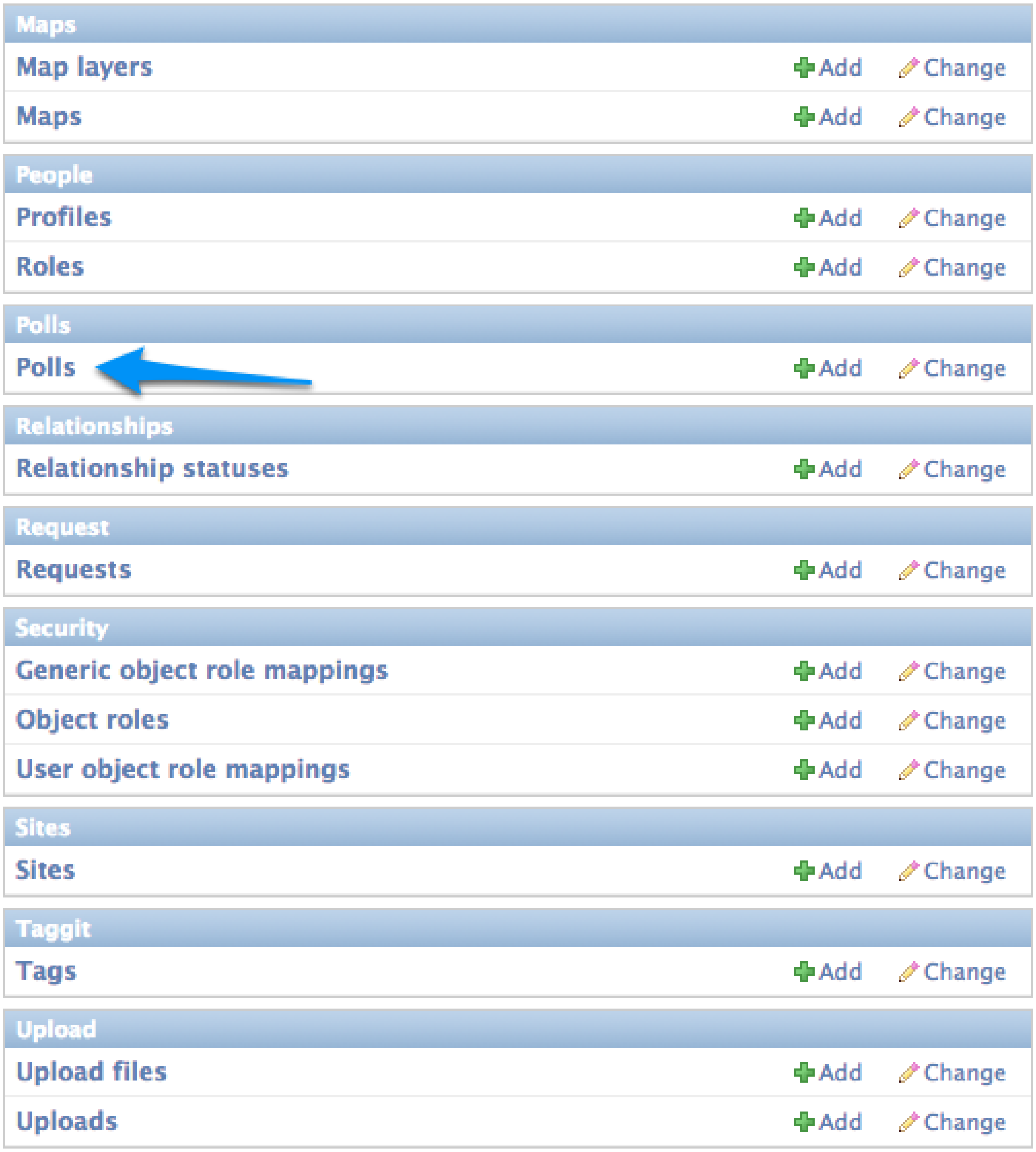
   From here, we want to create views to display the polls inside our GeoNode project. A view is a “type” of Web page in your Django application that generally serves a specific function and has a specific template. In our poll application, there will be the following four views:

   - Poll “index” page—displays the latest few polls.
   - Poll “detail” page—displays a poll question, with no results but with a form to vote.
   - Poll “results” page—displays results for a particular poll.
   - Vote action—handles voting for a particular choice in a particular poll.

   The first step of writing views is to design your URL structure. You do this by creating a Python module called a URLconf. URLconfs are how Django associates a given URL with given Python code.

   Let’s start by adding our URL configuration directly to the `urls.py` that already exists in your project at `<my_geonode>/urls.py`. Edit this file and add the following lines after the rest of the existing imports around line 80:

```
url(r'^polls/$', 'polls.views.index'),
url(r'^polls/(?P<poll_id>\d+)/$', 'polls.views.detail'),
url(r'^polls/(?P<poll_id>\d+)/results/$', 'polls.views.results'),
url(r'^polls/(?P<poll_id>\d+)/vote/$', 'polls.views.vote'),
```



Maps		
Map layers	Add	Change
Maps	Add	Change

People		
Profiles	Add	Change
Roles	Add	Change

Polls		
Polls	Add	Change

Relationships		
Relationship statuses	Add	Change

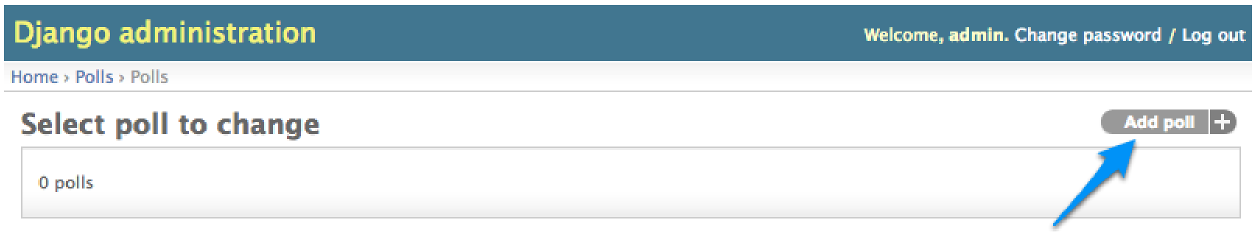
Request		
Requests	Add	Change

Security		
Generic object role mappings	Add	Change
Object roles	Add	Change
User object role mappings	Add	Change

Sites		
Sites	Add	Change

Taggit		
Tags	Add	Change

Upload		
Upload files	Add	Change
Uploads	Add	Change



Django administration

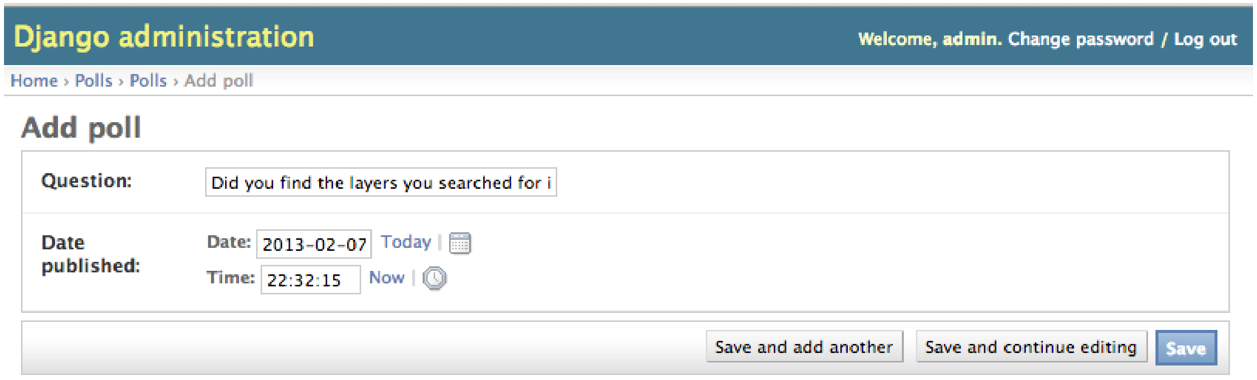
Welcome, **admin**. Change password / Log out

Home › Polls › Polls

Select poll to change

Add poll

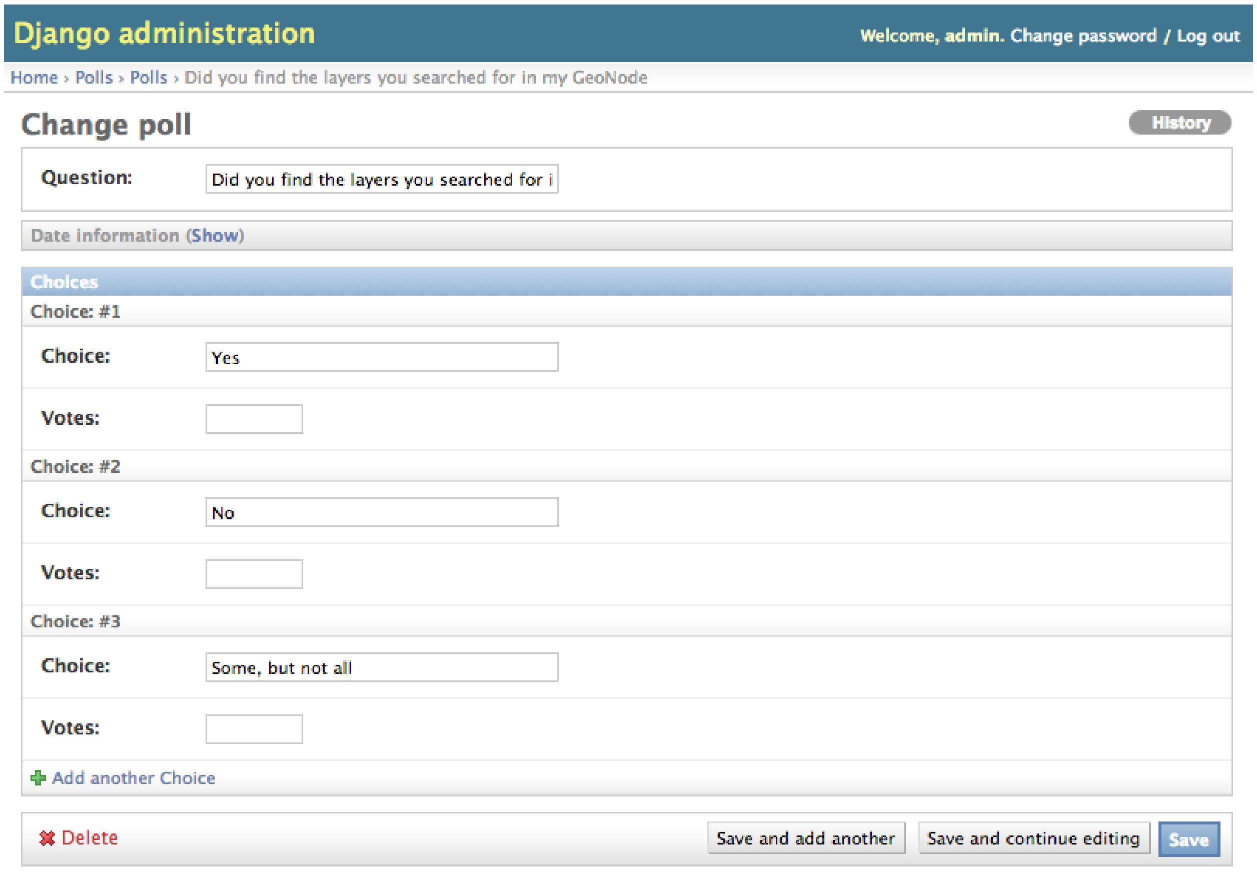
0 polls



Django administration
Welcome, admin. Change password / Log out
Home › Polls › Polls › Add poll
Add poll
Save and add another
Save and continue editing
Save

Question: Did you find the layers you searched for i

Date published: Date: 2013-02-07 Today | Time: 22:32:15 Now |



Django administration
Welcome, admin. Change password / Log out
Home › Polls › Polls › Did you find the layers you searched for in my GeoNode
Change poll
History
Date information (Show)
Choices
Add another Choice
Delete
Save and add another
Save and continue editing
Save

Question: Did you find the layers you searched for i

Choice: #1

Choice: Yes

Votes:

Choice: #2

Choice: No

Votes:

Choice: #3

Choice: Some, but not all

Votes:

Note: Eventually we will want to move this set of URL configurations inside the URLs app itself, but for the sake of brevity in this workshop, we will put them in the main `urls.py` for now. You can consult the Django tutorial for more information on this topic.

Next write the views to drive the URL patterns we configured above. Edit polls/views.py to that it looks like the following:

```
from django.template import RequestContext, loader
from polls.models import Poll
from django.http import HttpResponse
from django.http import Http404
from django.shortcuts import render_to_response

def index(request):
    latest_poll_list = Poll.objects.all().order_by('-pub_date')[:5]
    return render_to_response('polls/index.html',
        RequestContext(request, {'latest_poll_list': latest_poll_list}))

def detail(request, poll_id):
    try:
        p = Poll.objects.get(pk=poll_id)
    except Poll.DoesNotExist:
        raise Http404
    return render_to_response('polls/detail.html', RequestContext(request, {'poll': p}))

def results(request, poll_id):
    return HttpResponse("You're looking at the results of poll %s." % poll_id)

def vote(request, poll_id):
    return HttpResponse("You're voting on poll %s." % poll_id)
```

Note: We have only stubbed in the views for the results and vote pages. They are not very useful as-is. We will revisit these later.

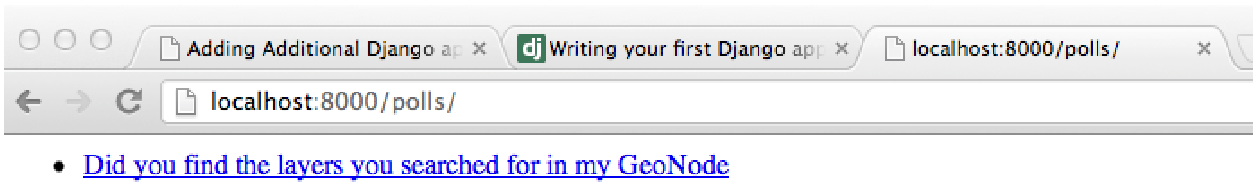
Now we have views in place, but we are referencing templates that do not yet exist. Let's add them by first creating a template directory in your polls app at `polls/templates/polls` and creating `polls/templates/polls/index.html` to look like the following:

```
{% if latest_poll_list %}
    <ul>
    {% for poll in latest_poll_list %}
        <li><a href="/polls/{{ poll.id }}/">{{ poll.question }}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>No polls are available.</p>
{% endif %}
```

Next we need to create the template for the poll detail page. Create a new file at `polls/templates/polls/detail.html` to look like the following:

```
<h1>{{ poll.question }}</h1>
<ul>
{% for choice in poll.choice_set.all %}
    <li>{{ choice.choice }}</li>
    {% endfor %}
    </ul>
```

You can now visit http://localhost:8000/polls/ in your browser and you should see the the poll question you created in the admin presented like this.



- Did you find the layers you searched for in my GeoNode

7. Update templates

   We actually want our polls app to display as part of our GeoNode project with the same theme, so let's update the two templates we created above to make them extend from the `site_base.html` template we looked at in the last section. You will need to add the following two lines to the top of each file:

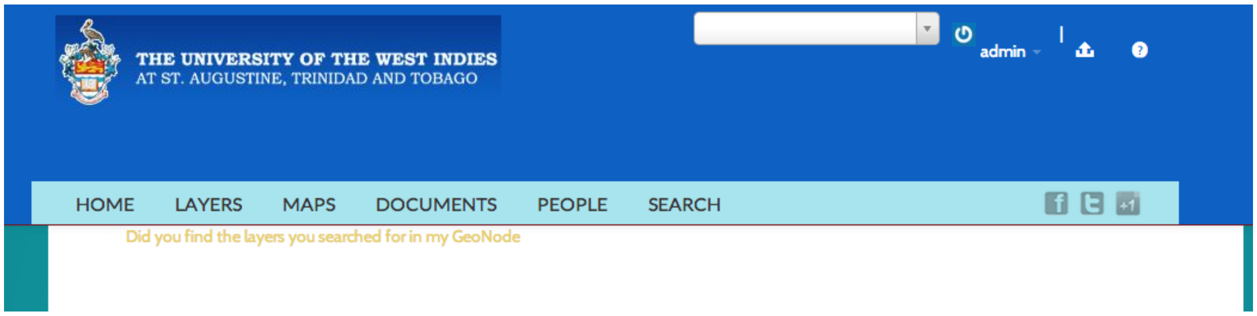
```
    {% extends 'site_base.html' %}
    {% block body %}
```

   And close the block at the bottom of each file with:

```
    {% endblock %}
```

   This tells Django to extend from the `site_base.html` template so your polls app has the same style as the rest of your GeoNode, and it specifies that the content in these templates should be rendered to the body block defined in GeoNode's `base.html` template that your `site_base.html` extends from.

   You can now visit the index page of your polls app and see that it is now wrapped in the same style as the rest of your GeoNode site.


THE UNIVERSITY OF THE WEST INDIES
AT ST. AUGUSTINE, TRINIDAD AND TOBAGO
admin
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH

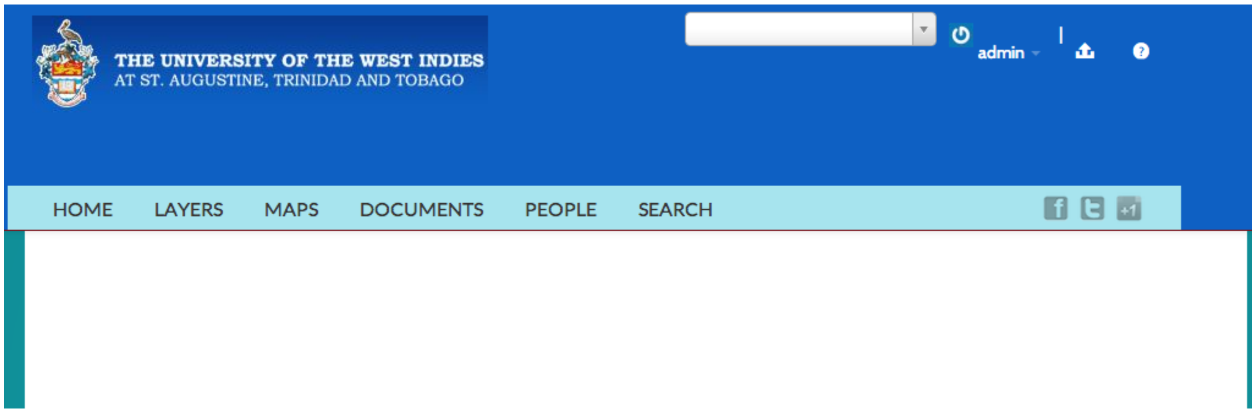
Did you find the layers you searched for in my GeoNode

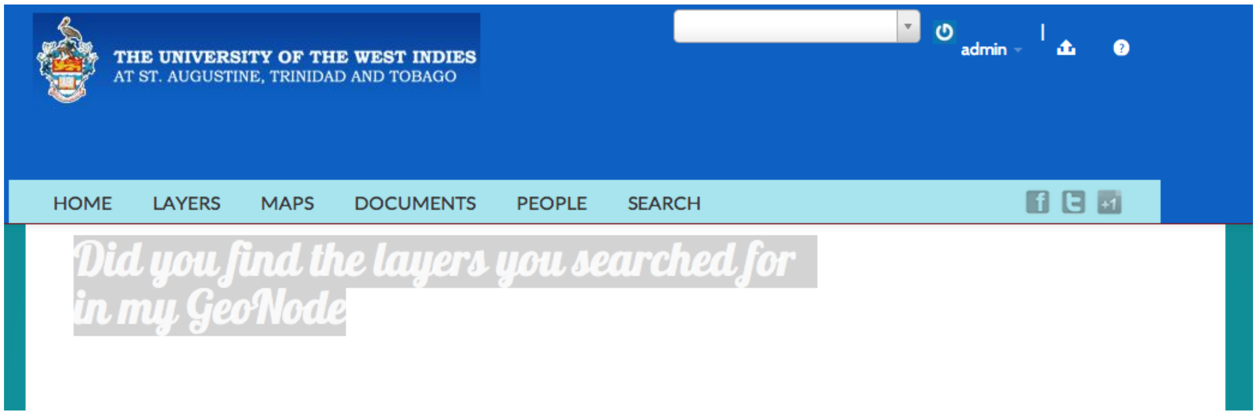
   If you click on a question from the list you will be taken to the poll detail page.

   It looks like it is empty, but in fact the text is there, but styled to be white by the Bootswatch theme we added in the last section. If you highlight the area where the text is, you will see that it is there.

Now that you have walked through the basic steps to create a very minimal (though not very useful) Django app and integrated it with your GeoNode project, you should pick up the Django tutorial at part 4 and follow it to add the form for actually accepting responses to your poll questions.

We strongly recommend that you spend as much time as you need with the Django tutorial itself until you feel comfortable with all of the concepts. They are the essential building blocks you will need to extend your GeoNode project by adding your own apps.


THE UNIVERSITY OF THE WEST INDIES
AT ST. AUGUSTINE, TRINIDAD AND TOBAGO
admin
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH


THE UNIVERSITY OF THE WEST INDIES
AT ST. AUGUSTINE, TRINIDAD AND TOBAGO
admin
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH
Did you find the layers you searched for in my GeoNode

Adding a 3rd party blog app Now that we have created our own app and added it to our GeoNode project, the next thing we will work through is adding a 3rd party blog app. There are a number of blog apps that you can use, but for purposes of this workshop, we will use a relatively simple, yet extensible app called Zinnia. You can find out more information about Zinnia on its website or on its GitHub project page or by following its documentation. This section will walk you through the minimal set of steps necessary to add Zinnia to your GeoNode project.

1. Install Zinnia

   The first thing to do is to install Zinnia into the virtualenv that you are working in. Make sure your virtualenv is activated and execute the following command:

```
$ pip install django-blog-zinnia
```

   This will install Zinnia and all of the libraries that it depends on.

2. Add Zinnia to INSTALLED_APPS

   Next add Zinnia to the INSTALLED_APPS section of your GeoNode projects `settings.py` file by editing `<my_geonode>/settings.py` and adding 'django.contrib.comments' to the section labeled "Apps Bundled with Django" so that it looks like the following:

```
# Apps bundled with Django
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.sites',
'django.contrib.admin',
'django.contrib.sitemaps',
'django.contrib.staticfiles',
    'django.contrib.messages',
    'django.contrib.humanize',
    'django.contrib.comments',
```

And then add the `tagging`, `mptt` and `zinnia` apps to the end of the INSTALLED_APPS where we previously added a section labeled "My GeoNode apps". It should like like the following:

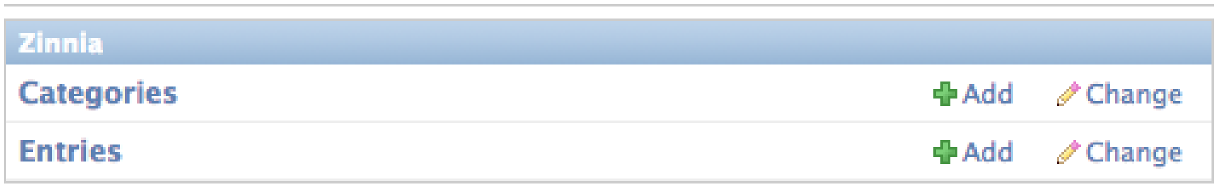
```
    # My GeoNode apps
    'polls',
    'tagging',
    'mptt',
    'zinnia',
```

3. Synchronize models

   Next you will need to run `syncdb` again to synchronize the models for the apps we have just added to our project's database. This time we want to pass the `--all` flag to `syncdb` so it ignores the schema migrations. Schema migrations are discussed further in GeoNode's documentation, but it is safe to ignore them here.

```
    $ python manage.py syncdb --all
```

   You can now restart the development server and visit the Admin interface and scroll to the very bottom of the list to find a section for Zinnia that allows you to manage database records for Categories and Blog Entries.



4. Configure project

   Next we need to configure our project to add Zinnia's URL configurations. Add the following two URL configuration entries to the end of `<my_geonode>/urls.py`:

```
    url(r'^blog/', include('zinnia.urls')),
    url(r'^djcomments/', include('django.contrib.comments.urls')),
```

   If you visit the main blog page in your browser at http://localhost:8000/blog/ you will find that the blog displays with Zinnia's default theme as shown below.

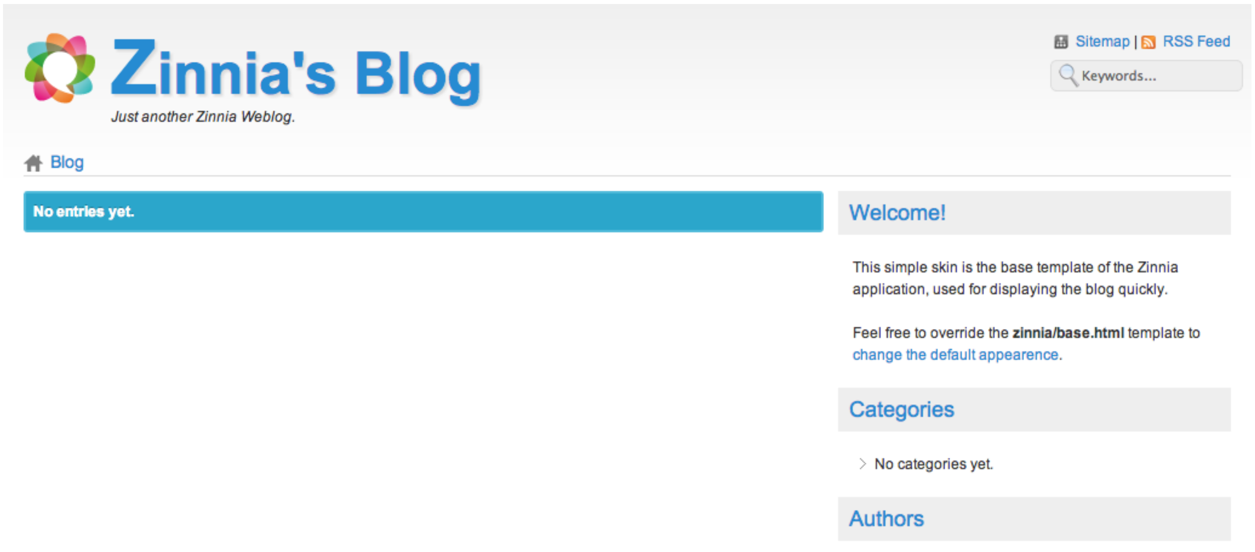
   Note: If you are not able to visit the main blog page, you will have to set `USE_TZ = True` in settings.py. Restart the server and try again!

   This page includes some guidance for us on how to change the default theme.

5. Change default theme

   The first thing we need to do is to copy Zinnia's `base.html` template into our own project so we can modify it. When you installed Zinnia, templates were installed to `/var/lib/geonode/lib/python2.7/site-packages/zinnia/templates/zinnia/`. You can copy the base template by executing the following commands:

```
    $ mkdir <my_geonode>/templates/zinnia
    $ cp /var/lib/geonode/lib/python2.7/site-packages/zinnia/templates/zinnia/base.html <my_geonode>
```



No entries yet.

Welcome!

This simple skin is the base template of the Zinnia application, used for displaying the blog quickly.

Feel free to override the **zinnia/base.html** template to change the default appearence.

Categories

No categories yet.

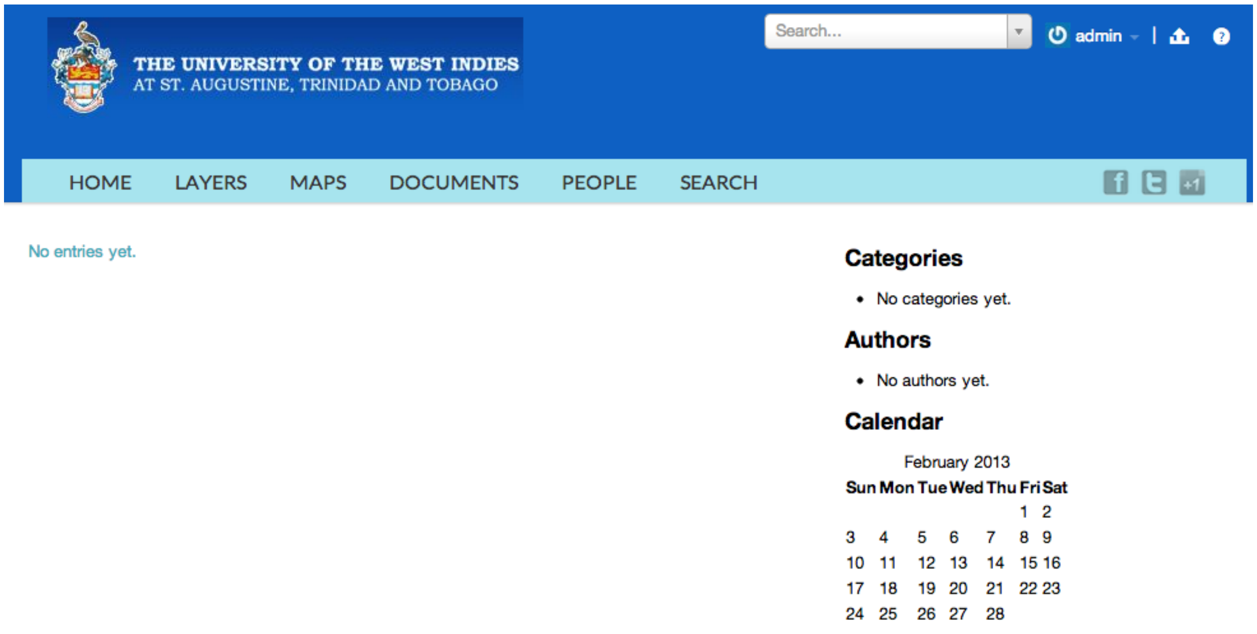
Authors

Then you need to edit this file and change the topmost line to read as below such that this template extends from our projects `site_base.html` rather than the zinnia `skeleton.html`:

```
{% extends "site_base.html" %}
```

Since Zinnia uses a different block naming scheme than GeoNode does, you need to add the following line to the bottom of your site_base.html file so that the content block gets rendered properly:

```
{% block body %}{% block content %}{% endblock %}{% endblock %}
```


THE UNIVERSITY OF THE WEST INDIES
AT ST. AUGUSTINE, TRINIDAD AND TOBAGO
Search...
admin
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH

No entries yet.

Categories

- No categories yet.

Authors

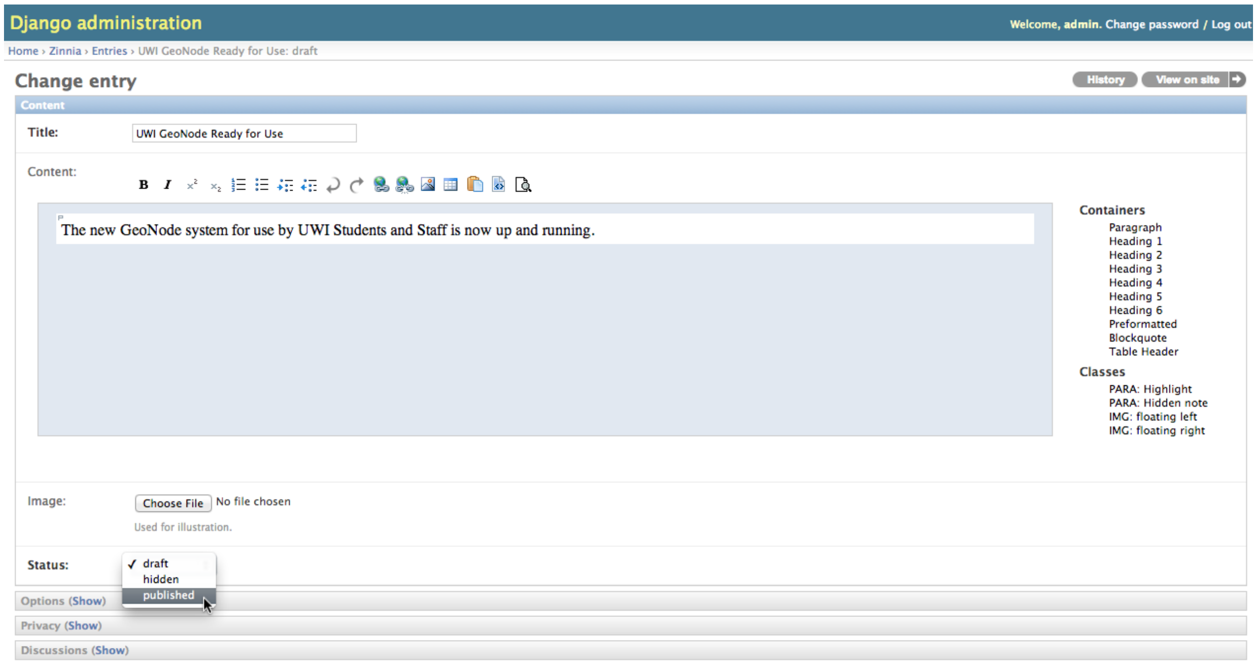
- No authors yet.

Calendar

February 2013

Sun	Mon	Tue	Wed	Thu	Fri	Sat
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28		

You can see that there are currently no blog entries, so let's add one. Scroll to the bottom of the interface and click the *Post an Entry* link to go to the form in the Admin interface that lets you create a blog post. Go ahead and fill out the form with some information for testing purposes. Make sure that you change the Status dropdown to "published" so the post shows up right away.



Change entry

Title: UWI GeoNode Ready for Use

Content:

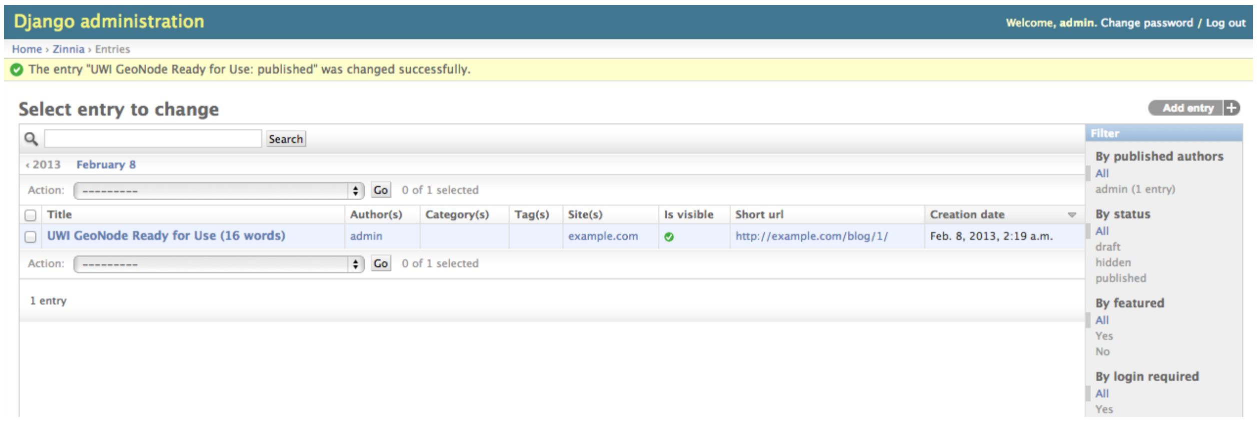
The new GeoNode system for use by UWI Students and Staff is now up and running.

Image: No file chosen

Used for illustration.

Status: ✓ draft, hidden, published

You can explore all of the options available to you as you create your post, and when you are done, click the *Save* button. You will be taken to the page that shows the list of all your blog posts.



The entry "UWI GeoNode Ready for Use: published" was changed successfully.

Select entry to change

0 of 1 selected

Title	Author(s)	Category(s)	Tag(s)	Site(s)	Is visible	Short url	Creation date
UWI GeoNode Ready for Use (16 words)	admin			example.com	✓	http://example.com/blog/1/	Feb. 8, 2013, 2:19 a.m.

1 entry

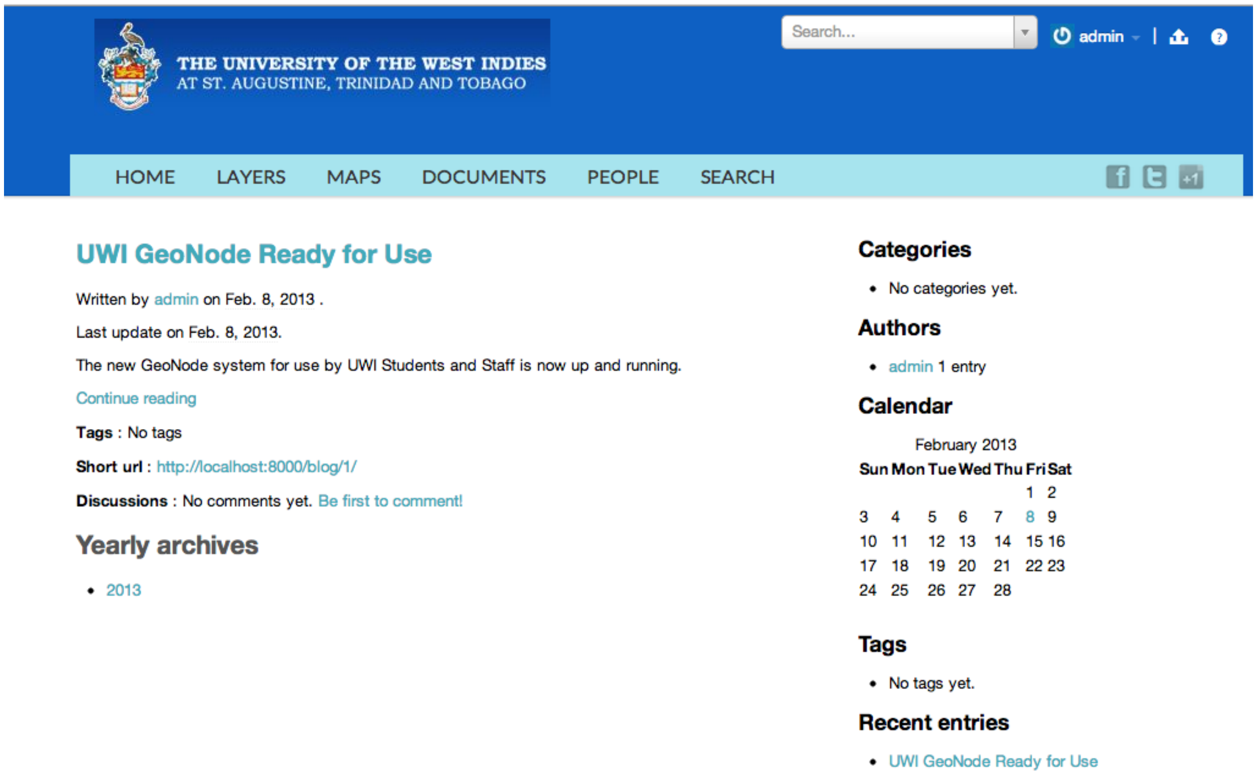
You can then visit your blog post/entry at http://localhost:8000/blog/.

And if you click on the blog post title, you will be taken to the page for the complete blog post. You and your users can leave comments on this post and various other blog features from this page.

6. Integrate app into your site

The last thing we need to do to fully integrate this blog app (and our polls app) into our site is to add it to the options on the navbar. To do so, we need to add the following block override to our Projects `site_base.html`:

```
{% block extra-nav %}
<li id="nav_polls">
    <a href="/polls/">Polls</a>
</li>
<li id="nav_blog">
    <a href="{% url 'zinnia_entry_archive_index' %}">Blog</a>
```


THE UNIVERSITY OF THE WEST INDIES
AT ST. AUGUSTINE, TRINIDAD AND TOBAGO
Search...
admin
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH

UWI GeoNode Ready for Use

Written by admin on Feb. 8, 2013 .

Last update on Feb. 8, 2013.

The new GeoNode system for use by UWI Students and Staff is now up and running.

Continue reading

Tags : No tags

Short url : http://localhost:8000/blog/1/

Discussions : No comments yet. Be first to comment!

Yearly archives

- 2013

Categories

- No categories yet.

Authors

- admin 1 entry

Calendar

February 2013

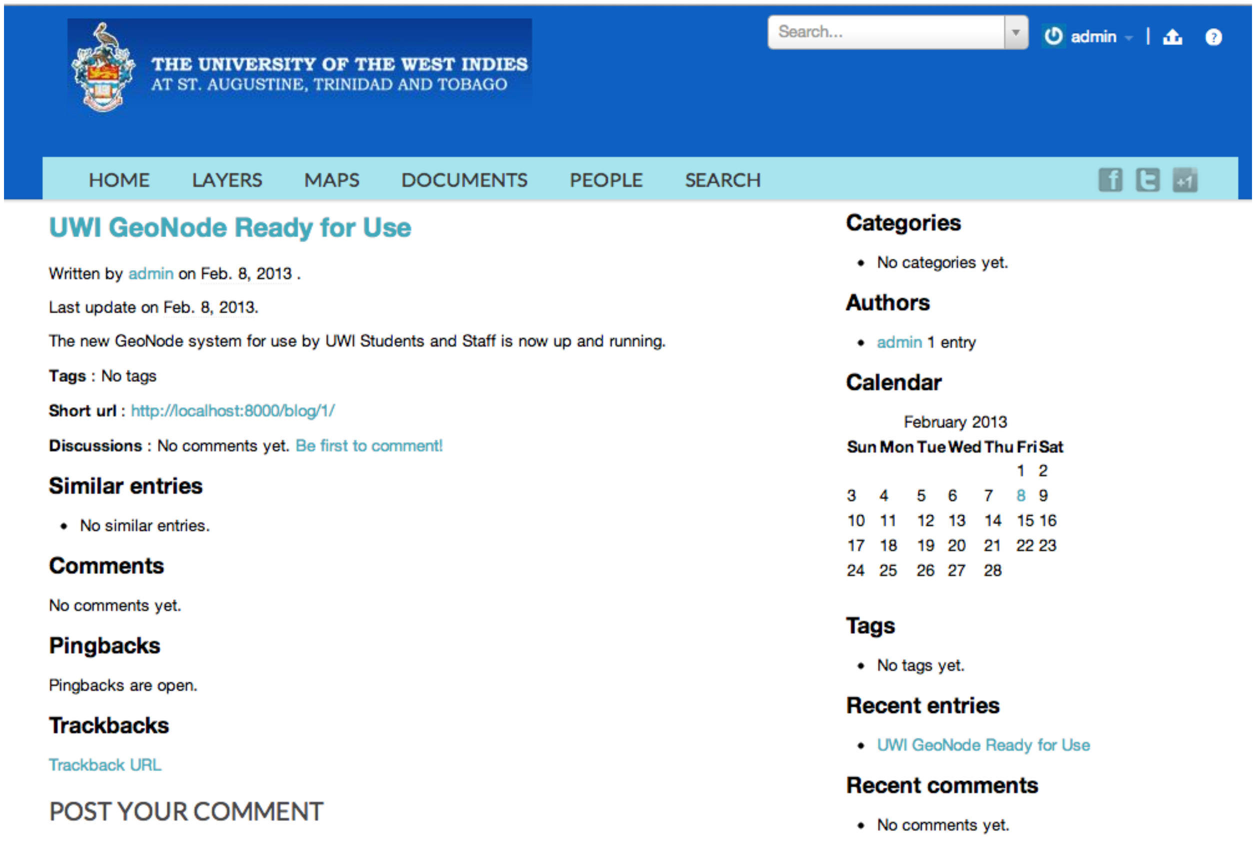
Sun	Mon	Tue	Wed	Thu	Fri	Sat
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28		

Tags

- No tags yet.

Recent entries

- UWI GeoNode Ready for Use


THE UNIVERSITY OF THE WEST INDIES
AT ST. AUGUSTINE, TRINIDAD AND TOBAGO
Search...
admin
HOME
LAYERS
MAPS
DOCUMENTS
PEOPLE
SEARCH

UWI GeoNode Ready for Use

Written by admin on Feb. 8, 2013 .

Last update on Feb. 8, 2013.

The new GeoNode system for use by UWI Students and Staff is now up and running.

Tags : No tags

Short url : http://localhost:8000/blog/1/

Discussions : No comments yet. Be first to comment!

Similar entries

- No similar entries.

Comments

No comments yet.

Pingbacks

Pingbacks are open.

Trackbacks

Trackback URL

POST YOUR COMMENT

Categories

- No categories yet.

Authors

- admin 1 entry

Calendar

February 2013

Sun	Mon	Tue	Wed	Thu	Fri	Sat
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28		

Tags

- No tags yet.

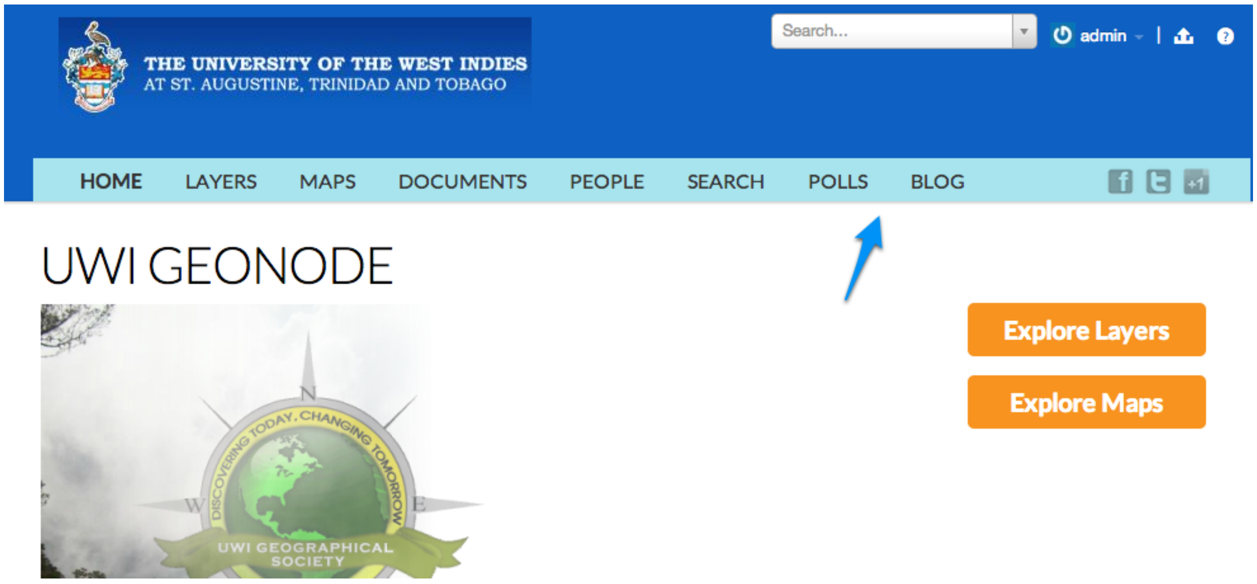
Recent entries

- UWI GeoNode Ready for Use

Recent comments

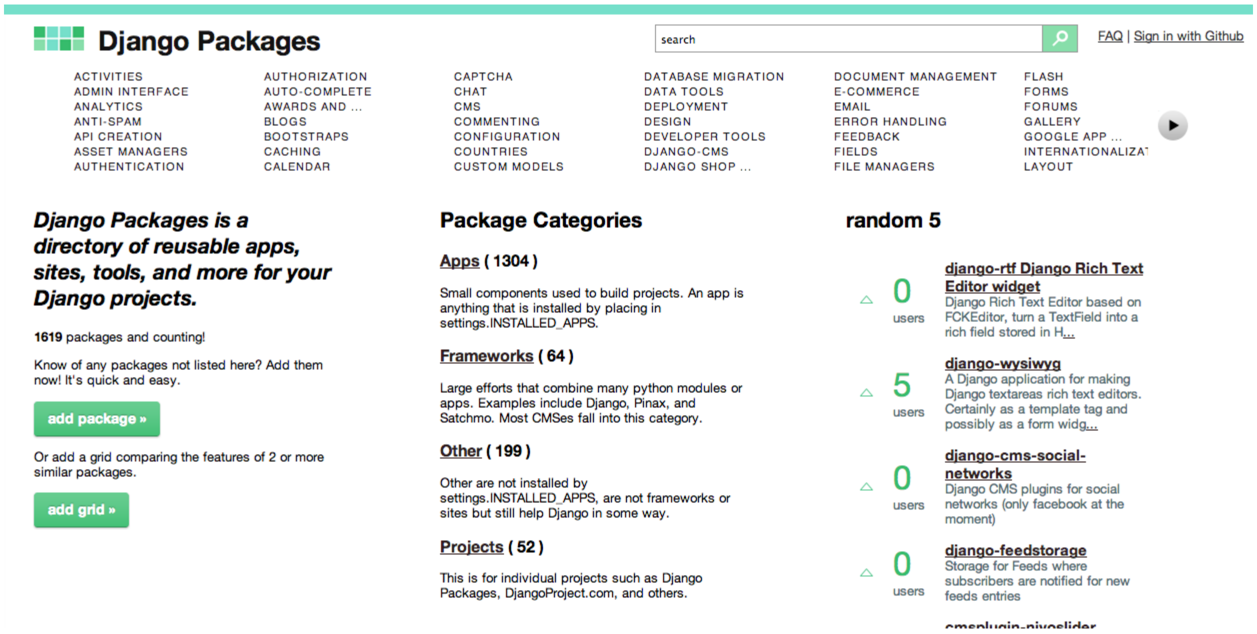
- No comments yet.

```
    </li>
    {% endblock %}
```



At this point, you could explore options for tighter integration between your GeoNode project and Zinnia. Integrating blog posts from Zinnia into your overall search could be useful, as well as including the blog posts a user has written on their Profile Page. You could also explore the additional plugins that go with Zinnia.

Adding other apps Now that you have both written your own app and plugged in a 3rd party one, you can explore sites like Django Packages to look for other modules that you could plug into your GeoNode project to meet your needs and requirements. For many types of apps, there are several options and Django Packages is a nice way to compare them. You may find that some apps require significantly more work to integrate into your app than others, but reaching out to the app's author and/or developers should help you get over any difficulties you may encounter.



Django Packages is a directory of reusable apps, sites, tools, and more for your Django projects.

1619 packages and counting!

Know of any packages not listed here? Add them now! It's quick and easy.

Or add a grid comparing the features of 2 or more similar packages.

Package Categories

Apps (1304)

Small components used to build projects. An app is anything that is installed by placing in settings.INSTALLED_APPS.

Frameworks (64)

Large efforts that combine many python modules or apps. Examples include Django, Pinax, and Satchmo. Most CMSes fall into this category.

Other (199)

Other are not installed by settings.INSTALLED_APPS, are not frameworks or sites but still help Django in some way.

Projects (52)

This is for individual projects such as Django Packages, DjangoProject.com, and others.

random 5

0 users **django-rtf Django Rich Text Editor widget**
Django Rich Text Editor based on FCKEditor, turn a TextField into a rich field stored in H...

5 users **django-wysiwyg**
A Django application for making Django textareas rich text editors. Certainly as a template tag and possibly as a form widg...

0 users **django-cms-social-networks**
Django CMS plugins for social networks (only facebook at the moment)

0 users **django-feedstorage**
Storage for Feeds where subscribers are notified for new feeds entries

cmsplugin-nivoslider

Integrating your project with other systems Your GeoNode project is based on core components which are interoperable and as such, it is straightforward for you to integrate with external applications and services. This section will walk you through how to connect to your GeoNode instance from other applications and how to integrate other services into your GeoNode project. When complete, you should have a good idea about the possibilities for integration, and have basic knowledge about how to accomplish it. You may find it necessary to dive deeper into how to do more complex integration in order to accomplish your goals, but you should feel comfortable with the basics, and feel confident reaching out to the wider GeoNode community for help.

OGC services Since GeoNode is built on GeoServer which is heavily based on OGC services, the main path for integration with external services is via OGC Standards. A large number of systems, applications and services support adding WMS layers to them, but only a few key ones are covered below. WFS and WCS are also supported in a wide variety of clients and platforms and give you access to the actual data for use in GeoProcessing or to manipulate it to meet your requirements. GeoServer also bundles GeoWebCache which produces map tiles that can be added as layers in many popular web mapping tools including Google Maps, Leaflet, OpenLayers and others. You should review the reference material included in the first chapter to learn more about OGC Services and when evaluating external systems make sure that they are also OGC Compliant in order to integrate as seamlessly as possible.

See *Using GeoNode with other applications* for examples with desktop applications. More examples will be shared for integration with existing databases, mapping server and sharing platform such as CKAN.

GeoNode debugging techniques

GeoNode can be difficult to debug as there are several different components involved:

- Browser - includes HTML/CSS issues, JavaScript, etc.
- Django - GeoNode HTML views and web APIs
- GeoServer - Core Wxx services and Platform REST APIs

When attempting to diagnose a specific problem, often the order of investigation mirrors the order above - that is, start with the client: Is this a bug in code running on the browser. If not, step to the next level: the Django responses to client requests. Often this is possible via the browser using the correct tools. Many requests require Django communications with GeoServer. This is the next stage of investigation if a specific bug does not appear to originate in Django or the client.

The following section covers techniques to help diagnose and debug errors.

Debugging GeoNode in the Browser This section covers some techniques for debugging browser and Django related response bugs using the Firefox web browser extension named Firebug. The concepts covered apply to other browser's tools but may vary in terminology.

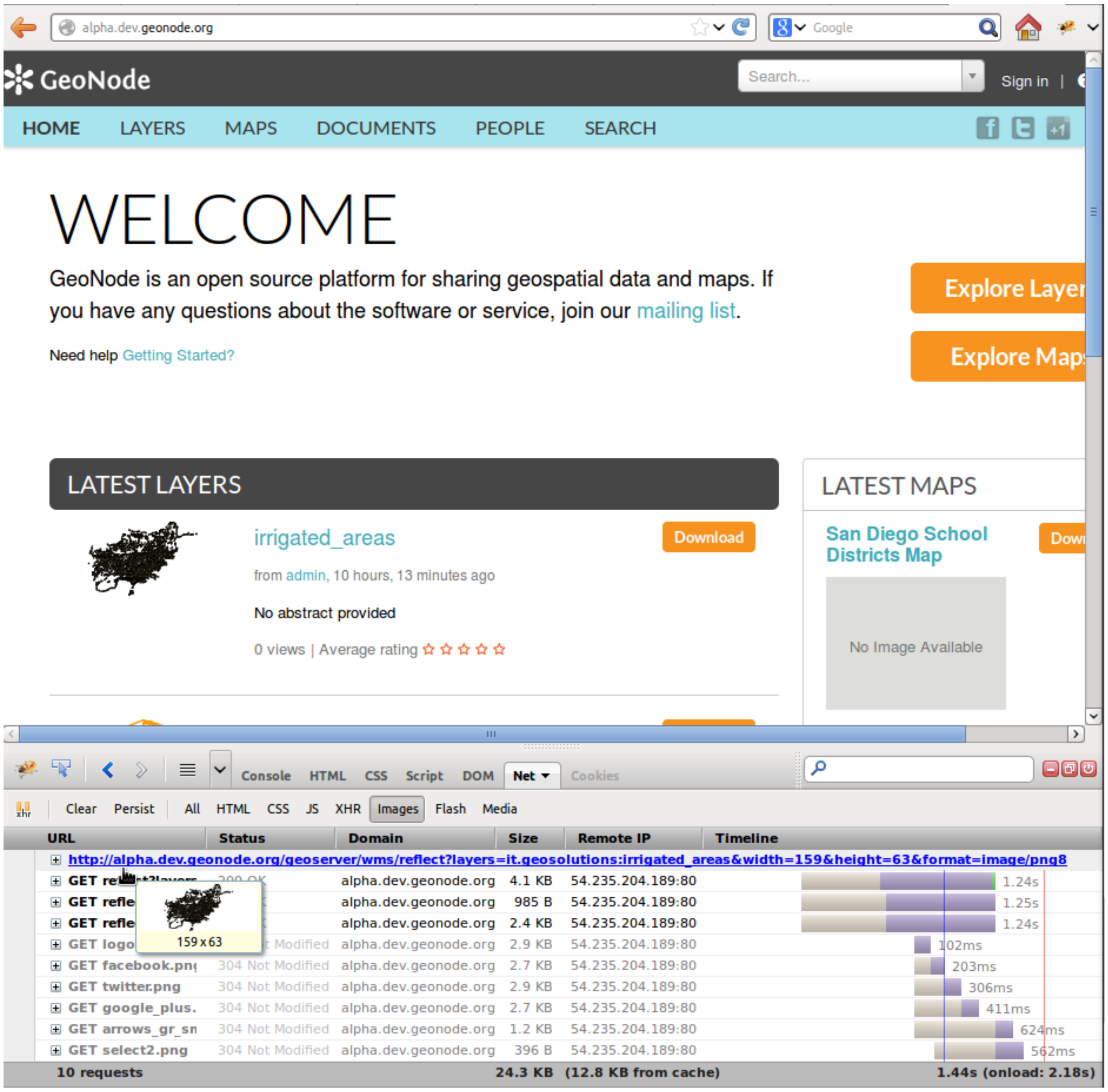
Another Firefox extension worth noting is 'jsonview'. This extension supports formatted viewing of JSON responses and integrates well with Firebug.

References:

- https://getfirebug.com/faq/
- http://jsonview.com/

Net Tab The net tab allows viewing all of the network traffic from the browser. The subtabs (like the selected "Images" tab) allow filtering by the type of traffic.

In this screen-shot, the mouse hover displays the image content and the full URL requested. One can right-click to copy-paste the URL or view in a separate tab. This is useful for obtaining test URLs. The grayed out entries show that the resource was cached via conditional-get (the 304 not modified). Other very useful advanced information includes



alpha.dev.geonode.org

GeoNode

Search... Sign in |

HOME LAYERS MAPS DOCUMENTS PEOPLE SEARCH

WELCOME

GeoNode is an open source platform for sharing geospatial data and maps. If you have any questions about the software or service, join our mailing list.

Need help Getting Started?

Explore Layer

Explore Map

LATEST LAYERS

irrigated_areas

Download

from admin, 10 hours, 13 minutes ago

No abstract provided

0 views | Average rating

LATEST MAPS

San Diego School Districts Map

No Image Available

Console HTML CSS Script DOM Net Cookies

Clear Persist All HTML CSS JS XHR Images Flash Media

```
URL                  Status            Domain                Size    Remote IP            Timeline
http://alpha.dev.geonode.org/geoserver/wms/reflect?layers=it.geosolutions:irrigated_areas&width=159&height=63&format=image/png8
GET reflect?layers   200 OK            alpha.dev.geonode.org 4.1 KB  54.235.204.189:80    1.24s
GET refle                              alpha.dev.geonode.org 985 B   54.235.204.189:80    1.25s
GET refle                              alpha.dev.geonode.org 2.4 KB  54.235.204.189:80    1.24s
GET logo             t Modified        alpha.dev.geonode.org 2.9 KB  54.235.204.189:80    102ms
GET facebook.pn(     304 Not Modified  alpha.dev.geonode.org 2.7 KB  54.235.204.189:80    203ms
GET twitter.png      304 Not Modified  alpha.dev.geonode.org 2.9 KB  54.235.204.189:80    306ms
GET google_plus.     304 Not Modified  alpha.dev.geonode.org 2.7 KB  54.235.204.189:80    411ms
GET arrows_gr_sn     304 Not Modified  alpha.dev.geonode.org 1.2 KB  54.235.204.189:80    624ms
GET select2.png      304 Not Modified  alpha.dev.geonode.org 396 B   54.235.204.189:80    562ms
10 requests                                          24.3 KB (12.8 KB from cache)        1.44s (onload: 2.18s)
```

159 x 63

Fig. 3.83: Firebug Net Tab

the size of the response and the loading indicator graphics on the right. At the bottom, note the total size and timing information.

Net Tab Exercises

1. Go to layers/maps/search pages and look at the various requests. Note the XHR subtab. Look at the various request specific tabs: headers, params, etc.
2. Use the 'disable browser cache' option and see how it affects page loads. Discuss advantages/challenges of caching.

DOM Tab The DOM tab displays all of the top-level window objects. By drilling down, this can be a useful way to find out what's going on in a page.

In this example, the mouse is hovering over the `app` object. Note the high level view of objects and their fields. The console tab allows interacting with the objects.

DOM Tab Exercises

1. Drill down in the DOM tab.
2. Use the console to interactively exercise jquery.
3. Use the console to interact with the app/map or other page objects

Script Tab The script tab allows viewing scripts and debugging.

The screen-shot displays a breakpoint set at line 3, the current code is stopped at line 8 and the mouse hover is displaying the value of the variable 'class_list'. On the right, the 'Watch' tab displays the various variables and scopes and offers a drill down view similar to the DOM view. The stack tab displays the execution stack context outside the current frame.

Script Tab Exercises

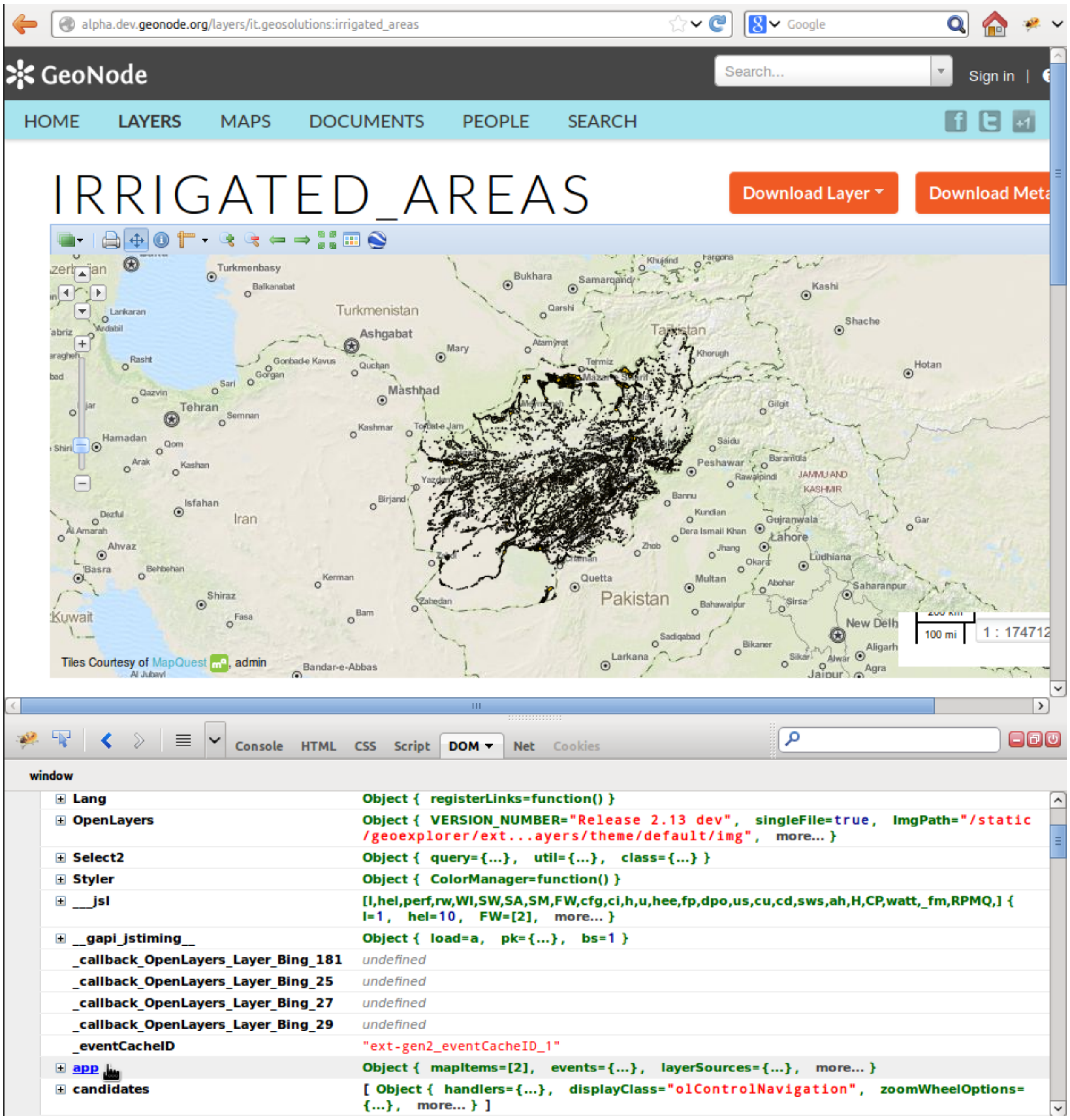
1. Step through some code
2. Look at various features: variables, scopes, DOM drill-down

HTML Tab The HTML tag allows viewing and drilling down into the DOM. This is an incredibly useful feature when doing CSS or HTML work.

The screen-shot displays a search result 'article' element highlighted with padding and margin in yellow and purple. The DOM structure is displayed on the left and the right panel displays the specific style rules while the computed tab displays the effective style rules. The layout tab displays rulers and property values while the DOM tab displays a debug/DOM-like view of the actual object's properties.

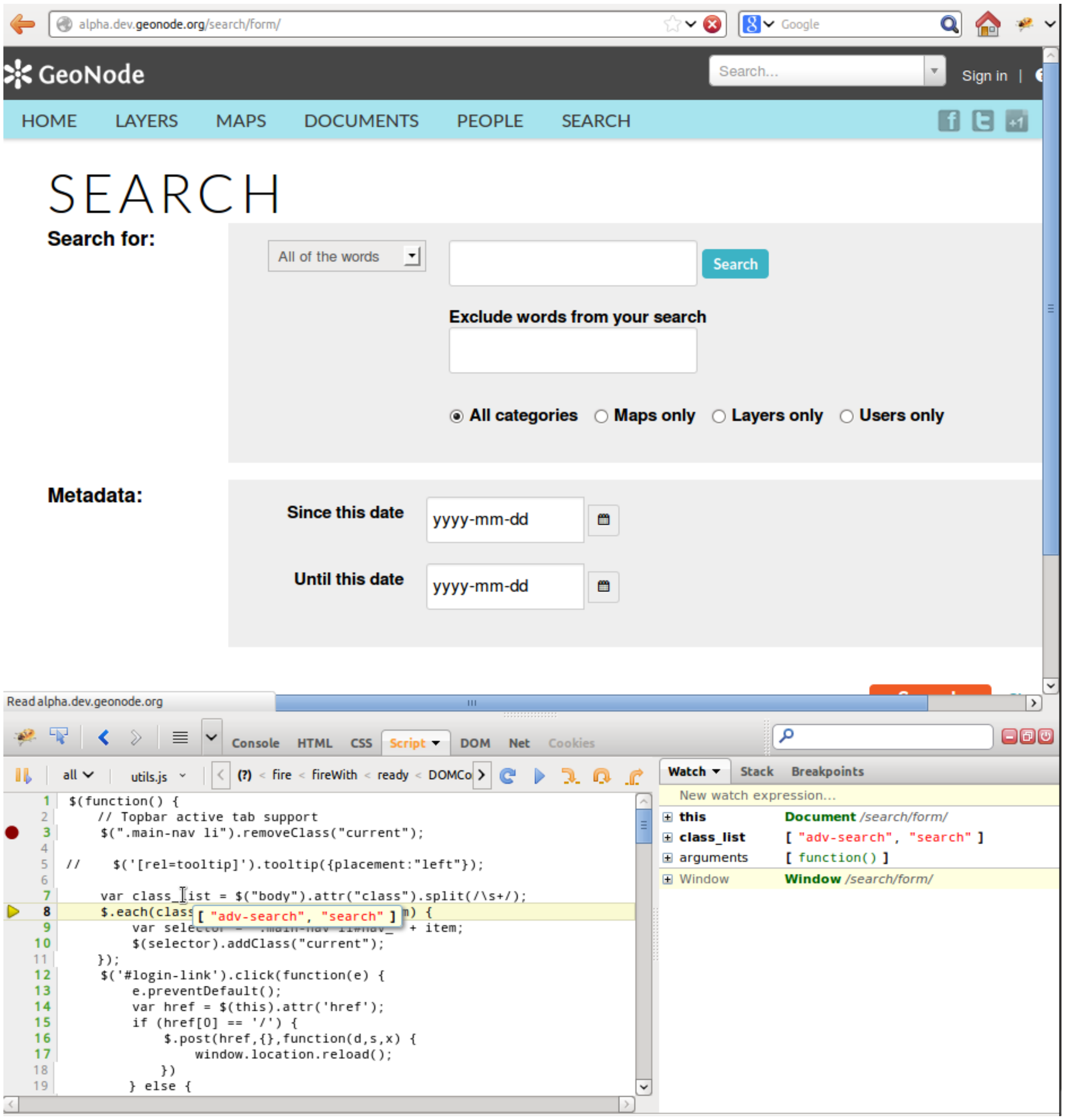
HTML Tab Exercises

1. Identify elements, look at the tabs on the right
2. Change styles, add new rules and styles
3. Edit existing HTML elements via the raw and tree-view



```
Lang                                   Object { registerLinks=function() }
OpenLayers                             Object { VERSION_NUMBER="Release 2.13 dev", singleFile=true, ImgPath="/static
                                       /geoexplorer/ext...ayers/theme/default/img", more... }
Select2                                Object { query={...}, util={...}, class={...} }
Styler                                 Object { ColorManager=function() }
__jsl                                  [l,hel,perf,rw,WI,SW,SA,SM,FW,cfg,ci,h,u,hee,fp,dpo,us,cu,cd,sws,ah,H,CP,watt,_fm,RPMQ,] {
                                       l=1, hel=10, FW=[2], more... }
__gapi_jstiming__                      Object { load=a, pk={...}, bs=1 }
_callback_OpenLayers_Layer_Bing_181    undefined
_callback_OpenLayers_Layer_Bing_25     undefined
_callback_OpenLayers_Layer_Bing_27     undefined
_callback_OpenLayers_Layer_Bing_29     undefined
_eventCacheID                          "ext-gen2_eventCacheID_1"
app                                    Object { mapItems=[2], events={...}, layerSources={...}, more... }
candidates                             [ Object { handlers={...}, displayClass="olControlNavigation", zoomWheelOptions=
                                       {...}, more... } ]
```

Fig. 3.84: Firebug DOM Tab



SEARCH

Search for: All of the words [ ] Search

Exclude words from your search [ ]

[✓ All categories] Maps only Layers only Users only

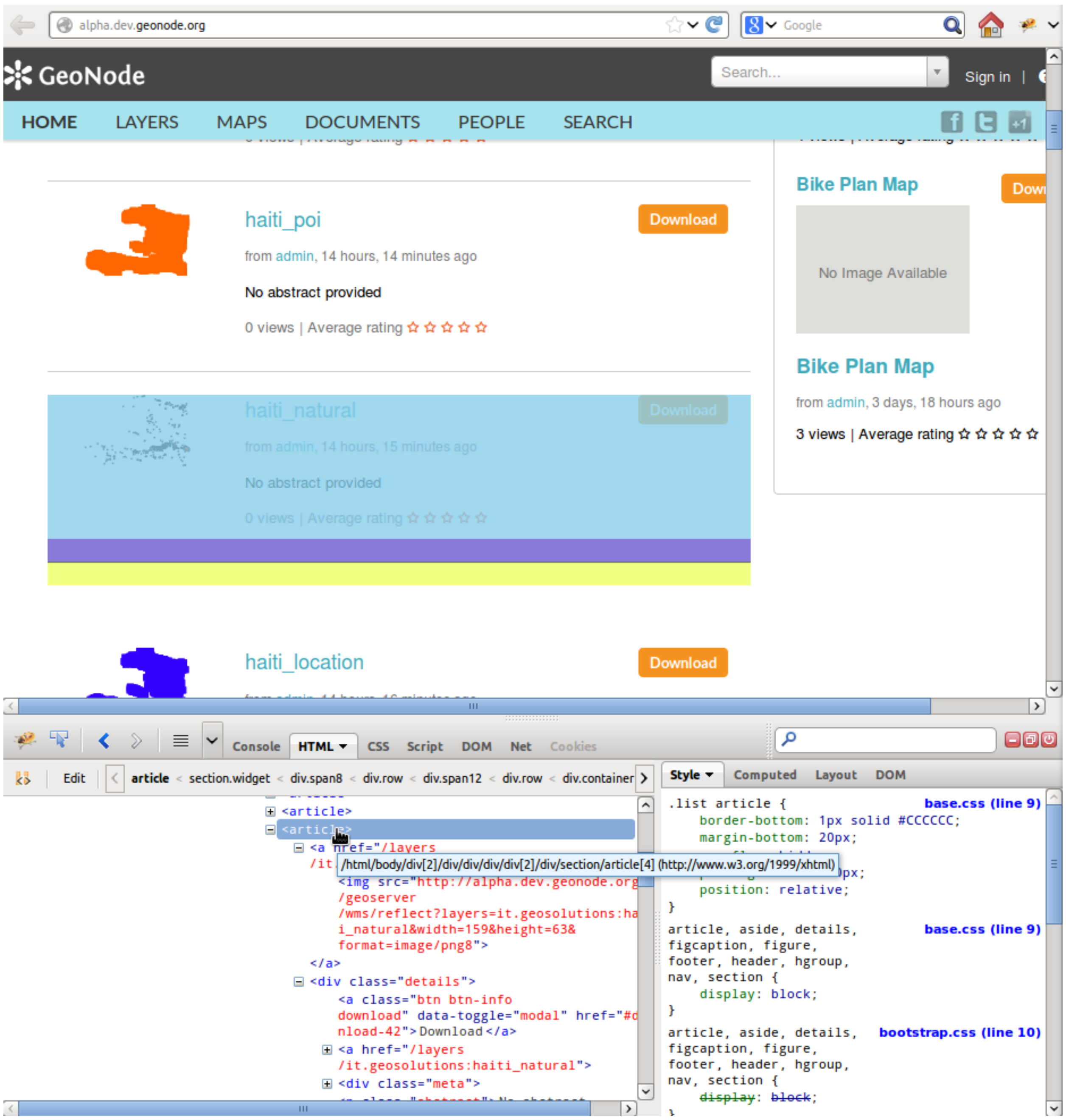
Metadata:

Since this date yyyy-mm-dd

Until this date yyyy-mm-dd

```
$(function() {
    // Topbar active tab support
    $(".main-nav li").removeClass("current");

//    $('[rel=tooltip]').tooltip({placement:"left"});

    var class_list = $("body").attr("class").split(/\s+/);
    $.each(class[ "adv-search", "search" ]m) {
        var selector = ".main-nav li#nav_" + item;
        $(selector).addClass("current");
    });
    $('#login-link').click(function(e) {
        e.preventDefault();
        var href = $(this).attr('href');
        if (href[0] == '/') {
            $.post(href,{},function(d,s,x) {
                window.location.reload();
            })
        } else {
```

this	Document /search/form/
class_list	["adv-search", "search"]
arguments	[function()]
Window	Window /search/form/



haiti_poi
Download
from admin, 14 hours, 14 minutes ago
No abstract provided
0 views | Average rating ☆ ☆ ☆ ☆ ☆

haiti_natural
Download
from admin, 14 hours, 15 minutes ago
No abstract provided
0 views | Average rating ☆ ☆ ☆ ☆ ☆

haiti_location
Download

Bike Plan Map
No Image Available
Bike Plan Map
from admin, 3 days, 18 hours ago
3 views | Average rating ☆ ☆ ☆ ☆ ☆

article < section.widget < div.span8 < div.row < div.span12 < div.row < div.container

```
<article>
<article>
  <a href="/layers
  /it
      <img src="http://alpha.dev.geonode.org
      /geoserver
      /wms/reflect?layers=it.geosolutions:ha
      i_natural&width=159&height=63&
      format=image/png8">
  </a>
  <div class="details">
      <a class="btn btn-info
      download" data-toggle="modal" href="#d
      nload-42">Download</a>
      <a href="/layers
      /it.geosolutions:haiti_natural">
      <div class="meta">
```

/html/body/div[2]/div/div/div/div[2]/div/section/article[4] (http://www.w3.org/1999/xhtml)

```
.list article {                      base.css (line 9)
    border-bottom: 1px solid #CCCCCC;
    margin-bottom: 20px;
    px;
    position: relative;
}
article, aside, details,             base.css (line 9)
figcaption, figure,
footer, header, hgroup,
nav, section {
    display: block;
}
article, aside, details,        bootstrap.css (line 10)
figcaption, figure,
footer, header, hgroup,
nav, section {
    ~~display~~: ~~block~~;
```

Debugging GeoExplorer In case you want to debug the GeoExplorer behaviour in your browser with Firebug of Chromium Developer toolbar, you may do the following:

Install Boundless Suite:

```
$ git clone git://github.com/boundlessgeo/suite.git
$ cd suite
$ git submodule update --init --recursive
```

Run GeoExplorer in debug mode:

```
$ cd geoexplorer
$ ant debug
```

Check if GeoExplore is running at this url: http://localhost:9080

Edit the layers/templates/layers/layer_geoext_map.html file and replace this line:

```
{% include "geonode/geo_header.html" %}
```

with this one:

```
{% include "geonode/geo_header_debug.html" %}
```

Debugging GeoNode's Python Components

Logging References:

- http://docs.python.org/2/library/logging.html
- https://docs.djangoproject.com/en/1.4/topics/logging/

Logging is controlled by the contents of the logging data structure defined in the `settings.py`. The default settings distributed with GeoNode are configured to only log errors. During development, it's a good idea to override the logging data structure with something a bit more verbose.

Output In production, logging output will go into the apache error log. This is located in `/var/log/apache2/error.log`. During development, logging output will, by default, go to standard error.

Configuring

- Ensure the 'console' handler is at the appropriate level. It will ignore log messages below the set level.
- Ensure the specific logger you'd like to use is set at the correct level.
- If attempting to log SQL, ensure `DEBUG=True` in your `local_settings.py`.

Debugging SQL

- To trace all SQL in django, configure the `django.db.backends` logger to `DEBUG`
- To examine a specific query object, you can use the `query` field: str(Layer.objects.all().query)
- You can gather more information by using `django.db.connection.queries`. When `DEBUG` is enabled, query SQL and timing information is stored in this list.

Hints

- Don't use print statements. They are easy to use in development mode but in production they will cause failure.
- Take advantage of python. Instead of:

```
logging.info('some var ' + x + ' is not = ' + y)
```

  Use:

```
logging.info('some var %s is not = %s', x, y)
```

Excercises:

1. Enable logging of all SQL statements. Visit some pages and view the logging output.
2. Using the python shell, use the `queries` object to demonstrate the results of specific queries.

PDB Reference:

- http://docs.python.org/2/library/pdb.html

For the adventurous, `pdb` allows for an interactive debugging session. This is only possible when running in a shell via `manage.py runserver` or `paver runserver`.

To set a breakpoint, insert the following code before the code to debug.

..code-block:: python

import pdb; pdb.set_strace()

When execution reaches this statement, the debugger will activate. The commands are noted in the link above. In addition to those debugger specific commands, general python statements are supported. For example, typing the name of a variable in scope will yield the value via string coersion. Typing "n" will execute the next line, "c" wil continue the execution of the program, "q" will quit.

Debugging GeoServer Resources:

- http://docs.geoserver.org/stable/en/user/advanced/logging.html
- http://docs.geoserver.org/stable/en/user/production/troubleshooting.html

This section does not attempt to cover developer-level debugging in GeoServer as this is a much larger topic involving many more tools. The goal here is to provide 'black-box' techniques to help resolve and report problems.

Logging GeoServer logging, while sometimes containing too much information, is the best way to start diagnosing an issue in GeoNode once the other. To create a proper error report for use in requesting support, providing any contextual logging information is critical.

When using a standard geoserver installation, the GeoServer logs are located at `/usr/share/geoserver/data/logs/geoserver.log`. The properties files that control the varying rules are also located here.

Exercises

1. Switch logging levels for various loggers.
2. Look at the different logging profiles and discuss the loggers and levels.
3. Learn how to read stacktraces, nested or otherwise.

Advanced Troubleshooting JVM diagnostics and advanced troubleshooting techniques are covered in the GeoServer documents linked to above. When providing information for a bug report, these can be helpful but in-depth Java knowledge is required to fully comprehend the output from some of these tools.

Exercises

1. Look at jstack output

Using Django to Help Debug The gsconfig library provides a rich interface to interacting with GeoServer's REST API. This allows high-level functions as well as viewing raw REST responses.

```
cat = Layer.objects.gs_catalog
cat.get_layers() # list of gsconfig layer objects
# OR, for a specific layer
lyr = Layer.objects.get(id=1)
lyr.resource # specfic gsconfig layer object
lyr.resource.fetch() # get the XML from REST
lyr.resource.dom # reference to the parsed XML
from xml.etree.ElementTree import tostring
tostring(lyr.resource.dom)
```

GeoNode APIs

OGC Web Services

Overview At its core, GeoNode provides a standards-based platform to enable integrated, programmatic access to your data via OGC Web Services, which are essential building blocks required to deploy an OGC-compliant spatial data infrastructure (SDI). These Web Services enable discovery, visualization and access your data, all without necessarily having to interact directly with your GeoNode website, look and feel/UI, etc.

OGC Web Services:

- operate over HTTP (GET, POST)
- provide a formalized, accepted API
- provide formalized, accepted formats

The OGC Web Services provided by GeoNode have a mature implementation base and provide an multi-application approach to integration. This means, as a developer, there are already numerous off-the-shelf GIS packages, tools and webapps (proprietary, free, open source) that natively support OGC Web Services.

There are numerous ways to leverage OGC Web Services from GeoNode:

- desktop GIS
- web-based application
- client libraries / toolkits
- custom development

Your GeoNode lists OGC Web Services and their URLs at `http://localhost:8000/developer`. You can use these APIs directly to interact with your GeoNode.

The following sections briefly describe the OGC Web Services provided by GeoNode.

Web Map Service (WMS) WMS provides an API to retrieve map images (png, JPEG, etc.) of geospatial data. WMS is suitable for visualization and when access to raw data is not a requirement.

WFS WFS provides provides an API to retrieve raw geospatial vector data directly. WFS is suitable for direct query and access to geographic features.

WCS WCS provides provides an API to retrieve raw geospatial raster data directly. WCS is suitable for direct access to satellite imagery, DEMs, etc.

CSW CSW provides an interface to publish and search metadata (data about data). CSW is suitable for cataloguing geospatial data and making it discoverable to enable visualization and access.

WMTS WMTS provides an API to retrive pre-rendered map tiles of geospatial data.

WMC WMC provides a format to save and load map views and application state via XML. This allows, for example, a user to save their web mapping application in WMC and share it with others, viewing the same content.

GeoServer REST API

GeoServers Import and Print APIs

GeoNode's Ad-Hoc API

Setting up a GeoNode development environment

This module will lead you through the steps necessary to install a GeoNode development environment.

GeoNode Development Tools Python is not a demamding programming language in terms of specific or advanced development tools. In the chapters *Install GeoNode for Development* and *GeoNode debugging techniques* we have already seen how to setup a geonode instance with the debug tools.

A setup that many GeoNode developers use is composed by four parts:

- a shell with the dev server running
- a shell with the python environment (ran with `‘paver start_django‘`) preferably using ipython
- a text editor for the code
- a browser with firebug/dev tools to inspect the pages and the connections

Many people just code using simple text editors like Vim, although there are some more featured and user friendly text editors or advanced IDEs that can be used.

Text editors like Sublime Text or TextMate offer advanced text interactions with language specific shortcuts, syntax highlight and functions.

The IDEs are full featured frameworks with debug tools, shell, code completion and suggestion, they are powerful but tend to be more complex to use and learn. The most famous are Pycharm and Eclipse with Pydev.

Git Repository Setup

Pavement.py and Paver Paver is a python module that automates repetitive tasks like running documentation generators, moving files around, testing and downloading things using the convenience of Python's syntax and massive library of code. GeoNode comes with several paver tasks which save administrators and developers from having to manually perform repetitive operations from the command line. The tasks are stored in the pavement.py file in your GeoNode root directory and can be run with `'paver <task_name>'` from that directory.

Pavement Tasks Here's a list of Pavement tasks maintained by the GeoNode development team.

deb `'paver deb'`

Creates debian packages.

Use the `key` option (or its shorter version `-k`) to specify the GPG key to sign the package with.

Use the `ppa` option (or its shorter version `-p`) to specify the PPA the package should be published to.

package `'paver package'`

Creates a distributable tarball for GeoNode.

reset `'paver reset'`

Resets the GeoNode development environment by deleting the development database and re-deploying the Geoserver data directory.

reset_hard `'paver reset_hard'`

Cleans the local GeoNode git repository and removes untracked directories.

test `'paver test'`

Runs the GeoNode unit tests.

test_integration `'paver test_integration'`

Runs the GeoNode integration tests.

setup `'paver setup'`

Installs GeoNode's Python dependencies using pip.

setup_data `'paver setup_data'`

Loads sample gis data from the gisdata python package.

Use the `type` option (or its shorter version `-t`) to only import a specific data type. Supported types are "vector", "rastor", and "time."

setup_geoserver `'paver setup_geoserver'`

Downloads Geoserver and the Jetty Runner and then moves the Geoserver data directory to the correct location.

start `'paver start'`

Starts the GeoNode development web server and Geoserver.

start_django `'paver start_django'`

Starts the GeoNode development web server on the local machine.

Use the `bind` option (or its shorter version `-b`) to bind the development server to a specific IP address and port number.

start_geoserver `'paver start_geoserver'`

Runs the local Geoserver using Jetty.

stop `'paver stop'`

Stops the GeoNode development web server and Geoserver.

stop_django `'paver stop_django'`

Stops the GeoNode development web server.

stop_geoserver `'paver stop_geoserver'`

Stops Geoserver.

sync `'paver sync'`

Synchronizes the database according the GeoNode models and loads the GeoNode sample data.

static `'paver static'`

Note: This task requires the Node Package Manager to be installed.

Downloads and installs GeoNode's static file dependencies and creates the production assets.

upgrade_db `'paver upgrade_db'`

Updates database schemas from legacy GeoNode versions.

Use the `version` option (or its shorter version `-v`) to specify the GeoNode version when running this task.

Manually Deploying your Development Environment

Testing in GeoNode

The community encourages the Test Driven Development (TDD) and the contribution to write new tests to extend the coverage. Ideally every model, view, and utility should becovered by tests.

GeoNode has Unit, Integration and Javascript tests. The Unit tests are located

in the tests file of every django app (Maps, Layers, Documents, Catalogue, Search, Security etc).

The Integration, CSW and smoke tests are located under the tests folder).

The tests are meant to be ran using the SQLite database, some of them may fail using PostgreSQL or others.

If running them in development mode make sure to have the jetty server shut down otherwise the test could get stuck. To make sure it is run:

```
    $ paver stop
```

Unit Tests

To run the unit tests make sure you have the virtualenv active (if running GeoNode under virtualenv) then run:

```
    $ paver test
```

This will produce a detailed test report.

It's possible to run just specific apps tests by using the django command:

```
$ python manage.py test app/tests.py
```

For example:

$ python manage.py test geonode.maps.tests

To run a single testcase or method (omit the method name to run the whole class), for example:

$ python manage.py test geonode.maps.tests:MapsTest.test_maps_search

These tests are based on the Python/django unit test suite.

Integration Tests

To run the unit tests make sure you have the virtualenv active (if running GeoNode under virtualenv) then run:

```
    $ paver test_integration
```

To run the csw integration test run:

```
$ paver test_integration -n geonode.tests.csw
```

Like the unit tests above, it is also possible to test specific modules, for example:

$ paver test_integration -n geonode.tests.integration:GeoNodeMapTest.test_search_result_detail

To test with with coverage:

$ python manage.py test geonode.maps.tests – –with-coverage –cover-package=geonode.maps

These tests are based on the Python/django unit test suite.

Javascript Tests

Make a GeoNode release

Making a GeoNode release requires a quite complex preparation of the environment while once everything is set up is a really easy and quick task. As said the complex part is the preparation of the environment since it involves, the generation of a password key to be uploaded to the Ubuntu servers and imported in launchpad.

If you have already prepared the environment then jump to the last paragraph.

Before start, make sure to have a pypi and a launchpad account.

Before doing the release, a GeoNode team member who can already make release has to add you as a launchpad GeoNode team member.

Creating and importing a gpg key A gpg key is needed to push and publish the package. There is a complete guide on how to create and import a gpg key

Preparing the environment Make sure to have a Ubuntu 12.04 machine. Install the following packages in additon to the python virtulenv tools:

```
$ sudo apt-get install git-core git-buildpackage debhelper devscripts
```

Get the GeoNode code (from master) in a virtuelnv:

```
$ mkvirtualenv geonode
$ git clone https://github.com/GeoNode/geonode.git
$ cd geonode
```

Edit the .bashrc file and add the following lines (the key ID can be found in "your personal keys" tab:

```
export GPG_KEY_GEONODE="your_gpg_key_id"
export DEBEMAIL=yourmail@mail.com
export EDITOR=vim
export DEBFULLNAME="Your Full Name"
```

then close and:

```
$ source .bashrc
```

Type "env" to make sure all the variables are correctly exported

Set the correct git email:

```
$ git config --global user.email "yourmail@mail.com"
```

Register on Pypi with your Pypi credentials:

```
$ python setup.py register
```

Make the release The followings are the only commands needed if the environment and the various registrations have already been done.

Make sure to have pulled the master to the desired commit. Edit the file geonode/__init__.py at line 21 and set the correct version.

Install GeoNode in the virtualenv (make sure to have the virtualenv activated and be in the geonode folder):

```
$ pip install -e geonode
```

Publish the package:

```
$ cd geonode
$ paver publish
```

The last command will:

- Tag the release and push it to GitHub
- Create the debian package and push it at ppa:geonode/testing in launchpad
- Create the .tar.gz sources and push them to Pypi
- Update the changelog and commit it to master

Introduction to GeoNode development Learn about GeoNode's core components, its Architecture, the tools it is developed with and the standards it supports.

Development Prerequsites and Core Modules Learn about the pre-requisites you will need in order to develop with GeoNode. Take a look at its core modules and how they work together to provide a complete web mapping tool.

Install GeoNode for Development Prepare your development environment by installing GeoNode in dev mode.

Customized GeoNode Projects Learn how existing projects leverage GeoNode and create your own GeoNode based project.

GeoNode debugging techniques Learn how to debug GeoNode instances and projects.

Setting up a GeoNode development environment Learn how to set up a GeoNode development environment so you can contribute to GeoNode's core.

GeoNode APIs Learn about the APIs GeoNode leverages and provides.

Organizational Learn about GeoNode's development process and how to work with the GeoNode community.

Make a GeoNode release Learn how to make a release of GeoNode

3.2 Reference

The Reference section provides details about the internals of the GeoNode project. It has background information about components that make up GeoNode, the security system, APIs and much more.

3.2.1 Reference documentation

Here you will find information about each and every component of Geonode, for example geoserver, geonode settings, security, etc.

Architecture

Components

Security and Permissions

GeoNode APIs

Localization

GeoNode Django Apps

JavaScript in GeoNode

Settings

Architecture

The Big Picture

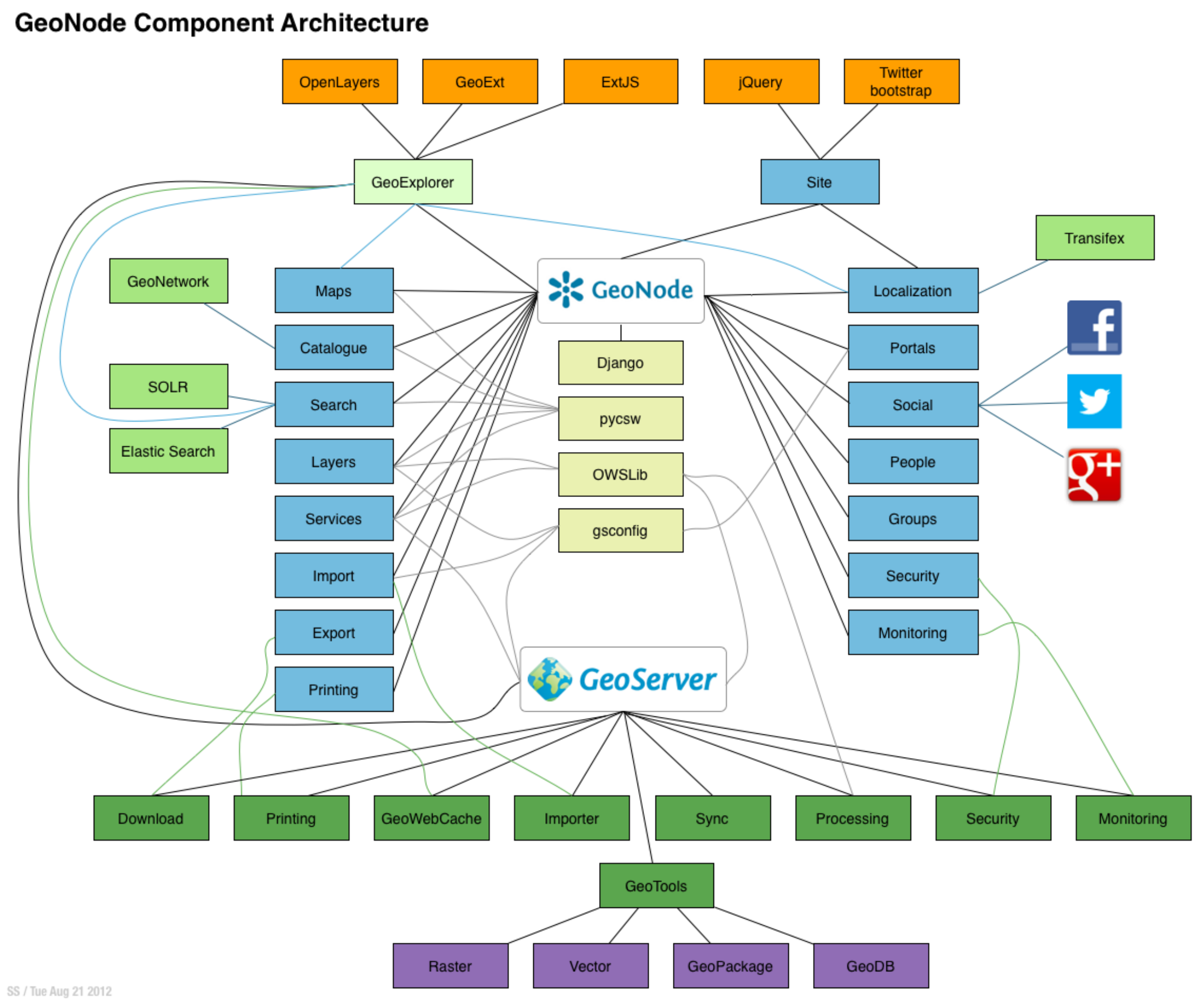


Fig. 3.85: GeoNode Component Architecture

Django Architecture

www.djangoproject.com

MVC/MVT

MVC Model, View, Controller

MVT Model, View, Template

- Model represents application data and provides rich ORM functionality.
- Views are a rendering of a Model most often using the Django template engine.

- In Django, the controller part of this commonly discussed, layered architecture is a subject of discussion. According to the standard definition, the controller is the layer or component through which the user interacts and model changes occur.

More: http://reinout.vanrees.org/weblog/2011/12/13/django-mvc-explanation.html

WSGI

WSGI Web Server Gateway Interface (whis-gey)

- This is a python specification for supporting a common interface between all of the various web frameworks and an application (Apache, for example) that is 'serving'.
- This allows any WSGI compliant framework to be hosted in any WSGI compliant server.
- For most GeoNode development, the details of this specification may be ignored.

More: http://en.wikipedia.org/wiki/Wsgi

GeoNode and GeoServer

GeoNode uses GeoServer for providing OGC services.

- GeoNode configures GeoServer via the REST API
- GeoNode retrieves and caches spatial information from GeoServer. This includes relevant OGC service links, spatial metadata, and attribute information.

  In summary, GeoServer contains the layer data, and GeoNode's layer model extends the metadata present in GeoServer with its own.
- GeoNode can discover existing layers published in a GeoServer via the WMS capabilities document.
- GeoServer delegates authentication and authorization to GeoNode (see README).
- Data uploaded to GeoNode is first processed in GeoNode and finally published to GeoServer (or ingested into the spatial database).

More: http://geoserver.org

GeoNode and PostgreSQL/PostGIS

In production, GeoNode is configured to use PostgreSQL/PostGIS for it's persistent store. In development and testing mode, often an embedded sqlite database is used. The latter is not suggested for production.

- The database stores configuration and application information. This includes users, layers, maps, etc.
- It is recommended that GeoNode be configured to use PostgresSQL/PostGIS for storing vector data as well. While serving layers directly from shapefile allows for adequate performance in many cases, storing features in the database allows for better performance especially when using complex style rules based on attributes.

GeoNode and pycsw

GeoNode is built with pycsw embedded as the default CSW server component.

Publishing Since pycsw is embedded in GeoNode, layers published within GeoNode are automatically published to pycsw and discoverable via CSW. No additional configuration or actions are required to publish layers, maps or documents to pycsw.

Discovery GeoNode's CSW endpoint is deployed available at `http://localhost:8000/catalogue/csw` and is available for clients to use for standards-based discovery. See http://docs.pycsw.org/en/latest/tools.html for a list of CSW clients and tools.

Javascript in GeoNode

- GeoExplorer runs in the browser and talks with GeoNode and GeoServer's APIs using AJAX.
- jQuery is used for incremental enhancement of many GeoNode HTML interfaces.

Components

Architecture is based on a set of core tools and libraries that provide the building blocks on which the application is built. Having a basic understanding of each of these components is critical to your success as a developer working with GeoNode.

Lets look at each of these components and discuss how they are used within the GeoNode application.

Django

GeoNode is based on Django which is a high level Python web development framework that encourages rapid development and clean pragmatic design. Django is based on the Model View Controller (MVC) architecture pattern, and as such, GeoNode models layers, maps and other modules with Django's Model module and and these models are used via Django's ORM in views which contain the business logic of the GeoNode application and are used to drive HTML templates to display the web pages within the application.

GeoServer

GeoServer is a an open source software server written in Java that provides OGC compliant services which publish data from many spatial data sources. GeoServer is used as the core GIS component inside GeoNode and is used to render the layers in a GeoNode instance, create map tiles from the layers, provide for downloading those layers in various formats and to allow for transactional editing of those layers.

GeoExplorer

GeoExplorer is a web application, based on the GeoExt framework, for composing and publishing web maps with OGC and other web based GIS Services. GeoExplorer is used inside GeoNode to provide many of the GIS and cartography functions that are a core part of the application.

PostgreSQL and PostGIS

PostgreSQL and PostGIS are the database components that store and manage spatial data and information for GeoNode and the django modules that it is composed of, pycsw and GeoServer. All of these tables and data are stored within a geonode database in PostgreSQL. GeoServer uses PostGIS to store and manage spatial vector data for each layer which are stored as a separate table in the database.

pycsw

pycsw is an OGC CSW server implementation written in Python. GeoNode uses pycsw to provide an OGC compliant standards-based CSW metadata and catalogue component of spatial data infrastructures, supporting popular geospatial metadata standards such as Dublin Core, ISO 19115, FGDC and DIF.

Geospatial Python Libraries

GeoNode leverages several geospatial python libraries including gsconfig and OWSLib. gsconfig is used to communicates with GeoServer's REST Configuration API to configure GeoNode layers in GeoServer. OWSLib is used to communicate with GeoServer's OGC services and can be used to communicate with other OGC services.

Django Pluggables

GeoNode uses a set of Django plugins which are usually referred to as pluggables. Each of these pluggables provides a particular set of functionality inside the application from things like Registration and Profiles to interactivity with external sites. Being based on Django enables GeoNode to take advantage of the large ecosystem of these pluggables out there, and while a specific set is included in GeoNode itself, many more are available for use in applications based on GeoNode.

jQuery

jQuery is a feature-rich javascript library that is used within GeoNode to provide an interactive and responsive user interface as part of the application. GeoNode uses several jQuery plugins to provide specific pieces of functionality, and the GeoNode development team often adds new features to the interface by adding additional plugins.

Bootstrap

Bootstrap is a front-end framework for laying out and styling the pages that make up the GeoNode application. It is designed to ensure that the pages render and look and behave the same across all browsers. GeoNode customizes bootstraps default style and its relatively easy for developers to customize their own GeoNode based site using existing Boostrap themes or by customizing the styles directly.

Security and Permissions

GeoNode has the ability to restrict the access on your layers, maps, and documents to other users or group of users.

This section should help you to understand which restrictions are possible and what to take care of while using them.

Generally permissions can be set on all your uploaded data. Here´s an overview:

1. Users
   - Superuser permissions
   - Django admin interface permissions
2. Groups
   - Public
   - Public (invite only)
   - Private

3. Layers
   - View a layer
   - Download a layer
   - Change metadata for a layer
   - Edit layer's features
   - Edit styles for a layer
   - Manage a layer (update, delete, change permissions, publish/unpublish it)
4. Maps
   - View a map
   - Download a map
   - Change metadata for a map
   - Manage a map (update, delete, change permissions, publish/unpublish it)
5. Documents
   - View a document
   - Download a document
   - Change metadata for a document
   - Manage a document (update, delete, change permissions, publish/unpublish it)

To understand how this permissions can be set, you first have to know about the different kinds of users.

Permissions and GeoNode objects

Users GeoNode has two types of users:

- Unregistered users (anonymous)
- Registered users

An unregistered user is someone who is just visiting the site, but doesn't have any data uploaded yet. A registered user has already done that. But there are even more kinds of registered users! A registered user can have one or more of those three status:

- Superuser
- Staff
- Active

A superuser is usually generated directly after the installation of GeoNode via the terminal. When creating a *superuser* through the terminal it always has the status *active* and the status *staff* as well. It is also important to know that a superuser is a user that has all permissions without explicitly assigning them! That means that he is able to upload and edit layers, create maps etc. and can not be restricted from that! So the superuser is basically the administrator, who knows and has access on everything.

The status *staff* only implies that a user with this status is able to attend the *Django Admin Interface*. *Active* has no special meaning, it only says that there is a user and it is available. Instead of deleting this user, you could just unset the status *active*, your user will still be remaining, but it won´t show up.

There are several options to create a user:

- From the terminal: Here you can only create a *superuser*
- From the GeoNode interface (when GeoNode registration are open): A *normal* user will be created by signing up to GeoNode. It only has the status *active* so far!
- From the GeoNode interface (when GeoNode registration are closed): A superuser will be able to invite a user
- From the Django administrative interface: a new user can be created as well as the status of an already existing user can be changed, e.g make a generic user a superuser.

Groups In GeoNode you can assign permissions to groups, all the users that belong to the group will inherit its permissions.

If you are an administrator you can create a group in the dedicated tab and invite or assign users to it. The group will be available in the permissions widget in geonode and you will be able to assign object permissions to it.

Layers As mentioned above, a superuser or the layer owner or a user with management permissions on the layer should be able to restrict other users from the layer itself.

Generally there are the following types of permissions:

These are permissions that is possible to assign to a GeoNode layer:

- Who can view the layer
- Who can download the layer
- Who can change the metadata of the layer
- Who can edit data of the layer
- Who can edit styles of the layer
- Who can manage the layer (update, delete, change permissions, publish/unpublish it)

Each of these permissions can be assigned to:

- Anyone (only for who can view and download)
- One or more users
- One or more groups

A user with all of these permissions in the layer detail page will have a button to download the layer, a button to download its metadata, a button to change the layer permissions and an edit button that will display links to:

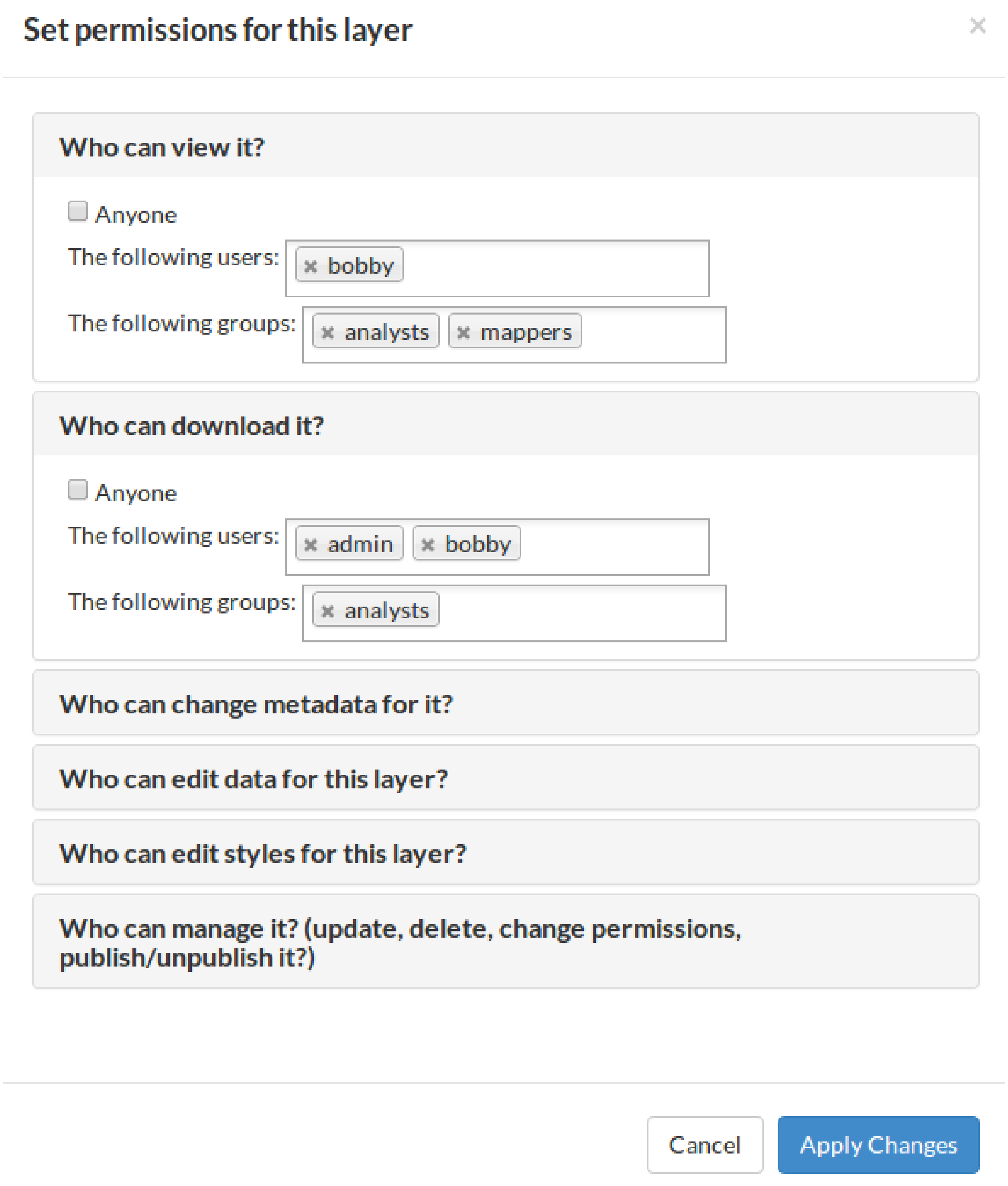
- Edit metadata
- Edit styles
- Manage styles
- Replace the layer
- Remove the layer

This can also be seen here:

If the layer is vectorial the user will be able also to edit the layer's features in a GeoNode map (the "Edit" tool should be enabled).

Now take a closer look on to the section *Edit Metadata*. All the following things can be edited in the metadata section:

- Owner
- Title



Set permissions for this layer

Who can view it?

[ ] Anyone

The following users: × bobby

The following groups: × analysts × mappers

Who can download it?

[ ] Anyone

The following users: × admin × bobby

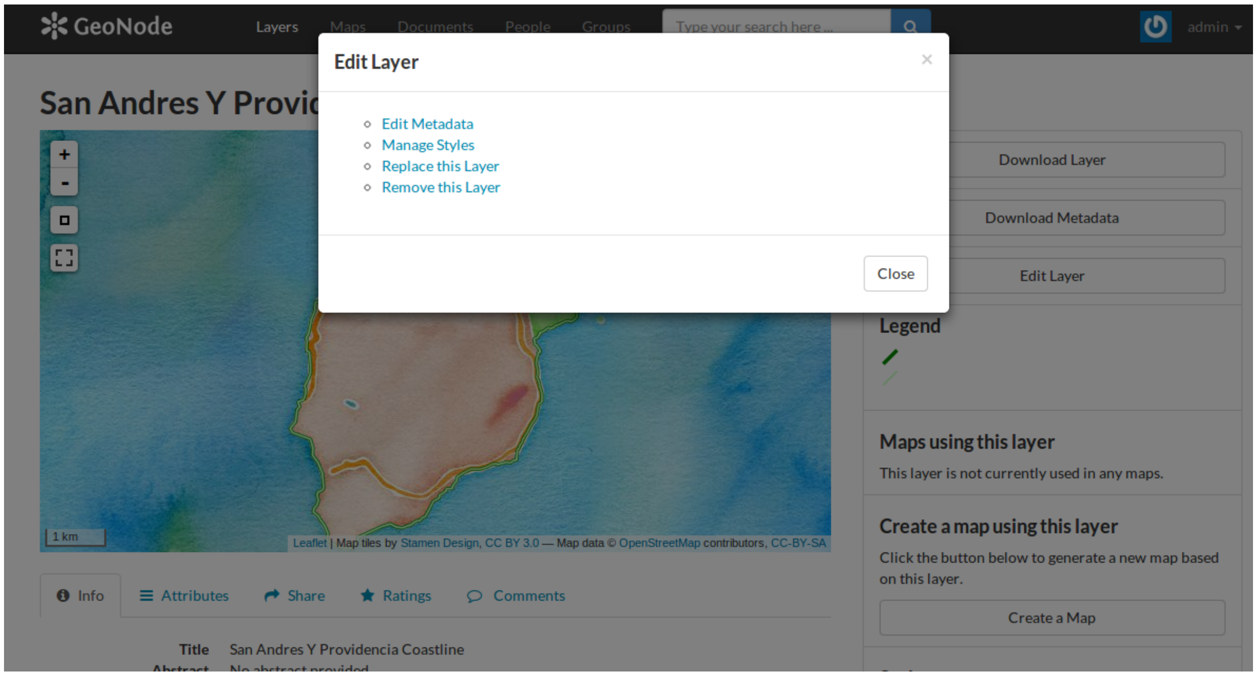
The following groups: × analysts

Who can change metadata for it?

Who can edit data for this layer?

Who can edit styles for this layer?

Who can manage it? (update, delete, change permissions, publish/unpublish it?)



Maps using this layer

This layer is not currently used in any maps.

Create a map using this layer

Click the button below to generate a new map based on this layer.

Create a Map

Title San Andres Y Providencia Coastline

Abstract No abstract provided

- Date
- Data type
- Edition
- Abstract
- Purpose
- Maintenance frequency
- Keywords region
- Restrictions
- Restrictions other
- Language
- Category
- Spatial representation type
- Temporal extent start
- Temporal extent end
- Supplemental information
- Distribution URL
- Distribution description
- Data quality statement
- Keywords
- Point of contact
- Metadata author

- Attributes (those can though not be changed!)

Maps Generally all the same applies to maps, but with fewer options:

- Who can view the map
- Who can download the map
- Who can change the metadata of the map
- Who can manage (delete, change permissions, publish/unpublish it, set map thumbnail)

The section *Edit metadata* is almost the same as for layers, with two more options:

- Metadata XML
- Thumbnail

In *Set map thumbnail* the thumbnail of the map can be set.

Documents The same permissions that can be used on layers can be used on the documents, with the exception of the edit data and edit styles permissions.

Require authentication to access GeoNode By default, unregistered users cannot view maps, layers, and documents on your site without being authenticated. GeoNode comes a security option that requires users to authenticate before accessing any page. To enable this option, set the `LOCKDOWN_GEONODE` setting to true in your `settings.py` file. You can fine-tune which url routes are white-listed (accessible by unregistered users) by listing the routes in the `AUTH_EXEMPT_URLS` tuple. See the *GeoNode Django Apps* documentation for more information.

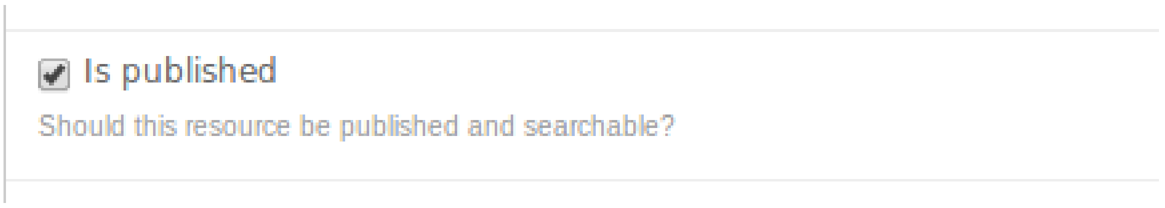
Publishing and unpublishing objects

By default GeoNode does not implement any kind of mechanism to publish/unpublish resources such as layer, maps and documents.

Setting the RESOURCE_PUBLISHING to True such a workflow is used, and by default new uploaded resources are unpublished.

It is possible for any GeoNode staff member that has permissions on the base/ResourceBase model to decide to publish/unpublish a layer, map or document.

The staff member can go to the resource base Django admin page, and publish or unpublish the resource by checking or unchecking the is_published field:


[✓] Is published

Should this resource be published and searchable?

When the resource is unpublished, it will be not available to any user, including administrators, in the GeoNode site. If the unpublished resource is a layer it will be considered in the GetCapabilities generated by GeoServer.

The unpublished resource will not be reachable by anyone using GeoNode search features. The only way to access to it is by the Django admin site, from where it will be eventually possible to publish again the resource by a staff member, or from the layer details page, accessible by any user with the publish_resourcebase permission on that layer.

GeoNode ah-hoc API

GeoNode provides JSON API which currently support the GET method. The API are also used as main serch engine.

API endpoints

GeoNode provides some enpoints and filtering.

- *"/api/base"* query on the ResourceBase table and returns combined results of Maps, Layers Documents and Services
- *"/api/layers"* query the Layer table
- *"/api/maps"* query the Map table
- *"/api/documents"* query the Document table
- *"/api/groups"* query the GroupProfile table (which contains the Groups)
- *"/api/profiles"* query the Profile table (which is the geonode authentication table)
- *"/api/categories"* query the Category table
- *"/api/keywords"* query the Tag table
- *"/api/featured"* query the ResouceBase table by limiting the items to the ones flagged as "featured" (listed in home page)

API filtering

The API allow filtering by adding django style model filters to the url.

As an example, filtering by title corresponds to a url like *"/api/layers?title__contains=grid"* It's also possible to filter by related tables like *"/api/layers?keywords__slug__exact=the-keyword"*

There are many possible filter, refer to the django filters guide.

API limit and pagination

It's possible to limite the number of the results returned by the API by adding a limit parameter like *"/api/layers?limit=10"* It's also possible to specify an offset so that the firts results will not be returned (together with the limit this makes a pagination logic), *"/api/layers?offset=5"*

So a query like *"/api/layers?offset=5&limit=10"* will return 10 results starting from the 6th found in the database.

API settings

You can configure how many results will be lists per page on the client (in the list pages and search page) by changing this line https://github.com/GeoNode/geonode/blob/master/geonode/settings.py#L643

And you can set the amount of data returned by default from the API (if the limit parameter is not set), the default is 0 which means no limit https://github.com/GeoNode/geonode/blob/master/geonode/settings.py#L646

Searching with Haystack

GeoNode is ready to use a complete full text search engine. Note that haystack will be used only on the base, layers, maps and documents API.

Once activated the full text API is reachable by appending "search" to the url, for example *"/api/base/search?limit=0&offset=0"*

Although the backend type is not mandatory, we suggest (for simplicity) to use Elastichsearch:

To activate the search backend make sure that you have a running instance of Elastichsearch, then uncomment the following line in the geonode settings:

https://github.com/GeoNode/geonode/blob/master/geonode/settings.py#L219

And activate the search through the settings at the line:

https://github.com/GeoNode/geonode/blob/master/geonode/settings.py#L607

Also uncomment and correct the address of elastichsearch if needed: https://github.com/GeoNode/geonode/blob/master/geonode/settings
https://github.com/GeoNode/geonode/blob/master/geonode/settings.py#L619

You can do some more customizations like:

- should the search skip the permissions filtering? https://github.com/GeoNode/geonode/blob/master/geonode/settings.py#L609
- should the search update the facets counts on every search you make or keep the standard behavior? https://github.com/GeoNode/geonode/blob/master/geonode/settings.py#L611
- How many results should the backend return by default? https://github.com/GeoNode/geonode/blob/master/geonode/settings.py#L

Localization

To enable a new language in GeoNode you have to do the following:

1. Install gettext:

```
sudo apt-get install gettext
```

2. Create a directory named locale in the root of your project:

```
mkdir locale
```

3. In the root of your project, run:

```
python manage.py makemessages -l fr
```

4. Navigate to the GeoNode dir and do:

```
cd src/GeoNodePy/geonode/maps; django-admin.py makemessages -l fr
cd src/GeoNodePy/geonode; django-admin.py makemessages -l fr
```

Optional steps:

1. Install django-rossetta:

```
http://code.google.com/p/django-rosetta/
```

2. Install django-modeltranslation
3. If you want to enable metadata in the other format too, make sure you have model translation installed and create a translations.py file like this:

```
from modeltranslation.translator import translator, TranslationOptions
from geonode.maps.models import Layer

class LayerTO(TranslationOptions):
    fields = (
             'title',
             'edition',
             'abstract',
             'purpose',
             'constraints_other',
             'distribution_description',
             'data_quality_statement',
             'supplemental_information',
             )

translator.register(FlatBlock, FlatBlockTO)
translator.register(Layer, LayerTO)
```

Developers Reference

Here you will find information about each and every component of Geonode, for example geoserver, geonode settings, security, etc.

GeoNode Django Apps

JavaScript in GeoNode

Settings

GeoNode Django Apps

The user interface of a GeoNode site is built on top of the Django web framework. GeoNode includes a few "apps" (reusable Django modules) to support development of those user interfaces. While these apps have reasonable default configurations, for customized GeoNode sites you will probably want to adjust these apps for your specific needs.

`geonode.base` - GeoNode core functionality

Stores core functionality used throughout the GeoNode application.

Template Tags num_ratings <object>

Returns the number of ratings an object has. Example usage:

```
{% load base_tags %}
{% num_ratings map as num_votes %}

<p>Map votes: {{num_votes}}.</p>
```

categories

Returns topic categories and the count of objects in each category.

`geonode.documents` - Document creation and management

Manages uploaded files that can be related to maps. Documents can be any type of file that is included in the ALLOWED_DOCUMENTS_TYPES setting.

`settings.py` Entries

ALLOWED_DOCUMENT_TYPES Default: `['doc', 'docx', 'xls', 'xslx', 'pdf', 'zip', 'jpg', 'jpeg', 'tif', 'tiff', 'png', 'gif', 'txt']`

A list of acceptable file extensions that can be uploaded to the Documents app.

MAX_DOCUMENT_SIZE Default: `2`

The maximum size (in megabytes) that an upload will be before it gets rejected.

`geonode.layers` - Layer creation and geospatial data management This Django app provides support for managing and manipulating single geospatial datasets known as layers.

Models

- Attribute - Feature attributes for a layer managed by the GeoNode.
- Layer - A data layer managed by the GeoNode
- Style - A data layer's style managed by the GeoNode

Views

- Creating, viewing, browsing, editing, and deleting layers and their metadata

Template Tags

featured_layers Returns the the 7 newest layers.

layer_thumbnail <layer> Returns the layer's thumbnail.

`manage.py` Commands

importlayers `python manage.py importlayers`

Brings a data file or a directory full of data files into a GeoNode site. Layers are added to the Django database, the GeoServer configuration, and the GeoNetwork metadata index.

updatelayers `python manage.py updatelayers`

Scan Geoserver for data that has not been added to GeoNode.

`geonode.maps` - Map creation and geospatial data management This Django app provides some support for managing and manipulating geospatial datasets. In particular, it provides tools for editing, viewing, and searching metadata for data layers, and for editing, viewing, and searching maps that aggregate data layers to display data about a particular topic.

Models

- Map - A collection of data layers composed in a particular order to form a map
- MapLayer - A model that maintains some map-specific information related to a layer, such as the z-indexing order.

Views The maps app provides views for:

- Creating, viewing, browsing, editing, and deleting Maps

These operations require the use of GeoServer to manage map rendering, as well as GeoExt to provide interactive editing and previewing of maps and data layers.

There are also some url mappings in the `geonode.maps.urls` module for easy inclusion in GeoNode sites.

`settings.py` Entries

OGC_SERVER Default: `{}` (Empty dictionary)

A dictionary of OGC servers and and their options. The main server should be listed in the 'default' key. If there is no 'default' key or if the `OGC_SERVER` setting does not exist GeoNode will raise an Improperly Configured exception. Below is an example of the `OGC_SERVER` setting:

```
OGC_SERVER = {
  'default' : {
      'LOCATION' : 'http://localhost:8080/geoserver/',
      'USER' : 'admin',
      'PASSWORD' : 'geoserver',
  }
}
```

BACKEND Default: `"geonode.geoserver"`

The OGC server backend to use. The backend choices are:

- `'geonode.geoserver'`

BACKEND_WRITE_ENABLED Default: `True`

Specifies whether the OGC server can be written to. If False, actions that modify data on the OGC server will not execute.

LOCATION Default: `"http://localhost:8080/geoserver/"`

A base URL from which GeoNode can construct OGC service URLs. If using Geoserver you can determine this by visiting the GeoServer administration home page without the /web/ at the end. For example, if your GeoServer administration app is at http://example.com/geoserver/web/, your server's location is http://example.com/geoserver.

PUBLIC_LOCATION Default: `"http://localhost:8080/geoserver/"`

The URL used to in most public requests from GeoNode. This settings allows a user to write to one OGC server (the LOCATION setting) and read from a seperate server or the PUBLIC_LOCATION.

USER Default: `'admin'`

The administrative username for the OGC server as a string.

PASSWORD Default: `'geoserver'`

The administrative password for the OGC server as a string.

MAPFISH_PRINT_ENABLED Default: `True`

A boolean that represents whether the Mapfish printing extension is enabled on the server.

PRINT_NG_ENABLED Default: `True`

A boolean that represents whether printing of maps and layers is enabled.

GEONODE_SECURITY_ENABLED Default: `True`

A boolean that represents whether GeoNode's security application is enabled.

GEOGIT_ENABLED Default: `False`

A boolean that represents whether the OGC server supports Geogig datastores.

WMST_ENABLED Default: `False`

Not implemented.

WPS_ENABLED Default: `False`

Not implemented.

DATASTORE Default: `''` (Empty string)

An optional string that represents the name of a vector datastore that GeoNode uploads are imported into. In order to support vector datastore imports there also needs to be an entry for the datastore in the `DATABASES` dictionary with the same name. Example:

```
OGC_SERVER = {
  'default' : {
     'LOCATION' : 'http://localhost:8080/geoserver/',
     'USER' : 'admin',
     'PASSWORD' : 'geoserver',
     'DATASTORE': 'geonode_imports'
  }
}

DATABASES = {
 'default': {
     'ENGINE': 'django.db.backends.sqlite3',
     'NAME': 'development.db',
 },
 'geonode_imports' : {
     'ENGINE': 'django.contrib.gis.db.backends.postgis',
     'NAME': 'geonode_imports',
     'USER' : 'geonode_user',
     'PASSWORD' : 'a_password',
     'HOST' : 'localhost',
     'PORT' : '5432',
  }
 }
```

GEOSERVER_CREDENTIALS Removed in GeoNode 2.0, this value is now specified in the `OGC_SERVER` settings.

GEOSERVER_BASE_URL Removed in GeoNode 2.0, this value is now specified in the `OGC_SERVER` settings.

CATALOGUE A dict with the following keys:

- ENGINE: The CSW backend (default is `geonode.catalogue.backends.pycsw_local`)
- URL: The FULLY QUALIFIED base url to the CSW instance for this GeoNode
- USERNAME: login credentials (if required)
- PASSWORD: login credentials (if required)

pycsw is the default CSW enabled in GeoNode. pycsw configuration directives are managed in the PYCSW entry.

PYCSW A dict with pycsw's configuration. Of note are the sections `metadata:main` to set CSW server metadata and `metadata:inspire` to set INSPIRE options. Setting `metadata:inspire['enabled']` to `true` will enable INSPIRE support. See http://pycsw.org/docs/configuration.html for full pycsw configuration details.

SITEURL Default: `'http://localhost:8000/'`

A base URL for use in creating absolute links to Django views.

DEFAULT_MAP_BASE_LAYER The name of the background layer to include in newly created maps.

DEFAULT_MAP_CENTER Default: `(0, 0)`

A 2-tuple with the latitude/longitude coordinates of the center-point to use in newly created maps.

DEFAULT_MAP_ZOOM Default: `0`

The zoom-level to use in newly created maps. This works like the OpenLayers zoom level setting; 0 is at the world extent and each additional level cuts the viewport in half in each direction.

`geonode.proxy` - Assist JavaScript applications in accessing remote servers This Django app provides some HTTP proxies for accessing data from remote servers, to overcome restrictions imposed by the same-origin policy used by browsers. This helps the GeoExt applications in a GeoNode site to access various XML documents from OGC-compliant data services.

Views

geonode.proxy.views.proxy This view forwards requests without authentication to a URL provided in the request, similar to the proxy.cgi script provided by the OpenLayers project.

geonode.proxy.views.geoserver This view proxies requests to GeoServer. Instead of a URL-encoded URL parameter, the path component of the request is expected to be a path component for GeoServer. Requests to this URL require valid authentication against the Django site, and will use the default `OGC_SERVER` `USER`, `PASSWORD` and `LOCATION` settings as defined in the maps application.

`geonode.search` - Provides the GeoNode search functionality. This Django app provides a fast search functionality to GeoNode.

Views

- search_api- Builds and executes a search query based on url parameters and returns matching results in requested format.

`geonode.security` - GeoNode granular Auth Backend This app provides an authentication backend for use in assigning permissions to individual objects (maps and layers).

`settings.py` Entries

LOCKDOWN_GEONODE Default: `False`

By default, the GeoNode application allows visitors to view most pages without being authenticated. Set `LOCKDOWN_GEONODE = True` to require a user to be authenticated before viewing the application.

AUTH_EXEMPT_URLS Default: `()` (Empty tuple)

A tuple of url patterns that the user can visit without being authenticated. This setting has no effect if `LOCKDOWN_GEONODE` is not True. For example, `AUTH_EXEMPT_URLS = ('/maps',)` will allow unauthenticated users to browse maps.

Template Tags

geonode_media <media_name> Accesses entries in MEDIA_LOCATIONS without requiring the view to explicitly add it to the template context. Example usage:

```
{% include geonode_media %}
{% geonode_media "ext_base" %}
```

has_obj_perm <user> <obj> <permission> Checks whether the user has the specified permission on an object.

```
{% has_obj_perm user obj "app.view_thing" as can_view_thing %}
```

Django's error templates GeoNode customizes some of Django's default error templates.

500.html If no custom handler for 500 errors is set up in the URLconf module, django will call django.views.defaults.server_error which expects a 500.html file in the root of the templates directory. In GeoNode, we have put a template that does not inherit from anything as 500.html and because most of Django's machinery is down when an INTERNAL ERROR (500 code) is encountered the use of template tags should be avoided.

JavaScript in GeoNode

GeoNode provides a number of facilities for interactivity in the web browser built on top of several high-quality JavaScript frameworks:

- Bootstrap for GeoNode's front-end user interface and common user interaction.
- Bower for GeoNode's front-end package management.
- ExtJS for component-based UI construction and data access
- OpenLayers for interactive mapping and other geospatial operations
- GeoExt for integrating ExtJS with OpenLayers
- Grunt for front-end task automation.
- GXP for providing some higher-level application building facilities on top of GeoExt, as well as improving integration with GeoServer.
- jQuery to abstract javascript manipulation, event handling, animation and Ajax.

GeoNode uses application-specific modules to handle pages and services that are unique to GeoNode. This framework includes:

- A GeoNode mixin class that provides GeoExplorer with the methods needed to properly function in GeoNode. The class is responsible for checking permissions, retrieving and submitting the CSRF token, and user authentication.
- A search module responsible for the GeoNode's site-wide search functionality.
- An upload and status module to support file uploads.
- Template files for generating commonly used html sections.
- A front-end testing module to test GeoNode javascript.

The following concepts are particularly important for developing on top of the GeoNode's JavaScript framework.

- Components - Ext components handle most interactive functionality in "regular" web pages. For example, the scrollable/sortable/filterable table on the default Search page is a Grid component. While GeoNode does use some custom components, familiarity with the idea of Components used by ExtJS is applicable in GeoNode development.

- Viewers - Viewers display interactive maps in web pages, optionally decorated with Ext controls for toolbars, layer selection, etc. Viewers in GeoNode use the GeoExplorer base class, which builds on top of GXP's Viewer to provide some common functionality such as respecting site-wide settings for background layers. Viewers can be used as components embedded in pages, or they can be full-page JavaScript applications.
- Controls - Controls are tools for use in OpenLayers maps (such as a freehand control for drawing new geometries onto a map, or an identify control for getting information about individual features on a map.) GeoExt provides tools for using these controls as ExtJS "Actions" - operations that can be invoked as buttons or menu options or associated with other events.

Settings

Here's a list of settings available in GeoNode and their default values. This includes settings for some external applications that GeoNode depends on.

Documents settings Here's a list of settings available for the Documents app in GeoNode.

ALLOWED_DOCUMENT_TYPES Default: `['doc', 'docx', 'xls', 'xlsx', 'pdf', 'zip', 'jpg', 'jpeg', 'tif', 'tiff', 'png', 'gif', 'txt']`

A list of acceptable file extensions that can be uploaded to the Documents app.

MAX_DOCUMENT_SIZE Default: `2`

Metadata settings

CATALOGUE A dict with the following keys:

- ENGINE: The CSW backend (default is `geonode.catalogue.backends.pycsw_local`)
- URL: The FULLY QUALIFIED base url to the CSW instance for this GeoNode
- USERNAME: login credentials (if required)
- PASSWORD: login credentials (if required)

pycsw is the default CSW enabled in GeoNode. pycsw configuration directives are managed in the PYCSW entry.

PYCSW

A dict with pycsw's configuration. Of note are the sections `metadata:main` to set CSW server metadata and `metadata:inspire` to set INSPIRE options. Setting `metadata:inspire['enabled']` to `true` will enable INSPIRE support. See http://docs.pycsw.org/en/latest/configuration.html for full pycsw configuration details.

DEFAULT_TOPICCATEGORY Default: `'location'`

The identifier of the default topic category to use when uploading new layers. The value specified for this setting must be present in the TopicCategory table or GeoNode will return a `TopicCategory.DoesNotExist` exception.

MODIFY_TOPICCATEGORY Default: `False`

Metadata Topic Categories list should not be modified, as it is strictly defined by ISO (See: http://www.isotc211.org/2005/resources/Codelist/gmxCodelists.xml and check the <CodeListDictionary gml:id="MD_MD_TopicCategoryCode"> element).

Some customisation it is still possible changing the is_choice and the GeoNode description fields.

In case it is absolutely necessary to add/delete/update categories, it is possible to set the MODIFY_TOPICCATEGORY setting to True.

Maps settings

DEFAULT_MAP_BASE_LAYER The name of the background layer to include in newly created maps.

DEFAULT_MAP_CENTER Default: `(0, 0)`

A 2-tuple with the latitude/longitude coordinates of the center-point to use in newly created maps.

DEFAULT_MAP_ZOOM Default: `0`

The zoom-level to use in newly created maps. This works like the OpenLayers zoom level setting; 0 is at the world extent and each additional level cuts the viewport in half in each direction.

MAP_BASELAYERS Default:

```
MAP_BASELAYERS = [{
"source": {
    "ptype": "gxp_wmscsource",
    "url": OGC_SERVER['default']['PUBLIC_LOCATION'] + "wms",
    "restUrl": "/gs/rest"
 }
  },{
    "source": {"ptype": "gxp_olsource"},
    "type":"OpenLayers.Layer",
    "args":["No background"],
    "visibility": False,
    "fixed": True,
    "group":"background"
  }, {
    "source": {"ptype": "gxp_osmsource"},
    "type":"OpenLayers.Layer.OSM",
    "name":"mapnik",
    "visibility": False,
    "fixed": True,
    "group":"background"
  }, {
    "source": {"ptype": "gxp_mapquestsource"},
    "name":"osm",
    "group":"background",
    "visibility": True
  }, {
    "source": {"ptype": "gxp_mapquestsource"},
    "name":"naip",
    "group":"background",
    "visibility": False
  }, {
    "source": {"ptype": "gxp_bingsource"},
    "name": "AerialWithLabels",
    "fixed": True,
    "visibility": False,
    "group":"background"
  },{
    "source": {"ptype": "gxp_mapboxsource"},
  }, {
    "source": {"ptype": "gxp_olsource"},
    "type":"OpenLayers.Layer.WMS",
    "group":"background",
    "visibility": False,
    "fixed": True,
    "args":[
      "bluemarble",
      "http://maps.opengeo.org/geowebcache/service/wms",
      {
        "layers":["bluemarble"],
        "format":"image/png",
        "tiled": True,
        "tilesOrigin": [-20037508.34, -20037508.34]
      },
      {"buffer": 0}
    ]

}]
```

A list of dictionaries that specify the default map layers.

LAYER_PREVIEW_LIBRARY Default: `"leaflet"`

The library to use for display preview images of layers. The library choices are:

- `"leaflet"`
- `"geoext"`

OGC_SERVER Default: `{}` (Empty dictionary)

A dictionary of OGC servers and and their options. The main server should be listed in the 'default' key. If there is no 'default' key or if the `OGC_SERVER` setting does not exist Geonode will raise an Improperly Configured exception. Below is an example of the `OGC_SERVER` setting:

```
OGC_SERVER = {
  'default' : {
      'LOCATION' : 'http://localhost:8080/geoserver/',
      'USER' : 'admin',
      'PASSWORD' : 'geoserver',
  }
}
```

BACKEND Default: `"geonode.geoserver"`

The OGC server backend to use. The backend choices are:

- `'geonode.geoserver'`

BACKEND_WRITE_ENABLED Default: `True`

Specifies whether the OGC server can be written to. If False, actions that modify data on the OGC server will not execute.

DATASTORE Default: `''` (Empty string)

An optional string that represents the name of a vector datastore that Geonode uploads are imported into. In order to support vector datastore imports there also needs to be an entry for the datastore in the `DATABASES` dictionary with the same name. Example:

```
OGC_SERVER = {
  'default' : {
     'LOCATION' : 'http://localhost:8080/geoserver/',
     'USER' : 'admin',
     'PASSWORD' : 'geoserver',
     'DATASTORE': 'geonode_imports'
  }
}

DATABASES = {
 'default': {
     'ENGINE': 'django.db.backends.sqlite3',
     'NAME': 'development.db',
 },
 'geonode_imports' : {
     'ENGINE': 'django.contrib.gis.db.backends.postgis',
     'NAME': 'geonode_imports',
     'USER' : 'geonode_user',
     'PASSWORD' : 'a_password',
     'HOST' : 'localhost',
     'PORT' : '5432',
  }
 }
```

GEOGIG_ENABLED Default: `False`

A boolean that represents whether the OGC server supports GeoGig datastores.

GEONODE_SECURITY_ENABLED Default: `True`

A boolean that represents whether Geonode's security application is enabled.

LOCATION Default: `"http://localhost:8080/geoserver/"`

A base URL from which GeoNode can construct OGC service URLs. If using Geoserver you can determine this by visiting the GeoServer administration home page without the /web/ at the end. For example, if your GeoServer administration app is at http://example.com/geoserver/web/, your server's location is http://example.com/geoserver.

MAPFISH_PRINT_ENABLED Default: `True`

A boolean that represents whether the Mapfish printing extension is enabled on the server.

PASSWORD Default: `'geoserver'`

The administrative password for the OGC server as a string.

PRINT_NG_ENABLED Default: `True`

A boolean that represents whether printing of maps and layers is enabled.

PUBLIC_LOCATION Default: `"http://localhost:8080/geoserver/"`

The URL used to in most public requests from Geonode. This settings allows a user to write to one OGC server (the LOCATION setting) and read from a seperate server or the PUBLIC_LOCATION.

USER Default: `'admin'`

The administrative username for the OGC server as a string.

WMST_ENABLED Default: `False`

Not implemented.

WPS_ENABLED Default: `False`

Not implemented.

TIMEOUT Default: `10`

The maximum time, in seconds, to wait for the server to respond.

SITEURL Default: `'http://localhost:8000/'`

A base URL for use in creating absolute links to Django views and generating links in metadata.

Proxy settings

PROXY_ALLOWED_HOSTS Default: `()` (Empty tuple)

A tuple of strings representing the host/domain names that GeoNode can proxy requests to. This is a security measure to prevent an attacker from using the GeoNode proxy to render malicious code or access internal sites.

Values in this tuple can be fully qualified names (e.g. ‘www.geonode.org’), in which case they will be matched against the request’s Host header exactly (case-insensitive, not including port). A value beginning with a period can be used as a subdomain wildcard: `.geonode.org` will match geonode.org, www.geonode.org, and any other subdomain of geonode.org. A value of ‘*’ will match anything and is not recommended for production deployments.

PROXY_URL Default `/proxy/?url=`

The url to a proxy that will be used when making client-side requests in GeoNode. By default, the internal GeoNode proxy is used but administrators may favor using their own, less restrictive proxies.

Search settings

DEFAULT_SEARCH_SIZE Default: `10`

An integer that specifies the default search size when using `geonode.search` for querying data.

Security settings

AUTH_EXEMPT_URLS Default: `()` (Empty tuple)

A tuple of url patterns that the user can visit without being authenticated. This setting has no effect if `LOCKDOWN_GEONODE` is not True. For example, `AUTH_EXEMPT_URLS = ('/maps',)` will allow unauthenticated users to browse maps.

LOCKDOWN_GEONODE Default: `False`

By default, the GeoNode application allows visitors to view most pages without being authenticated. If this is set to `True` users must be authenticated before accessing URL routes not included in `AUTH_EXEMPT_URLS`.

RESOURCE_PUBLISHING Default: `True`

By default, the GeoNode application allows GeoNode staff members to publish/unpublish resources. By default resources are published when created. When this settings is set to True the staff members will be able to unpublish a resource (and eventually publish it back).

Social settings

SOCIAL_BUTTONS Default: `True`

A boolean which specifies whether the social media icons and javascript should be rendered in GeoNode.

SOCIAL_ORIGINS Default:

```
SOCIAL_ORIGINS = [{
    "label":"Email",
    "url":"mailto:?subject={name}&body={url}",
    "css_class":"email"
}, {
    "label":"Facebook",
    "url":"http://www.facebook.com/sharer.php?u={url}",
    "css_class":"fb"
}, {
    "label":"Twitter",
    "url":"https://twitter.com/share?url={url}",
    "css_class":"tw"
}, {
    "label":"Google +",
    "url":"https://plus.google.com/share?url={url}",
    "css_class":"gp"
}]
```

A list of dictionaries that is used to generate the social links displayed in the Share tab. For each origin, the name and and url format parameters are replaced by the actual values of the ResourceBase object (layer, map, document).

CKAN_ORIGINS Default:

```
CKAN_ORIGINS = [{
    "label":"Humanitarian Data Exchange (HDX)",
    "url":"https://data.hdx.rwlabs.org/dataset/new?title={name}&notes={abstract}",
    "css_class":"hdx"
}]
```

A list of dictionaries that is used to generate the links to CKAN instances displayed in the Share tab. For each origin, the name and and abstract format parameters are replaced by the actual values of the ResourceBase object (layer, map, document). This is not enabled by default. To enabled, uncomment the following line: SOCIAL_ORIGINS.extend(CKAN_ORIGINS).

Upload settings

GEOGIG_DATASTORE_NAME Default: `None`

A string with the default GeoGig datastore name. This value is only used if no GeoGig datastore name is provided when data is uploaded but it must be populated if your deployment supports GeoGig.

UPLOADER Default:

```
{
    'BACKEND' : 'geonode.rest',
    'OPTIONS' : {
        'TIME_ENABLED': False,
        'GEOGIG_ENABLED': False,
    }
}
```

A dictionary of Uploader settings and and their values.

BACKEND Default: `'geonode.rest'`

The uploader backend to use. The backend choices are:

- `'geonode.importer'`
- `'geonode.rest'`

The importer backend requires the Geoserver importer extension to be enabled and is required for uploading data into GeoGig datastores.

OPTIONS Default:

```
'OPTIONS' : {
    'TIME_ENABLED': False,
    'GEOGIG_ENABLED': False,
}
```

TIME_ENABED Default: `False`

A boolean that specifies whether the upload should allow the user to enable time support when uploading data.

GEOGIG_ENABED Default: `False`

A boolean that specifies whether the uploader should allow the user to upload data into a GeoGig datastore.

User Account settings

REGISTRATION_OPEN Default: `False`

A boolean that specifies whether users can self-register for an account on your site.

THEME_ACCOUNT_CONTACT_EMAIL Default: `'admin@example.com'`

This email address is added to the bottom of the password reset page in case users have trouble un-locking their account.

Download settings

DOWNLOAD_FORMATS_METADATA Specifies which metadata formats are available for users to download.

Default:

```
DOWNLOAD_FORMATS_METADATA = [
    'Atom', 'DIF', 'Dublin Core', 'ebRIM', 'FGDC', 'ISO',
]
```

DOWNLOAD_FORMATS_VECTOR Specifies which formats for vector data are available for users to download.

Default:

```
DOWNLOAD_FORMATS_VECTOR = [
    'JPEG', 'PDF', 'png', 'Zipped Shapefile', 'GML 2.0', 'GML 3.1.1', 'CSV',
    'Excel', 'GeoJSON', 'KML', 'View in Google Earth', 'Tiles',
]
```

DOWNLOAD_FORMATS_RASTER Specifies which formats for raster data are available for users to download.

Default:

```
DOWNLOAD_FORMATS_RASTER = [
    'JPEG', 'PDF', 'png' 'Tiles',
]
```

3.3 Organizational

The Organizational section contains information about the GeoNode project itself, how to contribute, learn about the community, helpful links, about the patch review process, the project road map and other administrative items.

3.3.1 Organizational

This is information on existing projects, contributing to GeoNode (code, documentation, translation, ...) and the community itself.

About GeoNode - What is GeoNode, the big picture.

Roadmap Process - How GeoNode can move ahead into the future.

Community Resources - Lots of links, think of it like your personal GeoNode bookmarks.

Community Bylaws - Some rules to keeps us stronger.

GeoNode Projects - Who else is doing cool stuff with GeoNode.

GNIPS GeoNode Improvement Proposals.

Contributing

Helping out the GeoNode project is great and by contributing we all benefit and here is how:

Contributing to GeoNode is the best way to help out and here we show you how.

GeoNode Patch Review Process is where code review happens, explained for developers.

Patch Review criteria for extending GeoNode.

How to contribute to GeoNode's Translation and update an existing language or add a new one.

How to contribute to GeoNode's Documentation is outlined how to get started writing documentation.

How to write Documentation a work in progress outlining well ..how to write documentation.

How to Translate the Documentation describes how to translate the documentation.

About GeoNode

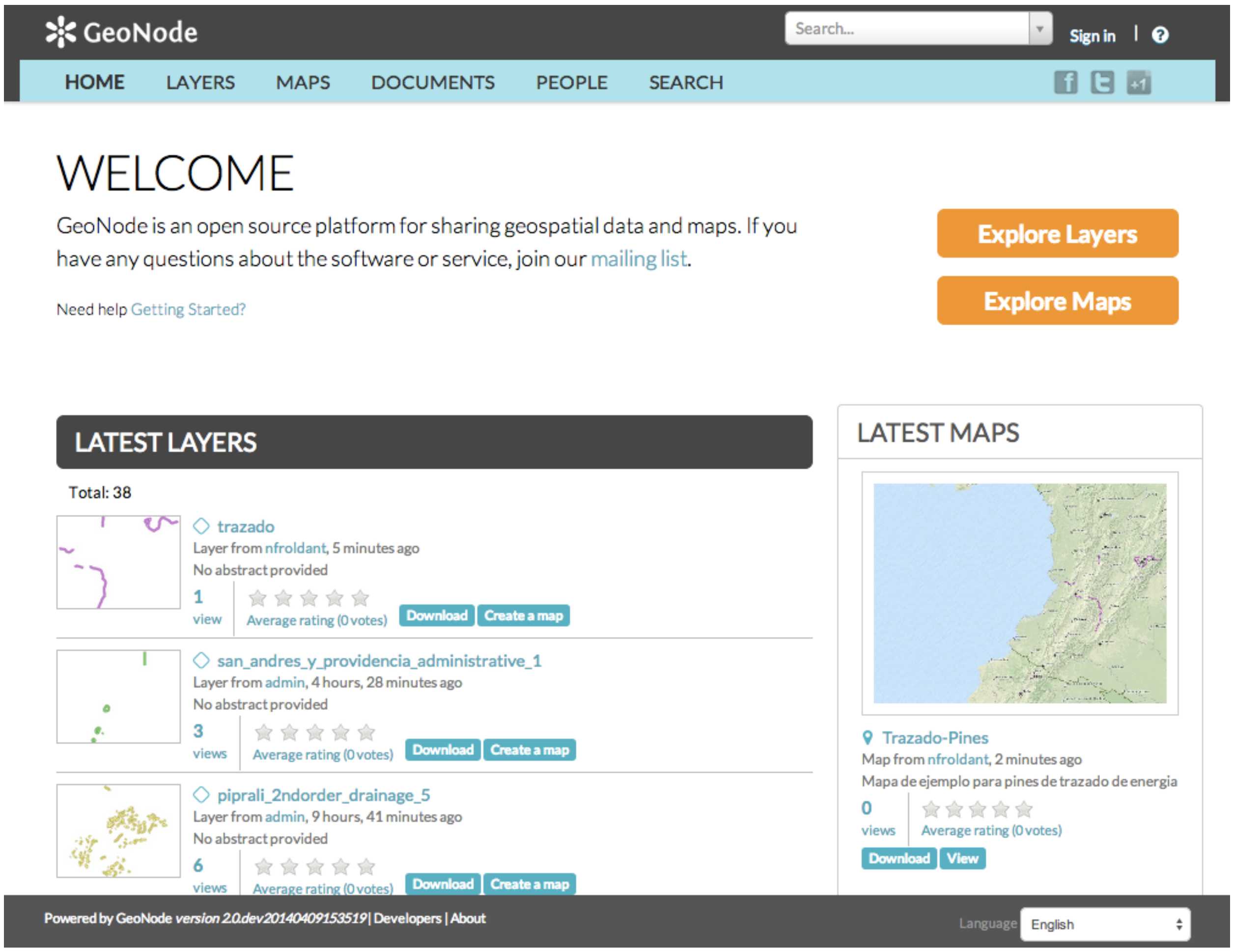
GeoNode is a geospatial content management system, a platform for the management and publication of geospatial data. It brings together mature and stable open-source software projects under a consistent and easy-to-use interface allowing non-specialized users to share data and create interactive maps.

Data management tools built into GeoNode allow for integrated creation of data, metadata, and map visualizations. Each dataset in the system can be shared publicly or restricted to allow access to only specific users. Social features like user profiles and commenting and rating systems allow for the development of communities around each platform to facilitate the use, management, and quality control of the data the GeoNode instance contains.

It is also designed to be a flexible platform that software developers can extend, modify or integrate against to meet requirements in their own applications.

Online demo A live demo of the latest stable build is available at demo.geonode.org.

Anyone may sign up for a user account, upload and style data, create and share maps, and change permissions. Since it is a demo site, we don't make any guarantee that your data and maps will always be there. But it should hopefully allow you to easily preview the capabilities of GeoNode.



WELCOME

GeoNode is an open source platform for sharing geospatial data and maps. If you have any questions about the software or service, join our mailing list.

Need help Getting Started?

LATEST LAYERS

Total: 38

trazado
Layer from nfroldant, 5 minutes ago
No abstract provided
1 view Average rating (0 votes)

san_andres_y_providencia_administrative_1
Layer from admin, 4 hours, 28 minutes ago
No abstract provided
3 views Average rating (0 votes)

piprali_2ndorder_drainage_5
Layer from admin, 9 hours, 41 minutes ago
No abstract provided
6 views Average rating (0 votes)

LATEST MAPS

Trazado-Pines
Map from nfroldant, 2 minutes ago
Mapa de ejemplo para pines de trazado de energia
0 views Average rating (0 votes)

Powered by GeoNode *version 2.0.dev20140409153519* | Developers | About

Geospatial data storage GeoNode allows the user to upload vector (currently only Shapefiles) and raster data in their original projections using a web form. Vector data is uploaded in ESRI Shapefile format and satellite imagery and other kinds of raster data are uploaded as GeoTIFFs.

Special importance is given to standard metadata formats like ISO 19139:2007. After the upload is finished, the user is presented with a form to fill in the metadata and it is made available using a CSW interface. Users may also upload a metadata XML document (in ISO, FGDC, or Dublin Core format) to fill in key GeoNode metadata elements automatically.

Similarly, GeoNode provides a web based styler, that lets the user change how the data looks and preview the changes in real time.

Data mixing, maps creation Once the data has been uploaded, GeoNode lets the user search for it geographically or via keywords and create maps.

All the layers are automatically reprojected to web mercator for maps display, making it possible to use different popular base layers, like Open Street Map, Google Satellite or Bing layers.

Once maps are saved, it is possible to embed them in any webpage or get a PDF version for printing.

GeoNode as a building block A handful of other Open Source projects extend GeoNode's functionality by tapping into the re-usability of Django applications. Visit our gallery to see how the community uses GeoNode: GeoNode Projects.

The development community is very supportive of new projects and contributes ideas and guidance for newcomers.

Convinced! Where do I sign? The next steps are to follow the quick_installation, read the tutorials and email geonode-users+subscribe@googlegroups.com to join the community. Thanks for your interest!

Note:

The registration process for the mailing list is the following:

1. Send an email to geonode-users+subscribe@googlegroups.com with any subject or content.
2. Get a confirmation notice from geonode-users+subconfirm@googlegroups.com and confirm by replying.
3. Once the above is complete, you are registered and you can start sending emails to the community.

Roadmap Process

The GeoNode Roadmap Process is designed to complement the more technical GeoNode [[Improvement Proposals]] and strives to make it easier for the various organizations invested in GeoNode to collaborate on features of common interest.

It is based on the roadmap items developed at the GeoNode Roadmapping Summit held in May 2011.

Overall, the process for adding items to the collective roadmap is as follows:

1. Organizational partner has an intent to add a feature to the roadmap.
2. Organizational partner communicates with the organizational partners list about the change to gauge interest and determine who else is committed to making it happen.
3. Organizational partner creates a feature specification on the wiki to further flesh out the idea.
4. Organizational partner finds a committer on the developer list to shepherd the roadmap item through the GeoNode [[Improvements Process | Improvement Proposals]].

Each roadmap item will go through four stages:

1. Descriptive Stage (under discussion/"Active")
2. Technical Stage
3. Development Stage
4. Released

After communicating on the organizational partners list the roadmap items enters the *Descriptive Stage* and must have a wiki page that lays out the description, user stories, and other interested parties. Optionally, the roadmap item will also include an idea of the difficulty and goals as well as any wireframes, technical diagrams, or prior art.

A roadmap item enters the *Technical Stage* once a committer has been found to shepherd the roadmap item through the [[GNIP | Improvement Proposals]] process, then the wiki page must contain a clear sense of the technical assumptions, requirements or dependencies, and suggested implementation. Some roadmap items may need to be divided into multiple independent GNIP proposals.

Once it passes through the [[GNIP | Improvement Proposals]] process, a roadmap item enters the *Development Stage* on its way to *Release*.

[[RoadMap-Items]]

Community Resources

Here you will find many links to resources on Github, external sites using GeoNode. Think of like your GeoNode Bookmarks.

Main Links These 3 top links are the GeoNode landing pages and will take you to all other information on GeoNode.

- Main HomePage and Blog http://geonode.org
- Documentation http://docs.geonode.org
- GitHub Code http://github.com/GeoNode

Community Contact Contact members and ask questions

- **Mailing Lists**
  - Users Forum https://groups.google.com/forum/#!forum/geonode-users
  - Developers Forum https://groups.google.com/a/opengeo.org/group/geonode-dev/topics
- Announcements https://groups.google.com/a/opengeo.org/forum/#!forum/geonode-announcements
- IRC irc://irc.freenode.net/geonode (logs?)
- **Social Media**
  - Blog http://geonode.org/blog/
  - Twitter https://twitter.com/geonode
  - Google+ https://plus.google.com/u/0/b/100587124776656797019/

Github Project Links

- Main Github page https://github.com/GeoNode/geonode
- Master Branch https://github.com/GeoNode/geonode/tree/master
- Issue Tracker https://github.com/GeoNode/geonode/issues
- Pull Requests https://github.com/GeoNode/geonode/pulls
- Wiki https://github.com/GeoNode/geonode/wiki
- GNIPs https://github.com/GeoNode/geonode/wiki/GeoNode-Improvement-Proposals
- geonode.org Homepage Code http://github.com/geonode/geonode.github.com (published to http://geonode.org)
- Releases http://dev.geonode.org/release/
- Localisation using Transifex https://www.transifex.com/projects/p/geonode

Demo Sites

- Demo GeoNode Site http://demo.geonode.org/
- Latest Beta Demo Site http://beta.dev.geonode.org/
- UI/UX Review Site https://sites.google.com/a/opengeo.org/geonode-ui/

Testing and Packaging

- **Testing**
  - Jenkins Continuous Integration Server http://geonode-testing.dev.opengeo.org:8090
  - Integration Test Suite https://github.com/GeoNode/geonode-integration
  - Travis-CI https://travis-ci.org/#!/GeoNode/geonode
  - Ohloh https://www.ohloh.net/p/geonode
- **Packages**
  - Debian Package https://github.com/GeoNode/geonode-deb
  - RPM Redhat https://github.com/GeoNode/geonode-rpm
  - Chef https://github.com/GeoNode/chef-geonode
  - PyPi https://pypi.python.org/pypi/GeoNode
  - **Launchpad**
    * https://launchpad.net/~geonode
    * https://launchpad.net/~geonode/+archive/release
    * https://launchpad.net/~geonode/+archive/testing
    * https://launchpad.net/~geonode/+archive/snapshots

Important Forks Full List here https://github.com/GeoNode/geonode/network/members

Hint: Look at the branches in these Forks

- https://github.com/opengeo/geonode
- https://github.com/gfdrr/geonode
- https://github.com/dwinslow/geonode
- https://github.com/ingenieroariel/geonode
- https://github.com/jj0hns0n/geonode
- https://github.com/ahocevar/geonode
- https://github.com/groldan/geonode
- https://github.com/AIFDR/geonode
- https://github.com/cga-harvard/cga-worldmap
- https://github.com/sopac/geonode
- https://github.com/dialog-it/geonode
- https://github.com/eldarion/geonode

Downstream Github Projects

- https://github.com/aifdr/tsudat2
- https://github.com/aifdr/riab
- https://github.com/gem/
- https://github.com/opengeo/mapstory
- https://github.com/dialog-it/anzsm
- https://github.com/CIGNo-project/CIGNo
- https://github.com/ROGUE-JCTD/rogue_geonode
- Many More ...

Additional Modules

- https://github.com/simod/geonode-documents
- https://github.com/GFDRR/geonode-registry
- https://github.com/eldarion/geonode/branches

Public Sites

- Risiko (Risk in a Box) demo site http://demo.riskinabox.org/
- CIGNo network: CNR - ISMAR Node http://cigno.ve.ismar.cnr.it/
- ... Many Many More

Community Bylaws

Committers The GeoNode community is divided into two groups - users and committers. There are no requirements or responsibilities to be a GeoNode user. To be a committer, you must be voted in by the existing committers (2 +1's and no -1's; a committer must initiate the vote.) Non-committers are encouraged to engage in discussions on the mailing lists, code review, and issue reports to qualify them to be voted in as committers. Committers (or PRIMARY AUTHORS) can be found in the [AUTHORS file](https://github.com/GeoNode/geonode/blob/dev/AUTHORS)

Committers must:

- Make useful contributions to the project in the form of commits at least once in a 6-month period, else they fall back to "committer emeritus" status. A committer emeritus has no special involvement in the project, but may request committer privileges from the current body of committers.
- Review code contributions, which may come from other committers or from users. Users must submit code externally to the main GeoNode repository (ie as a patch or a github pull request); committers can do this as well if they see review as particularly important (for example, a patch might affect a particularly crucial component of GeoNode, or a committer might be working in a part of the code that he is relatively unfamiliar with.) A review should result in either (a) instructions on how to bring the code to a more acceptable condition or (b) merging the changes in and notifying the submitter that this has been done.
- Committers also have the option to "self-review" and commit changes directly. It is at the discretion of individual committers when this is appropriate, but it should be rare - we encourage committers to only use this option when they deem a change extremely safe.

GeoNode Improvement Proposals (GNIPS) GNIPS If a committer thinks a proposed change to the software is particularly destabilizing or far-reaching, that committer can upgrade the ticket for that change to a GeoNode Improvement Proposal (GNIP). GNIP tickets are an opportunity for committers and users alike to provide feedback about the design of a proposed feature or architectural change. The proposal should be iteratively edited in response to community feedback.

To upgrade an issue to a GNIP, an active committer should give the ticket the 'GNIP' label in the issue tracker, and announce the issue on the developer mailing list.

If a ticket has a GNIP label, its patch can't be committed unless it also has the 'Approved' label. To be approved, it must pass community vote (see below).

When the GNIP is announced, other committers should review and provide feedback in the issue comments. Feedback should take the form of:

- +1 (with optional comment)
- -1, with mandatory rationale and suggestion for a better approach. The suggestion may be omitted if the objection doesn't have a straightforward solution - we don't want to withhold feedback just because problems with a proposal are hard to solve.

After receiving feedback, the proposal's author should discuss the feedback on the list if necessary and adjust the proposal in response.

A proposal can be Approved when there are 3 +1 responses (including the author's implicit approval) and no -1 responses from committers; and no feedback is offered in 3 days. If a proposal fails to receive multiple +1 responses within 5 days of the request for feedback it is rejected and the issue should be **closed** (but the author is free to draft similar proposals in the future.) Any committer may reverse or withdraw votes on a proposal until the proposal is closed.

If a user would like to submit a GNIP, they are welcome to write it as a ticket but should find an active committer willing to promote it to GNIP status.

Project Steering Committee In the event that a revision to these bylaws becomes necessary, authority for that decision lies with the currently presiding Project Steering Committee (PSC). The PSC at any time is made up of the top 7 committers over the past 365 days, by number of commits.

Contributing to GeoNode

If you are interested in helping us to make GeoNode, there are many ways to do so.

Participate in the Discussion GeoNode has a mailing list (https://groups.google.com/d/forum/geonode-users) where users can ask and answer questions about the software. There is also an IRC chat room #geonode on Freenode (http://freenode.net/) where users and developers can discuss GeoNode in real time. Sometimes users also post interesting tips for managing sites running GeoNode. If you want to help out with GeoNode, one easy way is to sign up for the mailing list and help answer questions.

Report Problems on the Issue Tracking System While we do our best to keep GeoNode fast and bug-free, problems can and do arise. Informative bug reports are a key part of the bug fixing process, so if you do run into a problem with GeoNode, please don't hesitate to report it on our bug tracker, available online at http://github.com/GeoNode/geonode/issues. Useful information for bug reports includes:

- What were you doing when the bug occurred? Does the problem occur every time you do that, or only occasionally?
- What were you expecting to happen? What happened instead?

- What version of the software are you using? Please also note any changes from the default configuration.
- If there is a data file involved in the bug (such as a Shapefile that doesn't render properly), please consider including it in the bug report. We do understand that not all data files are freely distributable.

To help us better address the issue you can tag the ticket with one or more lables that you can find on the side column.

Write Documentation GeoNode's documentation can always use improvement - there are always more questions to be answered. For managing contributions to the manual, GeoNode uses a process similar to that used for managing the code itself. The documentation is generated from source files in the *docs/* directory within the GeoNode source repository. See http://sphinx.pocoo.org/ for more information on the documentation system we use.

If you want to learn more about contributing to the documentation, please go ahead to the *How to contribute to GeoNode's Documentation*. We also have some guidelines to help with writing once you are set up *How to write Documentation*.

Provide Translations If GeoNode doesn't provide a user interface in your native language, consider contributing a new translation. To get started here are the instructions *How to contribute to GeoNode's Translation*.

Write Code Of course since GeoNode is an open source project we do encourage contributions of source code as well. If you are interested in making small changes, you can find an open ticket on http://github.com/GeoNode/geonode/issues, hack away, and get started on the Patch Review Process.

GeoNode Patch Review Process

This document outlines the code review process for GeoNode code. Each commit proposed for inclusion in GeoNode should be reviewed by at least one developer other than the author. For pragmatic reasons, some developers, referred to in this document as *core committers*, may commit directly to the GeoNode repository without waiting for review. Such changes are still subject to review, and may be reverted if they fail any of the [[Review Criteria]].

A related process is [[Improvement Proposals]]. While patch review protects code quality in the GeoNode project at a small granularity, the Improvement Proposal process is intended to promote coordinated design and feedback for larger modifications such as new features or architectural changes.

Goals By requiring a review of code coming into GeoNode, we aim to maintain code quality within the GeoNode project while still allowing contributions from the GeoNode community.

Review Criteria Patch reviews should adhere to the standards set in the [[Review Criteria]], a [[Project Steering Committee]] approved set of criteria for the inclusion of new code.

Process For contributors who do not have commit access to the GeoNode repository, the review process is as follows:

0. Find or create a ticket describing the feature or bug to be resolved.
1. Create changes to GeoNode code addressing the ticket.
2. Publish those changes and request a review from the GeoNode committers. The recommended format is a GitHub [pull request](http://help.github.com/pull-requests/). If you are unable or unwilling to provide a change as a branch on GitHub, please consult the developer's list for advice.

3. At least one GeoNode committer should review the submitted changes. If he finds the patch acceptable, the changes will be pulled into GeoNode. If problems are found, then he should list them clearly in order to allow the original author to update the submission (at which point we return to point 2 in this process.) In the case of a feature idea which is simply not suitable for inclusion as core GeoNode functionality, the patch may be rejected outright.

![Code Review Flow](http://geonode.org/wp-content/uploads/2011/05/code-review-flow.png)

The general workflow for code patches coming into GeoNode.

Note: If after a few days your patch has not been reviewed by any GeoNode committer, please feel free to bring it up either in the mailing list or the IRC channel. The GeoNode community (and it's committers) try to respond quickly and give adequate feedback to maintain the interest of new potential members. However, sometimes other responsibilities prevent us from responding quickly.

Core Committers It is assumed that core committers are familiar with the quality guidelines and capable of producing acceptable patches without the need for waiting on review. Therefore, core committers may make changes without requesting review first (although they are welcome to request review for code if they feel it is appropriate.) For commits made without prior review, committers should review the changes and revert them if they are in violation of the project quality guidelines.

Becoming a Core Committer In order for a developer to become a core committer, he must demonstrate familiarity with the quality guidelines for the GeoNode project by producing at least two patches which successfully pass review and are merged without requiring modification. A candidate for core committer-ship must be nominated by a member of the [[Project Steering Committee]], and approved via Apache consensus voting among PSC members.

Patch Review criteria

When a patch is rejected in the [[Patch Review Process]], it should be given a valid review.

This review should point out issues with the patch where it fails to meet one or more of the following criteria.

1. Major new features must be approved by the Improvement Process

GeoNode needs to be coherent software despite the diverse interests driving its development. Therefor, major new features need to first be approved according to the [[Improvement Process]].

If a patch fails by this criterion, then its developer is welcome to go through the improvement process to get approval. Otherwise, they can refactor their patch into a GeoNode extension.

1. Patches need sufficient documentation

We strive to keep GeoNode well-documented. If a patch contributes significant functionality to GeoNode that requires documentation to be understood, the patch review is an opportunity to hold the developer accountable for providing the adequate documentation.

1. New functionality needs to be internationalized

We strive to build GeoNode in a way that can be used in many different localities, by all languages. While there is no localization requirement for GeoNode besides providing default English text, new user-facing features need to be sufficiently internationalized so that others can write translations.

1. Design consistency

We strive to keep the default user interface for GeoNode appealing for new users and developer's approaching the project. If a patch significantly diminishes the user experience of the software, then a patch may be rejected with a review of how to improve it.

Note: Good design can sometimes be in the eye of the beholder. Developer's are encouraged to consult the community and/or a designer about the user interface design of their patches, and to be humble in their design criticisms of others.

1. Code should be covered by automated tests

To make development easier for others and guarantee software quality, we strive to have good automated test coverage in GeoNode. Patches may fail a review for having insufficient unit and/or integration tests.

Reviews saying that a patch has insufficient tests should offer actionable advice on how to improve those tests. This advice could be to improve code coverage. It may also be a list of possible cases that currently lack tests.

1. Patches should not have known bugs

   A patch may be rejected for having a known bug, (e.g.) one discovered by reading the code or testing it at the time of review.

2. Patches should meet GeoNode's code style guidelines

   New patches should meet GeoNode's code style guidelines. We follow different conventions per language:

   - In Java we use the GeoTools/GeoServer convention, essentially the [conventions recommended by Oracle](http://www.oracle.com/ technetwork/ java/codeconvtoc-136057.html) modified to make the recommended line length 100 columns instead of 80 to accommodate the long identifiers commonly used in GeoTools code. The GeoServer project provides an [Eclipse configuration](http://docs.geoserver.org/stable/en/developer/eclipse-guide/index.html#eclipse-preferences) which helps to stick to this convention.
   - In Python we use the conventions enumerated in [PEP8](http://www.python.org/dev/peps/pep-0008/). Many editors have plugins available to assist with conformance to this convention.
   - In JavaScript we use the OpenLayers conventions, described on the [OpenLayers wiki](http://trac.osgeo.org/openlayers/wiki/CodingStandards).

How to contribute to GeoNode's Translation

GeoNode uses Transifex to translate the webpage and code from english to any other language. To be able to contribute to the translation, you need to get an account on transifex.

Edit translations directly on Transifex webpage Here you will see how easy it is to update the translations directly on the transifex website.

1. Create account

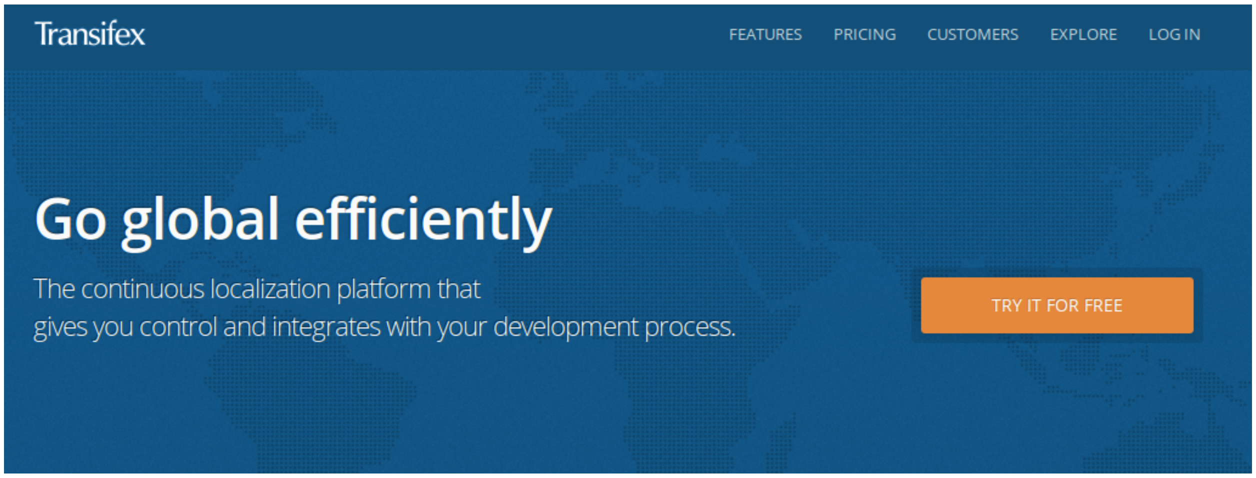
   Go to https://www.transifex.com and, click *Try it for free* and enter the needed information to create your free account.

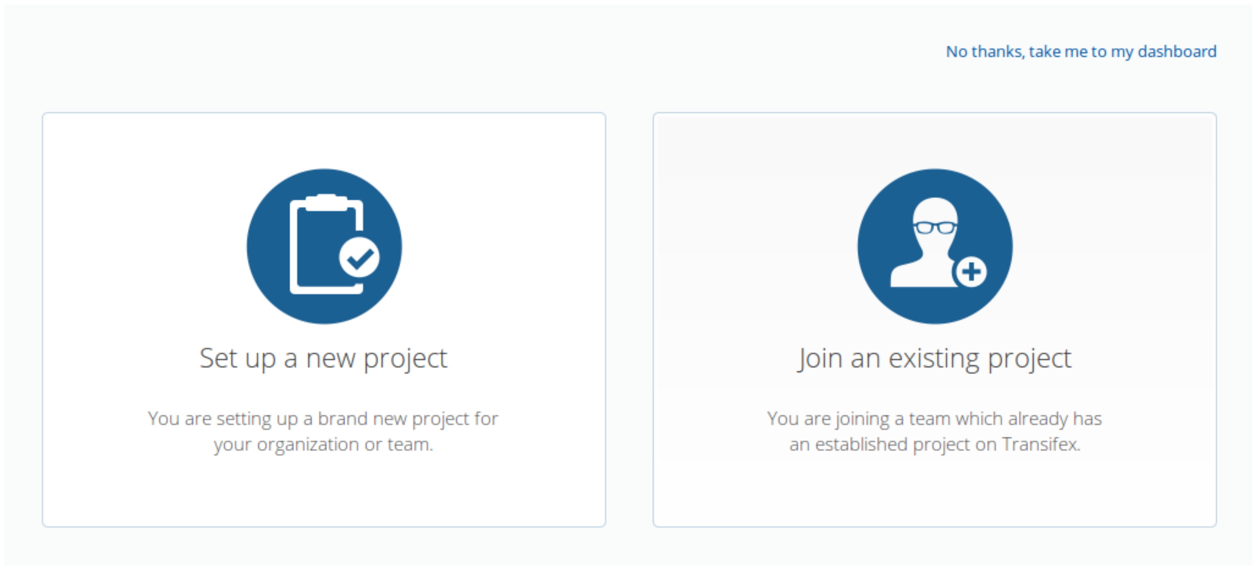
2. Join our project

   After activating the link you've got you will be asked whether you want to start a new project or to join an existing project.

   Click *join an existing project* and type *geonode* into the search bar. You will be directed to the geonode project site on transifex. To join the team, click on the language you want to add a translation in. Then a button *Join the team* will appaer. Hit it, and you will submit for permission to join.

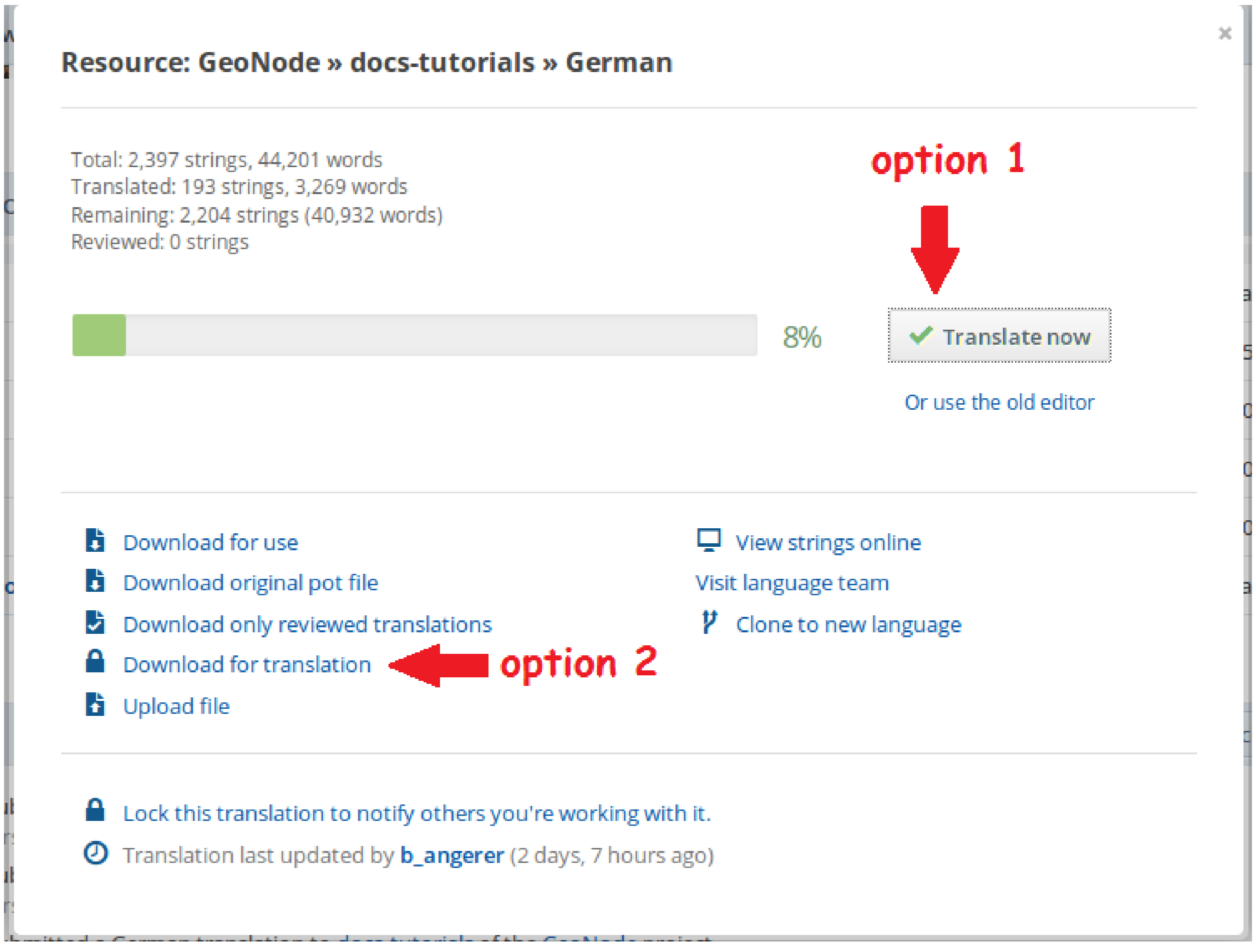
3. Wait for permission to update translations from translation leader (email)


Transifex
FEATURES
PRICING
CUSTOMERS
EXPLORE
LOG IN
Go global efficiently
The continuous localization platform that
gives you control and integrates with your development process.
TRY IT FOR FREE


No thanks, take me to my dashboard
Set up a new project
You are setting up a brand new project for your organization or team.
Join an existing project
You are joining a team which already has an established project on Transifex.

4. Start to Add a translation

   Click on the language where you want to add a translation. You'll see some bars, e.g. *javascript* and *master*. Choose the one you want to add a translation and you will see a dialog window like this



Resource: GeoNode » docs-tutorials » German

Total: 2,397 strings, 44,201 words
Translated: 193 strings, 3,269 words
Remaining: 2,204 strings (40,932 words)
Reviewed: 0 strings

Lock this translation to notify others you're working with it.
Translation last updated by b_angerer (2 days, 7 hours ago)

   Now you have two options to start translating. Either you use the transifex interface (*option 1*) or you download the file to translate locally (*option 2*).

   Option 1

   To start translating using the Transifex interface, hit *Translate now*. You'll see an interface like this

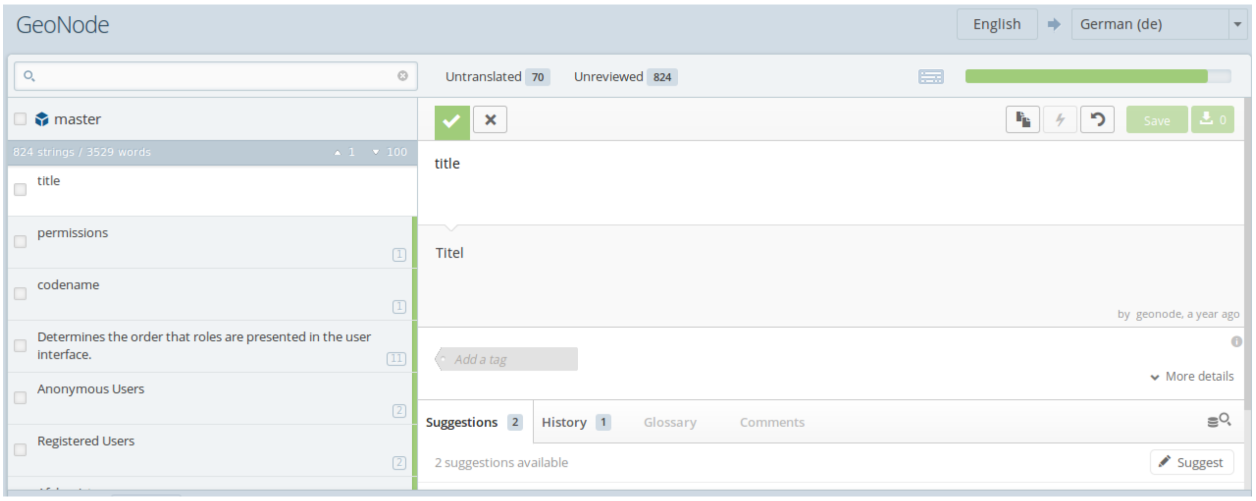
   Click *untranslated* and add your translation like shown below

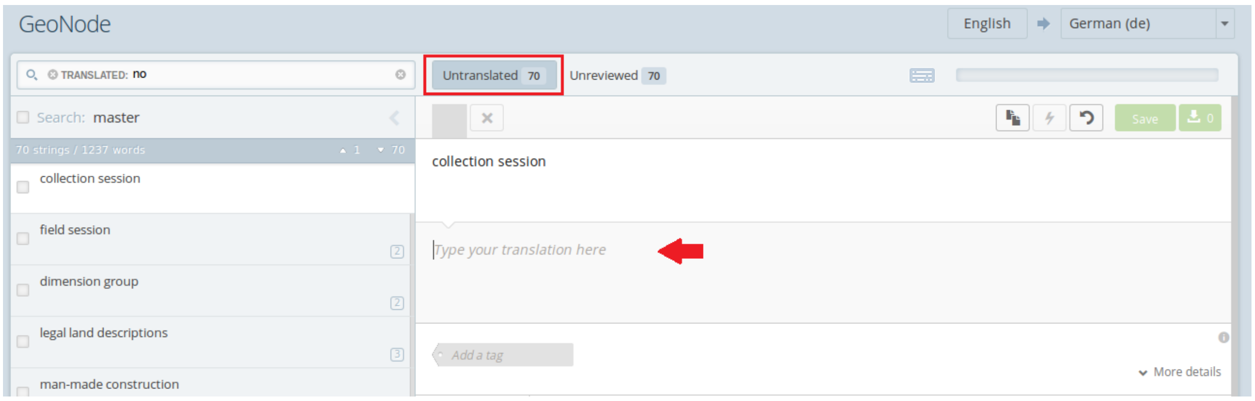
   When you stop translating, do not forget to hit the green *save* button at the top right!

   Option 2

   To translate in your favourite editor, download the file hitting *Download file for translation*. This will download a .po file, which includes the strings to be translated (*msgid*). Put your translation into the *msgstr* and when you're done, save the file and upload it to transifex (using the same dialog window as for the download).

Translate on local machine from github Using this options it is assumed that you have a local Geonode GitHub repository forked.


GeoNode
English
German (de)
Untranslated 70
Unreviewed 824
master
824 strings / 3529 words
title
permissions
codename
Determines the order that roles are presented in the user interface.
Anonymous Users
Registered Users
Save
title
Titel
by geonode, a year ago
Add a tag
More details
Suggestions 2
History 1
Glossary
Comments
2 suggestions available
Suggest


GeoNode
English
German (de)
TRANSLATED: no
Untranslated 70
Unreviewed 70
Search: master
70 strings / 1237 words
collection session
field session
dimension group
legal land descriptions
man-made construction
Save
collection session
Type your translation here
Add a tag
More details

Note: It is recommended to first create a new branch e.g. *translation* in your repository for your translations.

1. As a firt step, generate all of the needed .pot files (any time the master documentation changes):

```
$ cd docs
$ sphinx-build -b gettext . i18n/pot
```

2. Run the pre_translate.sh script to generate/merge (update) all of the .po files for each language:

```
$ sh i18n/scripts/pre_translate.sh
```

3. Do a pull from transifex to get latest translations:

```
$ tx pull -a
```

4. Now edit the .po files you need, make the translations and then run the post_translate.sh script:

```
$ vi i18n/it/LC_MESSAGES/index.po
$ sh i18n/scripts/post_translate.sh
```

5. Now you have to push the changed .po files and the appropriate .pot file (can be found in geonode/docs/i18n/pot) to your remote repository using:

```
$ git commit
$ git push
```

6. Now make a pull request and GeoNode will push your changes to transifex.

Only for transifex maintainers

Note: This section is only for the maintainers of a transifex group!

1. As a firt step, generate all of the needed .pot files (any time from master documentation changes):

```
$ cd docs
$ sphinx-build -b gettext . i18n/pot
```

2. Run the pre_translate.sh script to generate/merge (update) all of the .po files for each language:

```
$ sh i18n/scripts/pre_translate.sh
```

3. Do a pull from transifex to get latest translations:

```
$ tx pull -a
```

4. Now edit the .po files you need (here using Italian /it), make the translations and then run the post_translate.sh script:

```
$ vi i18n/it/LC_MESSAGES/index.po
$ sh i18n/scripts/post_translate.sh
```

5. Do a push to transifex to update translations on the server:

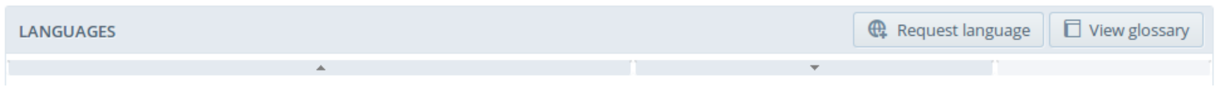
```
$ tx push -s -t
```

6. Finally build your html or pdf, and you should get the localized versions as well:

```
$ make html LANG=it
```

How to add a new language

To add a new language, click on *Request language* on the right top of the transifex webpage.



How to contribute to GeoNode's Documentation

If you feel like adding or changing something in the GeoNode documentation you are very welcome to do so. The documentation always needs improvement as the development of the software is going quite fast.

In order to contribute to the GeoNode documentation you should:

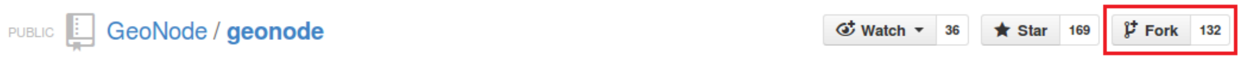
- Create an account on GitHub
- Fork the GeoNode repository
- Edit the files in the */docs* directory
- Submit pull requests

All these things can generally be done on the web, you won't need to download anything. But if you want to add images to the documentation you will have to do some more initial steps, because this can't be done on the web. To learn about how images can be added to your documentation and which additional steps have to be done, read the section *Add images*.

The general steps are explained in more detail below.

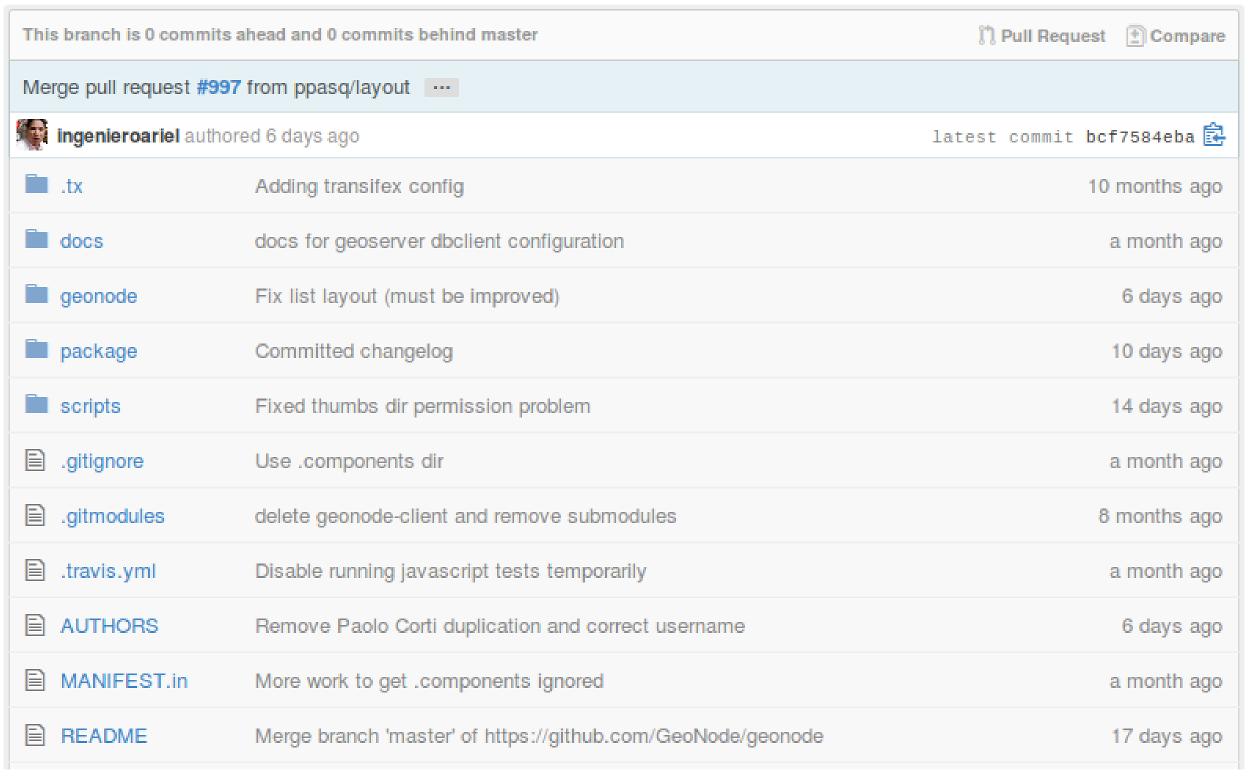
Create an account on GitHub The first step is to create an account on GitHub. Just go to Github, find a username that suits you, enter your email and a password and hit *Sign up for GitHub*. Now you've signed in, you can type *geonode* into the searching area at the top of the website. On top of the search results you will find the repository *GeoNode/geonode*. By clicking on it you will be entering the repository and will be able to see all the folders and files that are needed for GeoNode. The files needed for the documentation can be found in */docs*.

Fork a repository In order to make changes to these files, you first have to fork the repository. On the top of the website you can see the following buttons:



Click on the button *Fork* at the top right and the *geonode* repository will be forked. You should now be able to see your repository *your_name/geonode*. If you want to read more about how to fork a repository go to https://help.github.com/articles/fork-a-repo.

Edit files To make some changes to already exiting files or to create new files, go to your GitHub account. Under *repositories* you will find the geonode repository that you have forked. Click on it and you will again see all the folders and files GeoNode needs.



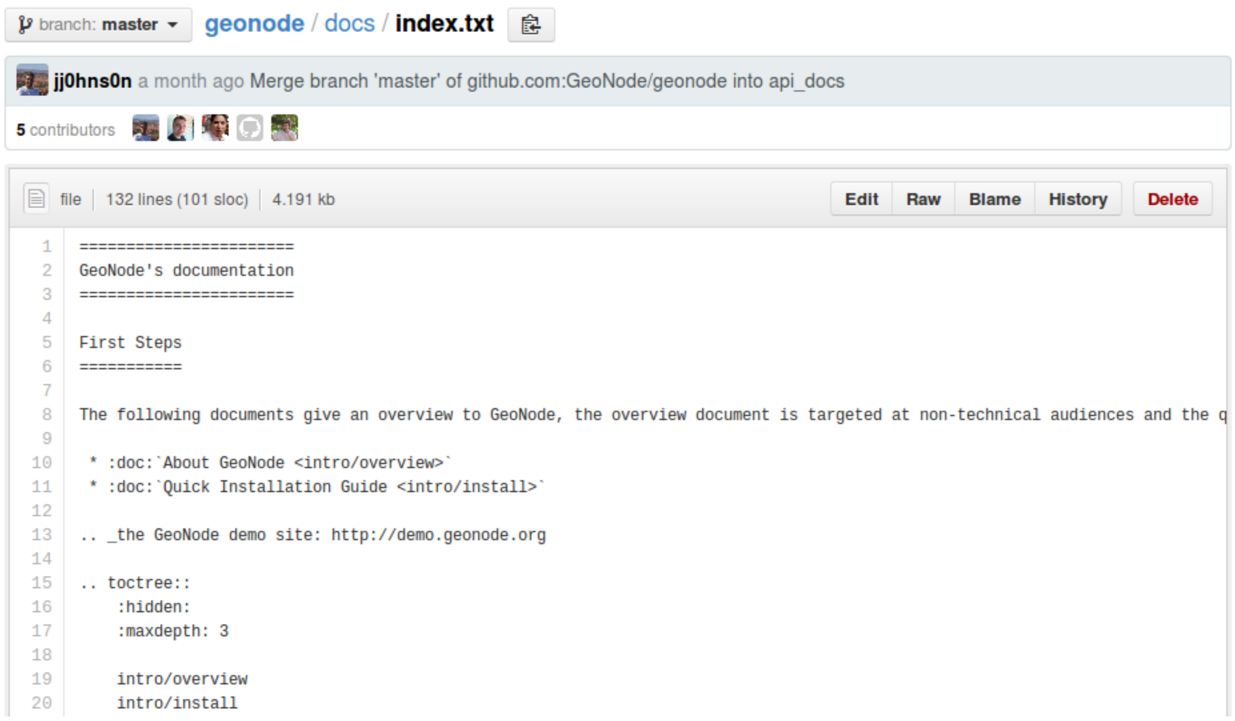
This branch is 0 commits ahead and 0 commits behind master | Pull Request | Compare

Merge pull request #997 from ppasq/layout

ingenieroariel authored 6 days ago | latest commit bcf7584eba

.tx	Adding transifex config	10 months ago
docs	docs for geoserver dbclient configuration	a month ago
geonode	Fix list layout (must be improved)	6 days ago
package	Committed changelog	10 days ago
scripts	Fixed thumbs dir permission problem	14 days ago
.gitignore	Use .components dir	a month ago
.gitmodules	delete geonode-client and remove submodules	8 months ago
.travis.yml	Disable running javascript tests temporarily	a month ago
AUTHORS	Remove Paolo Corti duplication and correct username	6 days ago
MANIFEST.in	More work to get .components ignored	a month ago
README	Merge branch 'master' of https://github.com/GeoNode/geonode	17 days ago

Click on the folder *docs* and search for the file that you want to edit. If you found it, click on it and you will be able to see the content of this file.



branch: master | geonode / docs / index.txt

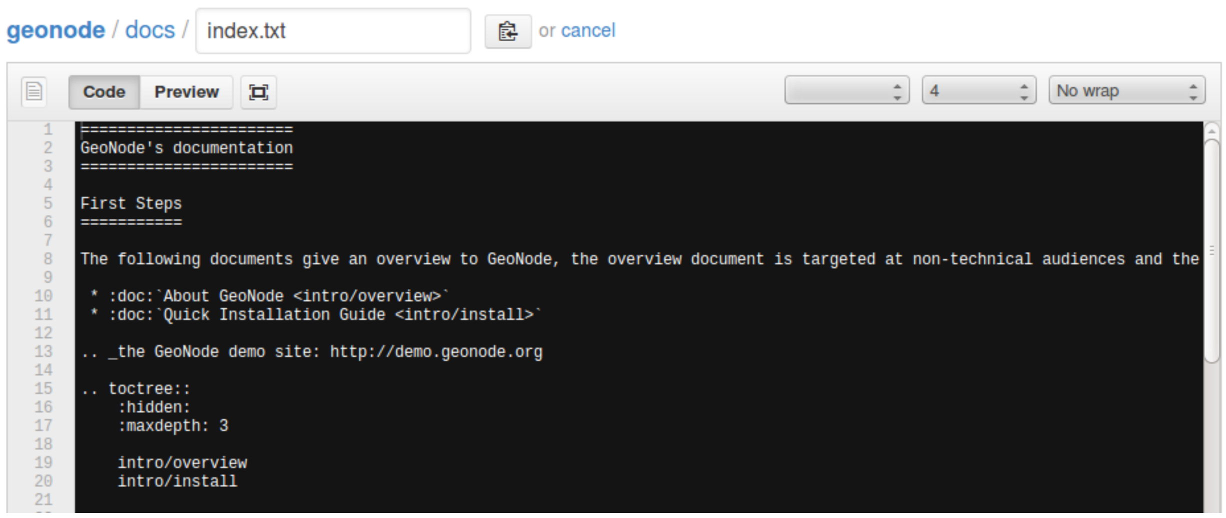
jj0hns0n a month ago Merge branch 'master' of github.com:GeoNode/geonode into api_docs

5 contributors

file | 132 lines (101 sloc) | 4.191 kb

```
=======================
GeoNode's documentation
=======================

First Steps
===========

The following documents give an overview to GeoNode, the overview document is targeted at non-technical audiences and the q

 * :doc:`About GeoNode <intro/overview>`
 * :doc:`Quick Installation Guide <intro/install>`

.. _the GeoNode demo site: http://demo.geonode.org

.. toctree::
    :hidden:
    :maxdepth: 3

    intro/overview
    intro/install
```

To make changes to this file, hit the button *edit* on the right top. You can now make your changes or add something to the existing content.

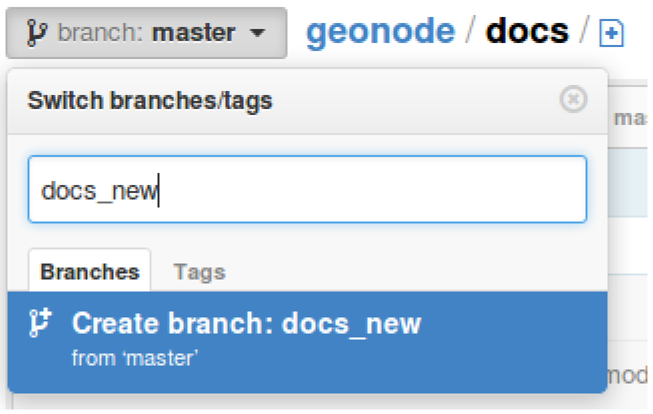


```
=======================
GeoNode's documentation
=======================

First Steps
===========

The following documents give an overview to GeoNode, the overview document is targeted at non-technical audiences and the

 * :doc:`About GeoNode <intro/overview>`
 * :doc:`Quick Installation Guide <intro/install>`

.. _the GeoNode demo site: http://demo.geonode.org

.. toctree::
   :hidden:
   :maxdepth: 3

   intro/overview
   intro/install
```

As you can see now, the documentation is written in *reStructeredText*, a lightweight markup language. To learn how to use it you should read the documentation that can be found here http://docutils.sourceforge.net/docs/user/rst/quickref.html. By hitting the *preview* button you will be able to see how your text it is going to look like on the web. To save your changes, click on *Commit Changes* at the bottom of the site. Now you've saved the changes in your repository, but the original geonode repository still doesn't know anything about that! In order to tell them that you have made some changes you have to send a *pull request* (as described below).

Create a new branch

If you are planning bigger changes on the structure of the documentation it is recommended to create a new branch and make your edits here. A new branch can be created by clicking on the button *branch: master* as shown here.

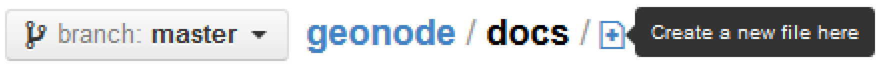


Just type the name of your new branch, hit enter and your branch will be created. To learn more about branches it is recommended to take a look here http://git-scm.com/book/en/Git-Branching-What-a-Branch-Is.

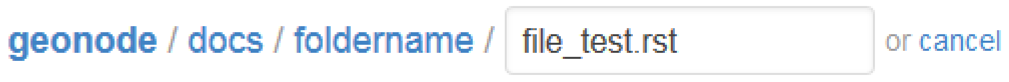
Note: Before you start editing make sure that you are in the right branch!

Create a new folder/file

If you want to add a completely new issue to the documentation, you have to create a new file (and maybe even folder). As you will see there is no possibility to create an empty folder. You always have to create a new file as well! This can be done here



If you click on *create new file here* you can first subdirect to another folder by typing the *foldername* followed by /. If this folder doesn't exist until now, one will be created. To create a new file in this folder just type *filename.rst* into the box and hit enter. A short example on how to manage this is given here http://i.stack.imgur.com/n3Wg3.gif.



Now a black box will appear where you can add your comments. To save the file, hit the green *Commit New File* button at the bottom.

Add images This section is about adding images to your documentation. Providing that you've read and done the steps described above you can now follow those further steps.

Install and set up Git

To add images to your documentation you have to get your repository onto your local machine. So far you only had your repository on the web. To be able to work on your local machine as well, you have to install *git*. To do so, type:

```
sudo apt-get install git
```

(Usually git has already been installed during geonode installation)

Before you go further you should do some setup steps (can be found here: https://help.github.com/articles/set-up-git).

Clone repository

We assume you have already forked the geonode repository. If not, please do so following _link and return back if ready.

Until now your repository only exists on the web! To get your forked repository on to your machine, you have to *clone* it. To do so, open a terminal, go to the folder where you want the projet to be and type:

```
git clone https://github.com/your_username/geonode.git my_geonode
```

Now change the active directory to the newly cloned geonode directory using:

```
cd my_geonode
```

To keep track of the original repository (the geonode repository where you forked from), you need to add a remote named *upstream*. Therefore type:

```
git remote add upstream https://github.com/GeoNode/geonode.git
```

By typing:

```
git fetch upstream
```

changes not present in your local repository will be pulled in without modifying your files.

Add folder with images

Warning: If you've already made some changes and commits to your repository on the web (during cloning the repository and now), you have to update your repository on the local machine!

Therefore you have to run the following commands:

```
git fetch origin

git merge
```

Or instead you could use:

```
git pull
```

Your repository should now be up to date! For more information on those commands go to http://git-scm.com/docs.

Note: If you've created a new branch, and you want to add the new folder to this branch, make sure you are working on this branch!

Typing:

```
git status
```

will show you the current branch. To change this you have to run this command (*your_branch* is the name of the branch you want to change in):

```
git checkout your_branch
```

Now you can easily add a new folder containing images to your repository. Go to the repository on your local machine and decide where you want your new folder containing the images to be (e.g in *docs_example*). There create a new folder (e.g. *images*) and add the images manually. Once you've done this, open a terminal and direct to to the folder *docs_example*. To add the folder *images* including all content to the repository, type:

```
git add images
```

If this command doesn't work, check your path, maybe it is incorrect!

Remark: In order to commit and push the folder, it must not be emtpy!

The next step is to commit the folder/files:

```
git commit -m 'Message'
```

Instead of 'Message' write something like 'add images'. To push the files to the repository type:

```
git push
```

Now you are able to see the folder on the web as well!

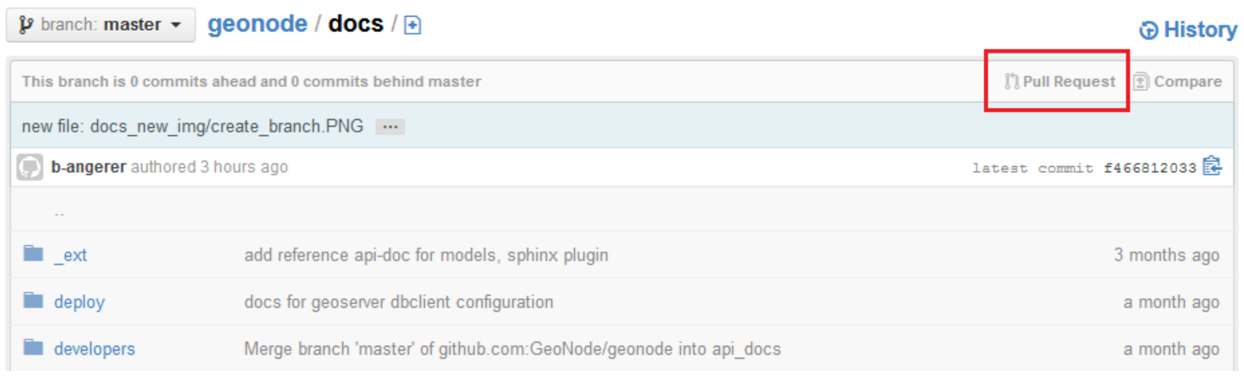
Include images

To include the images in to your documentation, you have to add the following lines to your file:

```
.. image:: images/test_img.png
```

Note: Be aware that everytime you commit something on the web, you have to make *git pull* on your machine, to keep it up to date!

Pull Request If you are done with your changes, you can send a pull request. This means, that you let the core developers know that you have done some changes and you would like them to review. They can hit accept and your changes will go in to the main line. The *pull request* can be found here.



This branch is 0 commits ahead and 0 commits behind master

new file: docs_new_img/create_branch.PNG ...

b-angerer authored 3 hours ago   latest commit f466812033

..

_ext	add reference api-doc for models, sphinx plugin	3 months ago
deploy	docs for geoserver dbclient configuration	a month ago
developers	Merge branch 'master' of github.com:GeoNode/geonode into api_docs	a month ago

How to write Documentation

GeoNode uses Restructured Text with Sphinx . Writing style should follow the same policies as geoserver does in their Documentation Style Guide

Sphinx Syntax

This page contains syntax rules, tips, and tricks for using Sphinx and reStructuredText. For more information, please see:

- RST Quick Reference
- Comprehensive guide to reStructuredText
- Sphinx reStructuredText Primer

Basic markup A reStructuredText document is written in plain text. Without the need for complex formatting, one can be composed simply, just like one would any plain text document. For basic formatting, see this table:

Format	Syntax	Output
Italics	`*italics*` (single asterisk)	*italics*
Bold	`**bold**` (double asterisk)	**bold**
Monospace	`` `` monospace `` `` (double back quote)	`monospace`

Sections, subtitles and titles Use sections to break up long pages and to help Sphinx generate tables of contents.

The top of the page (i.e. the title) should have an equals sign (=) above and below:

```
==============
Camel Spotting
==============
```

Level 2 section headers should have an equals sign (=) below the section name with same length as name:

```
I am a level 2 header
=====================
```

Level 3 sections should have a single minus symbol (-):

```
I am a level 3 header
---------------------
```

Level 4 sections should have a single dot,periol symbol (.):

```
I am a level 4 header
.....................
```

Level 5 sections should have a single dot,periol symbol (.):

```
I am a level 5 header
+++++++++++++++++++++
```

Page labels **Ensure every page has a label.** For example if the page is named `foo_bar.rst` then the page should have the label at the top of the file (line1)

```
..  _foo_bar:
```

Other pages can then link to that page by using the following code:

```
:ref:`foo_bar`
```

Linking Links to other pages should never be titled as "here". Sphinx makes this easy by automatically inserting the title of the linked document.

Using the following code:

```
:ref:`linking`
```

And here is the link in action *Linking* to use the link place this code some where in your open file:

```
.. _linking:
```

To insert a link to an external website:

```
`Text of the link <http://docs.geoserver.org/latest/en/docguide/style.html>`_
```

The resulting link would look like this: Text of the link

Lists There are two types of lists, bulleted lists and numbered lists.

A **bulleted list** looks like this:

- An item
- Another item
- Yet another item

This is accomplished with the following code:

```
* An item
* Another item
* Yet another item
```

A **numbered list** looks like this:

1. First item
2. Second item
3. Third item

This is accomplished with the following code:

```
#. First item
#. Second item
#. Third item
```

Note that numbers are automatically generated, making it easy to add/remove items.

List-tables Bulleted lists can sometimes be cumbersome and hard to follow. When dealing with a long list of items, use list-tables. For example, to talk about a list of options, create a table that looks like this:

Shapes	Description
Square	Four sides of equal length, 90 degree angles
Rectangle	Four sides, 90 degree angles

This is done with the following code:

```
.. list-table::
   :widths: 20 80
   :header-rows: 1

   * - Shapes
     - Description
   * - Square
     - Four sides of equal length, 90 degree angles
   * - Rectangle
     - Four sides, 90 degree angles
```

Notes and warnings To emphasize something Sphinx has two ways, a note and a warning. They work the same, and only differ in their coloring. You should use notes and warnings sparingly, however, as adding emphasis to everything makes the emphasis less effective.

Here is an example of a note:

Note: This is a note.

This note is generated with the following code:

```
.. note:: This is a note.
```

Similarly, here is an example of a warning:

Warning: Beware of dragons.

This warning is generated by the following code:

```
.. warning:: Beware of dragons.
```

Images Add images to your documentation when possible. Images, such as screen shots, are a very helpful way of making documentation understandable. When making screenshots, try to crop out unnecessary content (browser window, desktop, etc). Avoid scaling the images, as the Sphinx theme automatically resizes large images. It is also helpful to include a caption underneath the image.

This image is generated by the following code:

Fig. 3.86: *The GeoNode logo as shown on the homepage.*

```
.. figure:: img/logo.png
   :align: center

   *The GeoNode logo as shown on the homepage.*
```

In this example, the image file exists in the same directory as the source page. If this is not the case, you can insert path information in the above command.

External files Text snippets, large blocks of downloadable code, and even zip files or other binary sources can all be included as part of the documentation. To include files as part of the build process, use the following syntax:

```
:download:`An external file <readme.txt>`
```

The result of this code will generate a standard link to an `external file`

Reference files and paths Use the following syntax to reference files and paths:

```
:file:`myfile.txt`
```

This will output: `myfile.txt`.

You can reference paths in the same way:

```
:file:`path/to/myfile.txt`
```

This will output: `path/to/myfile.txt`.

For Windows paths, use double backslashes:

```
:file:`C:\\myfile.txt`
```

This will output: `C:\myfile.txt`.

If you want to reference a non-specific path or file name:

```
:file:`{your/own/path/to}/myfile.txt`
```

This will output: *your/own/path/to*/`myfile.txt`

Reference commands Reference commands (such as **make**) with the following syntax:

```
:command:`make`
```

Reference an element in a GUI Use the following syntax to direct a user to click a link or look to a certain area of the GUI:

```
:guilabel:`Main Menu`
```

This will output: *Main Menu*.

Menu traversal Direct a user through a menu with the following syntax:

```
:menuselection:`Start Menu --> Programs --> Geonode`
```

This will output *Start Menu* → *Programs* → *Geonode*.

Show Source Every page in the GeoNode documentation has a link for `Show Source` under the Table of Contents on the right side of the page. This allows for easy reverse engineering of unfamiliar markup. When in doubt, look at the source!

How to Translate the Documentation

All documentation is written in English as the master language but can be translated into your native language. This write up will show you how to translate the documentation. There are different methods to translate the documentation and here we list a few.

Translate using Github Locally Here we assume you have github running locally and that you are familiar with the command line.

1. Get latest version from Github:

```
$ git pull
```

2. Get latest version from transifex (it should be synced with Github, but just in case):

```
$ tx pull -a
```

3. Edit your files for example (if using vi):

```
$ vi i18n/it/LC_MESSAGES/index.po
```

4. Build the documentation now, before committing and pushing, build the doc to see if everything worked smoothly:

```
$ make html LANG=it
```

5. Push changes if everything worked well, push the changes:

```
$ tx push -s -t
```

CHAPTER 4

Need Help?

Having trouble? Cant find what you are looking for? We'd like to help!

- Search for information in the archives of the GeoNode mailing list, or post a question.
- Ask a question in the #geonode IRC channel using Freenode's web based client.
- Report bugs with GeoNode in our ticket tracker.