Wjordpress Documentation

Release 0.2.1

Chris Chang

November 11, 2014

Contents

Contents:

CHAPTER 1

Other Django-Wordpress Integration Projects

- Django-Wordpress – Interfaces directly with the WordPress database. Gives you models and views for directly working with WordPress. (Sunlight Foundation)
- Django-Wordpress (agiliq) – Another project that lets Django read WordPress database tables.
- Django Wordpress RSS – Uses the RSS feed as an api to proxy content from WordPress to Django. No models/database required. (Seattle Times)

CHAPTER 2

Webhooks

To automatically keep Django and WordPress in sync, you should install the HookPress plugin.

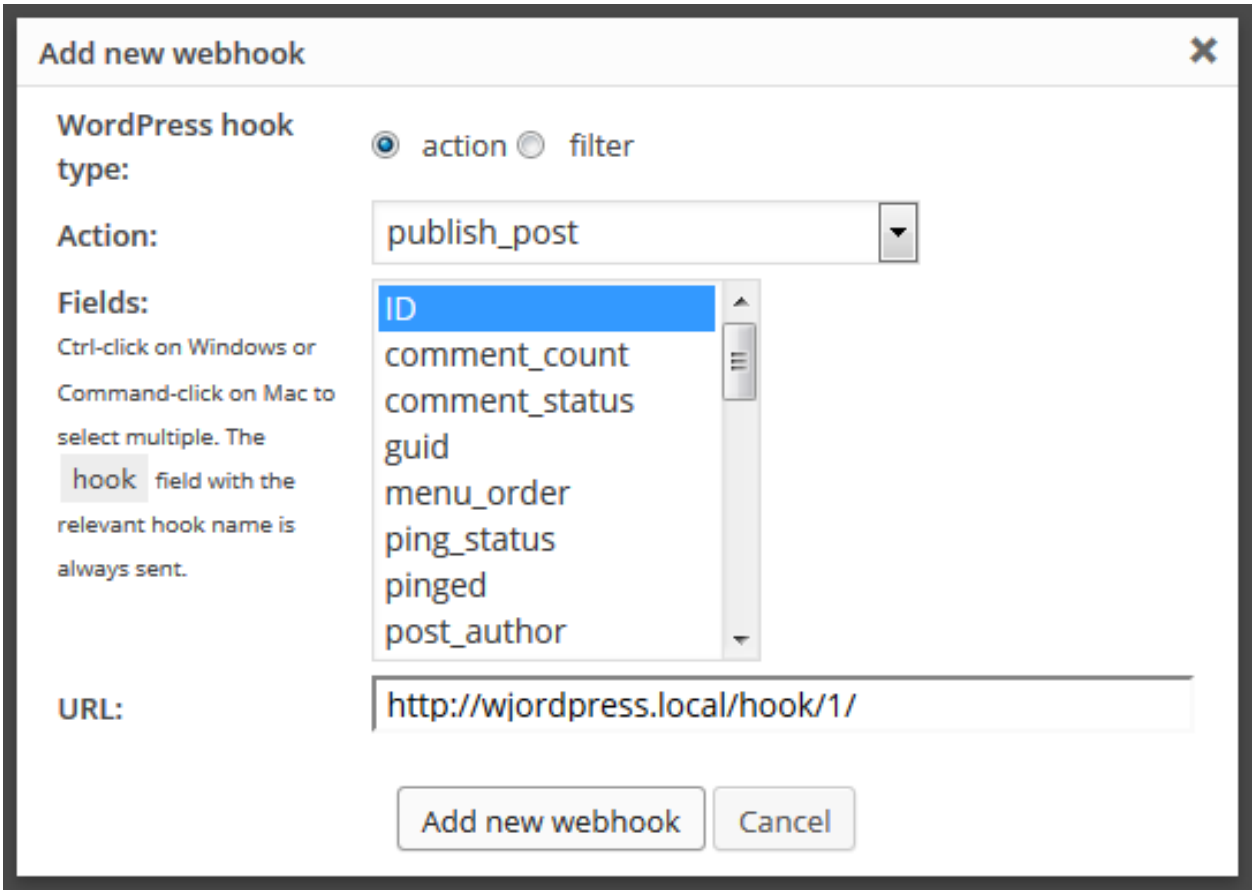
Saving will now be slower in WordPress because there are additional synchronous requests to the webhook on every save. Later, in a separate thread, Django asks WordPress about details of post. When you save a post, two webhook calls are made. One for the updated post, and one for the new revision.

2.1 Installing HookPress

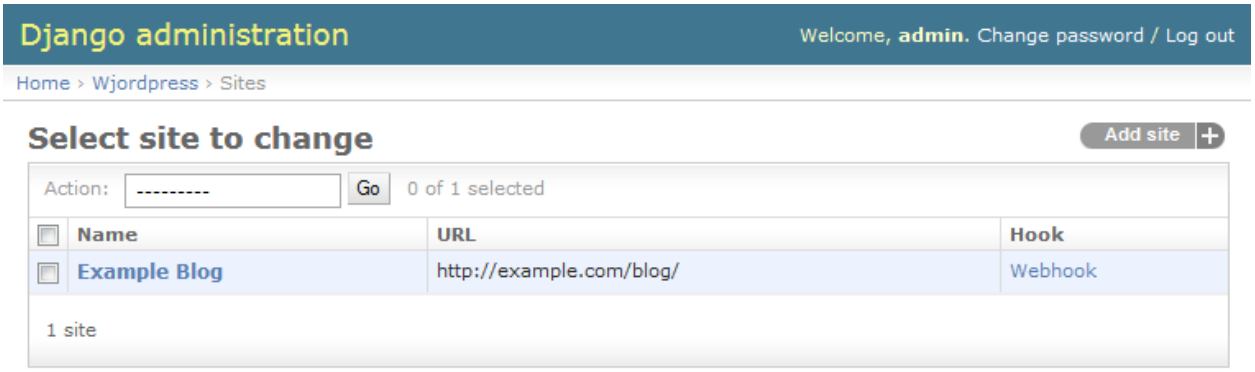
It's just like installing any other WordPress plugin. Starting from your admin dashboard, you go to "Plugins", then "Add New", search for "hookpress", and then install.

2.2 Configuring HookPress

Starting again from the admin dashboard, go to "Settings", then "Webhooks", then "Add webhook". At the least, you should set up a `publish_post` hook that sends the *ID* field back like this:



To get the url, go to the Wjordpress section of the Django admin and copy the "Webhook" url:



CHAPTER 3

WordPress Quirks

3.1 About Revisions

3.1.1 What happens when you update a post?

Let's say you have a post (ID=1). When you update it, it's saved and a copy of the post is created at the next available id (ID=2). So if you check the `posts/1` json, you'll see:

```
{
    ID: 1,
    status: "publish",
    type: "post",
    ...
}
```

And if you check the json for the revision, `posts/2`, it looks like:

```
{
    ID: 2,
    status: "inherit",
    type: "revision",
    ...
}
```

If you're using the save hook WordPress plugin, HookPress, you'll get two pings, one for the original post and one for the revision. If you look at the Django models, you'll find the post with ID=1 (`original = WPPost.objects.get(id=1)`) and the revision post (`revision = WPPost.objects.get(id=2)`) and the revision will be linked to the original (`revision.parent == original`). So the original ID is retained as the post is updated.

3.1.2 What happens you restore a revision?

When you revert to a revision, this is treated as if you updated the post. So the original id remains unchanged and another revision is created.

CHAPTER 4

Changelog

4.1 0.2.1

- Fix how things like author weren't set when creating/updating posts from webhook
- Better log viewing by letting you use your own browser json extensions
- Freshen requirements
- Fix this did not work in Python3

4.2 0.2.0

- Add ability to deal with image posts
- Add integrated logs so admins can debug communication
- Add a templatetag for quickly showing WordPress content
- Fix jsonfield packaging

4.3 0.1.0

- Initial release. Project can pull in WordPress posts

Since time immemorial, there has been effort after effort to make the Django admin friendlier for writers to use as a CMS interface. The goal always being: "Can we make Django as easy to use as WordPress?"

Well, instead of trying to make the Django admin ack like WordPress, why not use WordPress and feed the data into Django? That is the goal of Wjordpress.

CHAPTER 5

Installation

5.1 Adding Wjordpress to your Django Project

1. Install Wjordpress `pip install wjordpress` / update your requirements.
2. Add Wjordpress to your installed apps:

```
INSTALLED_APPS = [
    # ... your other installed apps
    'wjordpress',
]
```

3. Initialize database tables using `manage.py syncdb`. *Because the project is still in alpha, migrations have not been checked in.*
4. (Optional) For webhook support, add a route to Wjordpress's urls:

```
urlpatterns = patterns('',
    # ... your other url root patterns
    url(r'^_hooks/', include('wjordpress.urls',
        namespace='wjordpress', app_name='wjordpress')),
)
```

5.2 Setting Up Existing WordPress Site

1. Install the JSON REST API plugin
2. *(optional)* Install the HookPress plugin

5.3 Adding your first WordPress site

1. In the Django Admin, add a Wjordpress Site
2. For the URL input, use the same url you'd use to browse to the site
3. Save. Whenever you save a site in admin, the most recent 10 posts will be pulled.
4. You can add additional WordPress sites so one Django site can integrate with many WordPress sites.

CHAPTER 6

Next Steps

6.1 Automatically keep the Django site up to date

If you installed the HookPress WordPress plugin, you can set up a `save_post` webhook that will ping the Django site to update whenever you update a post. In the Django Admin change list for Wjordpress sites, there's a column, "Hook", for the url to use as the webhook url. In the WordPress admin, add a `save_post` hook to this url. Make sure the `ID` field is sent (this happens by default). See the *Webhooks* page for more detail.

6.2 Manually sync the Django site

Run the `manage.py wjordpress_fetch` management command.

6.3 Inspect communication between Django and WordPress

If you enabled logging when you added your WordPress site (this is on by default), you can see what communication has occurred between the two in the Django Admin at Wjordpress > Logs.

6.4 Embedding a WordPress site

Wjordpress comes with a templatetag so you can quickly insert a widget of recent posts. If your WordPress site was called "Mollusk Life", in your Django template HTML you would add something like:

```
{% load wjidget from wjordpress %}
{% wjidget "Mollusk Life" limit=5 %}
```

You need to add your own css to style the widget. All the css class names are namespaced with the `wjordpress-` prefix.

6.5 Using Your Own Django Models

If you want to sync WordPress content to your own models, you can write `post_save` signals. For an example, see the models and signals in the example app.