
django-feedmapper Documentation

Release 1.0

National Geographic

August 18, 2015

1	Installation	3
2	Parsers	5
3	Mapping	7
3.1	An example: users	7
4	Scheduling	11
4.1	Using django-celery	11
4.2	Using feedmapper_sync	12
5	Contributing	13
6	Issues	15
7	Contents	17
7.1	Reference	17
8	Indices and tables	19

django-feedmapper is a library for synchronizing data from feeds with Django models. The process of synchronizing the data requires the use of three pieces: a parser, a mapping, and a schedule.

Installation

Install from PyPI:

```
pip install django-feedmapper
```

Add feedmapper to your settings.py file:

```
INSTALLED_APPS = (  
    ...  
    'feedmapper',  
    ...  
)
```

If you are using South, run the migrations:

```
./manage.py migrate feedmapper
```

Otherwise, run syncdb:

```
./manage.py syncdb
```

Parsers

A parser defines methods for validating and parsing data from incoming feeds. There are two built-in parsers, *XMLParser* and *AtomParser*. You can write your own parser by subclassing the base *Parser* class.

Mapping

A mapping is written in JSON and describes how and when data from an incoming feed should be mapped to Django models. You can perform the following types of mappings:

- One field in a model to one field from a feed
- One field in a model to multiple fields from a feed
- One field in a model to a transformer method on the model

You can also set the following properties on a mapping through the Django admin:

- Data source
- Synchronization schedule
- Purge existing data

3.1 An example: users

Let's get into an example. Suppose we have the following incoming XML data and we want to map each `<user>` to Django's `User` model:

```
1  <?xml version="1.0" ?>
2  <auth>
3    <users>
4      <user>
5        <username>vader</username>
6        <first_name>Anakin</first_name>
7        <last_name>Skywalker</last_name>
8        <email>vader@sith.org</email>
9        <date_joined>2050-01-31T20:00-4:00</date_joined>
10     </user>
11    <user>
12      <username>kenobi</username>
13      <first_name>Obi-Wan</first_name>
14      <last_name>Kenobi</last_name>
15      <email>kenobi@jedi.org</email>
16      <date_joined>2000-01-31T20:00-4:00</date_joined>
17    </user>
18  </users>
19 </auth>
```

We need to specify a JSON map:

```
1 {
2   "models": {
3     "myapp.Thing": {
4       "nodePath": "users.user",
5       "identifier": "username",
6       "fields": {
7         "username": "username",
8         "email": "email",
9         "name": ["first_name", "last_name"],
10        "date_joined": {
11          "transformer": "convert_date",
12          "fields": ["date_joined"]
13        },
14      }
15    }
16  }
17 }
```

Let's break this down a bit. First, we can specify one or more models to map:

```
1 "models": {
2   "myapp.Thing": {
```

We need to tell the parser the path to all of the <user> elements:

```
1 "nodePath": "users.user",
```

If the mapping has purging turned off, we need to supply a unique identifier for Django ORM `get` calls. In this case our resulting ORM call would be `User.objects.get(username=username)`:

```
1 "identifier": "username",
```

Now the fun part. Mapping the fields:

```
1 "fields": {
2   "username": "username",
3   "email": "email",
4   "name": ["first_name", "last_name"],
5   "date_joined": {
6     "transformer": "convert_date",
7     "fields": ["date_joined"]
8   },
9 }
```

We've got example of all three types of field mappings here.

`username` and `email` are one-to-one mappings:

```
1 "username": "username",
2 "email": "email",
```

`name` is mapped to multiple fields. The parser will concatenate these fields, putting a space between them:

```
1 "name": ["first_name", "last_name"],
```

`date_joined` uses a transformer, which is simply a method defined on your model to do some manipulation to the incoming data before inserting it in a field. Here we tell the parser that the `date_joined` field should map to the `date_joined` field in the XML but use the `convert_date` method to transform the incoming data:

```
1     "date_joined": {
2         "transformer": "convert_date",
3         "fields": ["date_joined"]
4     },
```

Scheduling

There are two ways to schedule the synchronization of mappings.

4.1 Using django-celery

The first scheduling method, and the preferred, is to use `django-celery`. To take advantage of this scheduling method, take the following steps:

1. Install `django-celery`. If you've never done this before, it can be a little complicated. You'll want to read through the [official docs](#). An example of some basic settings is in `example/settings_celery.py`:

```
1 from .settings import *
2
3 import djcelery
4 djcelery.setup_loader()
5
6 INSTALLED_APPS += ('djcelery',)
7 CELERYBEAT_SCHEDULER = 'djcelery.schedulers.DatabaseScheduler'
8 BROKER_HOST = "localhost"
9 BROKER_PORT = 5672
10 BROKER_USER = "guest"
11 BROKER_PASSWORD = "guest"
12 BROKER_VHOST = "/"
```

2. Make sure you enable the Django database scheduler of `django-celery` by adding the following to your `settings.py` file:

```
CELERYBEAT_SCHEDULER = 'djcelery.schedulers.DatabaseScheduler'
```

Now every time you save a mapping, it will either create or update a matching `django-celery` `PeriodicTask` in the database. By default the periodic task will run once an hour. If you want to change this, visit the `PeriodicTask` in the Django admin (`/admin/djcelery/periodictask/` by default) and modify the interval or crontab settings:

Change periodic task

Name:	<input type="text" value="Huffington Post Politics"/>
	Useful description
Task (registered):	<input type="text"/>
Task (custom):	<input type="text" value="feedmapper.tasks.feec"/>
<input checked="" type="checkbox"/> Enabled	
Schedule	
Interval:	<input type="text" value="-----"/> <input type="button" value="+"/>
Crontab:	<input type="text" value="0 * * (m/h/d)"/> <input type="button" value="+"/>
	Use one of interval/crontab
Arguments (Show)	
Execution Options (Show)	
<input type="button" value="✖ Delete"/>	

4.2 Using feedmapper_sync

Of course, not everyone has resources or need to use a message queue solution. The second scheduling method is by setting up a cron job and using the `feedmapper_sync` management command. Make sure you have the `DJANGO_SETTINGS_MODULE` environment variable set and add the following to your crontab:

```
* * * * * /full/path/to/bin/django-admin.py feedmapper_sync
```

If you only want to sync a subset of the mappings you can supply one or more mapping IDs to the management command:

```
* * * * * /full/path/to/bin/django-admin.py feedmapper_sync 3 8 22
```

Contributing

To contribute to django-feedmapper [create a fork](#) on github. Clone your fork, make some changes, and submit a pull request.

Issues

Use the [github issue tracker](#) for django-feedmapper to submit bugs, issues, and feature requests.

7.1 Reference

7.1.1 Parsers

class `feedmapper.parsers.Parser(mapping)`

Base parser class for mapping Django model fields to feed nodes.

generate_filter_kwargs (*filter_string*)

Convert a string to kwargs that can be passed to the Django ORM's filter method. For example, 'slug__icontains="darth", name="Anakin"' will get converted to {'slug__icontains': 'darth', 'name': 'Anakin'}.

notify_failure (*subject=None*)

Notify recipients, if specified, of an error during parsing.

validate_model_format (*model_string*)

Validate that a model in the JSON mapping is in the format app.model.

class `feedmapper.parsers.XMLParser(mapping)`

A parser for XML that does not follow any standard.

get_value (*node, path*)

Attempts to retrieve either the node text or node attribute specified.

join_fields (*node, fields*)

Joins the text for the specified fields.

parse ()

Traverses through the XML document and parses the data, applying it to the model specified in the Mapping.

class `feedmapper.parsers.AtomParser(mapping)`

An XML parser for the Atom standard.

7.1.2 Mappings

Indices and tables

- `genindex`
- `modindex`
- `search`

A

AtomParser (class in feedmapper.parsers), 17

G

generate_filter_kwargs() (feedmapper.parsers.Parser method), 17

get_value() (feedmapper.parsers.XMLParser method), 17

J

join_fields() (feedmapper.parsers.XMLParser method),
17

N

notify_failure() (feedmapper.parsers.Parser method), 17

P

parse() (feedmapper.parsers.XMLParser method), 17

Parser (class in feedmapper.parsers), 17

V

validate_model_format() (feedmapper.parsers.Parser method), 17

X

XMLParser (class in feedmapper.parsers), 17