Factory Boy Fun

Adam Johnson - me@adamj.eu

9th September 2014

What the problem?

- You've done this, right?

```
# tests/test_something.py
class MyTests(TestCase):
    fixtures = ['basic.json']
    def setUp(self):
        self.user = User.objects.create(
            username='adam',
            first_name='Adam',
            last_name='Johnson',
            email='adam@example.com'
        )
        # ...
    # (and some tests, I hope!)
```

```
# tests/test_something.py
class MyTests(TestCase):
    fixtures = ['basic.json']
    def setUp(self):
        self.user = User.objects.create(
            username='adam',
            first_name='Adam',
            last_name='Johnson',
            email='adam@example.com'
        )
```

Pain...

- Test data in two places!
- A quirky json file that has to be maintained separately!
- I *just* want a User but I have to give *every* detail!

Model-building can overtake testing

- Let's fix that... with a package ported from Ruby on Rails. (Trust me, it's gonna be okay!)

Factory Boy

- a fixtures replacement based on thoughtbots 'factory_girl'.

- http://factoryboy.readthedocs.org/en/latest/

Example factory

- Doesn't fit on one slide... here's the imports:

```
# app/factories.py
from datetime import datetime, timedelta
from random import randint
from django.template.defaultfilters import slugify
from factory import DjangoModelFactory, lazy_attribute

now = datetime.now
```

```
class User(DjangoModelFactory):
    class Meta:
        model = 'auth.User'
        django_get_or_create = ('username',)

    first_name = 'Adam'
    last_name = 'Johnson'

    @lazy_attribute
    def username(o):
        return slugify(o.first_name + '.' +
                       o.last_name))

    @lazy_attribute
    def email(self):
        return self.username + "@example.com")

    @lazy_attribute
    def date_joined(self):
        return now() - timedelta(days=randint(5, 50))

    @lazy_attribute
    def last_login(self):
        return self.date_joined + dt.timedelta(days=4))
```