Referencia administradores

Release 1

FAO Forestry
GeoSolutions

Jul 31, 2017

Contents

Contents:

CHAPTER 1

Hoja de ruta para la publicación del portal de diseminación del SNMB

Esta hoja de ruta pretende ser una guía para la publicación de datos en el servidor prototipo para el portal de diseminación de datos del Sistema Nacional de Monitoreo de Bosques, que consiste básicamente en un servidor de bases de datos PostgreSQL y una instancia de GeoServer corriendo sobre Apache Tomcat.

Se parte de los siguientes requisitos:

- existe un servidor con acceso público a internet
- es posible realizar tareas administrativas en dicho servidor
- disponemos de los datos listos para su publicación. En este caso se supone que son ficheros shapefile y geotiffs.

Los pasos a seguir serían los siguientes:

1. Inventario de los datos a publicar. Listado de capas indicando para cada capa:
    - formato: raster o vector, geotiff, ecw, shapefile, etc.
    - número de instancias temporales. Por ejemplo: 2005, 2007, 2011. Otro ejemplo: Diario desde 2002.
    - Tamaño aproximado de cada instancia. 1Mb, 10Mb, 100Mb, 1Gb, 10Gb, ...
    - periodicidad de la actualización: Diario, mensual, anual, cada años.
    - ¿se debe publicar con una leyenda específica? Sí / No. En caso afirmativo indicar el formato en que se puede exportar la leyenda: SLD, QGIS, ArcMap, etc.
2. Instalación del servidor: PostgreSQL, GeoServer y portal. Incluye la configuración del portal para las distintas herramientas, como feedback, estadísticas, etc.
3. Carga de los datos en PostgreSQL. Transferencia de los ficheros al servidor en un directorio temporal para su carga inmediata en PostgreSQL. Una vez cargados, los fichero se eliminan del servidor ya que sólo se accederá a la copia existente en la base de datos.
4. Publicación de los datos de PostgreSQL desde GeoServer, con la simbología por defecto.
5. Configuración de las capas publicadas por GeoServer en el portal.
6. Configuración de la simbología de las capas. Configuración de metatileado para etiquetado de forma correcta.

7. Configuración del aspecto temporal: habilitación de la dimensión temporal en GeoServer, configuración en el portal.
8. Personalización del portal. Creación de una versión con los plugins de interés, cambios de estilo, cabezal, logos.
9. Optimizaciones
    - A nivel informático: parámetros máquina virtual, tomcat, etc.
    - A nivel de simbología: leyendas más sencillas y rápidas de dibujar para escalas más pequeñas.
    - A nivel de datos: Generación de versiones más ligeras de las capas vectoriales consultables. Optimizaciones de los datos raster.
    - A nivel de protocolo: Utilización de GeoWebCache (caché de teselas)
10. Programación de copias de seguridad
11. Robustecimiento del servicio. Herramientas de monitoreo, scripts whatchdog.

CHAPTER 2

Instalación del portal

Note: Obtener aquí la última versión del fichero unredd-portal.war.

La instalación del portal se realiza mediante el copiado del fichero .war de la aplicación al directorio `webapps` de la instancia Tomcat. Por ejemplo:

```
sudo cp unredd-portal.war /var/lib/tomcat/webapps/portal.war
```

En respuesta a esta acción, Tomcat descomprimirá los contenidos del WAR en un directorio con el mismo nombre que el fichero .war y la aplicación se podrá acceder en:

http://localhost:8080/portal/

Para exponer la aplicación por el puerto HTTP 80 a través de Apache2, editar el fichero `/etc/apache2/sites-enabled/000-default.conf` y añadir lo siguiente bajo `<VirtualHost *:80>`:

```
ProxyPass        /portal ajp://localhost:8009/portal
ProxyPassReverse /portal ajp://localhost:8009/portal
```

Reiniciar el servidor:

```
sudo service apache2 restart
```

Y acceder a:

http://localhost/portal/

Configuración del portal

Para la personalización del portal es necesario crear un directorio de configuración en `/var/portal`. Se puede encontrar un directorio de configuración de ejemplo en el portal desempaquetado en la operación anterior, en `/var/lib/tomcat/portal/webapps/portal/WEB-INF/default_config`. La forma más fácil de crear el directorio de configuración es tomar el de ejemplo como base:

```
sudo mkdir /var/portal
sudo cp -R /var/tomcat/webapps/portal/WEB-INF/default_config/* /var/portal/
```

Y luego, hay que indicar a la aplicación cuál es la nueva ubicación. Editar */etc/default/tomcat*, añadiendo la opción PORTAL_CONFIG_DIR a JAVA_OPTS:

```
...
PORTAL_CONFIG_DIR=/var/portal
...
JAVA_OPTS="... -DPORTAL_CONFIG_DIR=$PORTAL_CONFIG_DIR"
```

El resultado final de */etc/default/tomcat7* sería:

```
JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
GEOSERVER_DATA_DIR=/var/geoserver
PORTAL_CONFIG_DIR=/var/portal

JAVA_OPTS="-server -Xms1560m -Xmx2048m -XX:PermSize=256m -XX:MaxPermSize=512m -
↪XX:+UseConcMarkSweepGC -XX:NewSize=48m -Dorg.geotools.shapefile.datetime=true -
↪Duser.timezone=GMT -DGEOSERVER_DATA_DIR=$GEOSERVER_DATA_DIR -Dfile.encoding=UTF-8 -
↪DMINIFIED_JS=true -DPORTAL_CONFIG_DIR=$PORTAL_CONFIG_DIR"
```

Y reiniciamos tomcat:

```
sudo service tomcat7 restart
```

Funcionalidades con acceso a base de datos

Para algunas funcionalidades, como la herramienta de feedback o las estadísticas, el portal interactúa con una base de datos. Para configurar el acceso a la base de datos será necesario configurar dos cosas: la conexión a la base de datos y el esquema en el que se meten las tablas que necesita el portal.

Conexión

La conexión se configura en un fichero `META-INF/context.xml` existente dentro del directorio de la aplicación, en `webapps`, por ejemplo:

```
<Context copyXML="true">

    <!-- Default set of monitored resources -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>

    <!-- Uncomment this to disable session persistence across Tomcat restarts -->
    <!--
    <Manager pathname="" />
    -->

    <!-- Uncomment this to enable Comet connection tacking (provides events
         on session expiration as well as webapp lifecycle) -->
    <!--
    <Valve className="org.apache.catalina.valves.CometConnectionManagerValve" />
    -->

        <Resource name="jdbc/unredd-portal" auth="Container" type="javax.sql.
↪DataSource"
```

```
                driverClassName="org.postgresql.Driver" url="jdbc:postgresql://
↪postgis.unredd:5432/spatialdata"
                username="spatial_user" password="unr3dd" maxActive="20" maxIdle="10"
                maxWait="-1" />

</Context>
```

En concreto en el elemento `Resource`. Dicho elemento debe tener:

- Un atributo `name` con valor "jdbc/unredd-portal", que es el que buscará el portal.
- Un atributo `url` con la URL para conectar a la base de datos.
- Atributos `username` y `password` con las credenciales para conectar a la URL anterior.

Si se instala un .war genérico lo más probable es que este recurso no esté adaptado a la situación de nuestro servidor. Por ejemplo, si tenemos el servidor de base de datos en el mismo servidor que corre el portal, la URL deberá hacer referencia a 127.0.0.1, en lugar de "postgis.unredd". Para configurarlo correctamente tendremos que modificar este fichero. Sin embargo, no podemos modificarlo ahí porque cuando se despliegue un nuevo war, los contenidos que hay en el directorio `webapps` serán eliminados y reemplazados por los contenidos del nuevo .war. Es por esto que Tomcat copia ese fichero a `/var/tomcat/conf/Catalina/localhost/portal.xml`, para poder modificarlo allí y que no se sobreescriba cada vez que se despliega un nuevo .war.

Basta entonces con editar ese fichero y cambiar la URL, usuario y password a nuestras necesidades:

```
<Context copyXML="true">

    <!-- Default set of monitored resources -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>

    <!-- Uncomment this to disable session persistence across Tomcat restarts -->
    <!--
    <Manager pathname="" />
    -->

    <!-- Uncomment this to enable Comet connection tacking (provides events
         on session expiration as well as webapp lifecycle) -->
    <!--
    <Valve className="org.apache.catalina.valves.CometConnectionManagerValve" />
    -->

        <Resource name="jdbc/unredd-portal" auth="Container" type="javax.sql.
↪DataSource"
                driverClassName="org.postgresql.Driver" url="jdbc:postgresql://127.0.
↪0.1:5432/mibasededatos"
                username="miusuario" password="mipassword" maxActive="20" maxIdle="10"
                maxWait="-1" />

</Context>
```

Una vez editado el fichero hay que reiniciar el portal como se explica aquí: *Reinicio del portal*.

Esquema

Todas las funcionalidades que necesitan apoyo de la base de datos acceden a tablas con nombre conocido en un esquema que se configura en el fichero "portal.properties", situado en el directorio de configuración del portal, mediante la propiedad `db-schema`.

Así, para configurar estas funcionalidades hay que seguir dos pasos:

1. Especificar el esquema con la propiedad db-schema. Ver *portal.properties*.
2. Crear las tablas de nombre conocido según la funcionalidad que se desee instalar.
   - Servicio de estadísticas, ver *Instalación*
   - Herramienta de feedback, ver *Configuración*

Reinicio del portal

Cuando se modifican datos relativos a la base de datos es necesario reiniciar el portal. Esto se hace forzando a que Tomcat redespliegue la aplicación y puede hacerse copiando el fichero de nuevo en `/var/tomcat/webapps/` o, más simple, mediante el comando touch:

```
$ touch /var/tomcat/webapps/portal.war
```

Este comando cambia la fecha y hora de última modificación del fichero, forzando así a Tomcat a que vuelva a desplegar el .war. A los pocos segundos el portal se habrá reiniciado.

Warning: En algunos casos, sobre todo cuando la conexión a la base de datos apunta a algún servidor que no existe, el comando `touch` puede no ser efectivo. En tales casos, es necesario reiniciar Tomcat:

```
sudo service tomcat7 restart
```

Verificación del despliegue

Para verificar que la inicialización del portal ha sido exitosa es conveniente realizar dos comprobaciones por orden.

1. La primera es acceder a la URL del navegador y visualizar el portal. Es posible que la primera petición devuelva algún tipo de error pero tras unos pocos segundos el portal se deberá mostrar normalmente.

   Warning: Cuando se reinicia el servicio de Tomcat entero en lugar de sólo el portal, se da el caso de que todas las aplicaciones se inicializan una detrás de otra. Por tanto, el tiempo que tarda el portal en iniciarse depende de cuántas aplicaciones haya además de ésta y de lo "pesadas" que sean. Por ejemplo, si tenemos GeoServer con muchos datos cargados, es probable que durante el reinicio de Tomcat el portal tome más tiempo que si estuviera sólo.

2. La segunda es verificar en los logs que la inicialización del portal y la visualización en el navegador no han dado ningún error. Para ver los errores que se dan internamente en el portal es necesario visualizar los logs. Cada vez que el portal se inicializa o se carga en un navegador, el sistema escribe información relevante en un fichero de log de Tomcat, en concreto en el fichero `/var/tomcat/logs/catalina.out`.

   El comando `less` nos permite visualizar este fichero fácilmente:

   ```
   $ less /var/tomcat/logs/catalina.out
   ```

   donde encontraremos algo como esto:

```
INFO: Despliegue del archivo portal.war de la aplicación web
2015-04-02 02:22:39 INFO  ConfigFolder:55 -␣
↪================================================================================
2015-04-02 02:22:39 INFO  ConfigFolder:56 - PORTAL_CONFIG_DIR: /var/portal
2015-04-02 02:22:39 INFO  ConfigFolder:57 -␣
↪================================================================================
2015-04-02 02:22:39 DEBUG ConfigFolder:73 - Reading portal properties file /var/
↪argentina/portal.properties
2015-04-02 02:22:39 DEBUG ConfigFolder:73 - Reading portal properties file /var/
↪argentina/portal.properties
```

En caso de que haya algún error nos encontraremos con algo así:

```
INFO: Despliegue del archivo portal.war de la aplicación web
2015-04-02 02:34:50 INFO  ConfigFolder:55 -␣
↪================================================================================
2015-04-02 02:34:50 INFO  ConfigFolder:56 - PORTAL_CONFIG_DIR: /var/argentina
2015-04-02 02:34:50 INFO  ConfigFolder:57 -␣
↪================================================================================
2015-04-02 02:34:50 DEBUG ConfigFolder:73 - Reading portal properties file /var/
↪argentina/portal.properties
2015-04-02 02:34:50 DEBUG ConfigFolder:73 - Reading portal properties file /var/
↪argentina/portal.properties
2015-04-02 02:34:50 ERROR FeedbackContextListener:66 - Database error notifying␣
↪the comment authors
org.fao.unredd.portal.PersistenceException: Database error
        at org.fao.unredd.portal.DBUtils.processConnection(DBUtils.java:41)
        at org.fao.unredd.portal.DBUtils.processConnection(DBUtils.java:14)
        at org.fao.unredd.feedback.DBFeedbackPersistence.
↪getValidatedToNotifyInfo(DBFeedbackPersistence.java:122)
        at org.fao.unredd.feedback.Feedback.notifyValidated(Feedback.java:83)
        at org.fao.unredd.feedback.servlet.FeedbackContextListener$1.
↪run(FeedbackContextListener.java:61)
        at java.util.TimerThread.mainLoop(Timer.java:512)
        at java.util.TimerThread.run(Timer.java:462)
Caused by: org.apache.tomcat.dbcp.dbcp.SQLNestedException: Cannot create␣
↪PoolableConnectionFactory (Conexión rechazada. Verifique que el nombre del Host␣
↪y el puerto sean correctos y que postmaster este aceptando conexiones TCP/IP.)
        at org.apache.tomcat.dbcp.dbcp.BasicDataSource.
↪createPoolableConnectionFactory(BasicDataSource.java:1549)
        at org.apache.tomcat.dbcp.dbcp.BasicDataSource.
↪createDataSource(BasicDataSource.java:1388)
        at org.apache.tomcat.dbcp.dbcp.BasicDataSource.
↪getConnection(BasicDataSource.java:1044)
        at org.fao.unredd.portal.DBUtils.processConnection(DBUtils.java:37)
        ... 6 more
Caused by: org.postgresql.util.PSQLException: Conexión rechazada. Verifique que␣
↪el nombre del Host y el puerto sean correctos y que postmaster este aceptando␣
↪conexiones TCP/IP.
        at org.postgresql.core.v3.ConnectionFactoryImpl.
↪openConnectionImpl(ConnectionFactoryImpl.java:215)
        at org.postgresql.core.ConnectionFactory.openConnection(ConnectionFactory.
↪java:64)
        at org.postgresql.jdbc2.AbstractJdbc2Connection.<init>
↪(AbstractJdbc2Connection.java:144)
        at org.postgresql.jdbc3.AbstractJdbc3Connection.<init>
↪(AbstractJdbc3Connection.java:29)
        at org.postgresql.jdbc3g.AbstractJdbc3gConnection.<init>
↪(AbstractJdbc3gConnection.java:21)
```

```
        at org.postgresql.jdbc4.AbstractJdbc4Connection.<init>
↪(AbstractJdbc4Connection.java:31)
        at org.postgresql.jdbc4.Jdbc4Connection.<init>(Jdbc4Connection.java:24)
        at org.postgresql.Driver.makeConnection(Driver.java:410)
        at org.postgresql.Driver.connect(Driver.java:280)
        at org.apache.tomcat.dbcp.dbcp.DriverConnectionFactory.
↪createConnection(DriverConnectionFactory.java:38)
        at org.apache.tomcat.dbcp.dbcp.PoolableConnectionFactory.
↪makeObject(PoolableConnectionFactory.java:582)
        at org.apache.tomcat.dbcp.dbcp.BasicDataSource.
↪validateConnectionFactory(BasicDataSource.java:1556)
        at org.apache.tomcat.dbcp.dbcp.BasicDataSource.
↪createPoolableConnectionFactory(BasicDataSource.java:1545)
        ... 9 more
Caused by: java.net.ConnectException: Connection refused
        at java.net.PlainSocketImpl.socketConnect(Native Method)
        at java.net.PlainSocketImpl.doConnect(PlainSocketImpl.java:351)
        at java.net.PlainSocketImpl.connectToAddress(PlainSocketImpl.java:213)
        at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:200)
        at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:366)
        at java.net.Socket.connect(Socket.java:529)
        at org.postgresql.core.PGStream.<init>(PGStream.java:61)
        at org.postgresql.core.v3.ConnectionFactoryImpl.
↪openConnectionImpl(ConnectionFactoryImpl.java:109)
        ... 21 more
```

Para más información, consulte *Portal: Configuración inicial*.

Resolución de problemas

En los casos en los que el portal no se despliegua correctamente, es necesario buscar información sobre lo que puede estar funcionando mal.

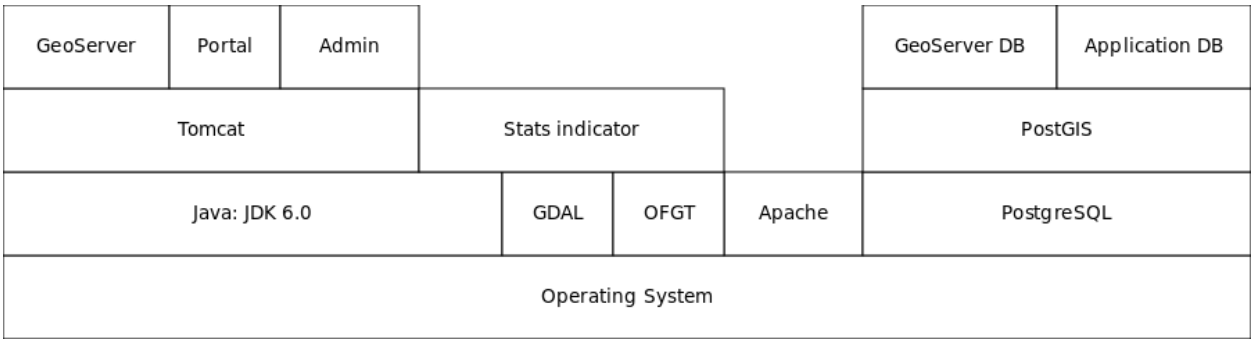
1. Lo primero y más sencillo es abrir una herramienta como FireBug, las herramientas para desarrolladores de Firefox o de Google Chrome y realizar de nuevo la operación que da problemas. A continuación podemos echar un vistazo a:
   (a) La pestaña Consola, para ver si hay algún mensaje de error.
   (b) La pestaña Red, para ver si hay algún recurso del portal que no está descargándose de forma correcta. En caso de encontrar algún recurso con error de carga que pueda ser sospechoso, es posible hacer clic en él con el botón derecho del ratón y abrirlo en una nueva ventana, de manera que el navegador nos reporte directamente el mensaje de error.
2. La segunda consiste en visualizar los logs como se explica en el punto anterior: *Verificación del despliegue*.

CHAPTER 3

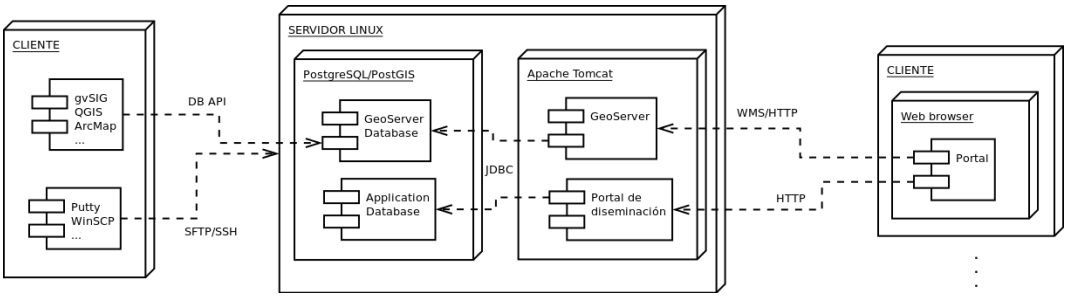
Arquitectura

Note:	Fecha	Autores
	24 Junio 2013	• Fernando González (fernando.gonzalez@fao.org)

El sistema incluye una serie de tecnologías que se apoyan unas sobre las otras, siendo la base el sistema operativo y estando en la parte superior las aplicaciones desarrolladas para el portal del SNMB. Es posible observar en el siguiente diagrama que tanto los portales desarrollados como GeoServer hacen uso de Java. De la misma manera la utilidad para crear estadísticas está también desarrollada en Java pero hace uso a su vez de herramientas del sistema como GDAL y OFT.



Cuando se realiza la instalación del sistema y éste se encuentra en funcionamiento, tenemos dentro de Tomcat una instancia de GeoServer y otra del portal, sobre el servidor de base de datos PostgreSQL/PostGIS tenemos dos bases de datos, una para almacenar los datos que sirve GeoServer y otra para los datos de la aplicación.



En general, se conectará al sistema de las siguientes formas:

- A Tomcat con un navegador, via HTTP, para visualizar el portal.
- A Tomcat con un navegador, via HTTP, para administrar GeoServer y configurar los servicios OGC, mapas, etc.
- Al sistema operativo desde un cliente SFTP (como WinSCP) para descargar ficheros de configuración, editarlos en local y volverlos a subir al servidor. También para copiar datos que se quieran publicar (shapefiles).
- Al sistema operativo desde un cliente SSH (como PuTTY) para realizar otras tareas administrativas que requieran ejecución de comandos en el servidor: carga de datos en postgreSQL, copias de seguridad, etc.
- A la base de datos espacial desde un cliente GIS para visualizar y editar los datos.

CHAPTER 4

Resumen administración

Note:

Fecha	Autores
24 Junio 2013	• Fernando González (fernando.gonzalez@fao.org)
18 Marzo 2014	• Víctor González (victor.gonzalez@geomati.co)

Java

- Directorio de instalación: /usr/lib/jvm/jdk1.6.0_45
- Enlace simbólico: /usr/lib/jvm/default-java
- Exntensiones imageIO y JAI instaladas

Tomcat 7

- Directorio de instalación: /var/apache-tomcat-7.0.47
- Enlace simbólico: /var/tomcat
- Variables de entorno: /var/tomcat/bin/setenv.sh
- Servicio tomcat7: (/etc/init.d/ubuntuTomcatRunner.sh, /etc/init.d/tomcat7)

Apache

Configuración proxy AJP (redirigir peticiones a Tomcat)

- Fichero de configuración: /etc/apache2/mods-available/proxy_ajp.conf
- Enlace simbólico al fichero anterior para habilitar el módulo: /etc/apache2/mods-enabled/proxy_ajp.conf
- Servicio apache2

Repositorio Ubuntu GIS

Se añade el repositorio "ppa:ubuntugis/ppa" para la instalación de GDAL y PostGIS. La instalación de GDAL y PostGIS se hace vía "apt-get".

GeoServer

- Instalado como war: /var/tomcat/webapps/geoserver.war
- Directorio de datos: /var/geoserver/data

Portal

Directorio de instalación: /var/portal

Dentro de dicho directorio:

- Configuración general: portal.properties
- Árbol de capas: layers.json
- Contenido estático: static/
- Imágenes internacionalizadas: static/loc/<lang>/images (donde <lang> corresponde con el idioma en el que se encuentran los recursos)
- HTML internacionalizado: static/loc/<lang>/html (donde <lang> corresponde con el idioma en el que se encuentran los recursos)
- Ficheros internacionalización de texto: messages/

Para desplegar el portal en el servidor basta con copiar el fichero `portal.war` en `/var/tomcat/webapps/portal.war`.

Una vez copiado, si Tomcat está arrancado y funcionando, el portal se desplegará de manera automática en unos instantes.

En el caso de que se quiera **actualizar** el portal con una nueva versión, bastará con sobreescribir el fichero `/var/tomcat/webapps/portal.war` con la nueva versión. De nuevo, si Tomcat está arrancado y funcionando, el portal se actualizará automáticamente en unos instantes.

PostgreSQL

Se instalan dos bases de datos. La primera de ellas es “app” y contiene los datos de configuración del portal. Está en desuso actualmente ya que toda la configuración se almacena en /var/portal.

La segunda es “geoserverdata” (“geoserver” en versiones anteriores) y contiene los datos que se desean publicar.

- Base de datos portal: app (en desuso actualmente)
- Base de datos geoserver: geoserverdata

Se configuran los usuarios “app” y “geoserver” para acceder a las bases de datos “app” y “geoserverdata” respectivamente.

CHAPTER 5

WinSCP

WinSCP es una herramienta para la copia de ficheros desde el servidor a la máquina local y viceversa.

Puede ser descargado desde esta dirección: http://winscp.net

TODO

CHAPTER 6

PuTTY

PuTTY es una herramienta que sirve para iniciar sesiones Secure SHell (SSH) en una máquina remota.

Puede descargarse desde aquí: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

TODO

CHAPTER 7

Introducción a Linux

Note:

Fecha	Autores
1 Septiembre 2012	• Fernando González (fernando.gonzalez@geomati.co) • Micho García (micho.garcia@geomati.co)
24 Junio 2013	• Fernando González (fernando.gonzalez@fao.org)

En el símbolo de sistema presentado es posible hacer uso de una serie de comandos que la mayor parte de sistemas linux tienen. Pero antes de ver los comandos es importante tener claro cómo se organiza el sistema de ficheros y cómo se referencian estos mediante rutas relativas y absolutas.

El sistema de ficheros linux se organiza jerárquicamente a partir de un directorio llamado "raíz" y que se denota por la barra inclinada hacia adelante (/). En linux los ficheros se referencian mediante rutas. Estas rutas pueden ser relativas o absolutas. Las absolutas comienzan por /, mientras que las relativas empiezan por el nombre de un subdirectorio, por . (directorio actual) o por .. (directorio padre).

Así pues, podemos tener rutas absolutas como:

```
/tmp
/home/geo
/home/geo/Escritorio
etc.
```

Note: En la documentación antepondremos el símbolo `$` a toda aquella instrucción que se puede ejecutar en la línea

de comandos de un sistema Linux. ¡Pero dicho símbolo no forma parte de la instrucción!

Las rutas absolutas se pueden utilizar desde cualquier directorio. Podemos listar los directorios anteriores con los siguientes comandos, independientemente del directorio en el que se esté:

```
$ ls /tmp
$ ls /home/geo
$ ls /home/geo/Escritorio
```

Las rutas relativas en cambio, parten del directorio actual. Si por ejemplo estamos en */home/geo*, podemos listar los directorios anteriores con los siguientes comandos:

```
$ ls ../../tmp
$ ls .
$ ls Escritorio
```

o "navegando" de forma más caprichosa:

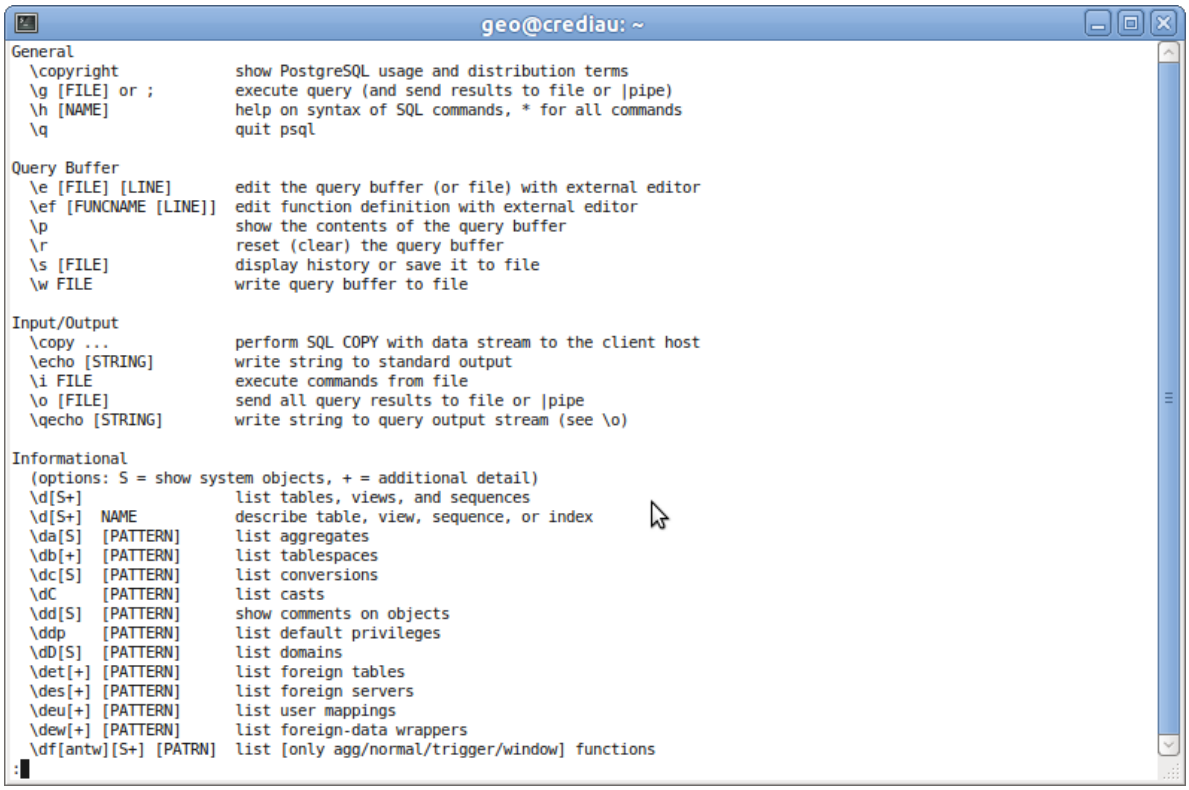
```
$ ls Escritorio/../../../tmp
$ ls ./././././././././././../geo
$ ls ../geo/Escritorio
```

A continuación mostramos algunos comandos útiles en linux:

- less: Visualiza un fichero de texto. La interacción es la misma que la descrita en el apartado "Ayuda de psql" anterior:

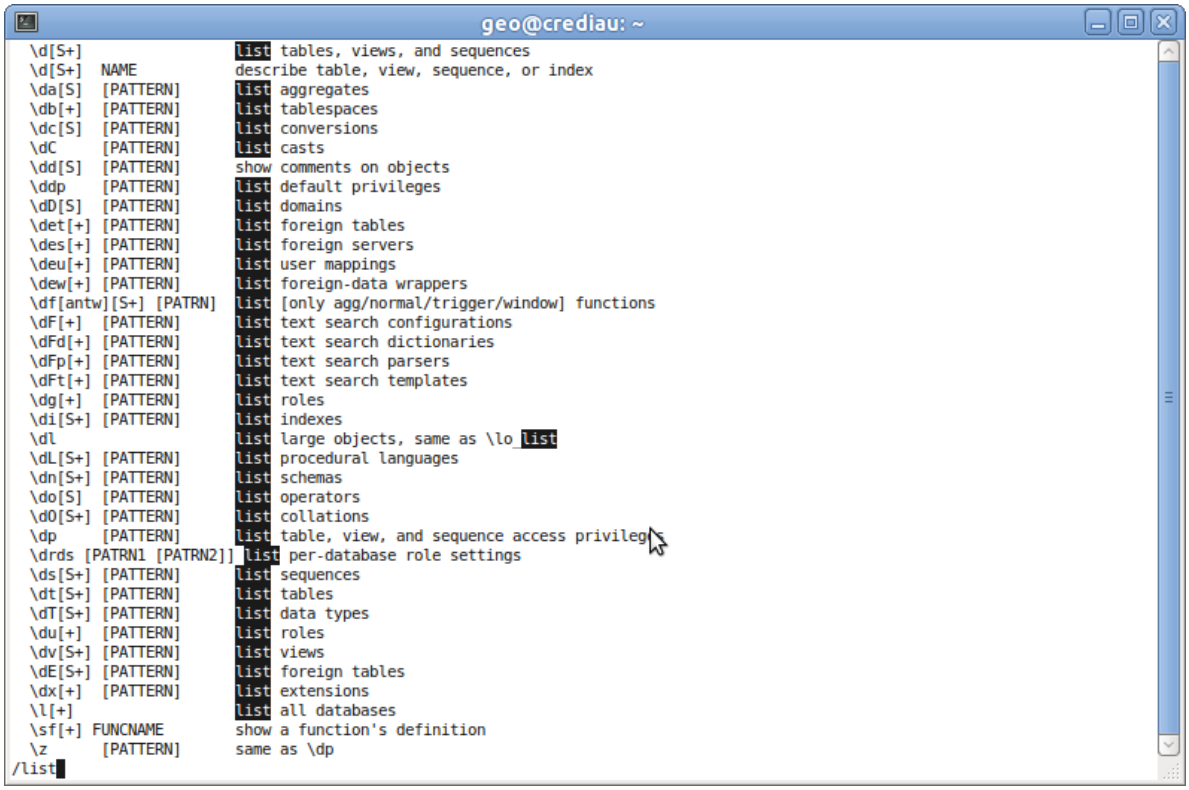
  ```
  $ less ruta_fichero_a_visualizar
  ```

  El fichero a visualizar se presenta de una manera muy común en los sistemas UNIX y que podemos identificar porque en la esquina inferior izquierda tenemos el signo de los dos puntos (:) seguido del cursor.



Podemos navegar por el contenido pulsando los cursores arriba y abajo, así como las teclas de página anterior y posterior.

También es posible hacer búsquedas utilizando el comando /texto. Una vez pulsamos intro, se resaltarán las coincidencias encontradas, como se puede ver en la siguiente imagen. Para navegar a la siguiente coincidencia es posible pulsar la tecla 'n' y para ir a la anterior Mayúsculas + 'n'

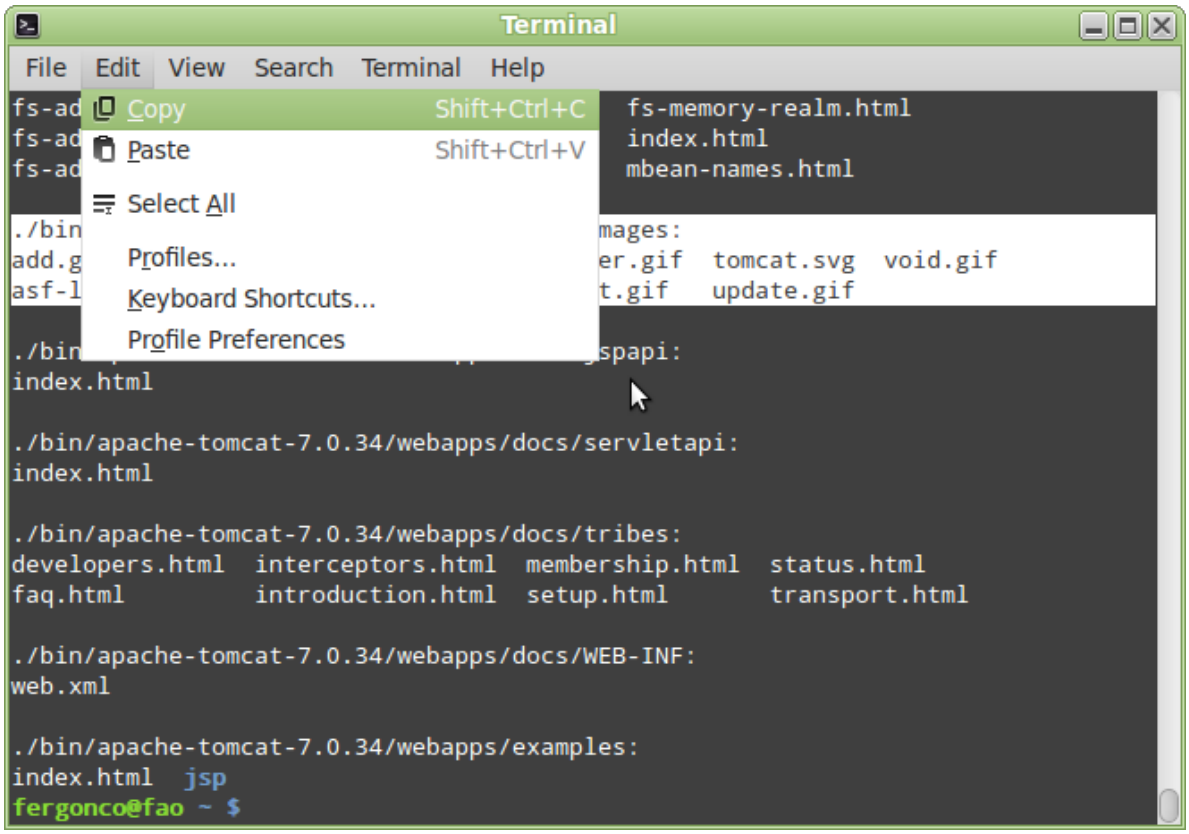


Para salir pulsar 'q'.

- nano: Permite editar ficheros. En la parte de abajo se muestran los comandos para salir, guardar el fichero, etc.:

```
$ nano ruta_fichero_a_editar
```

Cuando se trabaja en modo texto en linux, dependiendo de la aplicación terminal utilizada, es posible copiar y pegar texto de la forma habitual: seleccionando con el ratón y presionando una combinación de teclas. Sin embargo, esta combinación de teclas suele ser diferente a las habituales (`Ctrl+C`, `Ctrl+V`) ya que `Ctrl+C` tiene un significado distinto en el terminal: el de interrumpir el proceso que se está ejecutando. La combinación de teclas se puede averiguar si se utiliza un terminal con barra de menúes como el siguiente:



Si la aplicación terminal que se utiliza no incorpora menu, como `xterm`, siempre se puede utilizar un método bastante cómodo y siempre disponible en Linux que consiste en seleccionar el texto y pegar directamente con el botón central del ratón. Lo engañoso de este método es que el texto se pega en la posición del cursor y no allí donde se pincha.

Ejercicio: Crear un fichero con tu nombre y que contenga este apartado.

- locate: Localiza ficheros en el sistema operativo:

```
$ locate parte_del_nombre_del_fichero
```

Un aspecto a tener en cuenta en el uso de locate es que el sistema programa escaneos regulares del disco para construir un índice con los ficheros existentes y permitir así a *locate* responder al usuario sin tener que realizar la búsqueda en el sistema de ficheros, que toma mucho tiempo. Es por esto que *locate* funciona muy rápido pero puede que no encuentre los ficheros creados recientemente. Para estos, habrá que esperar a que se produzca un escaneo programado o lanzar un escaneo de forma manual con `updatedb`.

- find: Localiza ficheros en el sistema de archivos:

```
$ find ruta -name nombre_del_fichero
```

A diferencia de locate, el comando find recorrerá el sistema de archivos cada vez que se lo ejecute, sin emplear índices. Por esa razón, si bien es mucho más lento el resultado, puede hallar ficheros que no se hayan indexado, por ejemplo, los ficheros creados recientemente.

- id: Muestra la identidad actual del usuario:

```
$ id
```

- su: Permite autenticarse con un usuario distinto. El siguiente comando probablemente no funcionará porque es necesario tener permisos de superusuario para realizar *su*, ver el siguiente caso:

```
$ su postgres
```

- sudo: No es un comando en sí, sino que permite ejecutar el comando que le sigue con permisos de superusuario. Por ejemplo, para ejecutar el comando anterior con permisos de superusuario:

```
$ sudo su postgres
```

- passwd: Cambia el password de un usuario. Por ejemplo para cambiar el password de root:

```
$ sudo passwd root
```

- ssh: Acceso remoto en línea de comandos. Con SSH es posible entrar a un servidor remoto que tenga activado dicho acceso. Para ello es necesario especificar la dirección del servidor:

```
$ ssh 168.202.48.151
The authenticity of host '168.202.48.151 (168.202.48.151)' can't be established.
ECDSA key fingerprint is 9f:7c:a8:9c:8b:66:37:68:8b:7f:95:a4:1b:24:06:39.
Are you sure you want to continue connecting (yes/no)? yes
```

  En la salida anterior podemos observar como primeramente el sistema pregunta por la autenticidad de la máquina a la que queremos conectar. Tras responder afirmativamente el sistema nos comunica que el servidor al que vamos a conectarnos se añade a la lista de hosts conocidos, de manera que el mensaje anterior no volverá a aparecer la siguiente vez que se intente una conexión. A continuación el sistema pregunta el password del usuario "usuario":

```
Warning: Permanently added '168.202.48.151' (ECDSA) to the list of known hosts.
usuario@168.202.48.151's password:
```

  En caso de querer conectar con otro usuario es necesario prefijar el nombre de dicho usuario, seguido del carácter "@" antes de la dirección del servidor:

```
$ ssh otro_usuario@168.202.48.151
```

- scp: Copia ficheros al servidor:

```
$ scp fichero_origen directorio_destino
```

  El directorio puede ser una ruta normal o la cadena de conexión por SSH a un servidor remoto. Veamos varios ejemplos. El siguiente copia ficheros locales en el directorio */tmp* de un servidor remoto:

```
$ scp mi_fichero_local geo@geoportalcredia.org:/tmp
```

  El siguiente comando copia el fichero de vuelta:

```
$ scp geo@geoportalcredia.org:/tmp/mi_fichero_local .
```

  Se puede observar que el format de la URL remota es parecido al que se usa para conectar por cliente SSH. La única diferencia es que al final, separado por (:), encontramos una ruta en la máquina remota

  Ejercicio: Conectar a una máquina linux usando estos comandos.

  Ejercicio: Copiar el fichero creado en el apartado sobre `nano` en /tmp

  Ejercicio: Conectar al sistema linux desde windows y copiar un fichero cualquiera haciendo uso de putty.exe y scp.exe.

- zip: Comprime ficheros:

```
$ zip -r ruta_fichero.zip lista_de_ficheros_a_comprimir
```

  La opción -r hace que zip incluya los contenidos de los directorios que se encuentre en la lista de ficheros a compartir.

- unzip: Descomprime ficheros:

```
$ unzip ruta_fichero.zip
```

- chgrp: cambia el grupo de usuarios de un archivo o directorio en sistemas tipo UNIX. Cada archivo de Unix tiene un identificador de usuario (UID) y un identificador de grupo (GID) que se corresponden con el usuario y el grupo de quien lo creó.

  El usuario root puede cambiar a cualquier archivo el grupo. Los demás usuarios sólo pueden hacerlo con los archivos propios y grupos a los que pertenezca.:

```
$ chgrp nuevogrp archivo1 [ archivo2 archivo3...]

Cambia el grupo de archivo1 archivo2, etc. que pasará a ser nuevogrp

$ chgrp -R nuevogrp directorio

Cambia el grupo para que pase a ser nuevogrp a directorio, todos los archivos y␣
↪subdirectorios contenidos en él, cambiándolos también de forma recursiva en␣
↪todos archivos de los subdirectorios.
```

- chown: cambiar el propietario de un archivo o directorio:

```
$ chown nuevousr archivo1 [ archivo2 archivo3...]
$ chown -R nuevousr directorio
```

- chmod: permite cambiar los permisos de acceso de un archivo o directorio:

```
$ chmod [modificadores] permisos archivo/directorio
```

  Ejercicio: Quitarse el permiso de lectura sobre el fichero creado en el apartado de `nano`.

- wget: Utilizado para descargar ficheros de distintos servidores HTTP, HTTPS y FTP. Basta con teclear wget seguido de la dirección del fichero en internet:

```
wget http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf
```

  Entre las muchas opciones que soporta, la más frecuente es `-O <nombre_fichero>`, que permite dar un nombre distinto al fichero descargado:

```
wget http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf -O especificacion_
↪shapefile.pdf
```

  Ejercicio: Descargar el logo del portal de FAO (http://fao.org) con wget

Realizar el siguiente ejercicio:

1. Crear un fichero llamado `/tmp/copy-contents.sh` con las siguientes líneas (sustituyendo <servidor> y <nombre> por valores adecuados):

```
wget http://www.diva-gis.org/data/rrd/ARG_rrd.zip -O rails.zip
unzip rails.zip
scp * nfms@<servidor>:/tmp/<nombre>
```

2. Dar permisos de ejecución
3. Ejecutar

Ejercicio: Crear un fichero vacío en `/var/lib/postgresql`

De cuantas maneras es posible realizar esto?

1. Usando `sudo` para crear el fichero
2. Creando el fichero como `postgres`
3. Cambiando los permisos al directorio. ¡NO!

CHAPTER 8

Servicios web

Note:

Fecha	Autores
24 Junio 2013	• Fernando González (fernando.gonzalez@fao.org)

Existen múltiples definiciones del concepto de servicio web, ya que es un concepto muy general. En el caso que nos ocupa nos referiremos a servicios disponibles en la red que pueden ser accedido usando el protocolo HTTP, especificando unos parámetros y obteniendo una salida como resultado. Dichos servicios son componentes que realizan una tarea específica y que pueden ser combinados para construir servicios más complejos.

Al contrario que las aplicaciones monolíticas en las que los componentes están fuertemente acoplados, los sistemas basados en servicios web fomentan la independencia de los distintos elementos que forman la aplicación. Así, los componentes son servicios que exponen una API al resto para la colaboración en el contexto de la aplicación y que pueden ser intercambiados fácilmente por otros servicios que ofrezcan la misma API.

HTTP

Los servicios que se van a consumir durante el curso se construyen sobre el protocolo HyperText Transfer Protocol (HTTP), por lo que se van a ilustrar algunos conceptos del mismo.

Las interacciones HTTP se dan cuando un equipo cliente envía peticiones a un servidor. Estas peticiones incluyen un encabezado con información descriptiva sobre la petición y un cuerpo de mensaje opcional. Entre los datos del encabezado se encuentra el método requerido: GET, PUT, PUSH, etc. Está fuera del ámbito de esta documentación explicar las semánticas de los distintos métodos exceptuando la mención de que la barra de direcciones de los navegadores web realiza peticiones HTTP GET para descargarse el recurso especificado en la dirección. Por ejemplo:

- http://www.fao.org/fileadmin/templates/faoweb/images/FAO-logo.png

- http://www.diva-gis.org/data/rrd/ARG_rrd.zip
- http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf
- http://docs.geoserver.org/stable/en/user/gettingstarted/shapefile-quickstart/index.html

Siguiendo con el mismo ejemplo, mediante la barra de direcciones podemos descargar distintos tipos de contenidos: páginas HTML, ficheros de texto plano, ficheros XML, imágenes, vídeos, etc. Algunos de estos contenidos son directamente interpretados por el navegador mientras que para otros se ofrece la descarga del recurso. ¿En qué se basa el navegador para tomar estas decisiones?

Cada respuesta desde el servidor tiene también una cabecera en la que especifica el *Content-Type* de los datos que vienen en el cuerpo del mensaje de respuesta. El *Content-Type* puede ser por ejemplo:

- text/html
- text/plain
- text/xml
- image/gif
- video/mpeg
- application/zip

El navegador usa este valor para interpretar de una manera u otra el flujo de bytes que le envía el servidor, de manera que si en la cabecera aparece "image/gif" entenderá que está recibiendo una imagen y la mostrará al usuario, mientras que si lee "text/html" el navegador interpretará los bytes recibidos como una página HTML y la visualizará, la hará responder a los eventos del usuario, etc.

Por último, la respuesta incorpora un código con información adicional sobre lo que sucedió en el servidor al recibir la petición. El código más habitual usando el navegador es el 200, que informa que el contenido de la respuesta es aquello que se pidió. Otros códigos indican condiciones de errores, como el frecuente 404, que indica que el recurso que se ha pedido no existe en el servidor, o el 500 que indica que hubo un error en el servidor al procesar la petición.

Servicios del Open Geospatial Consortium (OGC)

El Open Geospatial Consortium es un consorcio industrial internacional que reúne centenares de compañias, agencias gubernamentales y universidades para la participación en la creación de estándares disponibles de forma pública.

En el contexto de los servicios web el OGC define los OGC Web Services (OWS), que son estándares construidos sobre el protocolo HTTP y que definen los parámetros, información de cabecera, etc. de las distintas peticiones y sus respuestas, así como la relación entre ellas. Por ejemplo, el estándar WMS, del que hablaremos posteriormente en profundidad, define una petición `GetMap` para obtener imágenes de datos almacenados en el servidor y define que la respuesta debe tener un *Content-Type* de imágenes (`image/png`, `image/gif`, etc.) y que la petición deberá incluir una serie de parámetros para poder generar la imagen: nombre de la capa, extensión a dibujar, tamaño de la imagen resultante, etc. Los OWS proporcionan una infraestructura interoperable, independiente de la implementación y de los proveedores para la creación de aplicaciones basadas en web y relacionadas con la información geográfica.

Entre las ventajas del uso de estándares podemos destacar:

- Es posible consumir servicios de otros proveedores, independientemente de la implementación concreta de estos.
- Existen desarrollos OpenSource que implementan estos estándares y permiten por tanto la publicación con interfaces estándar de forma sencilla.
- Multitud de herramientas OpenSource que permiten trabajar con los datos consumidos a través de estas interfaces.

Dos de los OWS más representativos son Web Map Service (WMS) y Web Feature Service (WFS), que vemos con algo más de detalle a continuación.

El estándar WMS define básicamente tres tipos de peticiones: *GetCapabilities*, *GetMap* y *GetFeatureInfo*. La primera de ellas es común para todos los OWS y devuelve un documento XML con información sobre las capas existentes en el servidor, los sistemas de coordenadas (CRS, Coordinate Reference System) soportados, etc.

Ejemplo:

```
http://www.cartociudad.es/wms/CARTOCIUDAD/CARTOCIUDAD?REQUEST=GetCapabilities
```

La petición GetMap obtiene imágenes con representaciones gráficas de la información geográfica existente en el servidor:

```
http://www.cartociudad.es/wms/CARTOCIUDAD/CARTOCIUDAD?REQUEST=GetMap&SERVICE=WMS&
↪VERSION=1.3.0&LAYERS=DivisionTerritorial&
        CRS=EPSG:25830&BBOX=718563.200906236,4363954.866694199,735300.5071689372,
↪4377201.249079251&WIDTH=701&
        HEIGHT=555&FORMAT=image/png&STYLES=municipio&TRANSPARENT=TRUE
```

Y por último, la petición GetFeatureInfo es capaz de proporcionar información sobre un punto:

```
http://www.cartociudad.es/wms/CARTOCIUDAD/CARTOCIUDAD?REQUEST=GetFeatureInfo&
↪SERVICE=WMS&QUERY_LAYERS=CodigoPostal&
        VERSION=1.3.0&INFO_FORMAT=application/vnd.ogc.gml&LAYERS=CodigoPostal&
↪CRS=EPSG:25830&
        BBOX=665656.9561496238,4410190.54853407,690496.231896245,4427113.624503085&
↪WIDTH=706&HEIGHT=481&
        FORMAT=image/png&STYLES=codigo_postal&TRANSPARENT=TRUE&I=475&J=204&FEATURE_
↪COUNT=10000&EXCEPTIONS=XML
```

Ejercicio: ¿Qué otra utilidad conocemos para visualizar los tres enlaces anteriores?

Por su parte el estándar WFS no trabaja con representaciones de los datos sino que lo hace con los datos mismos. Para ello define las siguientes llamadas:

- GetCapabilities: Al igual que todos los OWS, WFS admite la llamada GetCapabilities para obtener una lista de las capas y posibilidades que ofrece el servicio.
- DescribeFeatureType: Permite obtener un documento con la estructura de los datos.
- GetFeature: Permite realizar una consulta al sistema y obtener las entidades que cumplen los criterios de búsqueda.

Así, podemos ver qué capas hay en un servicio WFS:

```
http://www.cartociudad.es/wfs-comunidad/services?request=GetCapabilities&service=WFS
```

Consultar la estructura de una de ellas:

```
http://www.cartociudad.es/wfs-comunidad/services?request=DescribeFeatureType&
↪service=WFS&VERSION=1.0.0&
        TypeName=app:entidadLocal_&outputformat=text/xml;%20subtype=gml/3.1.1
```

Y efectivamente descargar algunas de sus entidades:

```
http://www.cartociudad.es/wfs-comunidad/services?REQUEST=GetFeature&SERVICE=WFS&
↪TYPENAME=app:entidadLocal_&
        NAMESPACE=xmlns%28app=http://www.deegree.org/app%29&VERSION=1.1.0&
↪EXCEPTIONS=XML&MAXFEATURES=10
```

CHAPTER 9

PostgreSQL

Note:

Fecha	Autores
1 Septiembre 2012	• Fernando González (fernando.gonzalez@geomati.co) • Micho García (micho.garcia@geomati.co)
24 Junio 2013	• Fernando González (fernando.gonzalez@fao.org) • Ramiro Mata (rmata@zonageo.com.ar) • Leandro Roncoroni (lronco@zonageo.com.ar)

Los contenidos de este punto son inicialmente traducciones de la documentación oficial de PostgreSQL que han sido extendidos posteriormente.

Introducción

El objetivo de este tutorial sobre *PostgreSQL* es que el usuario sea capaz de crear y eliminar bases de datos y acceder a ellas para la manipulación de los datos.

Por esto, los puntos siguientes están pensados para dar una introducción simple a *PostgreSQL*, a los conceptos básicos sobre bases de datos relacionales y al lenguaje SQL. No se requiere experiencia en sistemas UNIX ni en programación.

Tras el tutorial, es posible continuar el aprendizaje leyendo la documentación oficial del proyecto, en inglés, en la que se puede encontrar abundante información sobre el lenguaje SQL, el desarrollo de aplicaciones para PostgreSQL y la configuración y administración de servidores.

Arquitectura cliente/servidor

Al igual que el resto de componentes instalados, *PostgreSQL* utiliza un modelo cliente/servidor, ya explicado en la introducción.

Las aplicaciones cliente pueden ser de naturaleza muy diversa: una herramienta orientada a texto (psql), una aplicación gráfica (pgAdmin3), un servidor web que accede a la base de datos para mostrar las páginas web, o una herramienta de mantenimiento de bases de datos especializadas. Algunas aplicaciones de cliente se suministran con la distribución *PostgreSQL* mientras que otras son desarrolladas por los usuarios.

Creación de una base de datos

El primer paso para trabajar con *PostgreSQL* es crear una base de datos. Para ello es necesario ejecutar como usuario *postgres* el comando *createdb*:

```
$ sudo su postgres
$ createdb mibd
```

Si no se tiene acceso físico al servidor o se prefiere acceder de forma remota es necesario utilizar un cliente SSH. La siguiente instrución:

```
$ ssh geo@190.109.197.226
```

conecta al servidor 190.109.197.226 con el usuario *geo*.

Ejercicio: Conectar al sistema desde Windows y crear una base de datos.

Generalmente el mejor modo de mantener la información en la base de datos es utilizando un usuario distinto a *postgres*, que sólo debería usarse para tareas administrativas. Es posible incluso crear más de un usuario con diferentes derechos (SELECT, INSERT, UPDATE, DELETE) para tener un entorno más seguro. Sin embargo, esto queda fuera del ámbito de este tutorial y se conectará siempre con el usuario *postgres*.

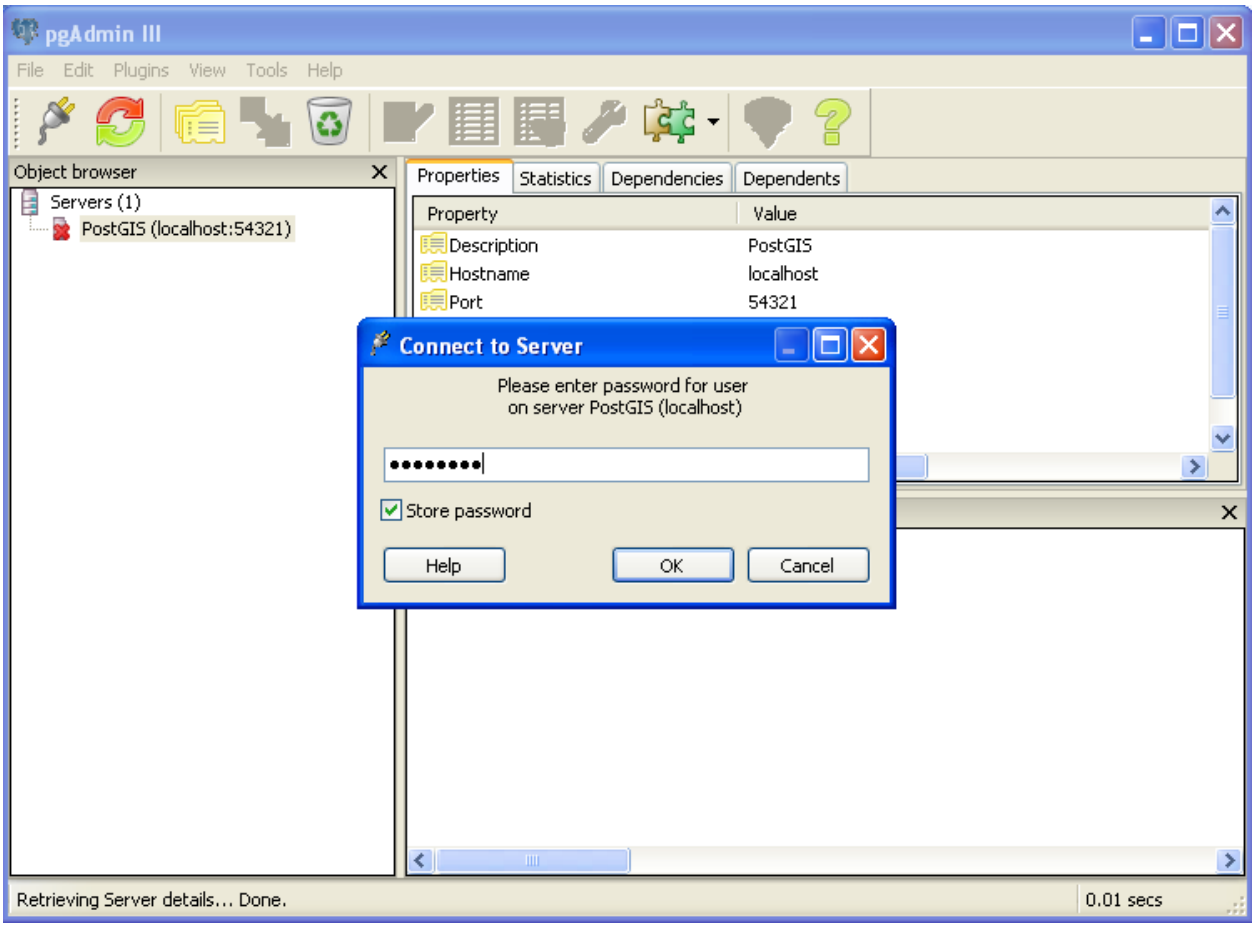
Acceso a una base de datos

Una vez la base de datos ha sido creada es posible utilizar un cliente para conectar a ella. Existen varias maneras:

- psql: el programa de terminal interactivo de PostgreSQL que permite introducir de forma interactiva, editar y ejecutar comandos SQL. Veremos más adelante qué es SQL. Es el que utilizaremos.
- una herramienta existente con interfaz gráfica, como pgAdmin, que veremos brevemente.
- una aplicación personalizada desarrollada con algún lenguaje para el que haya un driver de acceso. Esta posibilidad no se trata en esta formación.

Para conectar con pgAdmin se deberá seleccionar el menu File > Add Server y registrar el nuevo servidor con su dirección IP y el puerto en el que está escuchando (5432 por defecto). También habrá que indicar el nombre de usuario con el que se desea hacer la conexión.

Una vez se tiene configurada una entrada para la base de datos en pgAdmin, es posible conectar a dicho servidor haciendo doble click en dicha entrada.



Una vez creada, es posible selecionar la nueva base de datos y mostrar el árbol de objetos que contiene. Se puede ver el esquema "public" que no contiene ningún elemento.

Para seguir interactuando con la base de datos abriremos una ventana SQL clicando sobre el siguiente icono:

Que abrirá una ventana que permite enviar comandos SQL al servidor de base de datos. Probemos con los siguientes comandos:

```
SELECT version ();
SELECT current_date;
SELECT 2 + 2;
```

psql

También podemos conectar a la base de datos con psql. Podemos conectar con psql desde cualquier máquina que tenga una versión de psql compatible con el servidor. El propio servidor tiene dicho programa instalado y es obviamente compatible por lo que la mejor opción es acceder al servidor:

```
$ ssh geo@190.109.197.226
```

Es posible especificar al comando `psql` la base de datos a la que se quiere acceder, el usuario con el que se quiere realizar el acceso y la instrucción que se quiere ejecutar en el sistema. Los valores concretos utilizados dependerán de la configuración concreta del servidor. En adelante usaremos el usuario de base de datos `postgres` y la base de datos `geoserverdata`.

La siguiente instrucción invoca la función `version`:

```
$ psql -U postgres -d test_database -c "SELECT version ()"
                                                 version
-------------------------------------------------------------------------------------------
↪----------------------
 PostgreSQL 9.1.5 on x86_64-unknown-linux-gnu, compiled by gcc (Ubuntu/Linaro 4.6.3-
↪1ubuntu5) 4.6.3, 64-bit
(1 row)
```

Otros ejemplos:

```
$ psql -U postgres -d test_database -c "SELECT current_date"
    date
------------
 2012-09-11
(1 row)

$ psql -U postgres -d test_database -c "SELECT 2 + 2"
 ?column?
----------
        4
(1 row)
```

Todos estos comandos SQL pueden ser ejecutados usando otro parámetro del programa `psql`. La opción -f permite especificar un fichero que contiene instrucciones SQL. Así, por ejemplo sería posible crear un fichero en `/tmp/mi_script.sql` con el siguiente contenido:

```
SELECT version ();
SELECT current_date;
SELECT 2 + 2;
```

Y ejecutarlo con la instrucción:

```
$ psql -U geoserver -d geoserverdata -f /tmp/mi_script.sql

                                                    version
-------------------------------------------------------------------------------------------------
↪----------------
 PostgreSQL 9.1.11 on i686-pc-linux-gnu, compiled by gcc (Ubuntu/Linaro 4.6.3-
↪1ubuntu5) 4.6.3, 32-bit
(1 row)

    date
------------
 2014-02-11
(1 row)

 ?column?
----------
        4
(1 row)
```

Como se puede observar, se ejecutan todos los comandos del script sql uno detrás de otro.

Consola psql interactiva

También es posible, y conveniente para tareas de mayor complejidad, entrar al modo interactivo de `psql`. Para ello podemos omitir el parámetro -c:

```
$ psql -U postgres -d test_database
```

o conectar sin especificar la base de datos y usar el comando \c dentro de *psql*:

```
$ psql -U postgres
=# \c test_database
You are now connected to database "mibd" as user "postgres".
```

Note: Dado que psql es un programa en línea de comandos tenemos que diferenciar en la documentación las instrucciones que se deben de ejecutar en la línea de comandos del sistema operativo y la línea de comandos de psql. Las primeras, como se comentó en la introducción a Linux, vienen precedidas del símbolo del dólar ($) mientras que para las últimas utilizaremos un par de símbolos: =#. Es necesario prestar atención a este detalle durante el resto de la documentación.

En el resto de la documentación se seguirán enviando comandos SQL desde la línea de comandos del sistema operativo ($) usando el parámetro -c o el parámetro -f, como especificado anteriormente. Sin embargo, se especifica a continuación una mínima referencia sobre los comandos que se pueden ejecutar en la línea de comandos de postgresql (=#)

Para obtener el listado de las bases de datos existentes en el sistema, usar el comando \l:

```
=# \l
```

Y para listar tablas del esquema por defecto de la base de datos actual (*public*):

```
=# \dt
```

Si queremos listar las tablas que hay en otro esquema es posible utilizar la siguiente sintaxis:

```
=# \dt gis.*
```

Por último, para obtener información sobre cualquier objeto de la base de datos es posible utilizar el comando \d:

```
=# \d gis.categorias
```

Se puede añadir un + para obtener información más detallada:

```
=# \d+ gis.categorias
```

Ayuda de psql

Para una completa referencia de los comandos disponibles es posible usar el comando \?:

```
=# \?
```

que nos abrirá la ayuda. El formato de la ayuda es el mismo que el del comando *less*.

Cargando información desde shapefile: shp2pgsql

El parámetro -f es extremadamente útil cuando queremos usar PostgreSQL junto con su extensión espacial PostGIS para la carga de datos desde shapefile. Para ello contamos con `shp2pgsql`, que es capaz de generar un script SQL a partir de un shapefile que al ejecutar en PostgreSQL generará una tabla espacial con los mismos datos del shapefile.

La sintaxis básica es sencilla:

```
shp2pgsql <shapefile> <nombre_de_tabla_a_crear>
```

Por ejemplo:

```
$ shp2pgsql provincias.shp provincia
```

El comando anterior realmente muestra por pantalla el script, lo cual no es muy útil y además tarda mucho tiempo (con Ctrl+C es posible cancelar la ejecución en la mayoría de los casos). Para que realmente sea útil tenemos que almacenar los datos en un fichero que luego podamos pasar a psql con el parámetro -f. Esto lo podemos hacer mediante redireccionando la salida estándar a un fichero temporal:

```
$ shp2pgsql provincias.shp provincias > /tmp/provincias.sql
```

Es posible que durante este proceso obtengamos un error similar a éste:

```
Unable to convert data value to UTF-8 (iconv reports "Invalid or incomplete multibyte␣
↪or wide character"). Current encoding is "UTF-8". Try "LATIN1" (Western European),␣
↪or one of the values described at http://www.postgresql.org/docs/current/static/
↪multibyte.html.
```

lo cual quiere decir que la codificación utilizada para almacenar los textos en el fichero .dbf no es UTF-8, que es la que espera el programa `shp2pgsql` por defecto. También nos sugiere que intentemos LATIN1. Para decirle al programa qué codificacion utilizamos, podemos especificar el parámetro -W:

```
$ shp2pgsql -W LATIN1 provincias.shp provincias > /tmp/provincias.sql
```

Y si nuestros datos están en LATIN1 se generará el script sin ningún problema.

A continuación no tenemos más que cargar el fichero recién generado con psql:

```
$ psql -U postgres -d geoserverdata -f /tmp/provincias.sql
```

Tras la ejecución podemos ver con cualquier sistema GIS que soporte conexiones PostGIS 2.0 (como QGis) que se ha creado una tabla en PostreSQL/PostGIS con los mismos datos que contenía el shapefile.

El siguiente aspecto que tenemos que tener en cuenta, es que el sistema de referencia de coordenadas (CRS) no está especificado. Por ejemplo, ejecutando esta instrucción:

```
$ psql -U postgres -d geoserverdata -c "select * from geometry_columns"

 f_table_catalog | f_table_schema |      f_table_name       | f_geometry_column |␣
↪coord_dimension | srid |      type
-----------------+----------------+-------------------------+-------------------+-----
↪------------+------+-----------------
 geoserverdata   | public         | provincias              | geom              |    ␣
↪          2 |    0 | MULTIPOLYGON
```

podemos observar que la tabla recién creada tiene un campo srid, que indica el código EPSG del sistema de coordenadas utilizado, con valor igual a 0. Para evitar esto es posible utilizar el parámetro -s de `shp2pgsql`:

```
$ shp2pgsql -s 4326 provincias.shp provincias > /tmp/provincias.sql
```

que establecerá que nuestros datos están en EPSG:4326 (o el CRS que se especifique).

Por último, es recomendable crear nuestros datos en un esquema distinto de `public` para facilitar las copias de seguridad y las actualizaciones de PostGIS, por motivos que no se tratan en esta documentación:

```
$ psql -U postgres -d geoserverdata -c "create schema gis"
CREATE SCHEMA
$ shp2pgsql -s 4326 provincias.shp gis.provincias > /tmp/provincias.sql
```

Incluso es posible cargar en PostgreSQL el fichero resultante con una única línea, sólo enlazando la salida de `shp2pgsql` con la entrada de `psql` mediante una tubería de linux "|":

```
$ shp2pgsql -s 4326 provincias.shp gis.provincias | psql -U postgres -d geoserverdata
```

Por ejemplo los siguientes comandos cargan una serie de datos en PostGIS, en la base de datos `geoserver`:

```
$ psql -U postgres -d geoserver -c "create schema gis"
$ shp2pgsql -s 4326 -W LATIN1 /tmp/datos/ARG_adm0.shp gis.admin0 | psql -U postgres -
↪d geoserverdata
$ shp2pgsql -s 4326 -W LATIN1 /tmp/datos/ARG_adm1.shp gis.admin1 | psql -U postgres -
↪d geoserverdata
$ shp2pgsql -s 4326 -W LATIN1 /tmp/datos/ARG_adm2.shp gis.admin2 | psql -U postgres -
↪d geoserverdata
$ shp2pgsql -s 4326 -W LATIN1 /tmp/datos/ARG_rails.shp gis.ferrovia | psql -U␣
↪postgres -d geoserverdata
$ shp2pgsql -s 4326 -W LATIN1 /tmp/datos/ARG_roads.shp gis.vias | psql -U postgres -d␣
↪geoserverdata
$ shp2pgsql -s 4326 -W LATIN1 /tmp/datos/ARG_water_areas_dcw.shp gis.zonas_agua |␣
↪psql -U postgres -d geoserverdata
$ shp2pgsql -s 4326 -W LATIN1 /tmp/datos/ARG_water_lines_dcw.shp gis.lineas_agua |␣
↪psql -U postgres -d geoserverdata
```

Nótese que todos estos pasos se pueden simplificar en sólo dos, que cargarían todos los shapefiles de un directorio:

```
$ psql -U postgres -d geoserver -c "create schema gis"
$ for i in `ls /tmp/datos/*.shp`; do shp2pgsql -s 4326 $i gis.${i%.shp} | psql -U␣
↪postgres -d geoserverdata; done
```

El siguiente ejemplo crea una base de datos llamada `analisis` y dentro de ella un esquema llamado `gis`. Luego se instala la extensión PostGIS y por último se cargan en la base de datos todos los shapefiles existentes en el directorio `Escritorio/datos/analisis`:

```
$ psql -U postgres -c "create database analisis"
$ psql -U postgres -d analisis -c "create schema gis"
$ psql -U postgres -d analisis -c "create extension postgis"
$ for i in `ls /tmp/datos/analisis/*.shp`; do shp2pgsql -s 25830 $i gis.${i%.shp} |␣
↪psql -U postgres -d analisis; done
```

Creación de copias de seguridad

Un aspecto importante a la hora de administrar un servidor de base de datos es la creación de copias de seguridad.

Para hacer y restaurar la copia de seguridad se utilizan los comandos `pg_dump` y `pg_restore` en la línea de comandos del sistema operativo. El comando `pg_dump` tiene la siguiente sintaxis:

```
pg_dump <options> <database>
```

Entre las opciones más interesantes están:

- username: nombre del usuario con el que conectar a la base de datos para realizar la copia: –username=geo
- password: clave para conectar a la base de datos
- host: dirección del servidor de base de datos. Se puede omitir si es la máquina desde la cual se lanza el comando: –host=192.168.2.107
- schema: esquema que se quiere copiar. Si no se especifica se copiarán todos los esquemas.
- format: formato de la copia. Para obtener un formato compatible con `pg_restore` es necesario especificar "c": –format=c
- file: fichero donde queremos guardar la copia de seguridad: –file=/tmp/db.backup

Así, si accedemos a la base de datos "geoserverdata" con el usuario "geoserver" y quisiéramos hacer una copia del esquema "gis" podríamos ejecutar la siguiente instrucción desde la línea de comandos del servidor de base de datos:

```
$ pg_dump --username=geoserver --format=c --schema=gis --file=/tmp/gis.backup␣
↪geoserverdata
```

Dicho comando creará un fichero en `/tmp/gis.backup` con la copia de todos los datos que hay en el esquema "gis".

Para recuperar la copia se puede utilizar el comando `pg_restore`:

```
$ pg_restore --username=geoserver --dbname=geoserverdata /tmp/gis.backup
```

Si el esquema existe, el comando `pg_restore` dará un error por lo que si queremos reemplazar los contenidos del esquema deberemos renombrar el esquema primero con la siguiente instrucción:

```
$ psql --username=geoserver --dbname=geoserverdata --command="alter schema gis rename␣
↪to gis2"
```

Una vez la copia de seguridad ha sido recuperada de forma satisfactoria es posible eliminar el esquema renombrado:

```
$ psql --username=geoserver --dbname=geoserverdata --command="drop schema gis2 cascade
↪"
```

Warning: Para que todo este proceso se de sin problemas, es importante que los datos estén en un esquema distinto de "public", ya que algunas extensiones, como PostGIS, instalan tablas y funciones en dicho esquema y al hacer el backup estaremos incluyendo también estos objetos que luego no dejarán recuperar la copia.

Warning: También es muy importante guardar los ficheros con la copia de seguridad en una máquina distinta al servidor de bases de datos, ya que en caso de que haya algún problema con dicha máquina se pueden perder también las copias.

Más información

La página web de *PostgreSQL* se puede consultar aquí[1]. En ella hay abundante información en inglés[2], así como listas de correo en español[3].

También se puede descargar un curso de PostGIS de bastante difusión[4].

Referencias

[1] http://www.postgresql.org
[2] http://www.postgresql.org/docs/9.2/static/index.html
[3] http://archives.postgresql.org/pgsql-es-ayuda/
[4] http://blog.lookingformaps.com/2012/11/publicada-documentacion-del-curso-bases.html

CHAPTER 10

Indexación espacial

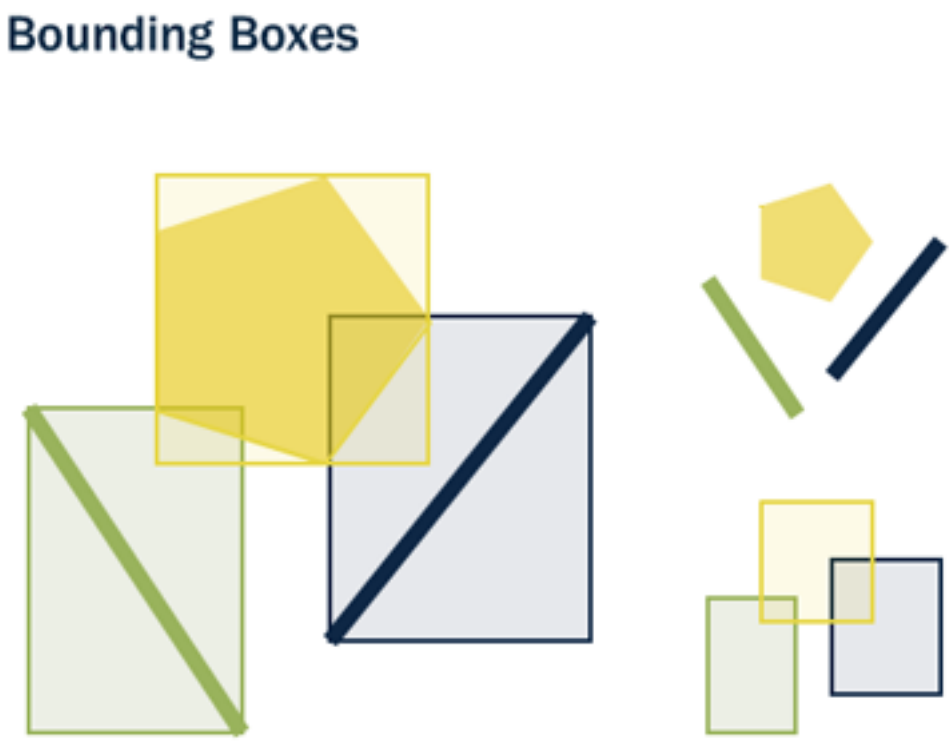
Note:

Fecha	Autores
1 Noviembre 2012	• Micho García (micho.garcia@geomati.co)
15 Octubre 2013	• Jorge Arévalo(jorge.arevalo@geomati.co)

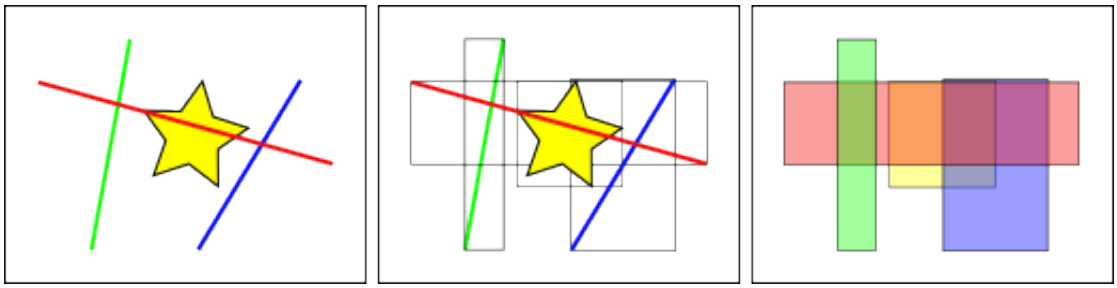
La indexación espacial es una de las funcionalidades importantes de las bases de datos espaciales. Los indices consiguen que las búsquedas espaciales en un gran número de datos sean eficientes. Sin idenxación, la búsqueda se realizaria de manera secuencial teniendo que buscar en todos los registros de la base de datos. La indexación organiza los datos en una estructura de arbol que es recorrida rapidamente en la búsqueda de un registro.

Como funcionan los índices espaciales

Las base de datos estándar crean un arbol jerarquico basados en los valores de las columnas. Los indice espaciales funcionan de una manera diferente, los índices no son capaces de indexar las geometrías, e indexarán las cajas (box) de las geometrías.



La caja (box) es el rectángulo definido por las máximas y mínimas coordenadas x e y de una geometría.



En la figura se puede observar que solo la linea intersecta a la estrella amarilla, mientras que si utilizamos los índices comprobaremos que la caja amarilla es intersectada por dos figuras la caja roja y la azul. El camino eficiente para responder correctamente a la pregunta **¿qué elemento intersecta la estrella amarilla?** es primero responder a la pregunta **¿qué cajas intersectan la caja amarilla?** usando el índice (consulta rápida) y luego calcular exactamente **¿quien intersecta a la estrella amarilla?** sobre el resultado de la consulta de las cajas.

Creación de indices espaciales

La síntaxis será la siguiente:

```
CREATE INDEX [Nombre_del_indice] ON [Nombre_de_tabla] USING GIST ([campo_de_
↪geometria]);
```

Esta operación puede requerir bastante tiempo en tablas de gran tamaño.

Uso de índices espaciales

La mayor parte de las funciones en PostGIS (ST_Contains, ST_Intersects, ST_DWithin, etc) incluyen un filtrado por indice automáticamente.

Para hacer que una función utilice el índice, hay que hacer uso del operador **&&**. Para las geometrías, el operador **&&** significa "la caja que toca (touch) o superpone (overlap)" de la misma manera que para un número el operador **=** significa "valores iguales"

ANALYZE y VACUUM

El planificador de PostGIS se encarga de mantener estadísticas sobre la distribución de los datos de cada columna de la tabla indexada. Por defecto PostgreSQL ejecuta la estadísticas regularmente. Si hay algún cambio grande en la estructura de las tablas, es recomendable ejecutar un `ANALYZE` manualmente para actualizar estas estadísticas. Este comando obliga a PostgreSQL a recorrer los datos de las tablas con columnas indexadas y actualizar sus estadísticas internas.

No solo con crear el índice y actualizar las estadísticas obtendremos una manera eficiente de manejar nuestras tablas. La operación `VACUUM` debe ser realizada siempre que un indice sea creado o después de un gran número de UPDATEs, INSERTs o DELETEs. El comando `VACUUM` obliga a PostgreSQL a utilizar el espacio no usado en la tabla que dejan las actualizaciones y los borrados de elementos.

Hacer un `VACUUM` es crítico para la eficiencia de la base de datos. PostgreSQL dispone de la opción `Autovacuum`. De esta manera PostgreSQL realizará VACUUMs y ANALYZEs de manera periodica en función de la actividad que haya en la tabla:

```
VACUUM ANALYZE [Nombre_tabla]
VACUUM ANALYZE [Nombre_tabla] ([Nombre_columna])
```

Esta orden actualiza las estadísticas y elimina los datos borrados que se encontraban marcados como eliminados.

Prácticas

1. Compare los resultados de obtener la población del barrio West Village así como el tiempo necesario para ejecutar cada operación. Use el operador de caja en una si y en otra no.

CHAPTER 11

Introducción a GeoServer

Note:

Fecha	Autores
1 Diciembre 2012	• Oscar Fonts (oscar.fonts@geomati.co)

La interfaz web de administración de GeoServer está accesible en:

> http://localhost/geoserver/

Nótese que es posible acceder remotamente a dicha interfaz, de tal manera que si la dirección de la máquina es por ejemplo *172.16.250.131*, es posible acceder desde cualquier otra máquina:

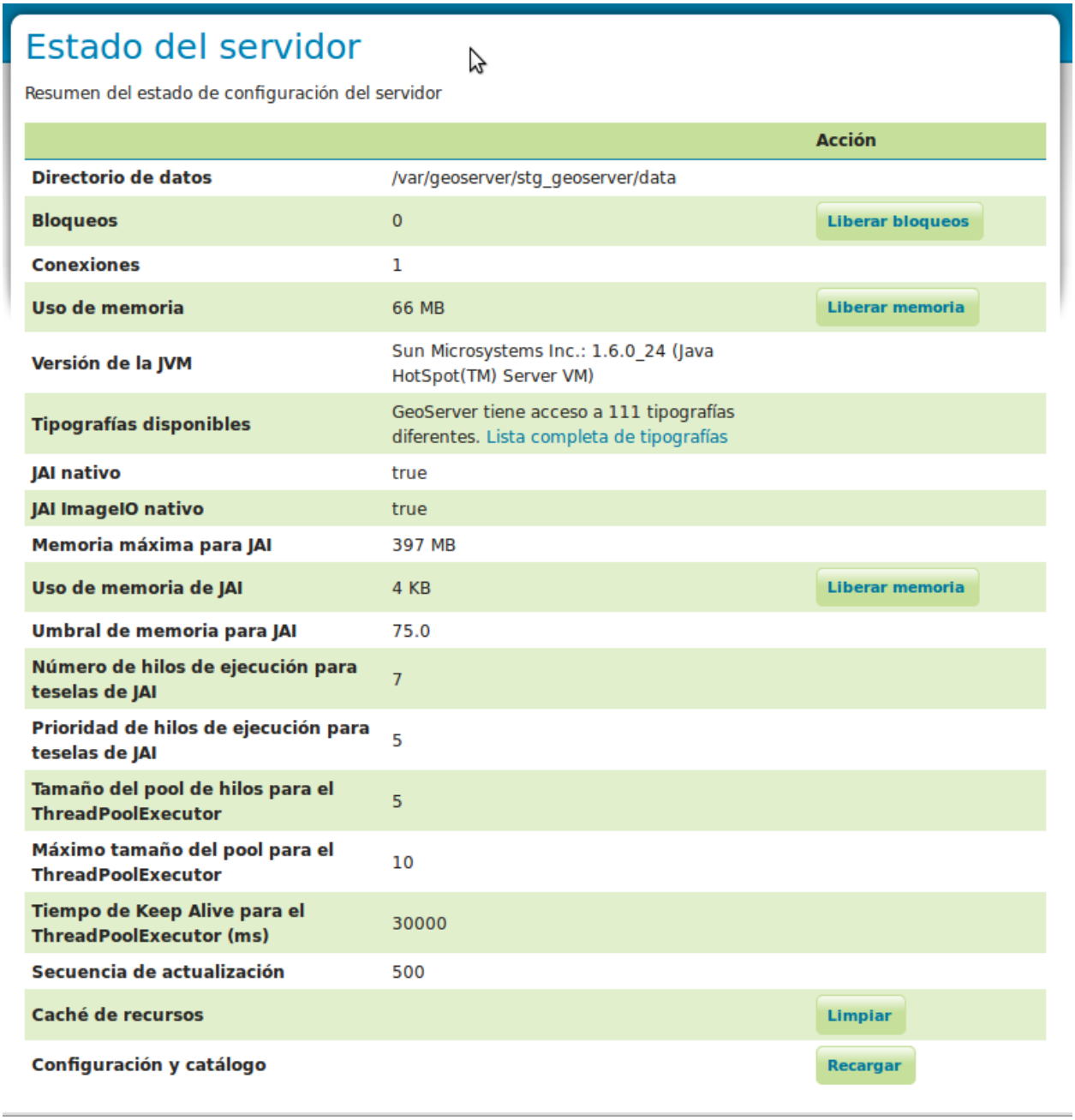
```
http://172.16.250.131/geoserver/
```

Para poder cambiar la configuración, es necesario identificarse con el usuario **admin** y contraseña **geoserver**.

La columna de la izquerda reune los enlaces hacia todas las páginas de configuración.

En este apartado veremos brevemente la primera sección, **Servidor**.

Estado del Servidor



Estado del servidor

Resumen del estado de configuración del servidor

		Acción
Directorio de datos	/var/geoserver/stg_geoserver/data	
Bloqueos	0	Liberar bloqueos
Conexiones	1	
Uso de memoria	66 MB	Liberar memoria
Versión de la JVM	Sun Microsystems Inc.: 1.6.0_24 (Java HotSpot(TM) Server VM)	
Tipografías disponibles	GeoServer tiene acceso a 111 tipografías diferentes. Lista completa de tipografías	
JAI nativo	true	
JAI ImageIO nativo	true	
Memoria máxima para JAI	397 MB	
Uso de memoria de JAI	4 KB	Liberar memoria
Umbral de memoria para JAI	75.0	
Número de hilos de ejecución para teselas de JAI	7	
Prioridad de hilos de ejecución para teselas de JAI	5	
Tamaño del pool de hilos para el ThreadPoolExecutor	5	
Máximo tamaño del pool para el ThreadPoolExecutor	10	
Tiempo de Keep Alive para el ThreadPoolExecutor (ms)	30000	
Secuencia de actualización	500	
Caché de recursos		Limpiar
Configuración y catálogo		Recargar

El directorio de datos

La información más importante de esta primera página es el **directorio de datos**, técnicamente conocido como *GEOSERVER_DATA_DIR*. Indica el directorio donde se almacenará toda la información relativa a la configuración de GeoServer. Por tanto, es una localización de la que convendrá realizar copias de seguridad.

En nuestro caso, el directorio de datos es `/var/geoserver/data`.

Máquina Java y JAI nativo

Para un óptimo rendimiento de GeoServer, es recomendable utilizar la máquina virtual de java de Oracle 1.6, e instalar las librerías nativas JAI y JAI ImageIO.

Los detalles sobre la instalación de GeoServer quedan fuera del alcance de esta guía. En caso necesario, se pueden consultar en la documentación técnica de referencia de la plataforma, y en la documentación oficial de GeoServer.

Logs de GeoServer

El `log` o **archivo de registro** de una aplicación es un fichero de texto donde se van almacenando detalles sobre la ejecución del mismo. Así, un archivo de `log` guarda un histórico con el detalle de las operaciones realizadas, con mayor o menor detalle. Generalmente, cuando ocurre un error de ejecución, se consulta este archivo para obtener detalles sobre las causas del mismo.

Información de Contacto

Esta información de contacto se utilizará en los documentos de *GetCapabilities* del servidor de mapas. Así pues, es una información que se hará pública, y que servirá a los consumidores de los geoservicios para contactar con sus responsables. Es por tanto importante rellenarla con datos significativos:


Información de contacto
Establezca la información de contacto para este servidor
Persona de contacto
X
Organización
Secretaría de Ambiente
Posición
NFMS4REDD Technical Contact
Tipo de dirección
Office
Dirección
Calle X
Ciudad
Asunción
Estado
Distrito Capital
Código postal o ZIP
País
Paraguay
Teléfono
1234
Fax
4321
E-mail
contact@seam.py
Enviar
Cancelar

Demostración: Es posible visualizar esta información desde gvSIG al realizar la conexión al servidor cargando una capa WMS por ejemplo.

Acerca de GeoServer

Esta es una página informativa donde se puede consultar la versión de GeoServer, así como enlaces a la web principal del proyecto, a la documentación, y al sistema de seguimiento de incidencias.

Gestión de usuarios

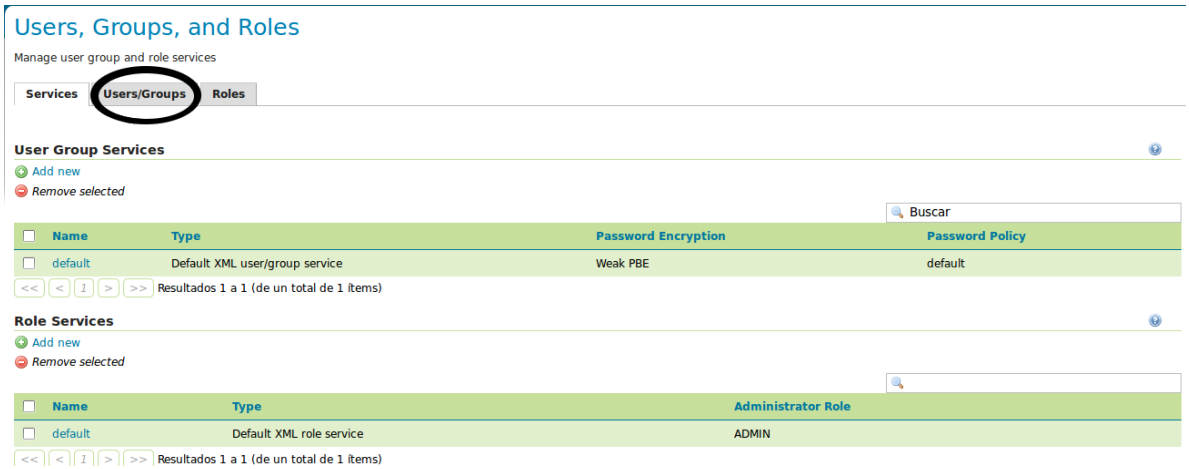
TODO: verificar que está actualizado

Existe la posibilidad de cambiar la contraseña del usuario *admin*, así como de crear nuevas cuentas de usuario con permisos de administración. Sin embargo en este punto nos centraremos únicamente en el cambio de contraseña del usuario *admin*.

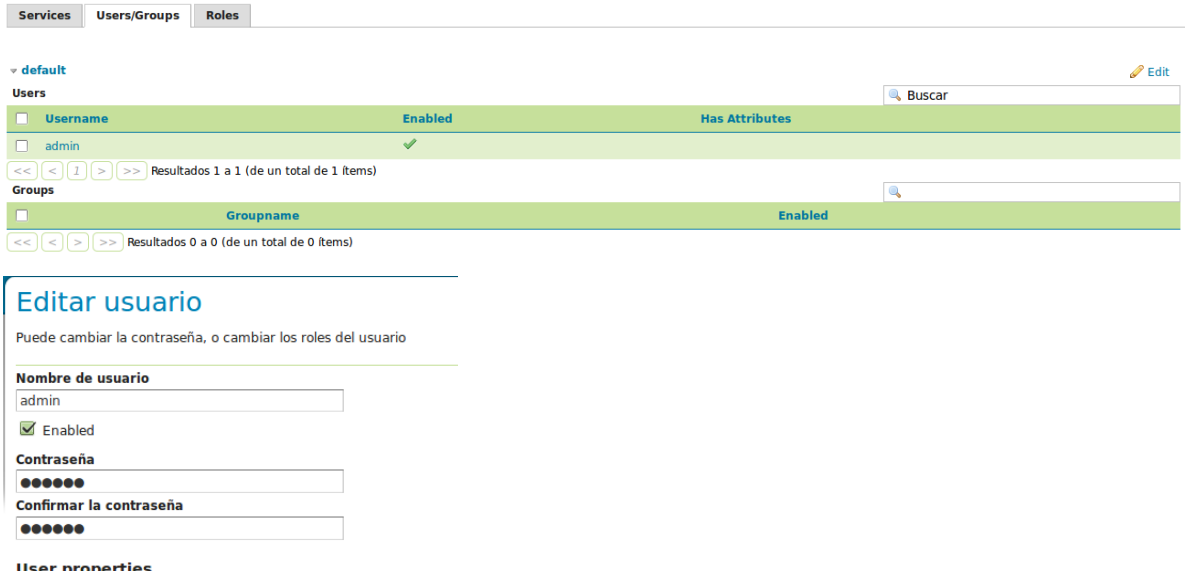
Para ello, hay que seguir los siguientes pasos:

1. lo primero que hay que hacer es seleccionar la entrada "Users, Groups, Roles" del apartado de "Seguridad".

2. En la pantalla resultante hay que seleccionar *Users/Groups*.



3. Y en ella pinchar sobre *admin* para poder editar su password:


Services
Users/Groups
Roles
default
Edit
Users
Buscar
Username
Enabled
Has Attributes
admin
Resultados 1 a 1 (de un total de 1 ítems)
Groups
Groupname
Enabled
Resultados 0 a 0 (de un total de 0 ítems)
Editar usuario
Puede cambiar la contraseña, o cambiar los roles del usuario
Nombre de usuario
admin
Enabled
Contraseña
Confirmar la contraseña
User properties

CHAPTER 12

GeoServer: Publicación de datos vectoriales

Note:

Fecha	Autores
1 Diciembre 2012	• Oscar Fonts (oscar.fonts@geomati.co)
24 Junio 2013	• Fernando González (fernando.gonzalez@fao.org)

La publicación de datos se realiza a partir de los enlaces bajo el apartado **Datos**.

Creación de un espacio de trabajo

Un espacio de trabajo es un contenedor para agrupar datos publicados. Por ejemplo, resulta útil para clasificar los datos publicados en diversas áreas temáticas. Vamos a crear un espacio de trabajo bajo el que publicar todos los datos del taller.

- En la página "Espacios de trabajo", hacer clic en "crear un nuevo espacio de trabajo".

Simplemente hay que introducir el nombre del espacio de trabajo, y un identificador URI. En nuestro caso, el URI puede ser cualquiera, pero debe ser un identificador único universal.

- Utilizaremos **capacitacion** como nombre, y **http://nfms4redd.org/capacitacion_[nombre]_[fecha]** como espacio de nombres (reemplazando [nombre] y [fecha]).

Una vez creado, nos debe aparecer en la lista de espacios de trabajo disponibles:

Creación de un almacén de datos

Una vez creado el espacio de trabajo, es posible crear capas en él. Sin embargo, GeoServer distingue dos conceptos relacionados con las capas: los almacenes de datos y las capas. El primer concepto representa la forma de encontrar los datos de la capa, datos de conexión a la base de datos, ruta en el sistema operativo donde se encuentran los ficheros, etc. El segundo concepto en cambio, contiene la información para la visualización: simbología, extensión de la capa, etc. Así pues, primeramente hay que crear un almacén de datos.

Un almacén de datos contiene la información necesaria para acceder a un determinado tipo de datos geográficos. En función del tipo de datos, será necesario crear un tipo de almacén distinto.

- En la página "Almacenes de datos", hacer clic en "Agregar nuevo almacén".

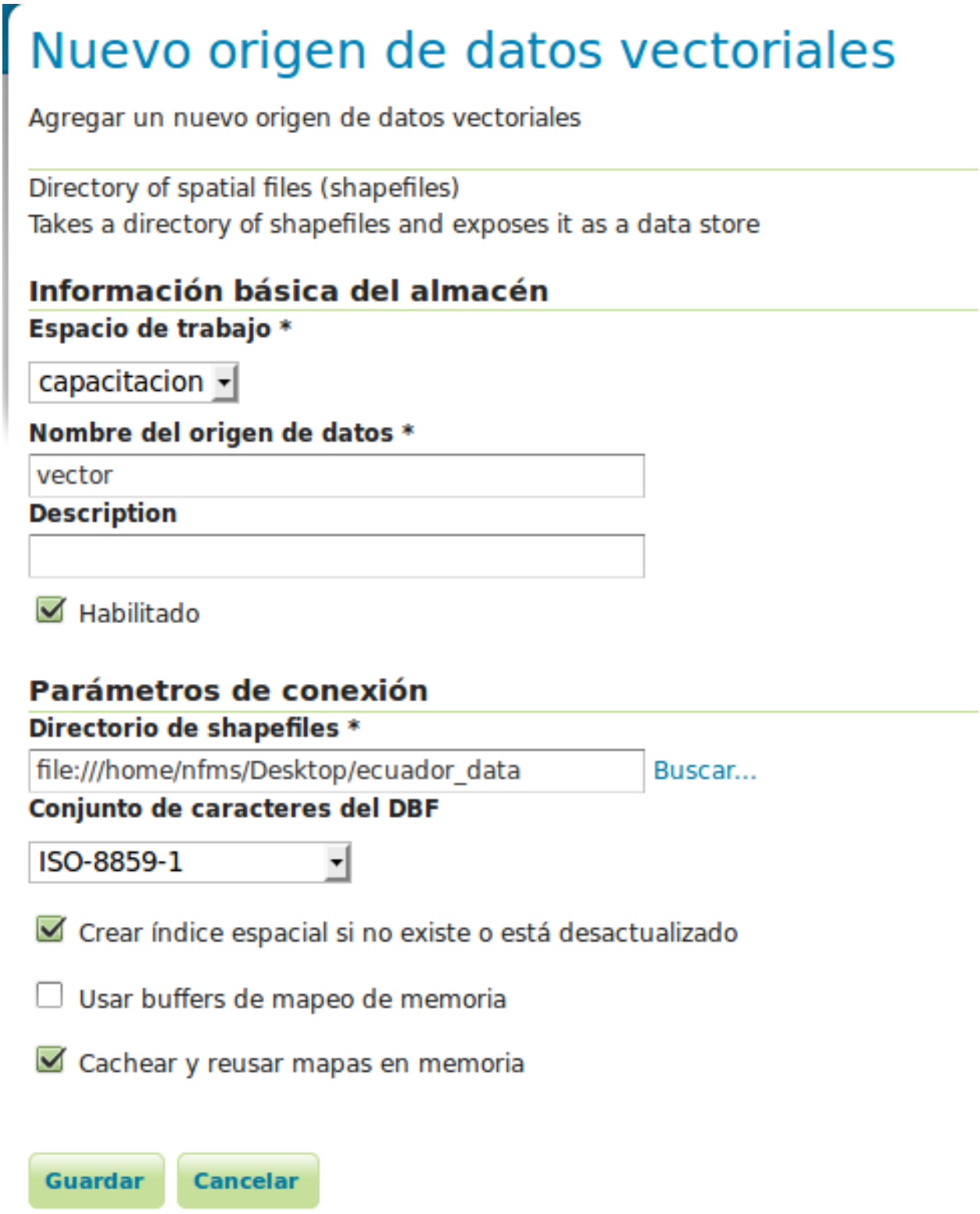
Aparece una lista de origenes de datos, separados en dos grandes bloques "Origenes de datos vectoriales" y "Origenes de datos raster". Aquí debemos escoger en función del tipo de datos que queremos acceder. Durante el taller utilizaremos los almacenes de tipo "Directory of spatial files (shapefiles)", "PostGIS", "GeoTIFF" e " ImageMosaic". Como vemos, también se puede conectar a servicios "WMS" y "WFS" remotos.

Para este ejercicio publicaremos una serie de *shapefiles*:

- Clicar en "Directory of spatial files (shapefiles)"
- Escoger **unredd** como espacio de trabajo
- Asignar el nombre **vector** al almacén de datos.
- También podemos añadir una descripción.

El dato más importante a introducir es el directorio donde están alojados los **shapefiles**.

- Hacer clic en el enlace "Buscar..." en "Directorio de Shapefiles", y navegar al directorio de datos vectoriales.
- Hacer clic en "Guardar".


Nuevo origen de datos vectoriales
Agregar un nuevo origen de datos vectoriales
Directory of spatial files (shapefiles)
Takes a directory of shapefiles and exposes it as a data store
Información básica del almacén
Espacio de trabajo *
capacitacion
Nombre del origen de datos *
vector
Description
Habilitado
Parámetros de conexión
Directorio de shapefiles *
file:///home/nfms/Desktop/ecuador_data
Buscar...
Conjunto de caracteres del DBF
ISO-8859-1
Crear índice espacial si no existe o está desactualizado
Usar buffers de mapeo de memoria
Cachear y reusar mapas en memoria
Guardar
Cancelar

Publicación de capas vectoriales

Tras la creación del almacén, GeoServer pasa automáticamente a la pantalla de publicación de capas, donde obtenemos una lista con las capas del almacén de datos recién creado que se pueden publicar y que, en este caso, se corresponden con los distintos shapefiles en el directorio. Publicaremos estas capas una a una.

- Clicar en "publicación" una a una en las capas que se quieren publicar.

Podemos escoger un nombre, título, resumen y palabras clave para describir mejor los datos a publicar.

Por ahora, nos centraremos en los campos "Sistema de referencia de coordenadas" y "Encuadres":

Sistema de referencia de coordenadas

En general, GeoServer tratará de leer el sistema de referencia en que están expresados los datos de forma automática. En ocasiones, GeoServer no puede identificarlo, y hay que declararlo manualmente, como en este caso.

Sabemos que los datos para todos los ficheros "shapefile" están espresados en el sistema de referencia **EPSG:4326**.

- En "SRS Declarado", clicar en "Buscar...", introducir "4326", y aplicar este SRS.

Encuadres

A continuación, hemos de declarar el ámbito geográfico cubierto por esta capa. Para datos que no vayan a cambiar su extensión en el futuro, se pueden calcular automáticamente a partir de los datos:

- Bajo "Encuadres", Clicar en "Calcular desde los datos" y "Calcular desde el encuadre nativo".
- Finalmente, clicar en "Guardar" para publicar la nueva capa "country".

Publicación del resto de capas

Usando el mismo procedimiento, publicaremos las otras capas del almacén:

- Clicar en "Agregar nuevo recurso" dentro de la página "Capas", y seleccionando el almacén **capacitacion:vector**.
- Repetir los pasos descritos anteriormente para cada capa.

Note: Para saber más...

- Documentación técnica NFMS: GeoServer > Adding Data to Geoserver > Adding Base Types

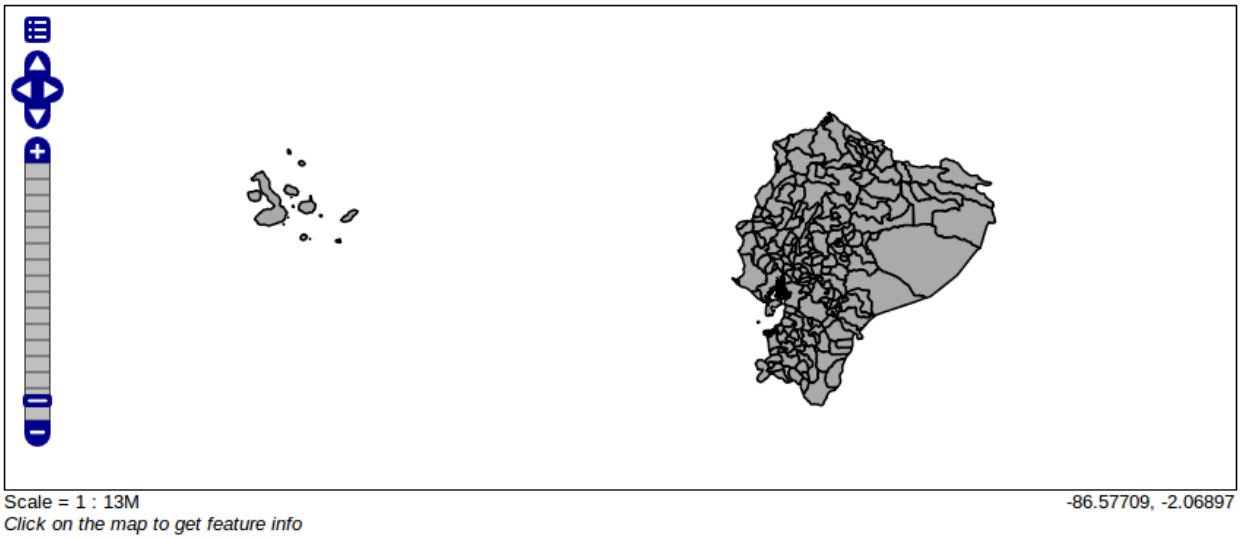
---

Previsualización de capas

Desde la página "Previsualización de capas", podemos acceder a los datos recién publicados en diversos formatos.

- Visualizar las nuevas capas utilizando el enlace "OpenLayers".

Observamos que las capas poligonales están simbolizadas en gris, mientras que las capas lineales aparecen azules. Estas son las simbolizaciones o **estilos** que GeoServer aplica por defecto. En el siguiente apartado veremos cómo crear nuestra propia simbolización.



Desde la página "Previsualización de capas" tambien tenemos acceso a los datos en muchos otros formatos, como KML, para visualizar sobre Google Earth.

Simbolización de capas vectoriales

Para agregar nuevos estilos, acceder a la página "Estilos", y clicar en "Agregar un nuevo estilo".

Los estilos se definen utilizando el formato XML estándar llamado SLD (*Styled Layer Descriptor*). Es un formato bastante prolijo, con multitud de elementos, que iremos descubriendo paso a paso. Generalmente se parte de un ejemplo ya existente, y se adapta a nuestras necesidades.

A continuación, una plantilla básica de SLD:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<StyledLayerDescriptor version="1.0.0"
    xsi:schemaLocation="http://www.opengis.net/sld StyledLayerDescriptor.xsd"
    xmlns="http://www.opengis.net/sld"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <NamedLayer>
    <Name>...nombre del estilo...</Name>
    <UserStyle>
      <FeatureTypeStyle>
        <Rule>...regla de simbolización 1...</Rule>
        <Rule>...regla de simbolización 2...</Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

A partir de esta plantilla, daremos un nombre al estilo, y añadiremos una o más reglas de simbolización.

Estilo para límites administrativos: Linea básica

Para los límites administrativos, utilizaremos una línea de color ocre, codificado como `#f0a020`, y de grosor 1 píxel.

Note: Utilidad en línea para generar códigos de colores

Así, la regla de simbolización se aplicará sobre los elementos lineales (`LineSymbolizer`), sobre los que definiremos dos parámetros para el trazo: `stroke` y `stroke-width`.

- A partir de la plantilla anterior, incluir la siguente regla de simbolización:

```
<Rule>
  <LineSymbolizer>
    <Stroke>
      <CssParameter name="stroke">#f0a020</CssParameter>
      <CssParameter name="stroke-width">1</CssParameter>
    </Stroke>
  </LineSymbolizer>
</Rule>
```

Importante: Antes de "Enviar" el estilo, es conveniente "Validar", para asegurarse que la sintaxis es la correcta, y evitar errores al aplicar el estilo a la capa.

A continuación, asignaremos este nuevo estilo a la capa.

- Desde la página "capas", seleccionar la capa a la que queremos aplicar el estilo.

- En la pestaña “Publicación”, bajo “Configuración WMS”, cambiar el estilo por defecto y seleccionar el estilo que acabamos de crear. Aparecerá una pequeña leyenda:  .
- Guardar los cambios.

Ahora, al previsualizar la capa obtendremos la nueva simbolización.

Múltiples simbolizadores: Etiquetado

Siguiendo los pasos anteriormente descritos, crearemos un nuevo estilo para una capa puntual.

En esta ocasión, simbolizaremos con un triángulo cada uno de los puntos de la capa y, adicionalmente, añadiremos una etiqueta con el nombre del punto, para lo cual utilizaremos dos simbolizadores: *PointSymbolizer* y *TextSymbolizer*.

Esta es la regla que debe aplicarse:

```
<Rule>
  <PointSymbolizer>
    <Graphic>
      <Mark>
        <WellKnownName>triangle</WellKnownName>
        <Fill>
          <CssParameter name="fill">#FF0000</CssParameter>
        </Fill>
      </Mark>
      <Size>6</Size>
    </Graphic>
  </PointSymbolizer>
  <TextSymbolizer>
    <Label>
      <ogc:PropertyName>Id</ogc:PropertyName>
    </Label>
    <Fill>
      <CssParameter name="fill">#000000</CssParameter>
    </Fill>
  </TextSymbolizer>
</Rule>
```

- Crear el nuevo estilo “etiquetado” aplicando los simbolizadores anteriores
- Validarlo
- Asignarlo a la capa
- Previsualizar la capa

Estilo para carreteras: Filtros

Para la capa de carreteras vamos a utilizar varias reglas de simbolización, dependiendo del valor del atributo `RTT_DESCRI`.

- Crear un nuevo estilo “roads”, con una simbolización de color rojo “#FF0000” y un grosor de línea de 4 píxeles.
- Añadir el siguiente filtro, justo antes del `LineSymbolizer`:

```
<ogc:Filter>
  <ogc:PropertyIsEqualTo>
    <ogc:PropertyName>RTT_DESCRI</ogc:PropertyName>
```

```
    <ogc:Literal>Primary Route</ogc:Literal>
  </ogc:PropertyIsEqualTo>
</ogc:Filter>
```

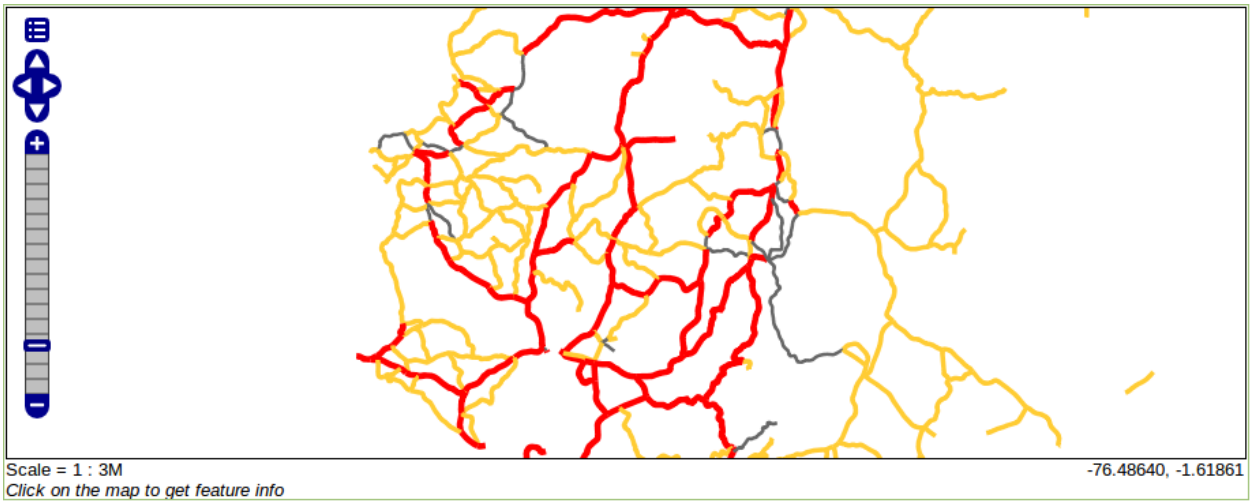
De este modo, la regla de simbolización sólo se aplicará sobre las líneas con `RTT_DESCRI` igual a `Primary Route`.

- Aplicar el nuevo estilo a la capa "roads", y previsualizar la capa.

Deberán mostrarse sólamente algunas de las carreteras, de color rojo.

A continuación, vamos a aplicar otras dos reglas, a otros dos tipos de carreteras:

- Volver a editar el estilo "roads".
- Copiar la regla de simbolización (`rule`) y pegar dos veces. Obtendremos tres reglas idénticas.
- Editar la segunda regla:
  - Cambiar el filtro para que coincida con las líneas con `RTT_DESCRI` igual a `Secondary Route`.
  - Cambiar el simbolizador para que utilice un color amarillo `#FFCC33` y un grosor de línea de 3 píxeles.
- Editar la tercera regla: * Cambiar el filtro para que coincida con las líneas con `RTT_DESCRI` igual a `Unknown`. * Cambiar el simbolizador para que utilice un color gris `#666666` y un grosor de línea de 2 píxeles.
- Validar el nuevo estilo, aplicar y previsualizar la capa "roads" de nuevo. Debería presentar un aspecto como este:



Note: Para saber más...

- Documentación técnica NFMS: GeoServer > Pretty Maps with GeoServer > Styling with SLD
- Manual de Usuario de GeoServer: Styling

CHAPTER 13

Optimización de GeoTIFF para su publicación

Note:

Fecha	Autores
1 Diciembre 2012	• Oscar Fonts (oscar.fonts@geomati.co)
24 Junio 2013	• Fernando González (fernando.gonzalez@fao.org)

Los datos raster generalmente contienen una gran cantidad de información, mucha más de la que se puede mostrar en una pantalla de una sola vez. Para que GeoServer pueda gestionar esta gran cantidad de datos de forma eficiente en diferentes situaciones, es necesario prestar atención a su optimización.

Imaginemos que queremos mostrar por pantalla una imagen raster de 10.000 x 10.000 píxeles. Puesto que la resolución de la pantalla es limitada, sólamente será capaz de mostrar, como máximo, un 1% de los píxeles totales del raster.

En lugar de leer todo el ráster, debemos incorporar mecanismos en que no sea necesario leer completamente todos los datos cada vez que visualizamos el ráster, sino sólamente a la porción de información que podemos visualizar. Esto se hace de dos modos:

- En situación de "zoom in", es conveniente poder acceder sólo a la porción de imagen que se va a mostrar, descartando el resto.
- En situación de "zoom out", es conveniente disponer de una o varias copias del ráster a resoluciones menores.

El formato interno de los ficheros GeoTIFF se puede procesar y prepararlo para estas dos situaciones.

Para ello utilizaremos las librerías GDAL desde la línea de comandos. En concreto, veremos las utilidades `gdalinfo`, `gdal_translate` y `gdaladdo`.

gdalinfo

Proporciona información sobre ficheros ráster.

- Abrir una consola (terminal).
- Acceder al directorio que contiene las imágenes landsat:

```
cd pry_workshop_data/raster/landsat/
```

- Ejecutar `gdalinfo` sobre la imagen de 1990:

```
gdalinfo landsat_1990.tif
```

Obtendremos información sobre el tamaño del fichero, el sistema de coordenadas, y la manera en que están codificadas las diferentes bandas internamente.

En concreto, observamos:

```
Band 1 Block=3069x1 Type=Byte, ColorInterp=Red
Band 2 Block=3069x1 Type=Byte, ColorInterp=Green
Band 3 Block=3069x1 Type=Byte, ColorInterp=Blue
```

Esto significa que la imagen está guardada en "tiras" de 1px de alto.

gdal_translate

Para optimizar el acceso en situaciones de "zoom in", podemos cambiar esta codificación interna para que almacene la información en bloques cuadrados de 512x512 píxeles. Ejecutar:

```
gdal_translate -co "TILED=YES" -co "BLOCKXSIZE=512" -co "BLOCKYSIZE=512" landsat_1990.
↪tif landsat_1990_tiled.tif
```

Veamos la información en la nueva imagen:

```
gdalinfo landsat_1990_tiled.tif
```

Ahora obtenemos:

```
Band 1 Block=512x512 Type=Byte, ColorInterp=Red
Band 2 Block=512x512 Type=Byte, ColorInterp=Green
Band 3 Block=512x512 Type=Byte, ColorInterp=Blue
```

gdaladdo

Para optimizar el acceso en situaciones de "zoom out", podemos añadir, internamente, una serie de imágenes a menor resolución:

```
gdaaddo landsat_1990_tiled.tif 2 4 8
```

Ejecutando de nuevo gdalinfo, observamos que para cada banda aparece esta nueva información:

```
Overviews of mask band: 1535x1535, 768x768, 384x384
```

La ventaja de utilizar la línea de comandos es que se puede crear un *script* para automatizar este procesado y aplicarlo masivamente a un gran conjunto de ficheros siempre que sea necesario.

Note: Para saber más...

- GDAL Utilities.

CHAPTER 14

GeoServer: Publicación de datos raster

Note:

Fecha	Autores
1 Diciembre 2012	• Oscar Fonts (oscar.fonts@geomati.co)
24 Junio 2013	• Fernando González (fernando.gonzalez@fao.org)

Almacen de datos GeoTIFF

- En la página "Almacenes de datos", hacer clic en "Agregar nuevo almacén".

Los datos raster para el taller se encuentran en formato GeoTIFF. A diferencia de los datos vectoriales, no tenemos un almacén de tipo "Directory of spatial files" para datos raster, así que deberemos crear un almacén distinto para cada una de las capas.

Comencemos con el primer fichero: Una clasificación de coberturas forestales.

- Escoger "GeoTIFF" bajo "Origenes de datos raster".
- En el formulario, utilizar "unredd" como espacio de nombres, y "forest_cover" como nombre de la capa. Opcionalmente, agregar una descripción.
- Clicar en "Buscar..." en "Parámetros de conexión", y navegar hasta el fichero `/home/unredd/Desktop/pry_workshop_data/raster/forest_cover_1990.tif`.
- Clicar en "Guardar".

Se presentará una nueva página con una *"lista"* de las capas a publicar: Sólo aparece un elemento, "forest_cover_1990", puesto que el almacén sólo contempla un fichero GeoTIFF.

Publicación de una capa GeoTIFF

Desde esta página,

- Clicar en "Publicación" para publicar.

Se presentará una página para rellenar los datos sobre la capa, similar a la que ya vimos para la creación de capas vectoriales.

En esta ocasión, GeoServer ha detectado automáticamente el sistema de referencia de coordenadas de la capa GeoTIFF. A diferencia de las capas vectoriales, no hará falta declarar manualmente el SRS y los encuadres, que ya tienen la información necesaria.

- Clicar en "Guardar".
- Previsualizar la nueva capa "forest_cover_1990" en OpenLayers.

En la misma página de previsualización, clicando sobre cada una de estas áreas, obtenemos una información numérica, `PALETTE_INDEX`. Se distinguen cinco valores distintos: Área sin datos (amarillo), Bosque Atlántico (verde), Bosque Chaqueño (azul), Superficie no forestal (magenta), y Masas de Agua (rojo). Esta combinación de colores de alto contraste permite distinguir claramente cada clase, pero obviamente no es la que mejor se asocia visualmente con el significado de cada categoría.

Simbolización Raster

Podemos asociar cada uno de los valores a un nuevo color que represente mejor cada clase:

Valor	Clase	Nuevo color deseado
0	Área sin datos	Transparente
1	Bosque Atlántico	Verde oscuro (#005700)
2	Bosque Chaqueño	Verde claro (#01E038)
3	Superficie no forestal	Amarillo pálido (#FFFF9C)
4	Masa de Agua	Azul (#3938FE)

A partir de esta tabla, crearemos un estilo SLD para la capa ráster.

- En la página "Estilos", "Agregar un nuevo estilo".
- Asignarle el nombre "forest_mask".
- Dejar el "Espacio de nombres" en blanco.

En lugar de escribir el SLD desde cero, podemos utilizar la opción "Copiar de un estilo existente".

- Utilizar "Copiar de un estilo existente" para cargar el estilo "raster".
- Sustituir el contenido de `RasterSymbolizer` por este otro:

```
<ColorMap type="values">
    <ColorMapEntry quantity="1" label="Bosque Atlantico" color="#005700" opacity="1"/>
    <ColorMapEntry quantity="2" label="Bosque Chaco" color="#01E038" opacity="1"/>
    <ColorMapEntry quantity="3" label="Zona no boscosa" color="#FFFF9C" opacity="1"/>
    <ColorMapEntry quantity="4" label="Masa de agua" color="#3938FE" opacity="1"/>
</ColorMap>
```

Este mapa de color asigna, a cada posible valor, un color y una etiqueta personalizada. El valor "0" (Área sin datos), al no aparecer en el mapa, se representará como transparente.

- "Validar" el nuevo SLD, "Enviar", y asignar como estilo por defecto a la capa "forest_cover_1990" (en la pestaña "Publicación").
- Previsualizar de nuevo la capa:

Publicación de un mosaico Raster temporal

Vamos a publicar una capa ráster con una imagen satelital RGB que pueda usarse como capa base de referencia.

En lugar de un solo fichero GeoTIFF, en esta ocasión disponemos de cuatro imagenes correspondientes a cuatro años distintos: 1990, 2000, 2005 y 2010.

Vamos a publicar las cuatro imágenes en como una sola capa, componiendo un "mosaico temporal".

- En la página "Almacenes de datos", hacer clic en "Agregar nuevo almacén".
- Escoger "ImageMosaic" bajo "Origenes de datos raster".
- Utilizaremos "landsat" como nombre para el almacen de datos.
- Este tipo de almacen no dispone de la utilidad "Buscar..." para indicar la localización de los datos, así que tendremos que escribirla a mano:

```
file:///home/unredd/Desktop/pry_workshop_data/raster/landsat/
```

- Clicar en "Guardar", y luego en "publicación" en la página siguiente.
- Ir a la pestaña "dimensions", para habilitar la dimensión "Time". Escoger "List" como opción de presentación.
- "Guardar" y previsualizar la capa.

Cómo se define la dimensión temporal

Si abrimos los contenidos de `pry_workshop_data/raster/landsat`, observamos los siguientes ficheros GeoTIFF, que contienen las imágenes para cada instante:

:file:`landsat_1990.tif` :file:`landsat_2000.tif` :file:`landsat_2005.tif` :file:`landsat_2010.tif`

Vemos que el nombre de todos los ficheros comienza por las mismas 8 letras `landsat_`, y que terminan con cuatro cifras indicando el año. De algún modo debemos indicar a GeoServer cómo están formados estos nombres, para que pueda extraer la información temporal a partir de ellos.

Esto se realiza mediante una serie de ficheros de *properties*:

`timeregex.properties`, cuyo contenido es:

```
regex=[0-9]{4}
```

Indica que la dimensión temporal está formada por 4 cifras.

`indexer.properties`, cuyo contenido es:

```
TimeAttribute=time
Schema=the_geom:Polygon,location:String,time:java.util.Date
PropertyCollectors=TimestampFileNameExtractorSPI[timeregex](time)
```

Indica que la marca temporal será obtenida aplicando *timeregex*, y se almacenará en un índice como atributo *time*.

Note: Para saber más...

- Documentación técnica NFMS: GeoServer > Advanced Raster data preparation and configuration > Adding an Image Mosaic to GeoServer
- Página sobre expresiones regulares.

Consumo del servicio temporal

Ahora que tenemos una capa temporal publicada podemos pasar a formar a consumirla con algún cliente estándar. Desafortunadamente gvSIG no es capaz de consumir la capa y QGIS no tiene soporte para la dimensión temporal. Sin embargo, es posible obtener las imágenes en los distintos instantes símplemente utilizando el navegador web. Para ello, las llamadas que se hacen deben incluir el parámetro *TIME*, como en los siguientes ejemplos:

```
http://168.202.48.83/geoserver/ows?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetMap&BBOX=-13.
↪910569,12.090411,5.395932,32.233551&
        TIME=2000&CRS=EPSG:4326&WIDTH=923&HEIGHT=885&LAYERS=capacitacion:test&STYLES=&
↪FORMAT=image/png&DPI=96&TRANSPARENT=TRUE

http://168.202.48.83/geoserver/ows?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetMap&BBOX=-13.
↪910569,12.090411,5.395932,32.233551&
        TIME=2005&CRS=EPSG:4326&WIDTH=923&HEIGHT=885&LAYERS=capacitacion:test&STYLES=&
↪FORMAT=image/png&DPI=96&TRANSPARENT=TRUE

http://168.202.48.83/geoserver/ows?SERVICE=WMS&VERSION=1.3.0&REQUEST=GetMap&BBOX=-13.
↪910569,12.090411,5.395932,32.233551&
        TIME=2010&CRS=EPSG:4326&WIDTH=923&HEIGHT=885&LAYERS=capacitacion:test&STYLES=&
↪FORMAT=image/png&DPI=96&TRANSPARENT=TRUE
```

Note: Para saber más...

- Documentación técnica NFMS: GeoServer > Advanced Raster data preparation and configuration > Processing with GDAL

CHAPTER 15

Pregeneración de teselas en GeoWebCache

Note:

Fecha	Autores
20 Enero 2014	• Fernando González (fernando.gonzalez@fao.org)

Pregeneración de teselas

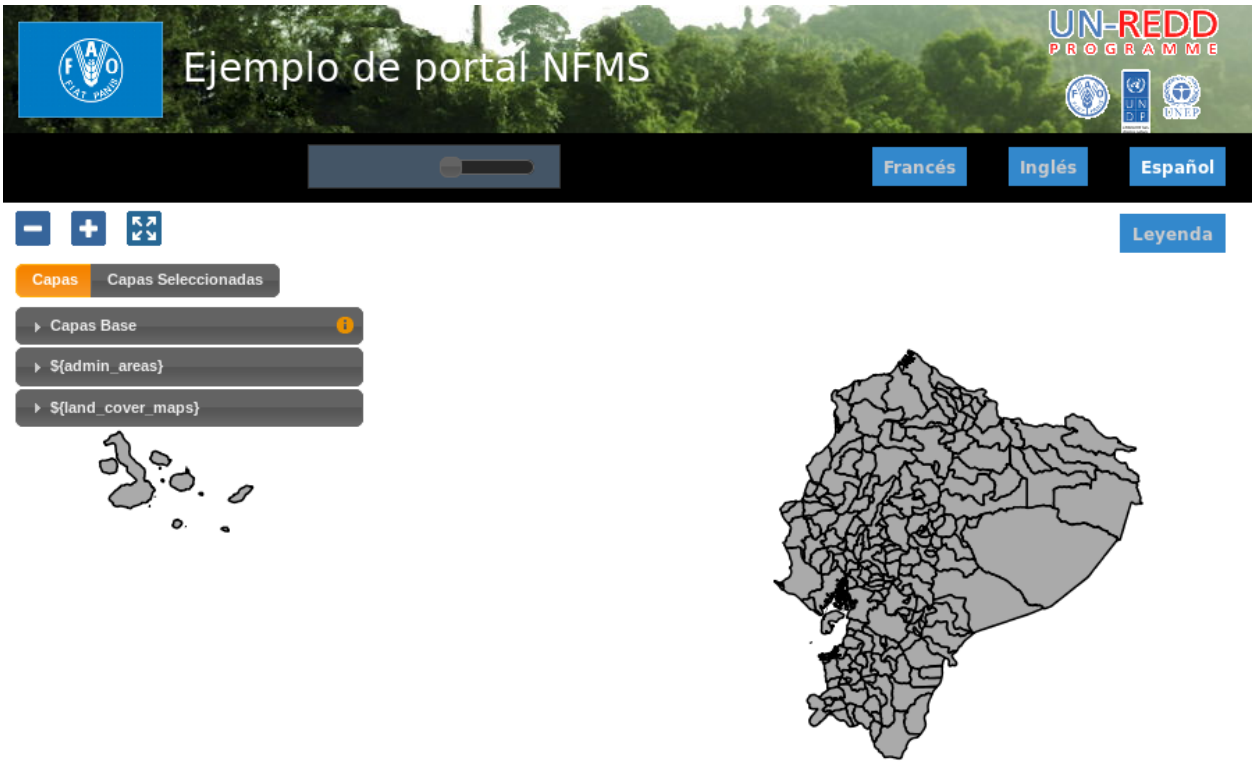
Hay dos maneras de generar las teselas de GeoWebCache. La primera forma consiste en generar progresivamente mientras se visualiza el mapa. En este caso, las teselas se almacenan en caché a medida que son solicitadas a través de la navegación por mapa (por ejemplo, en OpenLayers). La primera vez que se solicita una tesela, esta se servirá a la misma velocidad que en el caso de una solicitud WMS estándar, ya que ésta se ha de generar y guardar en la caché. La veces siguientes, la tesela ya estará generada y almacenada en la caché por lo que el tiempo de respuesta de la petición será mucho menor. La principal ventaja de este método es que no requiere ningún procesamiento previo y que sólo los datos que ha solicitado se almacenan en caché, ahorrando potencialmente espacio en disco. La desventaja de este método es que el mapa se visualiza con velocidad muy variable, lo que reduce la calidad de la experiencia del usuario.

La otra forma de rellenar la caché es mediante pregeneración. La pregeneración es el proceso en el que se generan y almacenan en caché todas las teselas deseadas. Cuando este proceso se usa inteligentemente, la experiencia de usuario mejora en gran medida ya que las teselas se encuentran todas pregeneradas y el usuario no tiene que sufrir tiempos de espera largos. La desventaja de este proceso es que la pregeneración puede ser muy costosa en tiempo y en espacio en disco.

En la práctica se utiliza una combinación de ambos métodos, pregenerando a ciertos niveles de zoom (o en determinadas zonas de los niveles de zoom) y dejando las teselas menos utilizadas sin pregenerar.

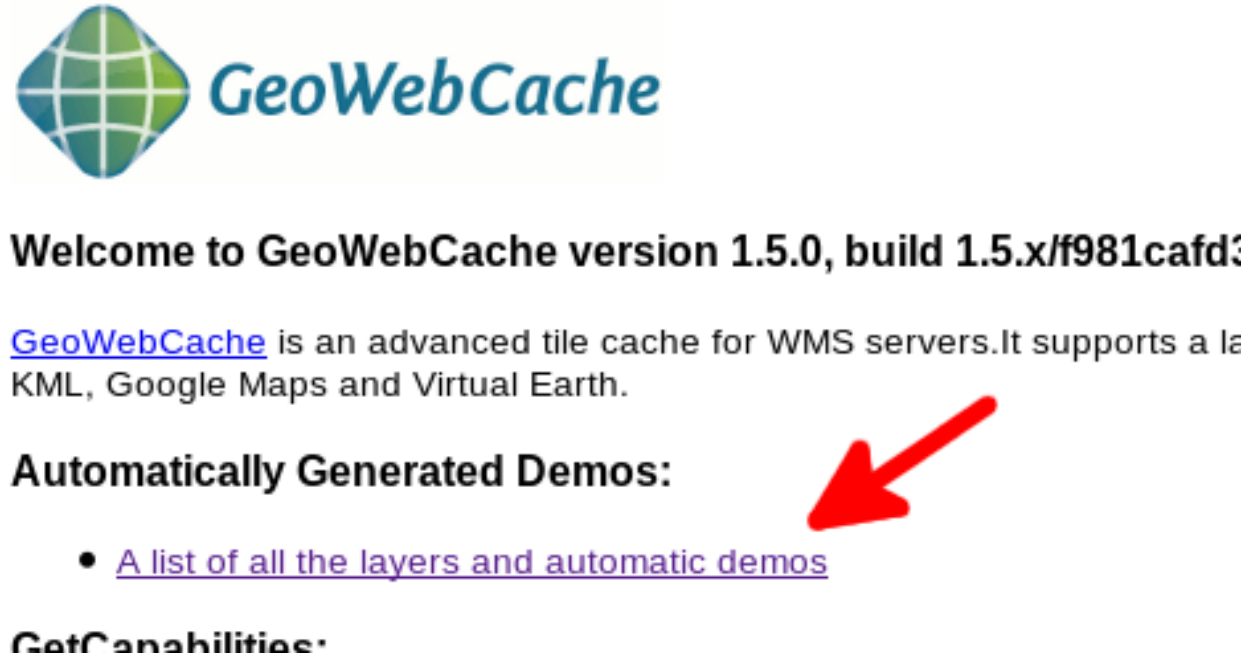
Ejemplo: pregeneración de unidades administrativas de Ecuador

En el ejemplo que nos ocupa vamos a pregenerar las teselas para el mapa de Ecuador que se puede ver en la figura, en el sistema de referencia EPSG:900913:

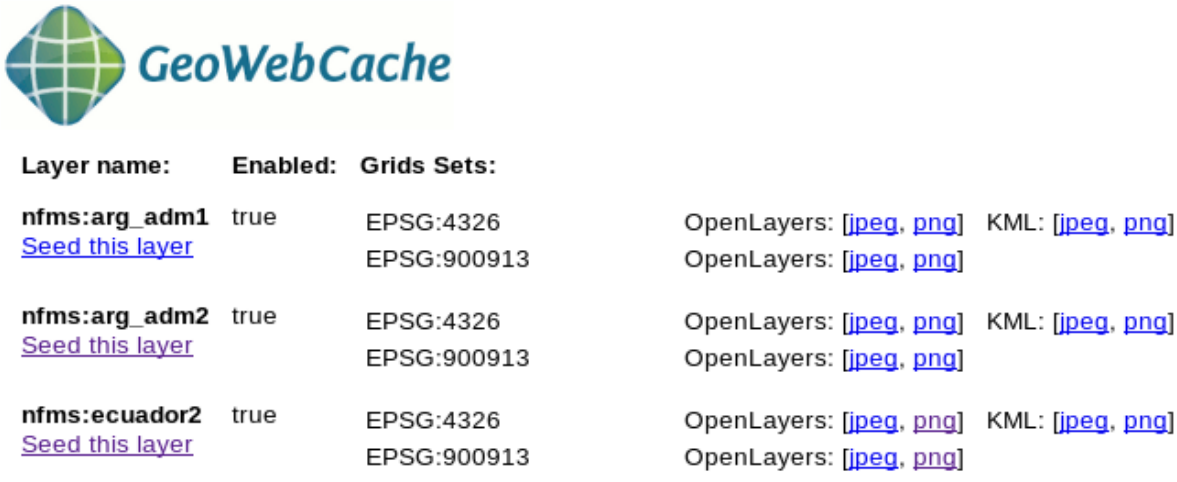


Para ello tenemos que acceder a la URL `geoserver/gwc` dentro de nuestro servidor, por ejemplo `http://127.0.0.1:8080/geoserver/gwc/` si estamos accediendo desde la máquina local.

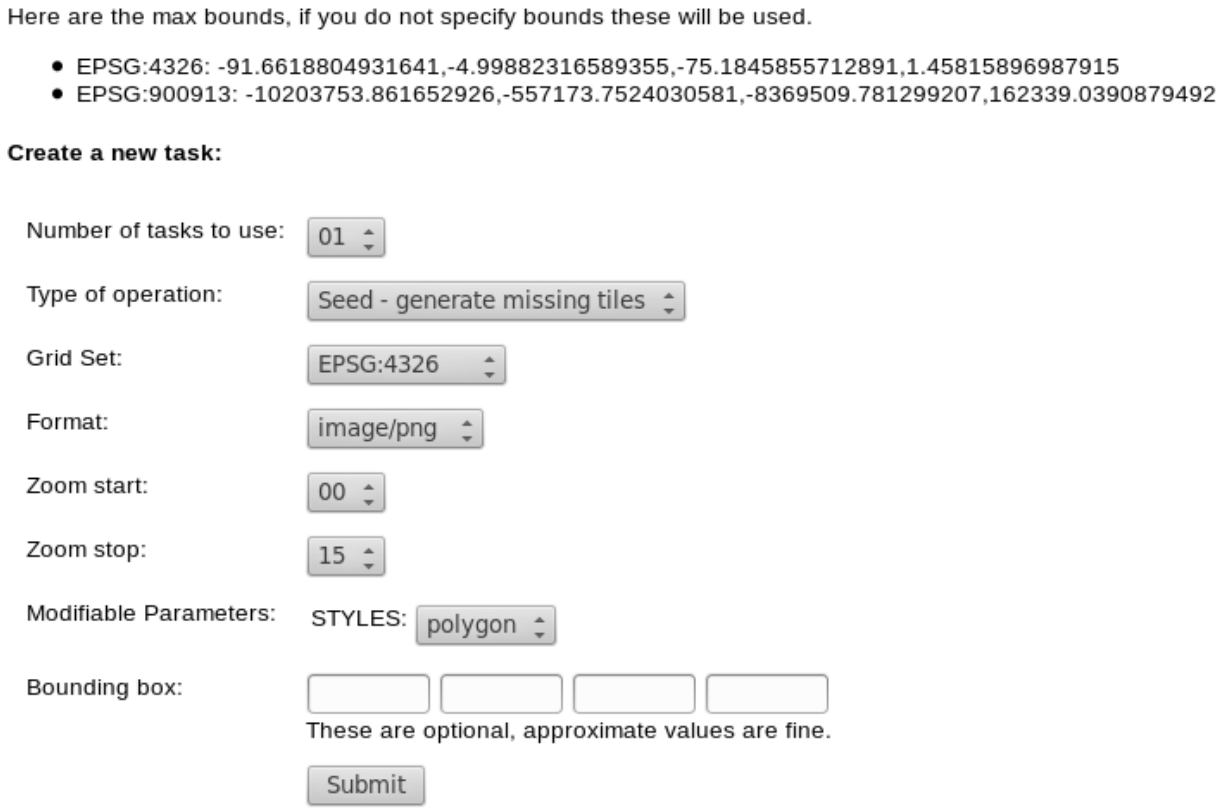
Nos aparecerá la página principal de GeoWebCache, y deberemos acceder al enlace donde se listan todas las capas:



Una vez seguido el enlace nos aparecerá una página con las capas existentes en la caché. Podemos observar cómo debajo de cada capa hay un enlace con el texto "Seed this layer", que nos permite pregenerar la caché.



Siguiendo dicho enlace llegaremos a la página que nos permite pregenerar la caché en un formulario al final de la misma:

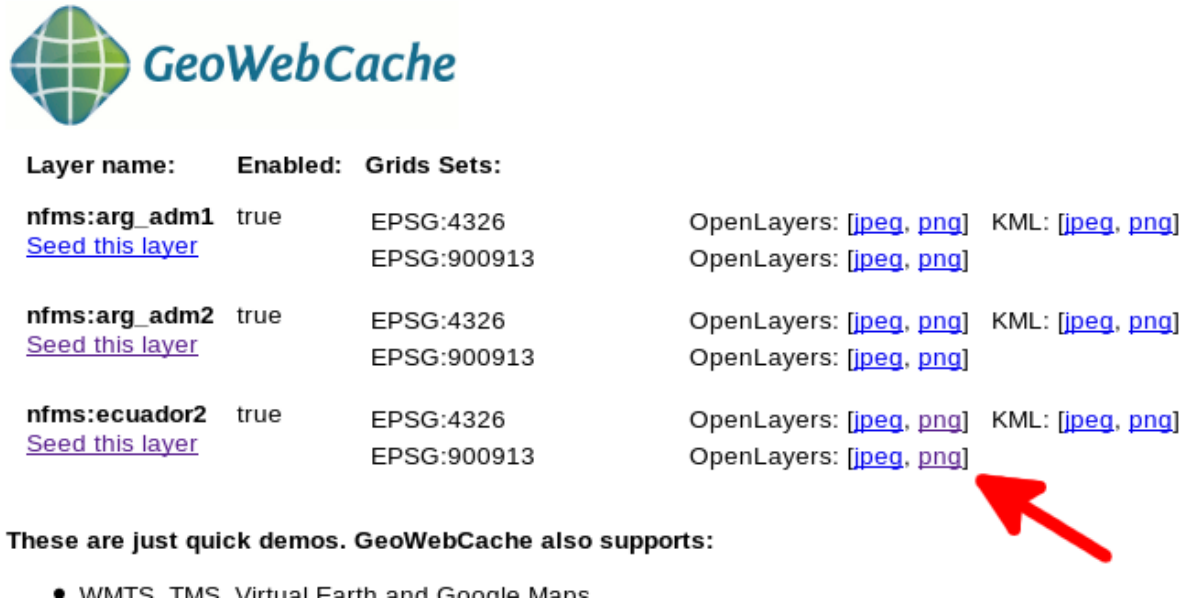


En él podemos observar que se nos piden distintos parámetros, de los que destacamos:

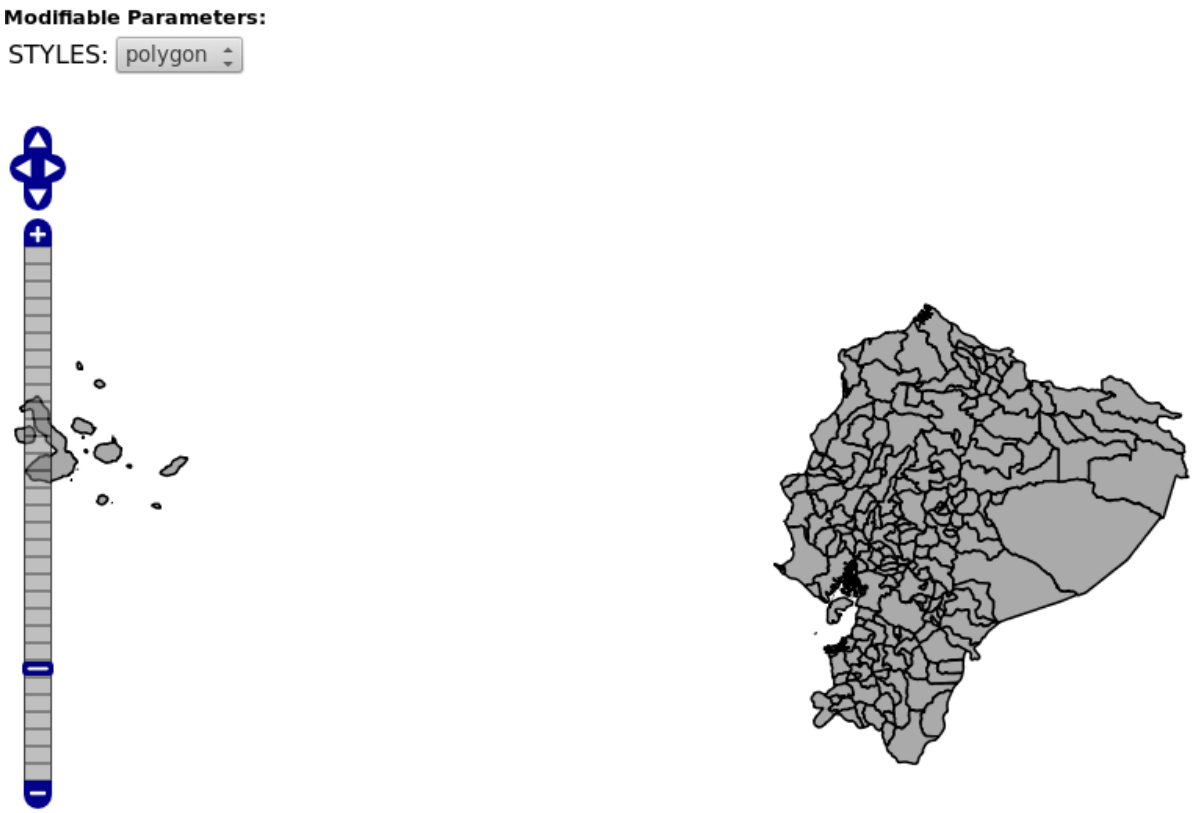
- Type of operation (Tipo de operación): Generalmente seleccionaremos siempre "Seed", o sea, pregeneración.
- Grid Set: En este punto seleccionaremos el sistema de referencia de nuestro mapa.
- Formato: Muy importante seleccionar el formato de imagen que estamos usando en las llamadas de nuestro mapa al servidor. En nuestro caso image/png
- Zoom start y Zoom stop: Esto son los niveles de zoom para los cuales se generarán las teselas.
- STYLES: El estilo con el que se generarán las teselas.
- Bounding box: Extensión de nuestros datos que define las teselas que se generarán. Por defecto se toma la extensión de la capa.

Casi todas las opciones son sencillas de seleccionar. Sin embargo, para los niveles de zoom nos puede surgir una duda ¿a qué escala corresponde cada nivel de zoom?

Si bien la respuesta exacta es complicada de obtener, es bastante sencillo hacerse una idea intuitiva. Para ello tenemos que volver a la página que lista las capas y ver que a la derecha aparecen demos con OpenLayers para los distintos sistemas de referencia y para cada formato de imagen:

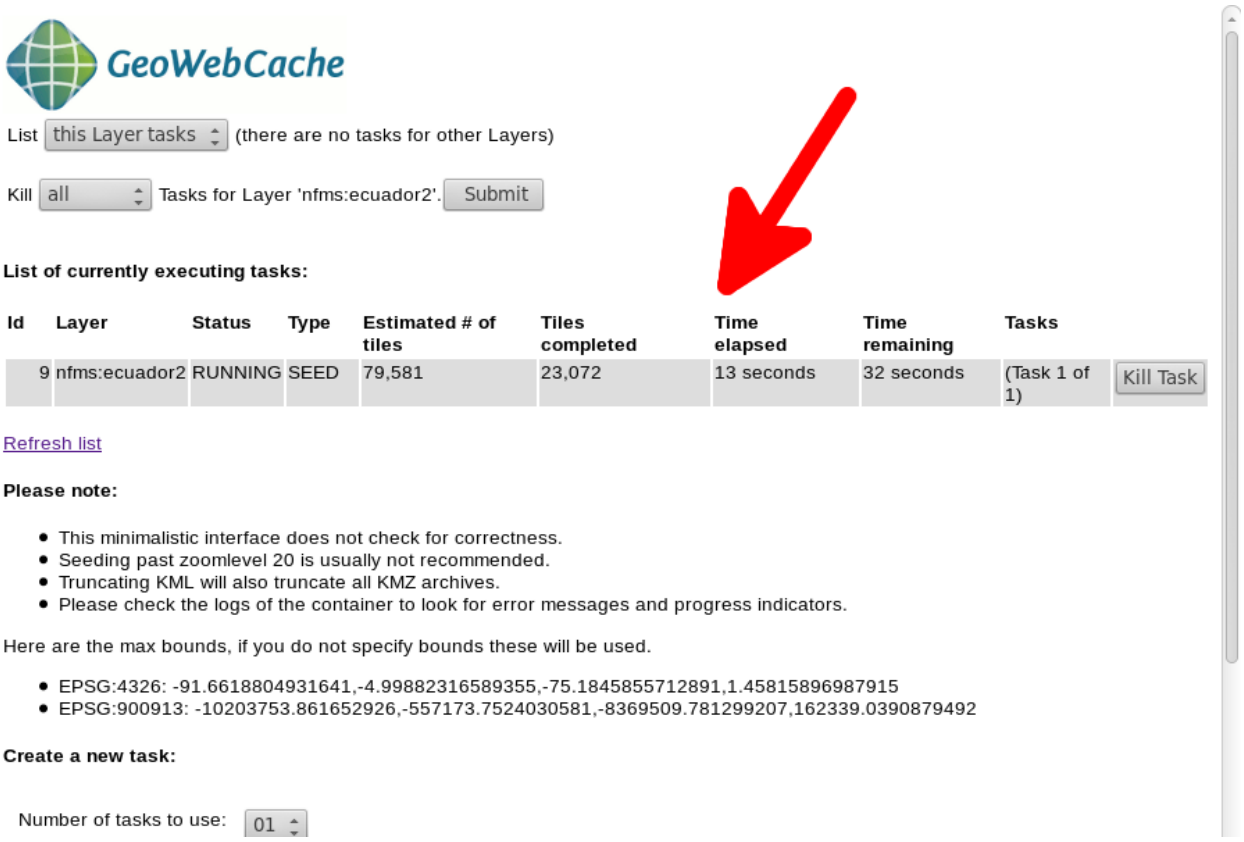


Pinchando en el que nos interesa EPSG:900913 y png podemos acceder a una página de demostración en la que aparece una barra de zoom con los niveles de GeoWebCache. Podemos navegar e identificar los niveles a los que queremos acceder, teniendo en cuenta que el más alejado corresponde con el nivel 0. Por ejemplo, la capa aparece dibujada inicialmente en el nivel de zoom 6:



Si nos movemos al nivel 12, podemos observar que tal vez sea el nivel máximo al cual queramos visualizar la imagen:

Con lo cual ya tenemos los parámetros necesarios para pregenerar la caché. Volviendo al formulario, podemos especificar los parámetros y, tras pulsar en el botón "Submit", se iniciará una tarea que se reporta más arriba en la página:



Una vez generada, cada vez que el usuario se mueva en un mapa entre los niveles 6 y 12 de zoom se obtendrán las imágenes desde la caché, por lo que la navegación será muy rápida, mientras que a distintos niveles de zoom la velocidad decrecerá porque el servidor tendrá que dibujar las teselas.

El resultado de la caché se guarda internamente en el directorio de datos de GeoServer. Así, si dicho directorio es `/var/geoserver/data`, la caché se almacena en `/var/geoserver/data/gwc`, en un subdirectorio para cada capa. El resultado de la pregerenación de la caché ocupa 87Mb en el directorio `/var/geoserver/data/gwc/nfms_ecuador2`.

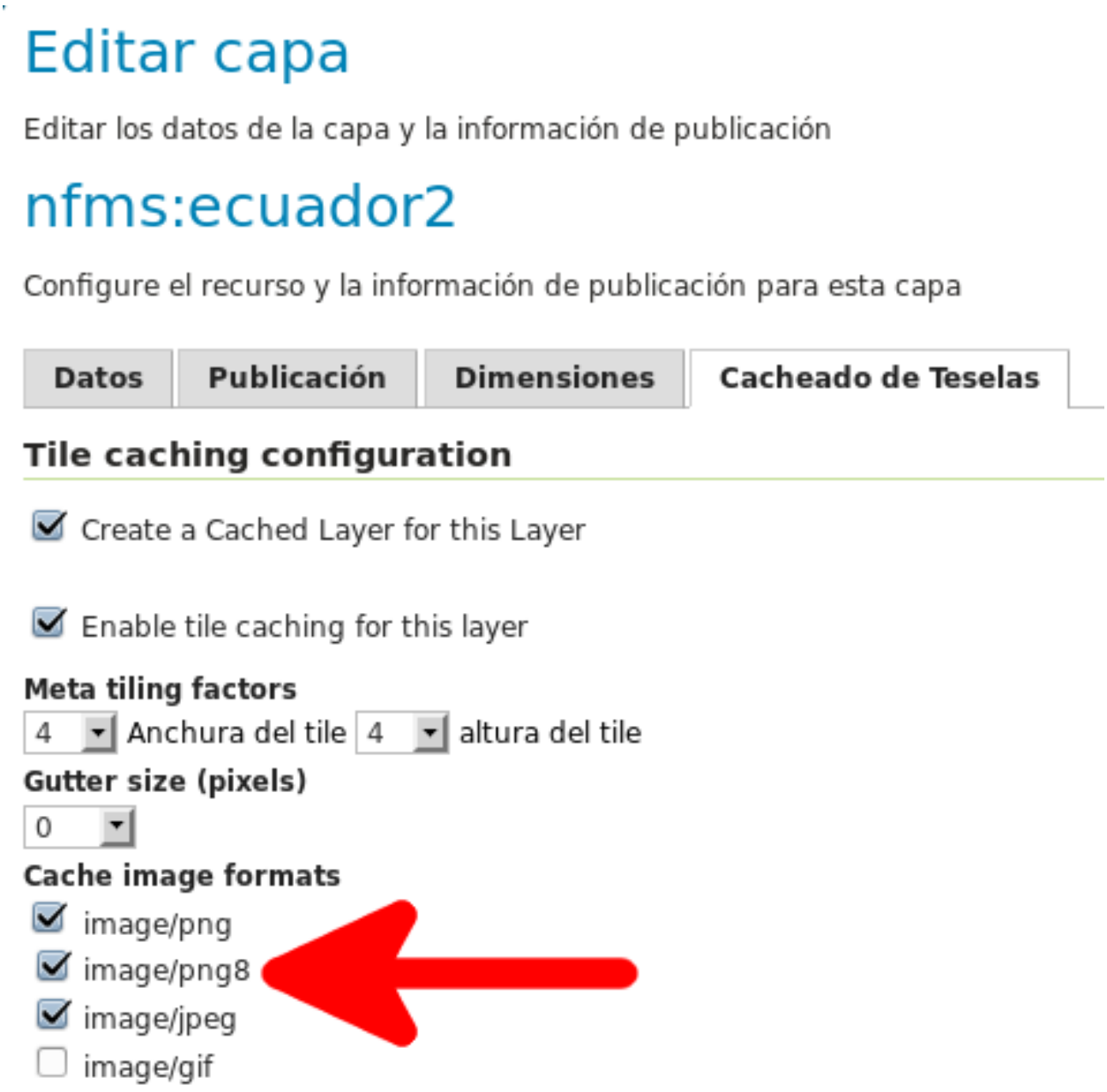
En la práctica, es bastante barato generar los primeros niveles, de 0 a 6, ya que al ser escalas muy pequeñas (zooms muy lejanos), con pocas teselas se cubre rápidamente la extensión de la capa. Sin embargo, es a escalas más grandes cuando cuesta cada vez más tiempo la generación de la caché para el nivel de zoom y más espacio almacenarlo. Por ejemplo, la generación del nivel 13, nos hace pasar de 87Mb a 319Mb. Y si observamos dentro del directorio `/var/geoserver/data/gwc/nfms_ecuador2` podemos ver que cada nivel de zoom casi cuadriplica el tamaño del nivel anterior:

```
12K     EPSG_900913_00
16K     EPSG_900913_01
24K     EPSG_900913_02
28K     EPSG_900913_03
32K     EPSG_900913_04
64K     EPSG_900913_05
104K    EPSG_900913_06
224K    EPSG_900913_07
608K    EPSG_900913_08
1.8M    EPSG_900913_09
5.2M    EPSG_900913_10
18M     EPSG_900913_11
62M     EPSG_900913_12
```

```
233M    EPSG_900913_13
```

Entre las prácticas que reducen el coste temporal y espacial de la caché está la de evitar la pregeneración de zonas sin interés. En el caso de Ecuador, es obvio que a partir del nivel de zoom 7 u 8, no tiene sentido pregenerar las teselas que corresponden al agua entre Islas Galápagos y el continente. Esto se puede regular con la opción `bounding box` del formulario de pregeneración.

Por último, en el caso de teselas cuyo renderizado no contenga más de 256 colores, es posible utilizar el formato PNG8, que ocupa algo menos que el formato PNG, lo cual se traduce en menor espacio para almacenar las teselas de la caché así como en menor tiempo de transmisión entre el servior y el mapa cliente. Para ello, hay que habilitar dicho formato en GeoServer, yendo a la pestaña "Cacheado de Teselas" de la capa y habilitando el formato en la sección "Cache image formats":



Tras guardar los cambios, seremos capaces de seleccionar dicho formato en el formulario de pregeneración. Como comparativa, el resultado de pregenerar los niveles de 6 a 12 es 78Mb, un 10% menor que los 87Mb correspondientes

al formato PNG.

Note: En caso de aplicar esta optimización, hay que asegurarse de que el cliente pide las teselas en el formato `image/png8`.

CHAPTER 16

Portal: Configuración inicial

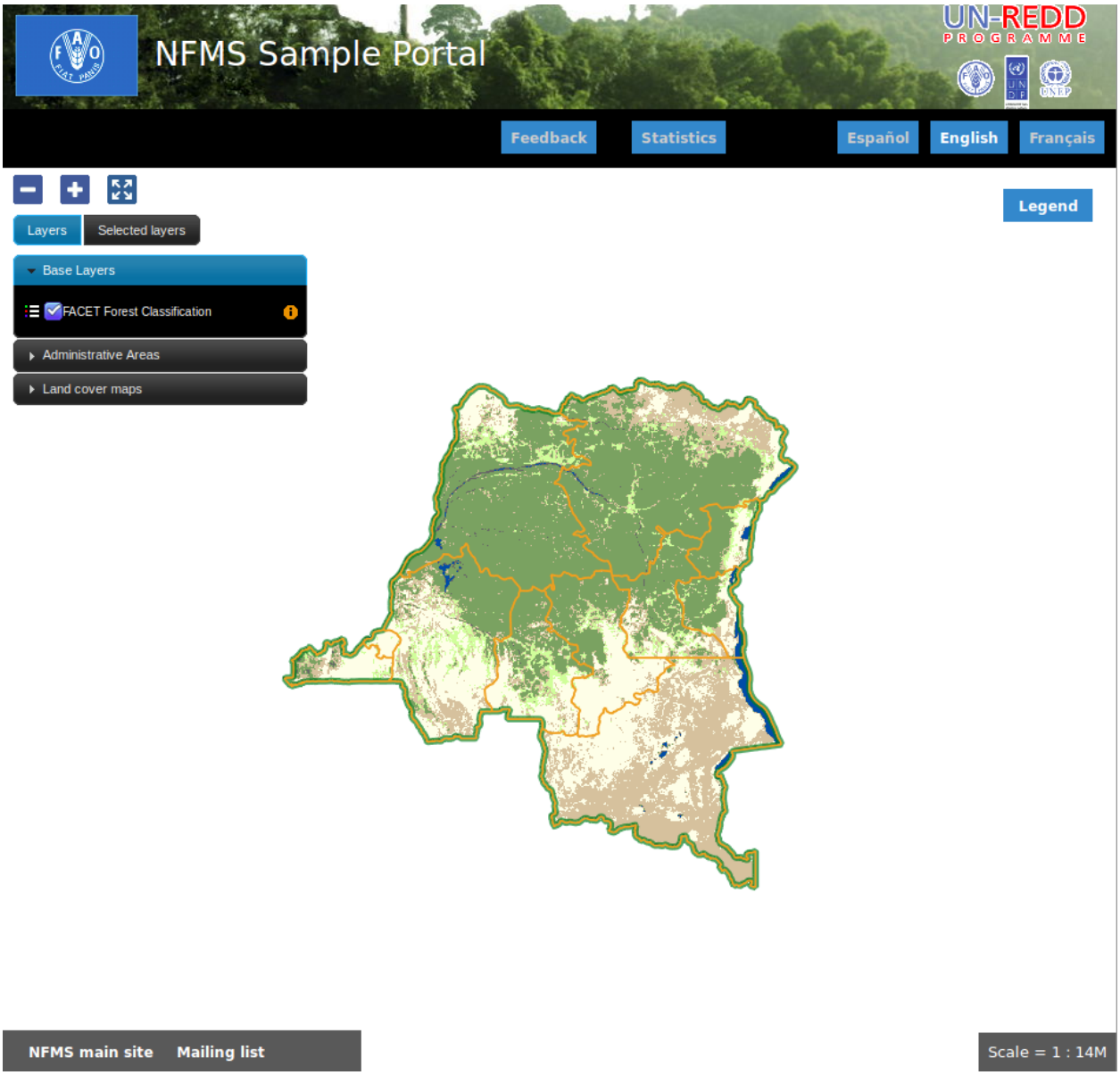
Note:

Fecha	Autores
1 Noviembre 2012	• Stefano Giaccio (Stefano.Giaccio@fao.org)
1 Diciembre 2012	• Oscar Fonts (oscar.fonts@geomati.co)
24 Junio 2013	• Fernando González (fernando.gonzalez@fao.org)

El Portal de NFMS es la aplicación pública de la plataforma, utilizada para la visualización de los recursos cartográficos y estadísticos.

El portal es personalizable a las necesidades de cada país, tanto en su aspecto como en los datos mostrados. La aplicación viene con una configuración por defecto que puede usarse como punto de partida.

En nuestro caso, una vez instalado el portal se obtendrá el siguiente aspecto inicial:



El Portal es accesible en:

http://localhost/portal/

El directorio de configuración

El directorio de configuración (`/var/portal`) tiene la siguiente estructura:

```
/var/portal
  |- portal.properties
  |- layers.json
  |- messages/
  |- modules/
  \- static/
      |- overrides.css
```

```
    |- img/
    \- loc/
        |- en/
        |   |- documents/
        |   |- html/
        |   \- images/
        \- es
            |- documents/
            |- html/
            \- images/
```

Los ficheros principales son:

- `portal.properties`, contiene parámetros generales del portal, como por ejemplo los idiomas soportados.
- `layers.json`, probablemente el fichero más importante, contiene la configuración de las capas de datos a mostrar.
- `messages/` contiene los textos de la aplicación traducidos a varios idiomas.
- `modules/` permite añadir modulos Javascript a la aplicación.
- `static/` contiene recursos estáticos.
- `static/loc`: recursos clasificados por idioma
- `static/overrides.css`: última hoja CSS cargada, ideal para sobreescribir otros estilos

portal.properties

- languages = {“en”: “English”, “fr”: “Franu00e7ais”, “es”: “Espau00f1ol”}

  Elemento JSON con los idiomas que soporta la aplicación

- languages.default = en

  Idioma por defecto.

- db-schema=portal_redd

  Nombre del esquema donde están las tablas necesarias para las distintas funcionalidades como estadísticas y feedback

- layers.rootFolder=/tmp

  Raíz de la configuración de estadísticas (experimental)

- info.layerUrl=http://demo1.geo-solutions.it/diss_geoserver/gwc/service/wms

  URL de las capas `queryable`

- info.queryUrl=http://demo1.geo-solutions.it/diss_geoserver/wms

  URL de las capas `queryable` contra la que hacer la petición GetFeatureInfo

- map.centerLonLat=24, -4

  Longitud y latitud del centro inicial del mapa

- map.initialZoomLevel=5

  Nivel de zoom inicial

layers.json

Define la estructura de capas del proyecto. Consiste en un elemento JSON con cuatro propiedades:

```
{
        "default-server" : "http://demo1.geo-solutions.it",
        "wmsLayers" : [],
        "portalLayers" : [],
        "groups" : []
}
```

- `default-server` define el servidor que se usará como base en caso de que la URL de las capas no incluyan servidor. Ver atributo `baseUrl` más abajo.
- `wmsLayers` define las capas que tendrá el mapa. El orden en el que estas capas aparecen en este array define el orden de las capas en el dibujado del mapa. Cada capa consistirá en un elemento que puede ser de tres tipos. El tipo por defecto es WMS y tiene las siguientes propiedades:
  - `id`: Identificador de la capa.
  - `type`: Tipo de la capa. Puede ser `wms`, `osm` (para Open Street Map) o `gmaps` (para Google Maps). Por defecto es `wms`.
  - `legend`: Nombre del fichero imagen con la leyenda de la capa. Estos ficheros se acceden en `static/loc/{lang}/images`. También es posible poner la cadena de carácteres `auto` y el portal intentará obtener la imagen automáticamente de GeoServer usando la petición `GetLegendGraphic` de WMS.
  - `sourceLink`: URL del proveedor de los datos.
  - `sourceLabel`: Texto con el que presentar el enlace especificado en `sourceLink`.

  En función del tipo de la capa se especificarán además otras propiedades. Para `wms`:
  - `baseUrl`: URL del servidor WMS que sirve la capa. Si se especifica una URL sin servidor, por ejemplo `/diss_geoserver/gwc/service/wms`, se usará `default-server`.
  - `wmsName`: Nombre de la capa en el servicio WMS.
  - `imageFormat`: Formato de imagen a utilizar en las llamadas WMS.
  - `queryType`: Protocolo usado para la herramienta de información: `wfs` o `wms`. En caso de ser `wfs` los siguientes parámetros son obligatorios: `queryGeomFieldName`, `queryFieldNames`, `queryFieldAliases` y `queryTimeField` (en caso de ser una capa temporal). En caso de ser `wms` no hay ningún parámetro adicional obligatorio, por lo que una capa que quiera usar WMS para la herramienta de información puede configurarse sólo con: `"queryType": "wms"`. El único requisito para capas con `"queryType": "wms"` es que el servidor codifique en EPSG:4326 la geometría de la respuesta al `GetFeatureInfo`; en caso contrario el objeto consultado no se podrá localizar en el mapa mediante zoom y resaltado. Si la capa no tiene un parámetro `queryType` la capa no será consultable.
  - `queryUrl`: URL base a utilizar en la petición de información. Base del servidor WMS o WFS a utilizar (según `queryType`). Si no se especifica se toma `baseUrl`.
  - `queryGeomFieldName`: Obligatorio en el caso de `"queryType": "wfs"`. El nombre del campo geométrico. Típicamente `geom`, `the_geom`, `geometry`, etc.
  - `queryFieldNames`: Obligatorio en el caso de `"queryType": "wfs"`. Nombres de los campos que se quieren obtener en la petición de info.
  - `queryFieldAliases`: Obligatorio en el caso de especificar `queryFieldNames`. Aliases de los campos especificados en `queryFieldNames`.

- `queryTimeFieldName`: Obligatorio en el caso de `"queryType": "wfs"` si la capa tiene varias instancias temporales. Sin uso en el caso de `"queryType": "wms"`.
- `queryHighlightBounds`: Sólo para el caso `"queryType": "wms"`. Si se desea resaltar solo el rectángulo que encuadra la geometría de los objetos consultados (`true`) o toda la geometría (`false`). Por defecto es `false`. En los casos en los que la geometría es muy grande puede ser conveniente ponerlo a `true` para que el proceso de resaltado sea más rápido.

Por ejemplo:

```
{
        "wmsLayers" : [
                {
                        "id" : "provinces",
                        "baseUrl" : "http://demo1.geo-solutions.it/diss_
↪geoserver/wms",
                        "wmsName" : "unredd:drc_provinces",
                        "imageFormat" : "image/png8",
                        "visible" : true,
                        "sourceLink" : "http://www.wri.org/publication/
↪interactive-forest-atlas-democratic-republic-of-congo",
                        "sourceLabel" : "WRI",
                        "queryable" : true
                }
        ],
        ...
}
```

Para `osm` (Open Street Map):

- `osmUrls`: lista de las URLs de los tiles. Usando `${x}`, `${y}` y `${z}` como variables.

Por ejemplo:

```
{
        "wmsLayers" : [
                {
                        "id" : "openstreetmap",
                        "type" : "osm",
                        "osmUrls" : [
                                "http://a.tile.openstreetmap.org/${z}/${x}
↪/${y}.png",
                                "http://b.tile.openstreetmap.org/${z}/${x}
↪/${y}.png",
                                "http://c.tile.openstreetmap.org/${z}/${x}
↪/${y}.png"
                        ]
                }
        ],
        ...
}
```

Para `gmaps` (Google Maps):

- `gmaps-type`: Tipo de capa Google: `ROADMAP`, `SATELLITE`, `HYBRID` o `TERRAIN`.

Por ejemplo:

```
{
        "wmsLayers" : [
                {
                        "id" : "google-maps",
                        "type" : "gmaps",
                        "gmaps-type" : "SATELLITE"
                }
        ],
        ...
}
```

- `portalLayers` define las capas que aparecen visibles al usuario. Una `portalLayer` puede contener varias `wmsLayers`. Cada `portalLayer` puede contener los siguientes elementos:
  - `id`: Identificador de la capa.
  - `label`: Texto con el nombre de la capa a usar en el portal. Si se especifica entre ${ }, se intentará obtener la traducción de los ficheros `.properties` existentes en el directorio `messages` del directorio de configuración del portal.
  - `infoFile`: Nombre del fichero HTML con información sobre la capa. El fichero se accede en `static/loc/{lang}/html`. En la interfaz gráfica se representa con un botón de información al lado del nombre de la capa.
  - `infoLink`: URL con la información sobre la capa. Igual que `infoFile` pero especificando una ruta absoluta. `infoFile` tiene preferencia sobre `infoLink`, por lo que si se define el primero, `infoLink` se ignorará.
  - `inlineLegendUrl`: URL con una imagen pequeña que situar al lado del nombre de la capa en el árbol de capas. También es posible poner la cadena de carácteres `auto` y el portal intentará obtener la imagen automáticamente de GeoServer usando la petición `GetLegendGraphic` de WMS.
  - `active`: Si la capa está inicialmente visible o no.
  - `layers`: Array con los identificadores de las `wmsLayers` a las que se accede a través de esta capa.
  - `timeInstances`: Instantes de tiempo en ISO8601 separados por comas.
  - `timeStyles`: Nombres de los estilos a utilizar para cada instancia temporal. Cada estilo se corresponde con aquella instancia temporal que ocupa la misma posición en la lista. Si no se especifica este parámetro se utilizará el estilo por defecto para todos los estilos.
  - `date-format`: Formato de la fecha para cada capa. Según la librería *Moment <http://momentjs.com/docs/#/displaying>*_. Por ejemplo: `DD-MM-YYYY`. Por defecto sólo el año (`YYYY`).
  - `feedback`: En el caso de que la herramienta de feedback esté instalada, si se quiere o no que la capa aparezca en dicha herramienta para permitir al usuario hacer comentarios sobre la capa.

  Por ejemplo:

```
{
        "wmsLayers" : [
                {
                        "id" : "wms_provinces",
                        "baseUrl" : "http://demo1.geo-solutions.it/diss_
↪geoserver/wms",
                        "wmsName" : "unredd:drc_provinces",
                        "imageFormat" : "image/png8",
```

```
                        "visible" : true,
                        "sourceLink" : "http://www.wri.org/publication/
↪interactive-forest-atlas-democratic-republic-of-congo",
                        "sourceLabel" : "WRI",
                        "queryable" : true
                }
        ],
        "portalLayers" : [
                {
                        "id" : "provinces",
                        "active" : true,
                        "infoFile" : "provinces_def.html",
                        "label" : "${provinces}",
                        "layers" : [ "wms_provinces" ],
                        "inlineLegendUrl" : "http://demo1.geo-solutions.
↪it/diss_geoserver/wms?REQUEST=GetLegendGraphic&VERSION=1.0.0&
↪FORMAT=image/png&WIDTH=20&HEIGHT=20&LAYER=unredd:drc_provinces&
↪TRANSPARENT=true",
                        "timeInstances" : "2007-03-01T00:00,2008-05-
↪11T00:00,2005-03-01T00:00",
                        "timeStyles" : "style2007,style2008,style2005",
                        "date-format" : "DD-MM-YYYY"
                }
        ],
        ...
}
```

- `groups` define la estructura final de las capas en el árbol de capas de la aplicación. Cada elemento de `groups` contiene:
  - `id`: Identificador del grupo.
  - `label`: Igual que en `portalLayer`
  - `infoFile`: Igual que en `portalLayer`
  - `infoLink`: Igual que en `portalLayer`
  - `items`. Array con los identificadores de otros grupos (con la misma estructura que este elemento; recursivo) o capas (`portalLayer`).

Por ejemplo:

```
{
        "wmsLayers" : [
                {
                        "id" : "wms_provinces",
                        "baseUrl" : "http://demo1.geo-solutions.it/diss_
↪geoserver/wms",
                        "wmsName" : "unredd:drc_provinces",
                        "imageFormat" : "image/png8",
                        "visible" : true,
                        "sourceLink" : "http://www.wri.org/publication/
↪interactive-forest-atlas-democratic-republic-of-congo",
                        "sourceLabel" : "WRI",
                        "queryable" : true,
                        "wmsTime" : "2007-03-01T00:00,2008-05-11T00:00,
↪2005-03-01T00:00"
                }
        ],
```

```
        "portalLayers" : [
                {
                        "id" : "provinces",
                        "active" : true,
                        "infoFile" : "provinces_def.html",
                        "label" : "${provinces}",
                        "layers" : [ "wms_provinces" ],
                        "inlineLegendUrl" : "http://demo1.geo-solutions.
↪it/diss_geoserver/wms?REQUEST=GetLegendGraphic&VERSION=1.0.0&
↪FORMAT=image/png&WIDTH=20&HEIGHT=20&LAYER=unredd:drc_provinces&
↪TRANSPARENT=true"
                }
        ],
        "groups" : [
                {
                        "id" : "base",
                        "label" : "${base_layers}",
                        "infoFile": "base_layers.html",
                        "items" : ["provinces"]
                }
        ]
}
```

Adaptación del aspecto gráfico

Cabecera de página

Veamos cómo modificar la imagen de fondo, bandera y título de la cabecera del portal:

Partiendo del PORTAL_CONFIG_DIR (generalmente en /var/portal):

- **Imagen de fondo**: Se encuentra en *static/img/right.jpg*. Sustituir este fichero por otro de igual nombre y formato (jpeg), de 92 píxeles de alto. El ancho puede variar, aunque se recomienda que sea tan ancho como sea posible, hasta los 1920 px de una pantalla de alta definición. Para conseguir un mejor efecto junto con la bandera, se recomienda rellenar de contenido (logotipos, fotografía) la parte más a la derecha de la imagen, hasta un máximo de 500 px. Utilizar un color de fondo liso para el resto de la imagen, que ocupe toda la franja de la izquierda, y que se corresponda con el color de fondo de la bandera.
- **Bandera**: Se encuentra en *static/img/left.jpg*. Sustituir este fichero por otro de igual nombre y formato (jpeg), de 92 píxeles de alto. El ancho puede variar, aunque se recomienda alrededor de los 200 px. Utilizar un color de fondo liso, correspondiente con la parte izquierda de la imagen de fondo, para dar una sensación de continuidad.
- **Título**: Se encuentra definido en los ficheros de mensajes, directorio *messages*, ficheros de nombre *messages_<lang>.properties*. Buscar la propiedad "title" en cada uno de los ficheros de idioma.

Favicon

Se conoce como *favicon* al icono que se muestra en el navegador en la barra de direcciones. Para personalizar el *favicon* del portal, basta con copiar la imagen en el directorio `static/img`. El nombre de la imagen sólo puede ser `favicon.ico` o `favicon.png`.

Estilos predefinidos (CSS)

En ciertos casos se requiere modificar los estilos que vienen predefinidos para OpenLayers, jQuery o cualquier otro. En estos casos, en lugar de modificar los estilos directamente en el fichero que se encuentra en `/var/tomcat/webapps/portal`, se ha de crear un nuevo fichero `overrides.css` en el directorio `/var/portal/static/css` que contenga las reglas CSS que se desean modificar.

De esta manera, tendrán preferencia las reglas que se escriban en `overrides.css` frente a cualquier otra que se encuentre en `/var/tomcat/webapps/portal`.

Además, cuando se despliegue una actualización del portal en Tomcat, el fichero `overrides.css` no se modificará, manteniendo así la personalización.

Soporte multiidioma

En los casos anteriores vemos algunas cadenas de texto entre los símbolos `${` y `}`. Estos elementos son sustituidos por mensajes de texto traducidos a cada idioma.

En el directorio `messages` contamos con un fichero `messages.properties` que contiene los mensajes por defecto. Son los textos que se usarán en caso de no encontrar mensajes traducidos a una lengua específica. Los ficheros para los distintos idiomas soportados llevan el código del idioma al final del nombre, según la nomenclatura ISO 639-1 de dos letras.

Como ejercicio:

- Buscar el elemento *title* en `messages_es.properties`.

Otro ejercicio:

- Traducir el texto del enlace añadido en `footer.tpl`

De la misma manera, Para añadir un nuevo idioma (por ejemplo, el guaraní):

- Editar `portal.properties` y añadir el elemento `"gn": "Guaraní"` a la propiedad `languages`:

```
languages = {"gn": "Guaraní", "es": "Español", "en": "English"}
```

- Copiar el fichero `messages_es.properties` con el nuevo nombre `messages_gn.properties`.
- Traducir los textos en `messages_gn.properties`.
- Reiniciar la aplicación para aplicar los cambios. Desde la linea de comandos:

```
sudo service tomcat6 restart
```

Configuración de una nueva capa

La definición de las capas a mostrar en el Portal se encuentra en el fichero `layers.json`.

Contiene la información para asociar los elementos de la interfaz de usuario (panel con la lista de capas en la parte izquierda de la página) con las capas WMS publicadas en GeoServer, personalizar las leyendas, y definir cuáles de las capas son interrogables. También clasifica las capas por grupos.

El formato utilizado para este fichero de configuración es JSON (JavaScript Object Notation), que es un formato para la representación de datos. Está fuera del objetivo de esta guía el aprendizaje de JSON, pero se exponen a continuación algunas nociones básicas:

- Los valores en JSON pueden ser: números, cadenas de carácteres, booleanos, arrays, objetos y el valor nulo. Por ejemplo: 13, “hola mundo”, true, [12, 5, 2], {“id”:3}.
- Los objetos están delimitados por llaves ({ }) y contienen una serie de pares atributo-valor separados por comas. Los pares atributo/valor consisten en un nombre de propiedad entrecomillado, dos puntos y el valor. Por ejemplo podemos tener el siguiente elemento:

```
{
        "id":12,
        "nombre":"paco",
        "edad":55
}
```

  o incluso un elemento dentro de otro:

```
{
        "empresa":"zapatos smith",
        "propietario":{
                "id":12,
                "nombre":"john smith",
                "edad":55
        },
        "pais":"Argentina"
}
```

- Los arrays especifican sus valores entre corchetes ([]) y separados por comas.

```
[1, 2, 3, 4, 5]
```

```
[
        {
                "id":12,
                "nombre":"john smith",
                "edad":34
        },
        {
                "id":12,
                "nombre":"sarah smith",
                "edad":22
        },
        {
                "id":12,
                "nombre":"Clark Kent",
                "edad":43
        }
]
```

Note: Recursos JSON

- Introducción al formato JSON
- Validador de JSON
- Validador en línea de comandos: python -mjson.tool <fichero.json>

El fichero `layers.json` contiene tres secciones:

- `wmsLayers`
- `portalLayers`
- `groups`

En este apartado vamos a realizar dos ejercicios:

- En primer lugar, vamos a añadir la capa de límites administrativos al grupo existente de "admin_areas".
- En segundo lugar, añadiremos la capa "roads" en un nuevo grupo de capas.

Conexión WMS

Cada "wmsLayer" se corresponde con una de las capas publicadas en GeoServer, y describe la manera de conectarse al servidor para obtener los datos:

TODO link the reference and complete the reference if necessary

```
"wmsLayers": [
   {
    "id": "limites_administrativos",
    "baseUrl": "http://172.16.250.131/geoserver/gwc/service/wms",
    "wmsName": "capacitacion:limites_administrativos",
    "imageFormat": "image/png",
    "visible": true
  }
],
```

- Es posible copiar y pegar un elemento existente y reemplazar :

  - el nuevo "id" será distinto a todos los otros, por ejemplo: "limites_administrativos".
  - el nuevo "wmsName" será "capacitacion:limites_administrativos" (el nombre de la capa publicada en GeoServer).
  - la baseUrl debe apuntar al servidor geoserver donde hemos cargado la capa.

Capas del portal

Cada "portalLayer" representa una capa en el árbol de capas del portal y por tanto añade nuevos datos necesarios para mostrar la información en la interfaz gráfica.

```
"portalLayers": [
  {
    "id": "limites_administrativos",
    "active": true,
    "label": "${limites_administrativos}",
    "infoFile": "limites_def.html",
    "layers": ["country"],
    "inlineLegendUrl": "http://172.16.250.131/geoserver/wms?REQUEST=GetLegendGraphic&
↪VERSION=1.0.0&FORMAT=image/png&WIDTH=20&HEIGHT=20&LAYER=unredd:country&
↪TRANSPARENT=true"
  }
],
```

- Añadir un nuevo objeto en "context", de igual estructura y valores que "country", excepto los siguientes cambios:
  - el nuevo "id" será "regions".
  - como "label" se utilizará "${limites_administrativos}". De nuevo, esta etiqueta de sintaxis ${...} será sustituida por un texto en el idioma que corresponda, según los contenidos de "messages". Es la etiqueta que se mostrará en la interfaz gráfica.
  - en "infoFile" pondremos "administrative_boundaries_def.html". Esto creará un enlace a un documento con información sobre los datos (localizado en `static/loc/<idioma>/html/`).
  - en "layers" pondremos ["limites_administrativos"], haciendo referencia al nuevo *layer*.
  - en "inlineLegendUrl" estableceremos el parámetro LAYER así *LAYER=capacitacion:limites_administrativos*. Esto generará una imagen con la leyenda.

Grupos

Los "Groups" son una estructura recursiva (multinivel) para agrupar visualmente las capas en el panel. El "group" de primer nivel construye cada uno de los grupos de capas en forma de persiana desplegable, conteniendo una lista de "items" que hacen referencia a los contextos definidos anteriormente.

```
"groups" : [
        {
                "id" : "admin",
                "label" : "${admin_areas}",
                "items" : [ "countryBoundaries", "provinces" ]
        },
        ...
]
```

Nótese que en la propiedad "items", se hace referencia a las "portalLayers" definidas anteriormente. También, es posible dentro de dicha propiedad, añadir varios subgrupos de manera que las capas contenidas en éstos se visualicen dentro de una misma pestaña, pero agrupados visualmente bajo un título.

```
"groups" : [
        {
                "id" : "admin",
                "label" : "${admin_areas}",
                "items" : [
                        {
                                "id" : "admin1",
                                "label" : "Nacional",
                                "items": ["limite_nacional"]
                        }, {
                                "id" : "admin2",
                                "label" : "Regional",
                                "items": [ "provincias" ]
                        }
                ]
        },
        ...
]
```

- Añadir un nuevo elemento *{ "context": "limites_administrativos" }* a continuación de *{ "context": "country" }*. Esto incluirá la capa en el grupo de áreas administrativas.
- Finalmente, utilizar un validador JSON, para comprobar que la sintaxis del nuevo `layers.json` es correcta, y recargar la página.

Posición inicial del mapa y prefijo capas

Antes de añadir la capa de carreteras vamos a proceder a configurar la posición inicial del mapa. Para ello tenemos que editar el fichero `static/custom.js` y que contiene al principio del todo una declaración con los valores que nos interesa cambiar:

```
UNREDD.maxExtent = new OpenLayers.Bounds(-20037508, -20037508, 20037508, 20037508);
UNREDD.restrictedExtent = new OpenLayers.Bounds(-20037508, -20037508, 20037508,␣
↪20037508);
UNREDD.maxResolution = 4891.969809375;
UNREDD.mapCenter = new OpenLayers.LonLat(-9334782,-101119);
UNREDD.defaultZoomLevel = 0;

UNREDD.wmsServers = [
    "http://demo1.geo-solutions.it",
    "http://incuweb84-33-51-16.serverclienti.com"
];
```

Para la posición central del mapa tendremos que modificar el valor *UNREDD.mapCenter* y poner la coordenada central en Google Mercator (EPSG:900913 o EPSG:3857), que es el sistema de referencia que se usa en la aplicación web.

- Obtener la coordenada central del mapa en el sistema de coordenadas usado en el portal.

Para regular el nivel de zoom inicial es posible cambiar el valor *UNREDD.defaultZoomLevel*. Cuanto mayor es el nivel de zoom, más cercano es el zoom inicial.

Por último, es posible configurar en `UNREDD.wmsServers` una o más URLs correspondientes a nuestro servidor de manera que en el fichero `layers.json` basete especificar los atributos *baseUrl* con URLs relativas comenzando por

el carácter "/". Estas URLs se componen prefijando los servidores especificados en el valor *UNREDD.wmsServers*. Por otra parte, si el servidor tiene más de una dirección, es conveniente especificarlas todas, ya que algunos navegadores limitan la cantidad de peticiones que se pueden hacer simultáneamente a un servidor y éste sería un método para sobrepasar ese límite.

Ejercicio:

- Poner el servidor local en *UNREDD.wmsServers* y poner todas las capas del servidor como relativas.

Configuración de un nuevo grupo de capas

Repetiremos el ejercicio anterior para añadir la capa de ciudades, teniendo en cuenta que:

- Para el nuevo "layer", usaremos el id "ciudades" y la capa wms "capacitacion:ciudades". Además, añadiremos un nuevo atributo *"legend": "ciudades.png"* para mostrar la leyenda de la capa. Este atributo hace referencia a una imagen localizada en `static/loc/<idioma>/images/`.
- En el nuevo "context", será más sencillo, sólo contendrá los tres elementos *"id": "roads", "label": "${ciudades}", "layers": ["ciudades"]*.
- En "contextGroups", crearemos un nuevo grupo llamado "otros", con esta sintaxis:

```
{
  "group": {
    "label": "${other}",
    "items": [
        { "context": "roads" }
    ]
}
```

- Tras validar el JSON, y recargar la página, obtendremos la capa de carreteras bajo el nuevo grupo "Otros".

CHAPTER 17

Portal: Configuración de capas raster

Note:

Fecha	Autores
1 Noviembre 2012	• Stefano Giaccio (Stefano.Giaccio@fao.org)
1 Diciembre 2012	• Oscar Fonts (oscar.fonts@geomati.co)
24 Junio 2013	• Fernando González (fernando.gonzalez@fao.org)

El proceso de configuración de capas raster en el portal es exactamente el mismo que para las capas vectoriales, ya que tanto unas como otras se exponen a través de la interfaz WMS, que no hace diferencia entre raster y vector.

Creación de capas temporales

Es posible añadir capas con dimensión temporal de la manera habitual añadiendo un nuevo elemento en “layers” con las siguientes características:

- Identificador: “landsat”
- Etiqueta: “${landsat}”
- Nombre de la capa landsat en GeoServer.
- Formato de imagen: “image/jpeg”

Novedad: Puesto que esta capa dispone de dimensión temporal, debemos indicar al Portal los tiempos para los que se dispone de información. Esto lo realizaremos mediante el atributo "wmsTime" con los años 1990, 2000, 2005 y 2010. Así pues:

- Añadir un nuevo atributo "wmsTime": "1990-01-01T00:00:00.000Z,2000-01-01T00:00:00.000Z,2005-01-01T00:00:00.000Z,2010-01-01T00:00:00.000Z"
- Añadir el contexto para esta capa, con las siguientes características:
  - Identificador: "landsat"
  - Etiqueta: "${landsat}"
  - Capa: ["landsat"]
  - Inicialmente desactivada al cargar la página
- Añadir un nuevo grupo con las siguientes características:
  - Etiqueta: "${base_layers}"
  - Un único contexto: "landsat"

Al finalizar, el Portal deberá contener los nuevos elementos. Se deberá apreciar cómo, además del nuevo grupo de Capas Base, ha aparecido un nuevo control que permite seleccionar el año de visualización.

- Comprobar que, al desplazar el control de tiempo, se actualizan los contenidos de la capa "landsat".

CHAPTER 18

Portal: Configuración de capas temporales

Note:

Fecha	Autores
16 Marzo 2015	• Fernando González (fernando.gonzalez@fao.org)

La configuración de capas temporales consta de tres etapas:

1. Preparación de los datos
2. Publicación en GeoServer
3. Configuración de la capa en el portal

Los dos primeros pasos dependen del tipo de datos que se quiere publicar, vectorial o raster.

Preparación y publicación de capas vectoriales

La preparación consistirá en introducir una columna de tipo fecha con la fecha para la cual ese registro es válido. Esto se puede realizar para shapefiles o para bases de datos PostGIS indistintamente, pero el campo ha de ser de tipo fecha.

La publicación en GeoServer se hace exactamente igual al de cualquier otra capa y la diferencia es que una vez la capa está publicada, hay que ir a la pestaña "Dimensiones" de la capa en cuestión y habilitar el checkbox para Tiempo. En el combo que aparece como "Atributo" hay que seleccionar el campo fecha añadido anteriormente, y para "Presentación" hay que seleccionar "List". Una vez esta configuración se guarda, la capa ya está preparada para servirse en distintas instancias temporales.

Configuración de la capa en el portal

La configuración de la capa en el portal sólo implica añadir un atributo "timeInstances" al elemento `portalLayer`:

```
{
                "id" : "mascara_forestal",
                "label" : "Máscara forestal",
                "layers" : [ "forest_mask_wms" ],
                "timeInstances" : "2000-01-01T00:00:00.000Z,2005-01-01T00:00:00.000Z,
↪2010-01-01T00:00:00.000Z"
}
```

El formato del atributo es una lista de fechas separadas por comas en la que cada fecha tiene el siguiente formato: yyyy-MM-ddTHH:mm:ssZ, donde:

- yyyy es el año
- MM es el mes

- dd es el día del mes
- T separa fecha y hora
- HH es la hora en formato 24h
- mm son los minutos
- ss son los segundos
- Z indica el final de la fecha

Warning: Anteriormente este parámetro se llamába `wmsTime` y se configuraba en los elementos `wmsLayer` por lo que es posible encontrar algún fichero con el formato antiguo que esté configurado de esta manera.

CHAPTER 19

Resolviendo el problema de las etiquetas

Descripción del problema

Tenemos una capa de provincias, de la que disponemos de las geometrías del polígono y de un campo con el nombre de la provincia. Se quiere publicar en un servidor de mapas, de manera que se muestre el polígono de la provincia y en un punto interior el nombre de la misma.

Para ello, mediante el uso de GeoServer, publicamos la capa y le aplicamos el siguiente estilo:

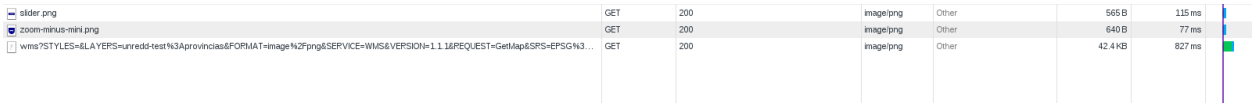
```
  <?xml version="1.0" encoding="UTF-8"?>
  <StyledLayerDescriptor version="1.0.0" xmlns="http://www.opengis.net/sld" xmlns:ogc=
↪"http://www.opengis.net/ogc"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/
↪XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/sld http://schemas.opengis.net/sld/1.0.0/
↪StyledLayerDescriptor.xsd">
<!--  simbologia_adm_prov_4326 -->
<!-- http://geo2.ambiente.gob.ar/geoserver/bosques_umsef_db/wms?service=WMS&version=1.
↪1.0&request=GetMap&layers=bosques_umsef_db:limites_provinciales&styles=&bbox=-73.
↪566302817,-52.394802778,-53.637962552,-21.777951173&width=333&height=512&
↪srs=EPSG:4326&format=application/openlayers&TIME=1999 -->

<NamedLayer>
  <Name>Blue lake</Name>
  <UserStyle>
    <Title>Blue lake</Title>
    <Abstract>A blue fill, solid black outline style</Abstract>
    <FeatureTypeStyle>
      <Rule>
        <Name>name</Name>
        <PolygonSymbolizer>
          <Fill>
            <CssParameter name="fill">
              <ogc:Literal>#ffffff</ogc:Literal>
            </CssParameter>
```

```
            <CssParameter name="fill">
              <ogc:Literal>1.0</ogc:Literal>
            </CssParameter>
          </Fill>
        </PolygonSymbolizer>
             <LineSymbolizer>
          <Stroke>
            <CssParameter name="stroke">#000000</CssParameter>
              <CssParameter name="stroke-width">1</CssParameter>
          </Stroke>
             </LineSymbolizer>
        <TextSymbolizer>
         <Geometry>
  <ogc:Function name="interiorPoint">
    <ogc:PropertyName>the_geom</ogc:PropertyName>
  </ogc:Function>
    </Geometry>
            <Label>
               <ogc:PropertyName>NAME_1</ogc:PropertyName>
             </Label>
             <LabelPlacement>
               <PointPlacement>
                 <AnchorPoint>
                   <AnchorPointX>0.5</AnchorPointX>
                   <AnchorPointY>0.5</AnchorPointY>
                 </AnchorPoint>
               </PointPlacement>
             </LabelPlacement>
             <Halo>
               <Radius>
                 <ogc:Literal>2</ogc:Literal>
               </Radius>
               <Fill>
                 <CssParameter name="fill">#FFFFFF</CssParameter>
               </Fill>
             </Halo>
             <Fill>
               <CssParameter name="fill">#000000</CssParameter>
             </Fill>
             <VendorOption name="goodnessOfFit">0</VendorOption>
             <VendorOption name="conflictResolution">false</VendorOption>
           </TextSymbolizer>
               </Rule>
       </FeatureTypeStyle>
     </UserStyle>
   </NamedLayer>
 </StyledLayerDescriptor>
```

De esta manera, publicada la capa, podremos observar a través de la previsualización de capas, que el aspecto de la capa es el siguiente:

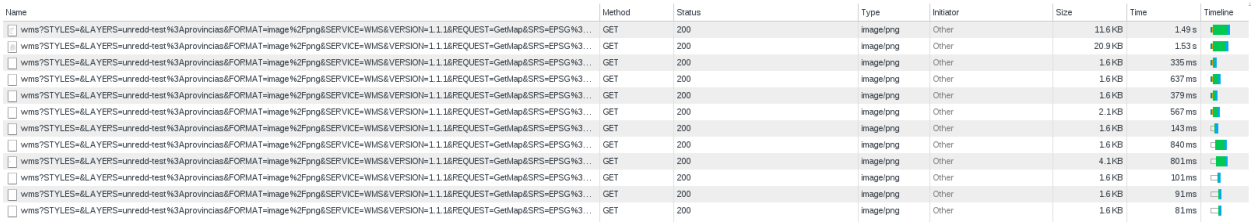

Jujuy
Salta
Formosa
Chaco
Misiones
Tucumán
Catamarca
Santiago del Estero
Corrientes
La Rioja
San Juan
Santa Fe
Córdoba
Entre Ríos
San Luis
Mendoza
Ciudad de Buenos Aires
La Pampa
Buenos Aires
Neuquén
Río Negro
Chubut
Santa Cruz
Tierra del Fuego

En este momento la capa se está sirviendo como una sola imagen. Esto lo podremos comprobar en la pestaña de `Red` de las herramientas de desarrollo de nuestro navegador (Google Chrome en este caso):

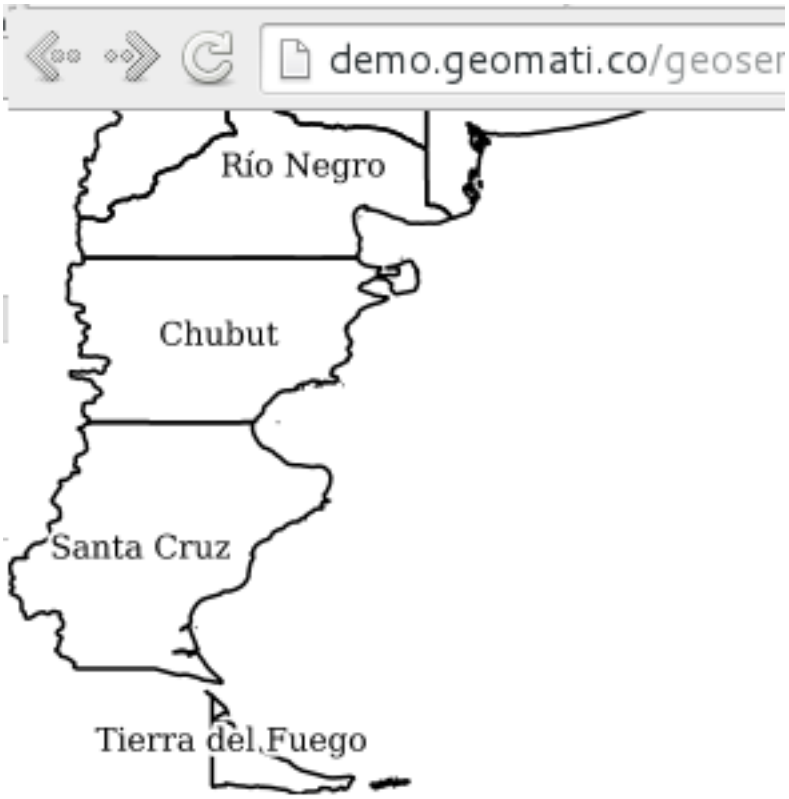


Para conseguir una mejor experiencia de usuario, se usa las peticiones teseladas, que consiste en particionar la imagen total en varias de menor tamaño, tiles (teselas), y de esta manera la descarga de la imagen se realiza en multiples imagenes que unidas componen la imagen total. En nuestro caso, para definir el uso de las teselas, podremos, desde el previsualizador de capas, en los parámetros adicionales:

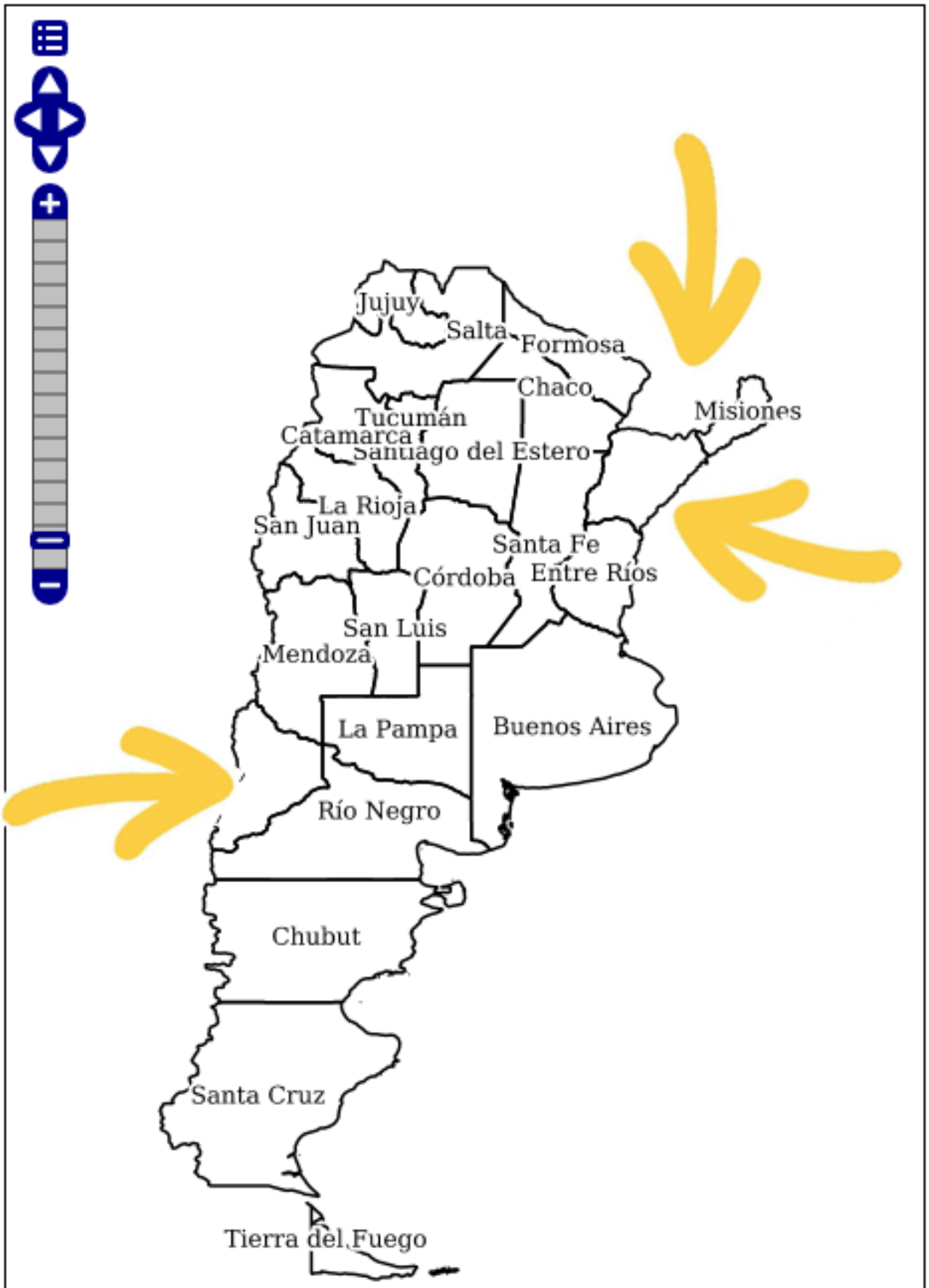
Desde la barra de parámetros opcionales, podremos definir que nos devuelva la imagen teselada, para ello en la opción **Tilling**, seleccionamos **Tiled**. Podremos comprobar en la pestaña de `Red` de nuestras herramientas de desarrollo que ahora en vez de realizar una única petición, realiza varias, una por cada tesela, para componer la imagen total:



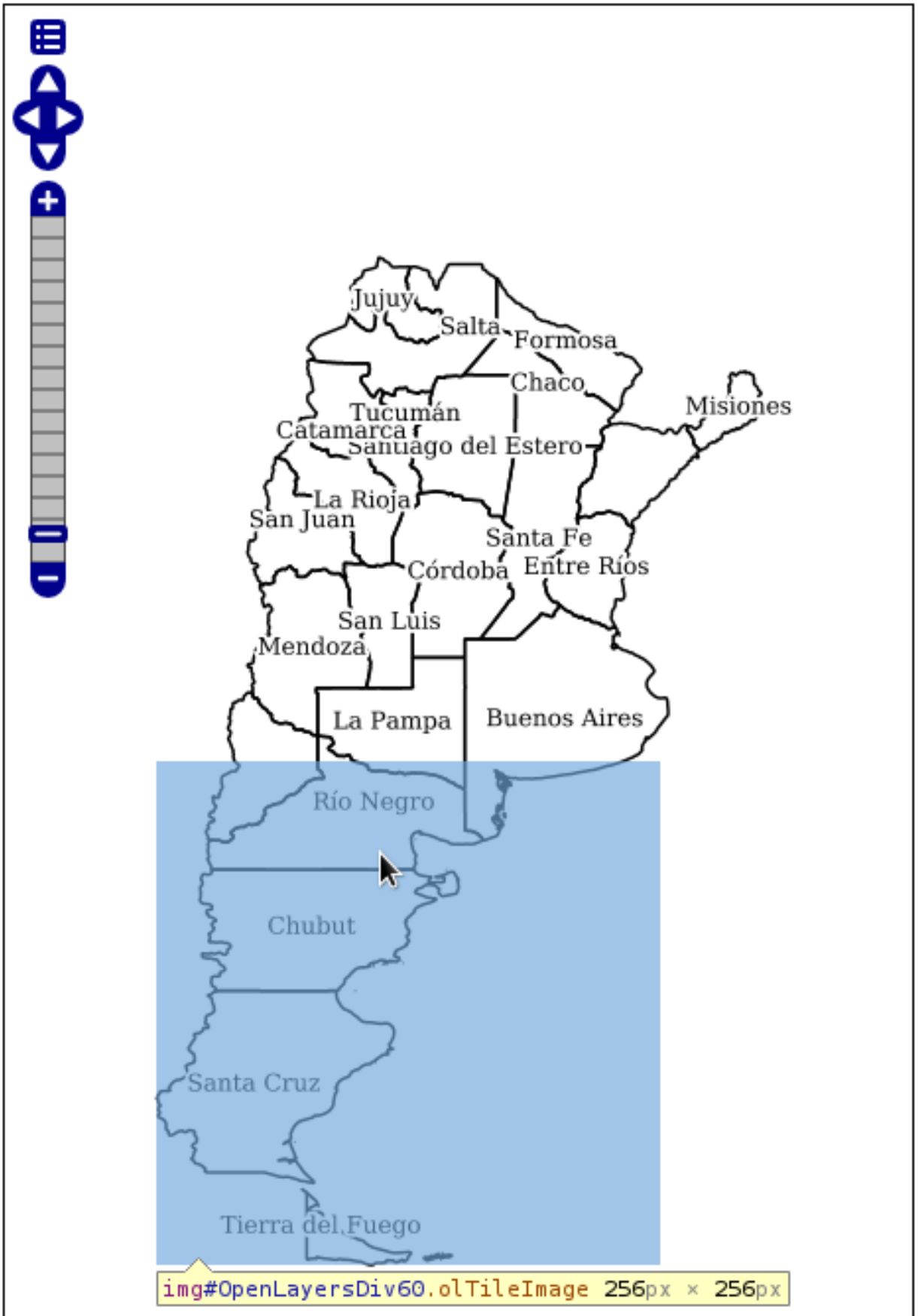
La diferencia es que en este último caso, cada petición realizada por el cliente, se trata de una parte de la imagen global:



El problema que aparece es que en algunas provincias, desaparecen las etiquetas con los nombres. Según variemos el zoom, podremos observar que estas etiquetas van apareciendo y desapareciendo de manera aleatoria.

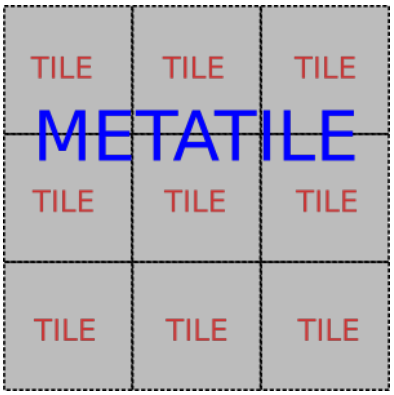

Jujuy
Salta
Formosa
Chaco
Misiones
Tucumán
Catamarca
Santiago del Estero
La Rioja
San Juan
Santa Fe
Entre Ríos
Córdoba
San Luis
Mendoza
La Pampa
Buenos Aires
Río Negro
Chubut
Santa Cruz
Tierra del Fuego

Esto es debido al proceso de creación de las teselas. Lo que sucede es que en el momento de definir la tesela, el servidor comprueba que la etiqueta esté completamente incluida en la tesela, y en caso contrario, delega la creación de la etiqueta en otra tesela, pero, si como en este caso, la etiqueta no vuelve a encajar dentro de otra tesela, esta etiqueta no será dibujada. Como se observa en la siguente figura, para la provincia por encima de Rio Negro y debajo de Mendoza, que aparece sin etiqueta, se puede ver como el límite de la tesela, coincide justo con la posición donde estaría la etiqueta.

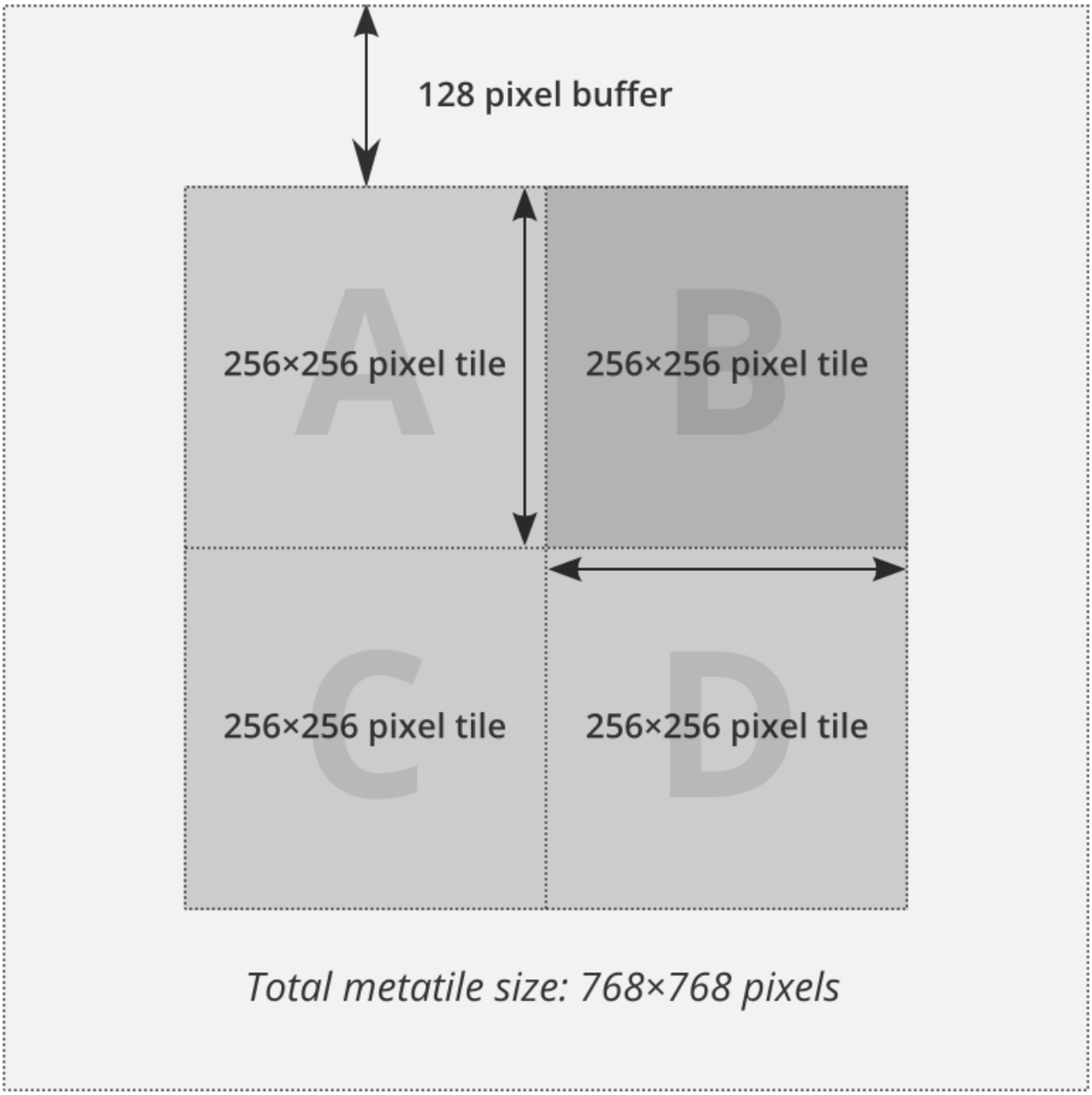

Jujuy
Salta
Formosa
Chaco
Misiones
Tucumán
Catamarca
Santiago del Estero
La Rioja
San Juan
Santa Fe
Córdoba
Entre Ríos
San Luis
Mendoza
La Pampa
Buenos Aires
Río Negro
Chubut
Santa Cruz
Tierra del Fuego
img#OpenLayersDiv60.olTileImage 256px × 256px

Solución 1: Metatileado

Las metatiles son tiles creadas de la combinación de varias tiles. Su beneficio es que el servidor de mapas genera el contenido de las tiles a partir de una metatile creada anteriormente y que después se partirá en el número de tiles necesario.



En la imagen anterior podemos observar una metatile 3x3, que está compuesta por 9 tiles. Además de la composición en N tiles, las metatiles disponen de una parte que es el buffer, que será tenida en cuenta a la hora de generar la metatile, pero que no se mostrará en las teselas resultantes de la metatile.



El objetivo de este buffer es permitir que las etiquetas, marcadores, etc, que se encuentran en los límites de la metatile, se dibujen correctamente ya que estos elementos serán tenidos en cuenta en la generación de las teselas de esa metatile.

Implementación del metatileado en GeoServer

GeoWebCache, integrado en GeoServer es el que aporta la solución del metatileado a este servidor de mapas. Para aplicar este:

vamos a la pestaña `Tile Caching` de nuestra capa publicada en GeoServer

en las opciones de `Tile cache configuration` selecciones los siguientes valores:

- Seleccionamos `Create a cached layer for this layer`
- Seleccionamos `Enable tile caching for this layer`
- En `Metatiling Factors` seleccionamos valores 4 `tiles wide` por 4 `tiles high`
- Seleccionamos un tamaño de Gutter (buffer) de 100 pixels

De esta manera, se generará una metatile de 4x4, con un buffer de 100px.

Ahora podremos consumir desde nuestro cliente la capa que se está sirviendo a través de GeoWebCache, bien usando la URL asociada a esta capa:

Note: http://example.com/geoserver/gwc/service/wms

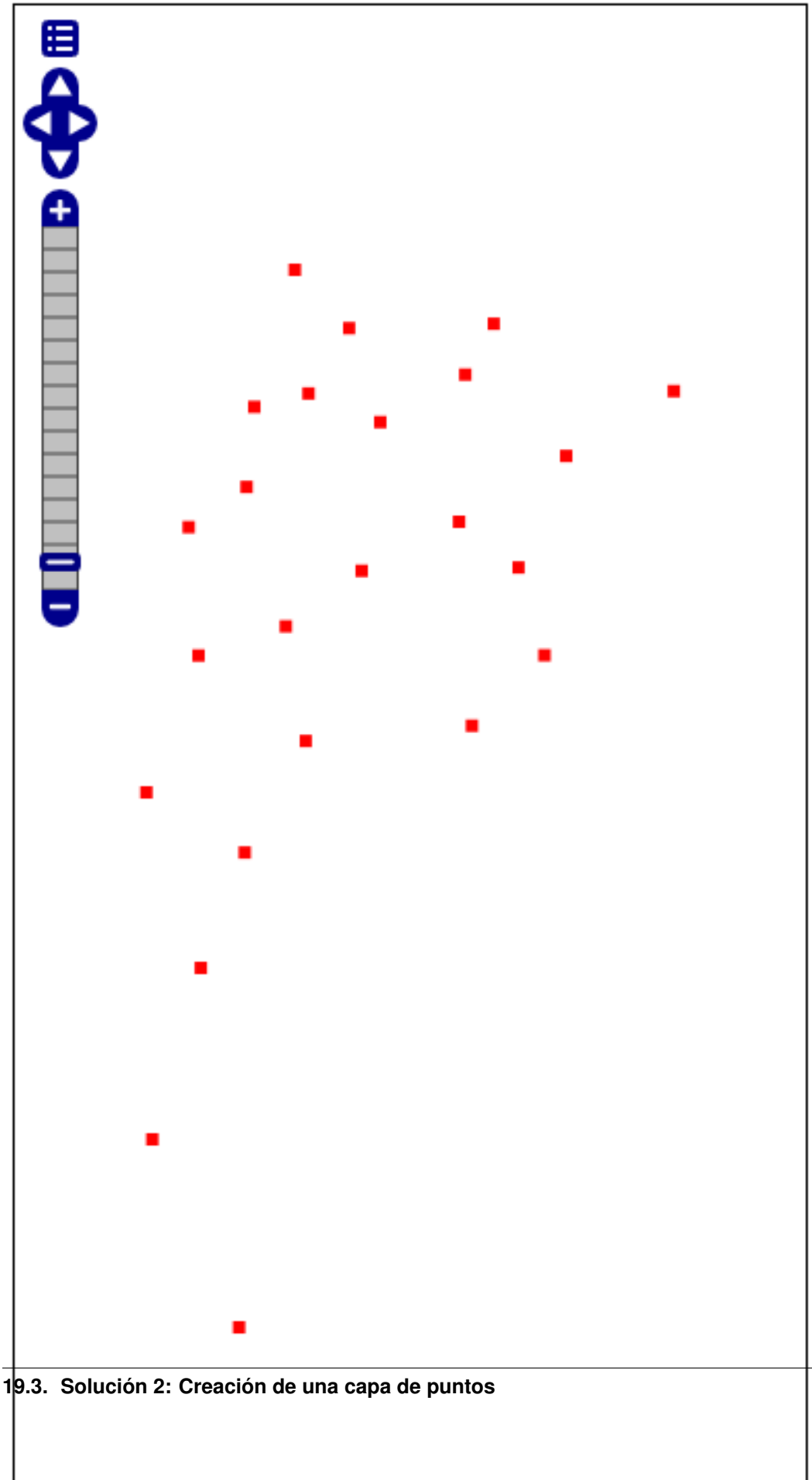
o mediante el uso de las peticiones GetMap del WMS siempre que GeoServer se encuentre correctamente configurado.

Note: Revisar documentación del uso de GeoWebCache dentro de GeoServer que se adjunta en la referencia de este capítulo

Solución 2: Creación de una capa de puntos

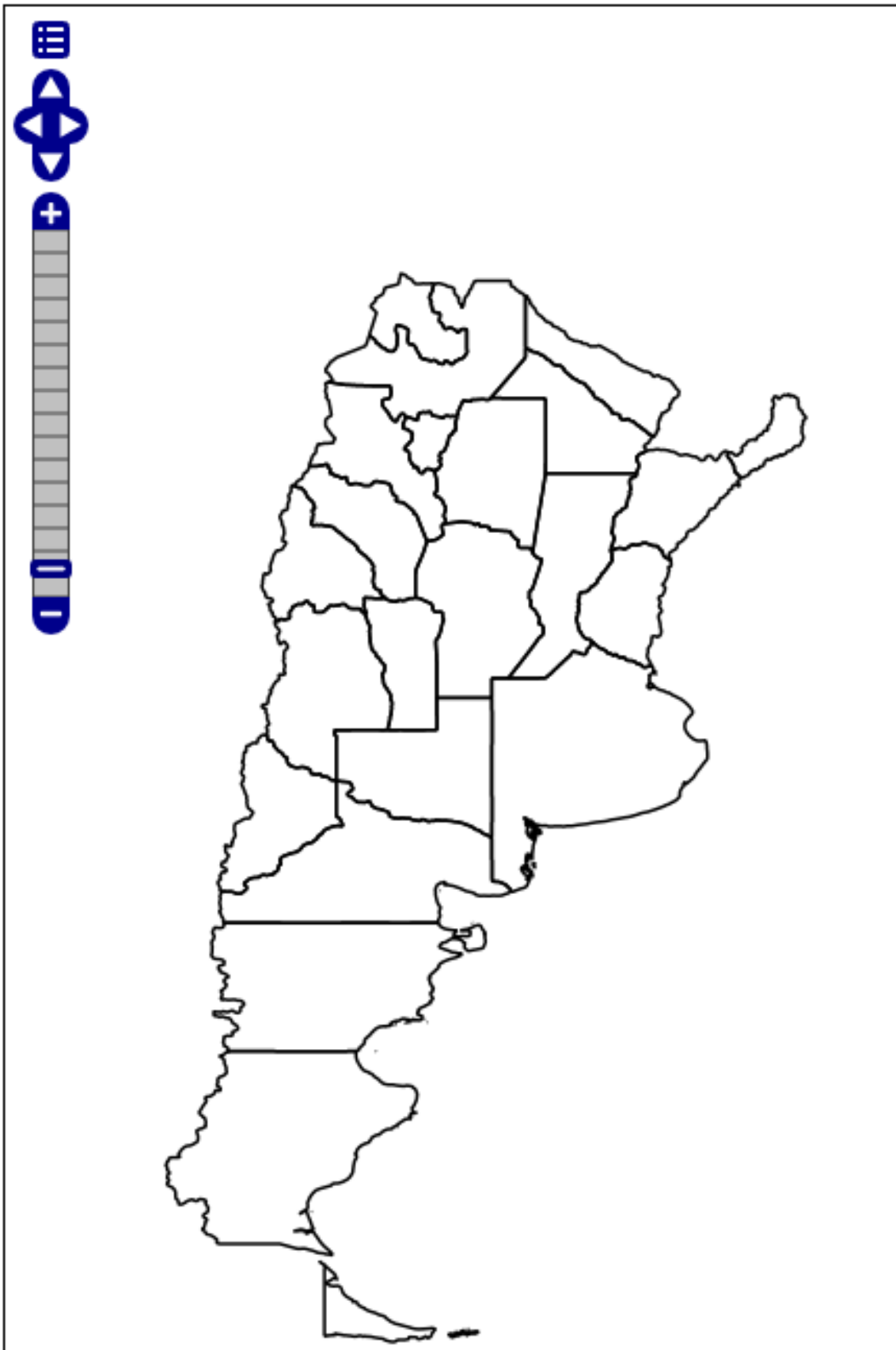
Otra solución al problema de las etiquetas de las capas es la separación de los elementos en varias capas y su posterior consumo. Para ello lo que se realizó es lo siguiente.

A partir de la capa de polígonos se obtiene una capa con los puntos interiores mediante el uso, por ejemplo, de un SIG de Escritorio como QGIS o mediante el uso de cualquier otra herramienta SIG como PostGIS, OGR, etc. El producto final del procesamiento será una capa de puntos.



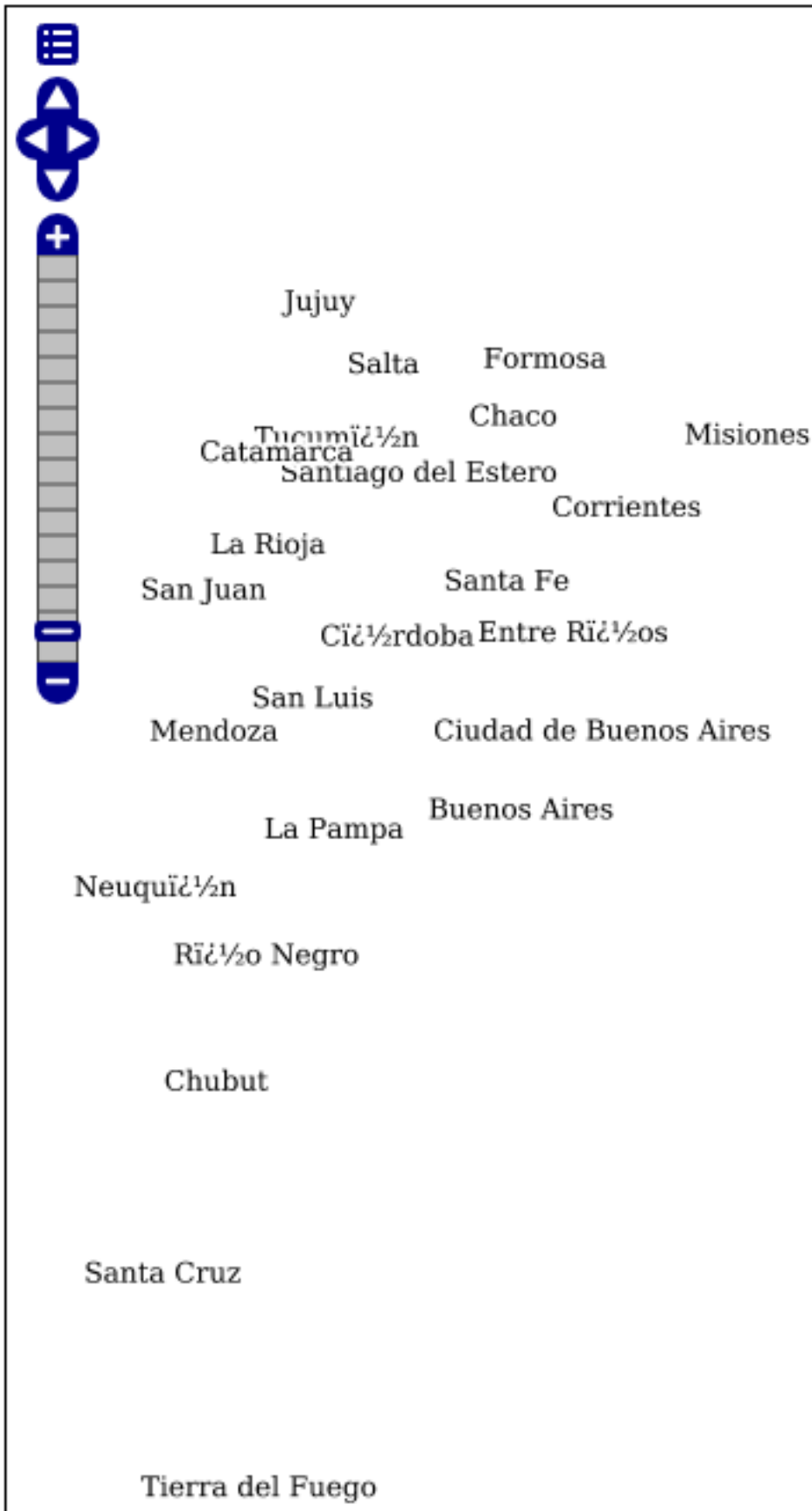
mos dividir el SLD con el estilo original en dos partes y aplicaremos estos estilos a cada una de las capas. Por un lado mostraremos los contornos de las provincias sobre la capa de provincias original:

```
<?xml version="1.0" encoding="UTF-8"?>
<StyledLayerDescriptor version="1.0.0" xmlns="http://www.opengis.net/sld" xmlns:ogc=
↪"http://www.opengis.net/ogc"
  xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/
↪XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/sld http://schemas.opengis.net/sld/1.0.0/
↪StyledLayerDescriptor.xsd">

  <NamedLayer>
    <Name>Blue lake</Name>
    <UserStyle>
      <Title>Blue lake</Title>
      <Abstract>A blue fill, solid black outline style</Abstract>
      <FeatureTypeStyle>
        <Rule>
          <Name>name</Name>
          <PolygonSymbolizer>
            <Fill>
              <CssParameter name="fill">
                <ogc:Literal>#ffffff</ogc:Literal>
              </CssParameter>
              <CssParameter name="fill">
                <ogc:Literal>1.0</ogc:Literal>
              </CssParameter>
            </Fill>
          </PolygonSymbolizer>
          <LineSymbolizer>
            <Stroke>
                <CssParameter name="stroke">#000000</CssParameter>
                <CssParameter name="stroke-width">1</CssParameter>
            </Stroke>
          </LineSymbolizer>
        </Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```



y por otro lado usaremos el estilo de las etiquetas con la capa de puntos que acabamos de generar:

```
<?xml version="1.0" encoding="UTF-8"?>
<StyledLayerDescriptor version="1.0.0" xmlns="http://www.opengis.net/sld" xmlns:ogc=
↪"http://www.opengis.net/ogc"
  xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/
↪XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/sld http://schemas.opengis.net/sld/1.0.0/
↪StyledLayerDescriptor.xsd">

  <NamedLayer>
    <Name>Blue lake</Name>
    <UserStyle>
      <Title>Blue lake</Title>
      <Abstract>A blue fill, solid black outline style</Abstract>
      <FeatureTypeStyle>
        <Rule>
          <Name>name</Name>
          <TextSymbolizer>
           <Label>
              <ogc:PropertyName>NAME_1</ogc:PropertyName>
            </Label>
            <LabelPlacement>
              <PointPlacement>
                <AnchorPoint>
                  <AnchorPointX>0.5</AnchorPointX>
                  <AnchorPointY>0.5</AnchorPointY>
                </AnchorPoint>
              </PointPlacement>
            </LabelPlacement>
            <Halo>
              <Radius>
                <ogc:Literal>2</ogc:Literal>
              </Radius>
              <Fill>
                <CssParameter name="fill">#FFFFFF</CssParameter>
              </Fill>
            </Halo>
            <Fill>
              <CssParameter name="fill">#000000</CssParameter>
            </Fill>
            <VendorOption name="goodnessOfFit">0</VendorOption>
            <VendorOption name="conflictResolution">false</VendorOption>
          </TextSymbolizer>
              </Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```


Jujuy
Salta
Formosa
Chaco
Misiones
Catamarca
Santiago del Estero
Corrientes
La Rioja
San Juan
Santa Fe
Entre Rï¿½os
San Luis
Mendoza
Ciudad de Buenos Aires
Buenos Aires
La Pampa
Neuquï¿½n
Rï¿½o Negro
Chubut
Santa Cruz
Tierra del Fuego

Ahora deberemos pinchar ambas capas por separado en nuestro visor, aplicando el teselado en la de provincias y mostrando sin teselar la de las etiquetas.

Referencias

Metatiles **[EN]**

Understanding Metatiles **[EN]**

MapProxy WMS Labeling **[EN]**

Using GeoWebCache **[EN]**

CHAPTER 20

Estadísticas

El sistema de estadísticas consta de dos partes. Por una parte, el servicio de estadísticas nos permite mostrar al usuario gráficas sobre objetos existentes en una capa del mapa. Por otra, el motor de cálculo nos permite generar los datos estadísticos de cobertura de forma automática.

Servicio de estadísticas

Este servicio nos permitirá obtener gráficos sobre objetos del mapa. Los datos a presentar en dichos gráficos pueden presentar una o más variables con datos en uno o más instantes temporales.

Para utilizar el servicio es necesario partir de la tabla con los datos a graficar. Esta tabla contendrá los datos a graficar. Requerirá:

- un campo identificador del objeto sobre el que se pide la información, por ejemplo, código de provincia
- un campo con la fecha del valor
- tantos campos como valores tenga ese objeto en esa fecha: superficie de bosque, superficie deforestada, etc.

En el siguiente ejemplo tenemos un campo identificador `id` y un campo `fecha`. Para cada objeto tenemos tres fechas en las que se mide la superficie de bosque y la superficie deforestada:

id	sup_bosque	sup_deforestada	fecha
1	400	100	2000
1	300	100	2005
1	200	100	2010
2	300	0	2000
2	350	100	2005
2	30	500	2010

Una vez se tiene la tabla con los datos hay que configurar las tablas de metadatos del servicio de estadísticas, que contiene información sobre el gráfico: título, unidades, tipo de gráfico; y permite enlazar al servicio con la tabla de datos anterior.

Instalación

Para que el servicio de estadísticas funcione es necesario:

1. Crear un esquema en la base de datos. En dicho esquema se introducirán todas las tablas usadas por el portal tanto para estadísticas como para otras funcionalidades.
2. Crear las tablas de metadatos `redd_stats_charts` y `redd_stats_variables` que definirán el aspecto del gráfico y asociarán las capas del mapa con la tabla de datos.

```
CREATE TABLE redd_stats_charts (
        id serial PRIMARY KEY,
        title character varying,
        subtitle character varying,
        layer_name character varying,
        division_field_id character varying,
        division_field_name character varying,
        table_name_data character varying,
        data_table_id_field character varying,
        data_table_date_field character varying,
        data_table_date_field_format character varying,
        data_table_date_field_output_format character varying
) WITH ( OIDS=FALSE );

CREATE TABLE redd_stats_variables (
        id serial NOT NULL PRIMARY KEY,
        chart_id integer,
        y_label character varying,
        units character varying,
        tooltipsdecimals integer,
        variable_name character varying,
        data_table_variable_field character varying,
        graphic_type character varying,
        priority integer
);
```

3. Configurar el esquema en el `portal.properties` existente en el directorio de configuración del portal:

```
db-schema=mi_esquema
```

4. Tras este último paso es necesario reinicializar el portal. Esto se puede realizar mediante el comando `touch` aplicado al war. Este comando modifica la fecha y hora del fichero que se le pasa como parámetro, lo que fuerza a Tomcat a redesplegar el war y reinicializar la aplicación:

```
$ touch /var/tomcat/webapps/portal.war
```

Configuración de la tabla de metadatos

Cada fila de la tabla de `redd_stats_charts` especificará un gráfico para los objetos de una capa del portal:

- id: identificador, rellenado automáticamente.
- title: título del gráfico
- subtitle: subtítulo del gráfico
- layer_name: Nombre en GeoServer de la capa para la que se ofrecerá el gráfico, con la forma espaciodetrabajo:nombrecapa. Por ejemplo: bosques:provincias

- division_field_id: Nombre del campo que identifica los objetos en la capa anterior. Es posible utilizar un campo que no es único si se desea tener el mismo gráfico para más de un objeto.
- division_field_name: Nombre del campo con el nombre del objeto en la capa anterior. Se incluirá en el título del gráfico.
- table_name_data: Nombre de la tabla con los datos.
- data_table_id_field: Nombre del campo identificador en la tabla de datos `table_name_data`.
- data_table_date_field: Nombre del campo fecha en la tabla de datos `table_name_data`.
- data_table_date_field_format: Formato del campo fecha si no es de tipo `date` según la función to_date de PostgreSQL . NULL si el campo fecha es de tipo `date`.
- data_table_date_field_output_format: Formato de las fechas en el eje X de la gráfica según la función to_date de PostgreSQL . Si se deja a NULL se tomará DD-MM-YYYY (día-mes-año).

Una vez el registro con el gráfico ha sido creado es necesario especificar cómo se presentan los datos en el gráfico en la tabla `redd_stats_variables`:

- id: identificador, rellenado automáticamente.
- chart_id: identificador del gráfico al que pertenece esta variable. Deberá tener valor del campo `id` del gráfico.
- y_label: Magnitud medida, por ejemplo "Superficie"
- units: unidades en las que se mide la magnitud, por ejemplo "Hectáreas"
- tooltipsdecimals: Número de decimales que se presentarán cuando el usuario interactúa con la gráfica
- variable_name: Nombre de la variable que aparecerá en el gráfico, por ejemplo "bosque cultivado".
- data_table_variable_field: Nombre del campo de la tabla de datos que contiene los valores de la variable anterior.
- graphic_type: Tipo de gráfico. Puede ser cualquier valor aceptado por la librería highcharts
- priority: Prioridad del eje. Para poder definir qué datos se presentan antes (p.ej.: líneas delante de gráfico de barras).

Caso práctico

En este ejemplo vamos a suponer que tenemos:

- Una tabla provincias con un campo `id_provincia` con tres provincias con identificador 1, 2 y 3.
- Una capa en GeoServer, publicando la tabla anterior con el nombre `provincias` en el espacio de trabajo `bosques`, es decir, con nombre `bosques:provincias`.
- La tabla convenientemente publicada en el portal, de manera es es posible mostrar el diálogo de información al pinchar en una de las provincias.

Es posible descargar los datos de ejemplo aquí, para su carga en PostGIS y la realización del caso práctico con ellos.

Queremos publicar los siguientes datos de cobertura forestal:

Provincia 1	1990	2000	2005
bosque nativo	100	98	78
bosque cultivado	1000	1100	1050

Provincia 2	1990	2000	2005
bosque nativo	590	ND	208
bosque cultivado	0	0	50

Provincia 3	1990	2000	2005
bosque nativo	2000	2300	2500
bosque cultivado	0	100	50

Lo primero será crear la tabla de datos con cualquier nombre significativo, por ejemplo `cobertura_forestal_provincias`. Suponemos que creamos todo en un esquema llamado estadísticas:

```
CREATE TABLE estadisticas.cobertura_forestal_provincias(
        id_provinc varchar,
        sup_nativo varchar,
        sup_cultivado varchar,
        anio date
);
```

Una vez la tabla está creada, es necesario introducir un registro por cada dato:

```
INSERT INTO estadisticas.cobertura_forestal_provincias VALUES ('1', 100, 1000, '1/1/
↪1990');
INSERT INTO estadisticas.cobertura_forestal_provincias VALUES ('1', 98, 1100, '1/1/
↪2000');
INSERT INTO estadisticas.cobertura_forestal_provincias VALUES ('1', 78, 1050, '1/1/
↪2005');

INSERT INTO estadisticas.cobertura_forestal_provincias VALUES ('2', 590, 0, '1/1/1990
↪');
INSERT INTO estadisticas.cobertura_forestal_provincias VALUES ('2', null, 0, '1/1/2000
↪');
INSERT INTO estadisticas.cobertura_forestal_provincias VALUES ('2', 208, 50, '1/1/2005
↪');

INSERT INTO estadisticas.cobertura_forestal_provincias VALUES ('3', 2000, 0, '1/1/1990
↪');
INSERT INTO estadisticas.cobertura_forestal_provincias VALUES ('3', 2300, 100, '1/1/
↪2000');
INSERT INTO estadisticas.cobertura_forestal_provincias VALUES ('3', 2500, 50, '1/1/
↪2005');
```

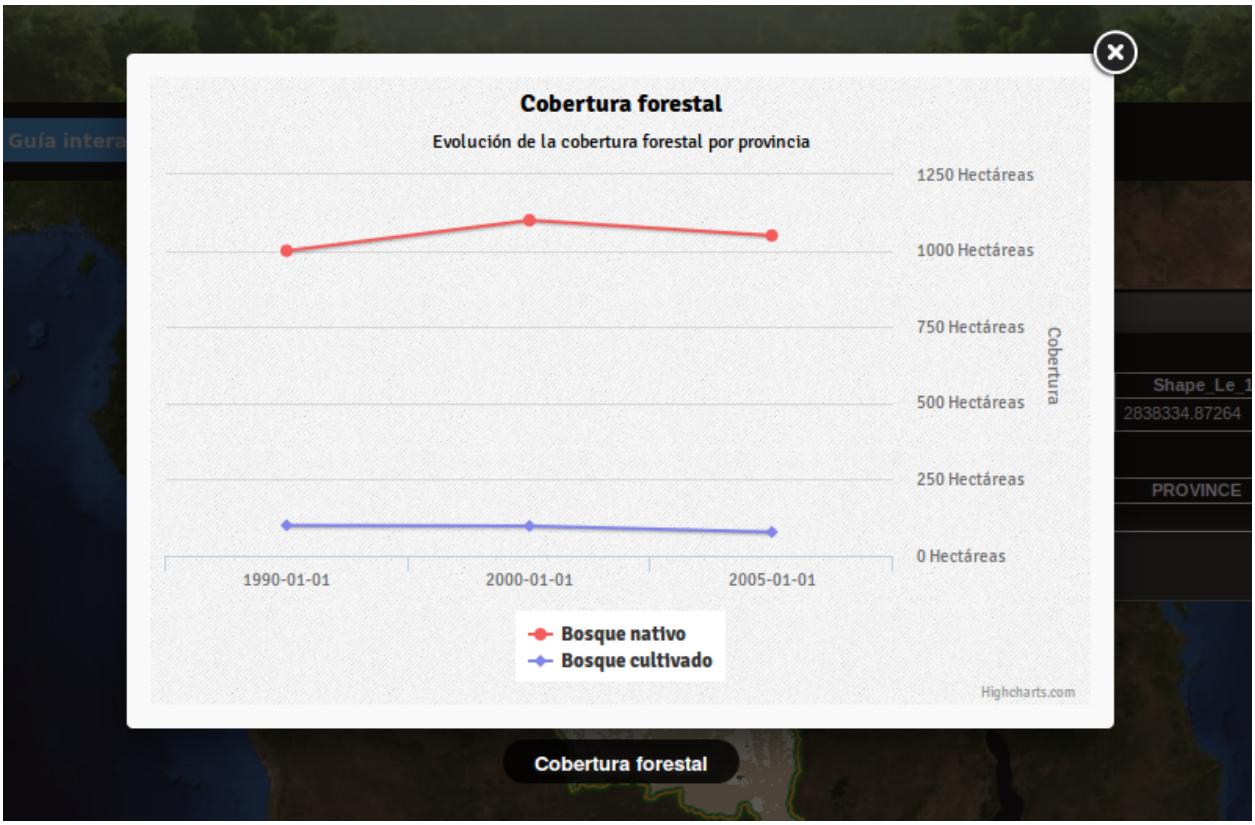
Por último crearemos el registro en la tabla de metadatos que enlaza estos datos con nuestra tabla de datos recién creada:

```
INSERT INTO estadisticas.redd_stats_charts VALUES (
        DEFAULT, -- id generado automaticamente
        'Cobertura forestal', --title
        'Evolución de la cobertura forestal por provincia', --subtitle
        'bosques:provincias', --capa en geoserver
        'id_provinc', -- nombre del campo identificador de la capa
        'estadisticas.cobertura_forestal_provincias', -- nombre de la tabla de datos
        'id_provinc', -- nombre del campo id
        'anio', -- nombre del campo fecha
        NULL, -- campo fecha de tipo date
        'YYYY-MM-DD'
);

INSERT INTO estadisticas.redd_stats_variables VALUES (
        DEFAULT, -- id generado automaticamente
        (select currval('estadisticas.redd_stats_charts_id_seq')), --nos da el id del␣
↪último INSERT en redd_stats_charts, es decir, nuestro gráfico
        'Cobertura', -- Nombre de la magnitud a medir
        'Hectáreas', -- Unidades de la magnitud a medir
```

```
        2, -- número de decimales a presentar
        'Bosque cultivado', -- Nombre de la variable
        'sup_nativo', --nombre del campo
        'line', --tipo de gráfico
        1 --prioridad
);
INSERT INTO estadisticas.redd_stats_variables VALUES (
        DEFAULT, -- id generado automaticamente
        (select currval('estadisticas.redd_stats_charts_id_seq')), --nos da el id del␣
↪último INSERT en redd_stats_charts, es decir, nuestro gráfico
        'Cobertura', -- Nombre de la magnitud a medir
        'Hectáreas', -- Unidades de la magnitud a medir
        2, -- número de decimales a presentar
        'Bosque nativo', -- Nombre de la variable
        'sup_cultivado', --nombre del campo
        'column', --tipo de gráfico
        2 --prioridad
);
```

Ahora, cuando el usuario pinche en una de las provincias:

1. el portal buscará en la tabla `estadisticas.redd_stats_charts` los registros que afectan a la capa `bosques:provincias` y encontrará el registro que acabamos de introducir.
2. el portal ofrecerá al usuario un botón para mostrar los datos de la tabla de datos asociada `estadisticas.cobertura_forestal_provincias`
3. el usuario pinchará en dicho botón
4. el portal leerá la tabla de variables, la tabla de datos y creará el gráfico que se ofrecerá al usuario



Motor de cálculo

El motor de cálculo son una serie de funciones PostgreSQL/PostGIS que permiten generar la tabla de datos que se presenta en los gráficos de forma automática, tomando como entrada:

- una tabla de polígonos sobre los cuales se quieren presentar las estadísticas, típicamente divisiones administrativas, con un campo identificador
- una tabla con la cobertura forestal en la que cada registro representa un area con la misma clasificación en un instante determinado.

Y produciendo:

- la tabla con los datos de cobertura en hectáreas para cada año y objeto existente en la primera capa.

Instalación

El motor de cálculo puede descargarse aquí. Para su instalación es necesario ejecutarlo en un intérprete de PostGIS, por ejemplo en línea de comandos:

```
$ psql -U spatial_user -d spatialdata -f redd_stats_calculator.sql
```

Esta ejecución instalará dos funciones, `redd_stats_calculo` y `redd_stats_run`. Esta última es la que se utilizará para iniciar el motor de cálculo.

Además de las funciones, el motor espera encontrar en el mismo esquema donde se encuentra la tabla de metadatos una tabla con las fajas en proyección EPSG:4326. Esta tabla deberá tener un campo `geom` con la geometria y un campo `srid` de tipo `integer` con el código SRID al que pertenece cada faja. Se puede ver un ejemplo en el caso práctico más abajo.

Una vez las funciones están instaladas y la tabla `redd_stats_fajas` está creada, podemos empezar a utilizarlo. Para hacerlo funcionar habrá que realizar dos pasos, 1) configurar la tabla de metadatos especificando esta vez TODOS los campos campos y 2) invocar al motor para que genere los gráficos.

Configuración de la tabla de metadatos

Además de los campos especificados para el servicio, será necesario especificar:

- table_name_division: nombre de la tabla que se publica por GeoServer con el nombre especificado en el campo `layer_name`.
- class_table_name: nombre de la tabla que tiene la clasificación forestal, con los polígonos de todos los años indicando la clasificación y la fecha en la fecha en la que es válido el polígono.
- class_field_name: nombre del campo en la tabla anterior que indica el tipo de clasificación para cada registro.
- date_field_name: nombre del campo que indica la fecha en la que el polígono es válido.

Invocación del motor para un gráfico determinado

El motor gráfico se invoca con la función `redd_stats_run`, que toma dos parámetros. El primero es el valor del campo `id` del registro de la tabla de metadatos cuyo gráfico queremos generar. El segundo es el esquema donde está esta tabla. La invocación se hace mediante una instrucción `SELECT`:

```
SELECT redd_stats_run(1, 'estadisticas');
```

Caso práctico

En este caso se parte de

- Una tabla `provincias` con un campo `id_provincia` como identificador.
- Una capa en GeoServer, publicando la tabla anterior con el nombre `provincias` en el espacio de trabajo `bosques`, es decir, con nombre `bosques:provincias`.
- La tabla convenientemente publicada en el portal, de manera es es posible mostrar el diálogo de información al pinchar en una de las provincias.
- Una tabla `cobertura` con los polígonos de la clasificación forestal y los campos:
  - un campo `clasificac` indicando el tipo de la clasificación
  - un campo `fecha` indicando el año de esa clasificación

Es posible descargar los datos de ejemplo aquí, para su carga en PostGIS y la realización del caso práctico con ellos.

En este caso no creamos la tabla de datos, ya que ésta la creará el motor directamente, y pasamos directamente a añadir el registro en la tabla de metadatos:

```
INSERT INTO estadisticas.redd_stats_metadata (
        title,
        subtitle,
        y_label,
        units,
        tooltipsdecimals,
        layer_name,
        table_name_division,
        division_field_id,
        class_table_name,
        class_field_name,
        date_field_name,
        table_name_data,
        graphic_type
) VALUES (
        'Cobertura forestal',
        'Evolución de la cobertura forestal por provincia',
        'Cobertura',
        'Hectáreas',
        2,
        'bosques:provincias',
        'estadisticas.provincias',
        'id_provinc',
        'estadisticas.cobertura',
        'clasificac',
        'instante',
        'estadisticas.cobertura_forestal_provincias_automatica',
        '2D'
);
```

Puede verse cómo en este caso se han especificado los parámetros `table_name_division`, `class_table_name`, `class_field_name`, `date_field_name`, que permitirán al motor acceder a los datos y generar la tabla automáticamente.

Para invocar el motor basta con ver el id asignado al registro recién insertado:

```
spatialdata=> SELECT id, title, table_name_data FROM estadisticas.redd_stats_metadata;
```

```
 id |       title        |                    table_name_data
----+--------------------+-------------------------------------------------------
  5 | Cobertura forestal | estadisticas.cobertura_forestal_provincias_automatica
(1 row)
```

y ejecutar la función `redd_stats_run()` con el código que nos interesa y el nombre del esquema donde está la tabla redd_stats_metadata, es decir `estadisticas`:

```
SELECT redd_stats_run(1, 'estadisticas');
```

Tras la ejecución, la tabla `estadisticas.cobertura_forestal_provincias_automatica` estará rellena con el resultado de los cálculos:

```
patialdata=> select * from estadisticas.cobertura_forestal_provincias_automatica ;
 division_id | variable  |   fecha    |    valor
-------------+-----------+------------+-------------
 1           | bosque    | 1999-01-01 | 6.37725e+07
 1           | bosque    | 2004-01-01 | 5.27672e+07
 1           | bosque    | 2010-01-01 | 8.30697e+07
 1           | no bosque | 1999-01-01 |  1.8682e+07
 1           | no bosque | 2004-01-01 | 2.97502e+07
 1           | no bosque | 2010-01-01 |           0
 2           | bosque    | 1999-01-01 |   4.982e+07
 2           | bosque    | 2004-01-01 | 3.54705e+07
 2           | bosque    | 2010-01-01 |           0
 2           | no bosque | 1999-01-01 | 4.55279e+07
 2           | no bosque | 2004-01-01 | 5.98773e+07
 2           | no bosque | 2010-01-01 | 9.53479e+07
 3           | bosque    | 1999-01-01 |  3.2107e+07
 3           | bosque    | 2004-01-01 | 2.52069e+07
 3           | bosque    | 2010-01-01 | 3.87162e+07
 3           | no bosque | 1999-01-01 | 6.60003e+06
 3           | no bosque | 2004-01-01 | 1.35093e+07
 3           | no bosque | 2010-01-01 |           0
(18 rows)
```

CHAPTER 21

Herramienta de feedback

La herramienta de feedback permite al usuario del portal proporcionar comentarios sobre determinadas regiones de las capas del mapa.

Configuración

La creación y almacenamiento de estos comentarios requiere de 1) una tabla en la base de datos y 2) una cuenta de e-mail para el intercambio de mensajes entre el sistema, el usuario y el técnico que valida el comentario.

Almacenamiento de los comentarios

Para almacenar los comentarios se utilizará una tabla llamada `redd_feedback` que debe existir en el esquema definido por la propiedad `db-schema` en el fichero `portal.properties`:

```
CREATE TABLE redd_feedback (
        id serial,
        geometry geometry('GEOMETRY', 900913),
        comment varchar NOT NULL,
        layer_name varchar NOT NULL,
        layer_date varchar,
        date timestamp NOT NULL,
        email varchar NOT NULL,
        verification_code varchar,
        language varchar,
        state int
);
```

Esta tabla alojará un registro por cada comentario hecho por el usuario. En ella podemos ver los siguientes campos:

- id: identificador de los comentarios
- geometry: geometría que identifica la región a la que afecta el comentario. Es dibujada por el usuario en el portal. En sistema de coordenadas EPSG:900913.

- comment: comentario realizado por el usuario
- layer_name: identificador de la `portalLayer` definida en el fichero layers.json que el usuario seleccionó para hacer el comentario.
- layer_date: instante para el cual el comentario se ha realizado. A tener en cuenta si la capa tiene varias instancias temporales.
- date: fecha en la que se realiza el comentario.
- email: email introducido por el usuario.
- verification_code: el código usado para verificar que la dirección de email proporcionada por el usuario está operativa. Ver *Flujo de trabajo de la herramienta Feedback*.
- language: el código del lenguaje seleccionado por el usuario en el momento de hacer el comentario. Por ejemplo "es" para español.
- state: estado en el flujo de trabajo. Uno de los siguientes:
  - 0 -> nuevo comentario. E-mail no confirmado.
  - 1 -> E-mail y comentario confirmado.
  - 2 -> Validado por un técnico.
  - 3 -> Validación notificada al usuario creador del comentario.

Comunicación

Para la comunicación entre usuario y técnico es necesario primero configurar la cuenta de correo que se utilizará para realizar el intercambio. Esto se hace en el portal.properties mediante las siguientes propiedades:

- feedback-mail-host. Nombre del servidor de correo. Por ejemplo `smtp.gmail.com` si se está utilizando una cuenta de gmail.
- feedback-mail-port. Puerto donde escucha el servidor de correo. `587` en el caso de gmail.
- feedback-mail-username. Cuenta de correo. Por ejemplo: `onuredd@gmail.com`.
- feedback-mail-password. Contraseña de la cuenta de correo.

Esta cuenta de correo se utilizará para contactar con el usuario que crea un nuevo mensaje, pero además será donde se reciban las notificaciones de que un nuevo comentario ha sido creado y está a la espera de ser validado por un técnico. Este correo llevará como título y cuerpo los contenidos especificados en las dos propiedades siguientes:

- feedback-admin-mail-title. Título del correo. Por ejemplo `Hay un nuevo comentario en el portal REDD`.
- feedback-admin-mail-text. Texto del correo. Por ejemplo `Se ha introducido un comentario en el portal de REDD con c\u00f3digo de verificaci\u00f3n "$code`. Nótese que la variable `$code` se reemplazará por el identificador que se le haya asignado al comentario en el campo `id` de la tabla.

Por último, la notificación a los usuarios de que su comentario ha sido validado no se realiza de inmediato sino que, de forma periódica, el sistema comprueba qué comentarios han sido validados por un técnico pero no han sido notificados todavía. El tiempo de espera entre dos comprobaciones se expresa con la siguiente propiedad:

- feedback-validation-check-delay. Tiempo que el sistema espera antes de volver a releer la tabla de comentarios y notificar a los usuarios cuyo comentario ha sido validado por un técnico. Se expresa en milisegundos, por lo que un valor de 600000 correspondería a 10 minutos, valor recomendado.

Por último, todos los mensajes enviados al usuario se deben adaptar al idioma que habla el usuario por lo que para personalizarlos es necesario establecer una serie de propiedades en los ficheros de mensajes que se pueden encontrar en el directorio `messages` del directorio de configuración de la aplicación. Las propiedades con sus valores iniciales es español son:

```
Feedback.all_parameters_mandatory=Todos los par\u00e1metros son obligatorios:
Feedback.error_sending_mail=Error enviando el correo:
Feedback.the_message_has_been_validated=El comentario ha sido validado.
Feedback.comment_not_found=No se encontr\u00f3 ning\u00fan comentario con el␣
↪c\u00f3digo
Feedback.mail-title=Comentario en el portal ONU-REDD
Feedback.verify-mail-text=Por favor, visite http://localhost:8080/unredd-portal/
↪verify-comment?lang=$lang&verificationCode=$code para confirmar el envío.
Feedback.validated-mail-text=El comentario con c\u00f3digo de verificaci\u00f3n "$code
↪", ha sido validado y puede consultarse en el portal.
Feedback.invalid-email-address=La direcci\u00f3n de correo especificada no es␣
↪v\u00e1lida
Feedback.no-geometries=Al menos se debe dibujar una geometr\u00eda
Feedback.verify_mail_sent=Se ha enviado un mensaje a la direcci\u00f3n de correo␣
↪especificada para confirmar el comentario.
Feedback.submit_error=No se pudo realizar el env\u00edo.
Feedback.no_layer_visible=Ninguna de las capas habilitadas para feedback est\u00e1␣
↪visible
Feedback.no-layer-selected=No se ha seleccionado ninguna capa
```

Flujo de trabajo de la herramienta Feedback

El flujo de trabajo habitual de la herramienta Feedback es el siguiente:

1. El usuario del portal abre el diálogo de feedback, dibuja una región e introduce un comentario y su dirección de e-mail y envía el formulario.
2. El sistema almacena el comentario en la tabla con `state` igual a 0 (nuevo). Se manda un correo al usuario con un enlace para confirmar el contenido del formulario y que la dirección de e-mail es correcta.
3. Cuando el usuario accede al enlace el sistema actualiza el campo `state` a 1 (confirmado) y envía un correo a la cuenta de correo configurada para que los técnicos sepan que hay un nuevo comentario en el sistema que espera a ser validado.
4. Un técnico forestal visualiza la entrada accediendo a la base de datos PostGIS, por ejemplo con QGIS, y puede opcionalmente marcar algunas entradas como validadas cambiando el valor de `state` a 2 (validado).
5. El sistema periódicamente comprueba los comentarios que hay en estado 2, validado pero no notificado al autor, y envía un mail automático indicándole que su comentario ha sido validado. Cuando consigue enviar este mensaje, actualiza el campo `state` a 3 (notificado), para no volver a procesarlo más.
6. El usuario recibe el mensaje indicándole que su entrada ha sido validada. Si se ha configurado en el portal una capa con los comentarios validados, el usuario podrá acceder al portal y ver su comentario allí.

Recomendaciones

Una vez la funcionalidad de Feedback ha sido instalada y está en funcionamiento se recomienda:

- Configurar la cuenta de correo en el cliente de correo habitual del técnico responsable. De esta manera se evitan los olvidos y los mensajes de nuevos comentarios siempre encontrarán alguien que los lea.

- Crear una vista SQL que seleccione todos los comentarios con `state` igual a 2 y añadirla como capa en el portal. De esta manera, cuando el técnico valida una entrada, ésta se muestra automáticamente en el portal.

CHAPTER 22

Herramienta de nota al pie

El plugin “footnote” muestra un texto en la parte inferior del portal, que puede contener a su vez un enlace. El plugin soporta multiidioma.

Para incorporar este plugin a un portal, se añade como dependencia en el `pom.xml`, como cualquier otro plugin:

```
<dependencies>
        ...
        <dependency>
                <groupId>org.fao.unredd</groupId>
                <artifactId>footnote</artifactId>
                <version>...</version>
        </dependency>
        ...
</dependencies>
```

Para configurar el texto a mostrar, editar el fichero `PORTAL_CONFIG_DIR/plugin-conf.json` y añadir la propiedad “footnote” bajo “default-conf”:

```
"footnote": {
    "text": "Texto a pie de página",
    "link": "http://snmb-desarrollo.readthedocs.org/",
    "align": "center"
}
```

Esta propiedad acepta tres atributos:

- “text”: Obligatorio. El texto a mostrar. Para portales en un sólo idioma, puede ser el texto a mostrar. Para portales multiidioma, será una referencia a la clave en el fichero de traducción (`PORTAL_CONFIG_DIR/messages/messages_*.properties`).
- “link”: Opcional. Si se quiere que el texto sea un enlace, indicar aquí la página a la que enlazar.
- “align”: Opcional. Indica la posición del texto: “left” (izquierda), “center” (centrado, por defecto) o “right” (derecha).

Ejemplo de configuración multiidioma

Configurar la propiedad "text" con una clave (por ejemplo, "footnote.text"):

```
"footnote": {
    "text": "footnote.text"
}
```

Añadir las traducciones usando dicha clave. Por ejemplo, en `PORTAL_CONFIG_DIR/messages/messages_es.properties`:

```
footnote.text=Esto es una nota a pie en espa\u00f1ol
```

O en `PORTAL_CONFIG_DIR/messages/messages_en.properties`:

```
footnote.text=This is an English footnote
```

CHAPTER 23

Scripts de administración

Note:

Fecha	Autores
19 Marzo 2014	• Fernando González (fernando.gonzalez@fao.org)
19 Marzo 2014	• Víctor González (victor.gonzalez@geomati.co)

Los *scripts* de administración se ejecutan desde la línea de comandos y sirven para:

- Cargar ficheros *.shp* como tablas en PostGIS.
- Añadir espacios de trabajo, almacenes de datos y capas en GeoServer, a partir de bases de datos PostGIS.
- Administrar el fichero *layers.json* de configuración del portal.

En este taller veremos cómo instalar los *scripts*, una descripción de los *scripts* y un ejemplo del flujo de trabajo más común.

Instalación

Para instalar los *scripts* de administración de forma que se puedan ejecutar desde cualquier directorio, únicamente hay que copiarlos en */usr/bin*:

```
$ sudo cp *.sh *.py /usr/bin
```

Para comprobar que se han instalado correctamente, se puede ejecutar el siguiente comando:

```
$ portal-layer-tree.py --help
```

Si se muestra el mensaje de ayuda, los scripts se han instalado correctamente.

Scripts

Ayuda

En primer lugar, todos los comandos aceptan la opción `--help`. Esta opción muestra un mensaje con una descripción del comando y qué opciones acepta. Así, cualquiera de los comandos disponibles que se explican aquí, se puede ejecutar con la opción `--help` para ver su descripción y sus opciones. Este mensaje de ayuda tiene la siguiente forma:

```
<modo de uso>

<descripción del comando>

Opciones:
        <opción 1>              <descripción de la opción 1>
        <opción 2>              <descripción de la opción 2>
        ...
```

Por ejemplo, si se ejecuta el comando `pg-load.sh --help` se obtiene el siguiente resultado:

```
uso: pg-load.sh --crs CRS [--schema SCHEMA] [--database DATABASE]
                [--encoding ENCODING] --file FILE

Carga un fichero .shp como una tabla de PostGIS.

Opciones:
        --crs           Sistema de coordenadas del fichero .shp
        --schema        Esquema donde se añadirá la nueva tabla. Valor por defecto:␣
↪nfms
        --database      Base de datos que contiene al esquema donde se añadirá la␣
↪nueva tabla. Valor por defecto: geoserverdata
        --encoding      Codificación de caracteres del fichero .dbf. Valor por␣
↪defecto: UTF8
        --file          Fichero .shp a añadir
```

donde se puede observar que la descripción del comando es *"Carga un fichero .shp como una tabla de PostGIS"* y que acepta las opciones `--crs`, `--schema`, `--database`, `--encoding` y `--file`, cada una con su correspondiente descripción. Además, es importante fijarse en el modo de uso:

```
uso: pg-load.sh --crs CRS [--schema SCHEMA] [--database DATABASE]
                [--encoding ENCODING] --file FILE
```

Es aquí donde se puede averiguar qué opciones son obligatorias y cuáles son opcionales. Las opcionales se muestran entre corchetes, mientras que las que son obligatoras no llevan corchetes. Así, vemos que para este *script*, `--crs` y `--file` son obligatorias, mientras que `--schema`, `--database` y `--encoding` son opcionales.

Descripción

Los *scripts* se pueden dividir en dos grupos: los *scripts* de *bash* (acabados en *.sh*) y los *scripts* de *Python* (comienzan con *portal-* y terminan en *.py*).

Los *scripts* de *bash* se utilizan para administrar las bases de datos PostGIS y los recursos de GeoServer. Estos *scripts*, a su vez, se pueden dividir en dos grupos. Por un lado los scripts que actúan sobre PostGIS:

- `pg-load.sh`: Este comando sirve para cargar un fichero *.shp* como una tabla en PostGIS.
- `multi-pg-load.sh`: Sirve para cargar todos los ficheros *.shp* de un directorio como diferentes tablas en PostGIS. Es muy útil cuando se quieren añadir muchos datos de una vez, por ejemplo, la primera vez que se despliega el portal de diseminación. **IMPORTANTE**: para utilizar este comando, todos los ficheros *.shp* deben estar en el mismo sistema de coordenadas.

Y por otro lado los que actúan sobre GeoServer:

- `add-layer.sh`: Sirve para añadir una nueva capa a GeoServer a partir de una tabla de PostGIS. Es posible que cuando se ejecute este comando se obtenga un error como este: `ERROR cargando capa '<layer_name>' (code 400): Trying to create new feature type inside the store, but no attributes were specified`. Este mensaje dice que la capa no existe en PostGIS. Esto puede ser porque PostGIS distingue mayúsculas y minúsculas para los nombres de la tabla. En caso de obtener este mensaje, se aconseja revisar en detalle el esquema, la base de datos y el nombre de la tabla.
- `multi-add-layer.sh`: Sirve para añadir varias capas a GeoServer a partir de diferentes tablas de PostGIS. Se puede utilizar en lugar de ejecutar varios comandos `add-layer.sh` seguidos para acelerar el proceso.
- `add-all-layers.sh`: Sirve para añadir a GeoServer todas las tablas de un esquema de base de datos de PostGIS. Este comando se puede utilizar cuando se han cargado por primera vez todos los datos en PostGIS y ninguna de las tablas está añadida todavía.

Es importante destacar que todos los *scripts* de *bash* tienen configurados por defecto los parámetros de conexión de la base de datos (*host*, puerto, esquema y nombre de base de datos) así como la configuración de GeoServer (espacio de trabajo y almacén de datos). En caso de omitir esas opciones al ejecutar los *scripts*, se tomarán esos valores por defecto. Para saber cuáles son los valores por defecto, simplemente basta con mostrar la ayuda del comando con la opción `--help`. Por ejemplo, para ver qué valores por defecto se toman para el comando `add-layer.sh`, podemos mostrar su ayuda:

```
uso: add-layer.sh --layer LAYER [--workspace WORKSPACE] [--datastore DATASTORE]

Añade una nueva capa a GeoServer a partir de una tabla de PostGIS.
El esquema que contiene la tabla debe haber sido añadido anteriormente a GeoServer␣
↪como un datastore.

Opciones:
        --layer         Tabla que se añadirá a GeoServer.
        --workspace     Espacio de trabajo de GeoServer que donde se añadirá la capa.␣
↪Valor por defecto: nfms
        --datastore     Nombre del almacén de datos que contiene la tabla a añadir.␣
↪Valor por defecto: geoserverdata
```

En ella podemos observar que el valor por defecto para el *workspace* es *nfms*, mientras que para el *datastore* es *geoserverdata*. En el caso de *layer* no se muestra ningún valor por defecto. Esto simplemente quiere decir que no existe un valor por defecto para esa opción. Nótese que `--layer` es obligatorio (no lleva corchetes en la descripción de uso).

Por su parte, los *scripts* de *Python* se utilizan exclusivamente para administrar el fichero *layers.json* de configuración del portal. Esto es así por simplicidad, ya que el manejo de ficheros *JSON* es mucho más simple con *Python* que con *bash*. El *script* de *Python* más sencillo es el que muestra el árbol de capas, `portal-layer-tree.py`:

```
$ portal-layer-tree.py
LAYER TREE
==========
root
        base
                innerbase
                        blue-marble (blue-marble)
                innerforest
                        forestClassification (forestClassification)
        admin
                countryBoundaries (countryBoundaries)
                provinces (provinces)
        landcover
                forest_mask (forest_mask)

MAP LAYER ORDER
===============
1. provinces
2. countryBoundaries
3. forest_mask
4. forestClassification
5. blue-marble
```

Como vemos, el árbol de capas tiene tres grupos *base*, *admin* y *landcover*. Es importante destacar que estos grupos tienen como padre a *root* que es el grupo que contiene todo, pero que no se mostrará en el portal. Cada grupo a su vez puede contener grupos, como *base*, que contiene *innerbase* y *innerforest*. Por último, los grupos contienen capas, que se muestran de la siguiente forma:

```
<capa_de_portal> (<capa_de_mapa 1>, <capa_de_mapa 2>, ...)
```

De esta manera es fácil distinguir las capas de portal de los grupos porque las capas siempre son las hojas del árbol y llevan las capas de mapa especificadas entre paréntesis.

Además, el script `portal-layer-tree.py` muestra el orden de las capas tal y como se mostrarán en el mapa. Así, en el ejemplo anterior la capa de provincias (*provinces*) se mostrará por encima de todas las demás, mientras que la capa base *blue-marble* estará debajo de todas las demás.

Por otro lado, existen *scripts* para gestionar los grupos de capas:

- `portal-add-group.py`: Añade un grupo vacío al árbol de capas del portal.
- `portal-set-group.py`: Modifica un grupo del árbol de capas. Con este *script* se puede modificar el nombre del grupo o moverlo a otro grupo.
- `portal-rm-group.py`: Elimina un grupo **vacío** del árbol de capas.

También existen *scripts* para las capas del portal:

- `portal-add-layer.py`: Añade una nueva capa al árbol de capas del portal.
- `portal-rm-layer.py`: Eliminar una capa del árbol de capas del portal.
- `portal-set-layer.py`: Modifica una capa. Este script es el más complejo de todos y acepta muchas opciones, todas ellas especificadas con detalle en la ayuda del *script*.

Avanzado

Por último, existen ciertos casos avanzados en los que se desea asociar más de una capa de mapa a una misma capa de portal. Esto puede ser útil, por ejemplo, para tener una capa de mapa que se dibuja mientras que otra, más simple, está

oculta y se utiliza para realizar peticiones. Para estos casos, es posible gestionar las capas de mapa a nivel individual con los siguientes *scripts*:

- `portal-add-map-layer`: Añade una nueva capa de mapa, asociándola con una capa de portal.
- `portal-rm-map-layer`: Eliminar una capa de mapa. Es necesario que la capa de portal asociada tenga más de una capa de mapa, ya que de lo contrario la capa de portal se quedaría sin ninguna capa de mapa.
- `portal-set-map-layer`: Modifica los valores de una capa de mapa. En el caso de que existan varias capas de mapa para una misma capa de portal, se ha de poder modificar una capa de mapa individualmente. En caso contrario (una única capa de mapa para una capa de portal), esta funcionalidad también está disponible con el *script* `portal-set-layer.py`.

Ejemplo

Carga de datos

En primer lugar, ya que la base de datos de PostGIS está vacía, hay que cargar ficheros *.shp*:

```
$ pg-load.sh --crs 4326 --file <fichero_shp>
```

En el caso de no especificar alguno de las opciones obligatorias, como en este caso `--crs` o `--file`, se mostraría un mensaje de error como el siguiente:

```
El sistema de referencia de coordenadas del fichero debe ser especificado. Ejemplo: --
↪crs 4326
```

En el caso de que haya que añadir todos los ficheros *.shp* de un directorio, es posible agilizar el proceso utilizando con el siguiente comando. Hay que recordar que es necesario que todos los ficheros *.shp* del directorio deben de estar en el mismo sistema de coordenadas:

```
$ multi-pg-load.sh --crs 4326 --folder <directorio>
```

Una vez se han cargado las capas en PostGIS, hay que añadirlas en GeoServer. En este punto, asumiremos que ya existe en GeoServer un espacio de trabajo y el almacén de datos correspondiente a la base de datos. Llegados a este punto podemos añadir las tablas de PostGIS a GeoServer una a una:

```
$ add-layer.sh --layer <nombre_de_la_tabla> --workspace <workspace> --datastore
↪<datastore>
```

Como se ha comentado anteriormente, en el caso de que el *workspace* sea *nfms* y el *datastore* sea *geoserverdata* es posible omitir esas opciones para utilizar los valores por defecto:

```
$ add-layer.sh --layer <nombre_de_la_tabla>
```

También es posible añadir varias tablas de manera interactiva:

```
$ multi-add-layer.sh
```

Este comando simplemente queda a la espera de que el usuario escriba el nombre de la tabla y pulse *Intro* tantas veces como necesite. Para terminar de añadir tablas es necesario pulsar *Ctrl + D*.

Además de introducir las tablas de manera interactiva, es posible también añadir con un único comando todas las tablas de una base de datos a GeoServer:

```
$ add-all-layers.sh
```

Configuración del portal

Una vez se han añadido los datos que necesitamos a la base de datos y GeoServer, podemos pasar a administrar las capas del portal. El caso más común es que ya exista un árbol de capas, bien porque sea el que viene por defecto como ejemplo cuando se despliega el portal, o bien porque esté ahí de una instalación anterior.

En cualquier caso, lo primero que haremos será mostrarlo para ver en qué estado tenemos las capas y grupos de nuestro portal. Esto lo haremos con el comando `portal-layer-tree.py`:

```
$ portal-layer-tree.py
LAYER TREE
==========
root
        base
                innerbase
                        blue-marble (blue-marble)
                innerforest
                        forestClassification (forestClassification)
        admin
                countryBoundaries (countryBoundaries)
                provinces (provinces)
        landcover
                forest_mask (forest_mask)


MAP LAYER ORDER
===============
1. provinces
2. countryBoundaries
3. forest_mask
4. forestClassification
5. blue-marble
```

Para este ejemplo supondremos que la única capa que nos es útil es *blue-marble* y que la queremos en el grupo *base*. Así que lo primero que haremos será eliminar todas las demás con el comando `portal-rm-layer.py` y volver a mostrar el árbol de capas:

```
$ portal-rm-layer.py --id forestClassification
$ portal-rm-layer.py --id countryBoundaries
$ portal-rm-layer.py --id provinces
$ portal-rm-layer.py --id forest_masl
$ portal-layer-tree.py
LAYER TREE
==========
root
        base
                innerbase
                        blue-marble (blue-marble)
                innerforest
        admin
        landcover


MAP LAYER ORDER
```

```
===============
1. blue-marble
```

Como podemos observar, al eliminar las capas se han quedado grupos vacíos que no vamos a necesitar. Estos grupos los eliminaremos con `portal-rm-group.py`:

```
$ portal-rm-group.py --id innerforest
$ portal-rm-group.py --id admin
$ portal-rm-group.py --id landcover
$ portal-layer-tree.py
LAYER TREE
==========
root
        base
                innerbase
                        blue-marble (blue-marble)


MAP LAYER ORDER
===============
1. blue-marble
```

Ahora podemos mover la capa *blue-marble* al grupo *base* y eliminar el grupo *innerbase*. Puesto que en este caso estamos modificando la capa (el grupo al que pertenece), utilizaremos el comando `portal-set-layer.py`:

```
$ portal-set-layer.py --id blue-marble --group base
$ portal-rm-group.py --id innerbase
$ portal-layer-tree.py
LAYER TREE
==========
root
        base
                blue-marble (blue-marble)


MAP LAYER ORDER
===============
1. blue-marble
```

Ahora que ya tenemos únicamente una capa base en el grupo que queríamos, podemos empezar a añadir nuevos grupos y capas para los datos que tenemos. Para empezar añadiremos dos grupos, uno para divisiones territoriales (*territoriales*) y otro para datos forestales (*forestales*). Esto lo haremos con el *script* `portal-add-group.py`:

```
$ portal-add-group.py --id territoriales --parent root --label "Divisiones␣
↪territoriales"
$ portal-add-group.py --id forestales --parent root --label "Datos forestales"
$ portal-layer-tree.py --id
LAYER TREE
==========
root
        base
                blue-marble (blue-marble)
        territoriales
        forestales


MAP LAYER ORDER
```

```
===============
1. blue-marble
```

Por último, podemos añadir capas a los grupos. Supondremos que anteriormente hemos añadido tres capas a GeoServer, dos con divisiones territoriales (provincias y municipios) y otro con datos de cobertura forestal. Añadiremos estas capas con `portal-add-layer.py`:

```
$ portal-add-layer.py --id provincias --url <url> --wmsName provincias --label
↪"Provincias" --group territoriales
$ portal-add-layer.py --id municipios --url <url> --wmsName municipios --label
↪"Municipios" --group territoriales
$ portal-add-layer.py --id cobertura --url <url> --wmsName cobertura --label
↪"Cobertura forestal" --group forestales
$ portal-layer-tree.py
LAYER TREE
==========
root
        base
                blue-marble (blue-marble)
        territoriales
                provincias (map-provincias)
                municipios (map-municipios)
        forestales
                cobertura (map-cobertura)


MAP LAYER ORDER
===============
1. map-cobertura
2. map-municipios
3. map-provincias
4. blue-marble
```

donde *<url>* es la URL del servicio WMS. Esta URL puede ser la URL de GeoServer en nuestro servidor (por ejemplo, `http://<servidor>/geoserver/nfms/wms`) o la URL de un servicio WMS externo.

Además, es importante destacar que cuando se añade una capa, se añade tanto una capa para el portal que se mostrará en el árbol de capas, como una capa para el mapa. Como se ha comentado anteriormente, las capas de portal van seguidas de las capas de mapa entre paréntesis. Además, para distinguirlas todavía más fácilmente, las nuevas capas del mapa se crean con el prefijo *map-*, como se puede ver arriba.

En este punto podemos querer cambiar el orden en que se mostrarán las capas. Por ejemplo, para hacer que la capa de provincias se muestre por encima de la de municipios podemos utilizar el comando `portal-set-map-layer.py`:

```
$ portal-set-map-layer.py --id map-provincias --order 2
$ portal-layer-tree.py
LAYER TREE
==========
root
        base
                blue-marble (blue-marble)
        territoriales
                provincias (map-provincias)
                municipios (map-municipios)
        forestales
                cobertura (map-cobertura)
```

```
MAP LAYER ORDER
===============
1. map-cobertura
2. map-provincias
3. map-municipios
4. blue-marble
```

Existen muchos más parámetros que se pueden configurar o cambiar tanto para las capas como para los grupos. Para ello podemos utilizar los *scripts* `portal-set-group.py`, `portal-set-layer.py` y `portal-set-map-layer.py`. Se recomienda ejecutar todos ellos con la opción `--help` para ver qué opciones admite y cómo utilizarlos.

CHAPTER 24

Autenticación

Note:

Fecha	Autores
2 Abril 2014	• Víctor González (victor.gonzalez@geomati.co)

Es posible que por diversos motivos se desee proteger el portal con usuario y contraseña. En este documento se explica cómo utilizar Apache para configurar la autenticación *HTTP Basic*.

En primer lugar, es necesario crear un directorio que contendrá el fichero de contraseñas:

```
$ sudo mkdir /etc/apache2/passwd
```

Posteriormente, deberemos añadir un usuario y contraseña a ese fichero:

```
$ sudo htpasswd -c /etc/apache2/passwd/basic nfms
New password:
Re-type new password:
Adding password for user nfms
```

Como se puede observar, el comando especifica el fichero donde se guardará la clave (`/etc/apache2/passwd/basic`) así como el usuario (*nfms*). Tras introducir la contraseña dos veces, el fichero está listo para ser usado.

Únicamente nos quedará configurar Apache para que utilice este fichero. Para ello editamos el fichero `/etc/apache2/conf.d/security`, añadiendo al final lo siguiente:

```
<Location />
        ProxyPass ajp://localhost:8009/
        ProxyPassReverse ajp://localhost:8009/
        AuthType Basic
```

```
        AuthName "NFMS4REDD"
        AuthUserFile /etc/apache2/passwd/basic
        Require valid-user
</Location>
```

donde *AuthUserFile* debe de coincidir con el fichero que hemos utilizado en el comando `htpasswd`.

Finalmente, únicamente nos queda recargar la configuración de Apache:

```
$ sudo service apache2 reload
```

CHAPTER 25

Checklist Rendimiento

Note:

Fecha	Autores
2 Julio 2015	• Oscar Fonts (oscar.fonts@geomati.co)

Esta lista de buenas prácticas se basa en la experiencia de múltiples instalaciones de GeoServer a lo largo de los años.

Pretende repasar las cuestiones básicas a tener en cuenta para un rendimiento aceptable de GeoServer, pero es una lista conservadora y razonable: No se han considerado situaciones excepcionales, ni se ha tratado de optimizar a toda costa. Se ha preferido conservar una instalación y mantenimiento lo más fáciles posible.

Esta lista se puede ir modificando o ampliando con todo aquello que se considere útil como buena práctica general. Los detalles sobre cómo realizar cada tarea están detallados a lo largo de este mismo manual.

Note: Para casos en que sea necesario profundizar: GeoServer on Steroids.

Requisitos Hardware

- Al menos 2 GB de RAM, hasta 4 si es posible.
- 4 núcleos de procesador.
- Si es posible, usar un disco lo más rápido posible (SSD local) para almacenar los datos y especialmente la GeoWebCache.

Sistema Operativo

- Utilizar versión más reciente de Ubuntu LTS: Ubuntu 14.04 Server 64 bits.
- Mantener actualizados los paquetes de sistema (apt-get upgrade & apt-get update).
- Activar la actualizaciones automáticas (unattended-upgrades).
- Instalar todo lo que sea posible del gestor de paquetes del propio SO (evitar en lo posible instalaciones manuales y repositorios de terceros).

Paquetes de Software

- Instalar GDAL (1.10)
- Instalar PostgreSQL (9.3) y PostGIS (2.1)
- Usar OpenJDK 7. Instalar manualmente las JAI e ImageIO nativas.
- Instalar Tomcat7 y configurar el /etc/default/tomcat convenientemente, reservando hasta 2 GB para la JVM, pero nunca más de la disponible, o el swapping nos arruinará el rendimiento).
- Usar la última versión de GeoServer 'stable' (2.7.x en Julio de 2015).

Configuración GeoServer

- Comprobar que el Data directory apunta a la ubicación deseada (típicamnete, /var/geoserver).
- Comprobar que la JVM que se está usando es la deseada (OpenJDK 7).
- Comprobar que native JAI y Native JAI ImageIO están a "true".
- Deshabilitar los servicios que no vayan a usarse.
- Borrar todos los workspaces, styles, datasources y layers que no se usen (típicamente, los que vienen de ejemplo).
- Configurar el logging a "PRODUCTION_LOGGING".
- Limitar el número de SRS del servicio WMS.
- Habilitar la creación de cachés en EPSG:900913 por defecto, donde:
  - Las capas Raster se cachearán en JPEG.
  - Las capas Vectoriales se cachearán en PNG8.
  - Los grupos de capas se cachearán en ambos formatos (JPEG, PNG8).
  - Si alguna capa ráster requiere de transparencia, se cambiará la configuración de GWC de JPEG a PNG8 sólo para ésa capa.

Datos Vectoriales

- Limpiar y simplificar los datos previamente:
  - Es preferible varios polígonos independientes que un sólo multipolígono muy complejo.

  - Evitar solapamiento y errores topológicos.
  - Simplificar las geometrías hasta una resolución lo suficientemente buena. Para geometrías muy detalladas, utilizar diferentes resoluciones a diferentes escalas.
  - Reproyectar al CRS en que se vayan a consumir más habitualmente los datos (para el Portal, EPSG:3857).
- Asegurarse que los datos están bien indexados:
  - Al importar los datos, utilizar la opción -I de shp2pgsql para generar automáticamente un índice espacial.
  - O bien, para datos ya cargados, comprobar que existe un índice o crearlo con CREATE INDEX [...] USING GIST.
  - Indexar también otros campos por los que se filtre habitualmente.
  - Comprobar que las tablas tienen definida una Primary Key (o, de no ser posible, activar OIDS para esa tabla).
- Tras actualizar una capa vectorial, ejecutar un VACUUM ANALYZE. Para capas que se editen contínuamente, programar la ejecución periódica de VACUUM ANALYZE.

Warning: Si GeoServer debe acceder a una Base de Datos PostGIS remota (alojada en otra máquina), asegurarse de que la conexión de red entre ambas máquinas es de excelente calidad, y si puede ser dedicada: la transferencia de datos va a ser masiva.

Datos Raster

- Usar el tipo de dato para cada celda lo más compacto posible. Por ejemplo, para un DEM quizá baste con un tipo Integer, y no haga falta un Double. El tipo Byte, con valores entre 0 y 255, suele ser más que suficiente para clasificaciones con unas pocas categorías, índices de vegetación, o indicadores en tanto por ciento.
- Reproyectar al CRS en que se vayan a consumir más habitualmente los datos (para el Portal, EPSG:3857), y recortar a la zona de interés (gdalwarp, gdalinfo).
- Convertir los datos a GeoTIFF sin comprimir, con tiling interno y con overviews (gdal_translate, gdaladdo).
- Cada fichero GeoTIFF debería rondar los 2 GB. Para mosaicos, trocear en caso de que pasen de 4 GB, o agrupar si quedan por debajo de 1 GB.
- Para coberturas enormes (por ejemplo, ortofoto nacional de gran resolución), hay que aplicar técnicas excepcionales: compresión interna JPEG, ImagePyramid.

Warning: No se recomienda que los datos raster (habitualmente GEOSERVER_DATA_DIR) estén en una unidad remota, puesto que la transferencia de datos para GeoTIFF sin comprimir es todavía más masiva que en el caso de datos vectoriales.

CHAPTER 26

Copias de seguridad

Note:

Fecha	Autores
10 Octubre 2014	• Fernando González (fernando.gonzalez@fao.org)

PostgreSQL

Ver documentación GeoTalleres: *Creación de copias de seguridad*

GeoServer

Ver documentación GeoTalleres, teniendo en cuenta que la instalación estándar del sistema establece `/var/geoserver/data` como directorio de datos de GeoServer: geoserver-backup

Portal

Las copias de seguridad de la configuración del portal son muy similares a las de GeoServer ya que la configuración está almacenada también en un directorio, generalmente `/var/portal`.

Warning: Antes de comenzar conviene revisar los permisos del directorio `/var/portal`, ya que el proceso se puede simplificar si dicho directorio pertenece al usuario administrador con el que vamos a realizar las siguientes operaciones. En cualquier caso, el directorio y todos sus ficheros deben estar accesibles en modo lectura para el usuario `tomcat7`, que es el que ejecuta la Tomcat7 que contiene el portal.

Ejecutando la siguiente instrucción podemos ver que el propietario del directorio del portal es `root`:

```
$ ls -l /var
nfms@nfms-server:/var/portal$ ls -l /var
total 56
drwxr-xr-x  9 tomcat7 tomcat7  4096 Oct  9 10:22 apache-tomcat-7.0.47
drwxr-xr-x  2 root    root     4096 May 27 06:47 backups
drwxr-xr-x 11 root    root     4096 May 27 06:46 cache
drwxrwsrwt  2 root    whoopsie 4096 May 27 06:25 crash
drwxr-xr-x  3 root    root     4096 Oct  9 09:15 geoserver
drwxr-xr-x 36 root    root     4096 Jun 10 18:36 lib
drwxrwsr-x  2 root    staff    4096 Aug 18  2013 local
lrwxrwxrwx  1 root    root        9 Dec 23  2013 lock -> /run/lock
drwxr-xr-x 12 root    root     4096 Oct  8 15:06 log
drwxrwsr-x  2 root    mail     4096 Dec 23  2013 mail
drwxr-xr-x  2 root    root     4096 Dec 23  2013 opt
drwxr-xr-x  6 root    root     4096 Jun  6 16:40 portal
lrwxrwxrwx  1 root    root        4 Oct  7 10:39 run -> /run
drwxr-xr-x  5 root    root     4096 Dec 23  2013 spool
drwxrwxrwt  2 root    root     4096 Aug 21 10:39 tmp
lrwxrwxrwx  1 root    root       20 Dec 23  2013 tomcat -> apache-tomcat-7.0.47
drwxr-xr-x  2 root    root     4096 Jan  9  2014 www
```

Suponiendo que accedemos con el usuario `nfms` podemos cambiar el propietario con la siguiente instrucción:

```
$ sudo chown -R nfms:nfms /var/portal
```

Nótese que es necesario usar `sudo` para cambiar el propietario del directorio ya que no pertenece a `nfms`. A partir de este momento ya pertenece a `nfms` y todas las operaciones siguientes pueden ser ejecutadas sin `sudo`. Si volvemos a mostrar los propietarios veremos que ha cambiado:

```
$ ls -l /var
nfms@nfms-server:/var/portal$ ls -l /var
total 56
drwxr-xr-x  9 tomcat7 tomcat7  4096 Oct  9 10:22 apache-tomcat-7.0.47
drwxr-xr-x  2 root    root     4096 May 27 06:47 backups
drwxr-xr-x 11 root    root     4096 May 27 06:46 cache
drwxrwsrwt  2 root    whoopsie 4096 May 27 06:25 crash
drwxr-xr-x  3 root    root     4096 Oct  9 09:15 geoserver
drwxr-xr-x 36 root    root     4096 Jun 10 18:36 lib
drwxrwsr-x  2 root    staff    4096 Aug 18  2013 local
lrwxrwxrwx  1 root    root        9 Dec 23  2013 lock -> /run/lock
drwxr-xr-x 12 root    root     4096 Oct  8 15:06 log
drwxrwsr-x  2 root    mail     4096 Dec 23  2013 mail
drwxr-xr-x  2 root    root     4096 Dec 23  2013 opt
drwxr-xr-x  6 nfms    nfms     4096 Jun  6 16:40 portal
lrwxrwxrwx  1 root    root        4 Oct  7 10:39 run -> /run
drwxr-xr-x  5 root    root     4096 Dec 23  2013 spool
drwxrwxrwt  2 root    root     4096 Aug 21 10:39 tmp
lrwxrwxrwx  1 root    root       20 Dec 23  2013 tomcat -> apache-tomcat-7.0.47
drwxr-xr-x  2 root    root     4096 Jan  9  2014 www
```

Así, para realizar una copia de seguridad, es necesario copiar este directorio, comprimido por comodidad y opti-

mización de espacio, a algún lugar fuera del servidor. Los siguientes comandos crearían una copia de la configuración en el fichero `/tmp/portal-backup.tgz`:

```
$ cd /var/portal
$ tar -czvf /tmp/portal-backup.tgz *
```

Nótese que el comando `tar`, encargado de la compresión, se debe ejecutar en el directorio $GEOSERVER_DATA. Las opciones `-czvf` especificadas significan:

- c: crear
- z: comprimir en zip
- v: verbose, muestra por pantalla los ficheros que se incluyen en la copia de seguridad
- f: fichero resultante, especificado a continuación

Warning: Es muy importante guardar los ficheros con la copia de seguridad en una máquina distinta al servidor que alberga el portal, ya que en caso de que haya algún problema con dicha máquina se pueden perder también las copias.

Para recuperar la configuración sólo tenemos que reemplazar el directorio `/var/portal` por los contenidos del fichero que contiene la copia. Para ello se puede descomprimir la copia de seguridad en un directorio temporal:

```
$ mkdir /tmp/copia
$ tar -xzvf /tmp/portal-backup.tgz --directory=/tmp/copia
```

A diferencia del comando `tar` que utilizamos para crear la copia de seguridad, ahora estamos usando la opción `x` (extraer) en lugar de `c` (crear) y estamos especificando con la opción `--directory` que queremos extraer la copia en el directorio `/tmp/copia`.

Una vez descomprimido sólo hay que reemplazar los contenidos del directorio `/var/portal` por los del directorio `/tmp/copia`. Por seguridad, moveremos los contenidos actuales a otro directorio temporal:

```
$ mkdir /tmp/portal_actual
$ mv /var/portal/* /tmp/portal_actual/
```

Tras estas dos instrucciones el directorio /var/portal estará vacío y tendremos los contenidos actuales en `/tmp/portal_actual/` y la copia en `/tmp/copia`. Por tanto, sólo tenemos que copiar los contenidos de `/tmp/copia` a `/var/portal`:

```
$ cp -R /tmp/copia/* /var/portal
```

Por último, quedaría reiniciar el portal, que se ejecuta dentro de Tomcat7:

```
$ sudo service tomcat7 restart
```

Versiones actuales del software

Además de los directorios de configuración, es conveniente guardar una copia de los programas instalados, básicamente:

- Fichero WAR del portal.
- Scripts de administración.

La forma más sencilla es guardar una copia de los ficheros que se instalan y repetir la instalación de los mismos en caso necesario.