Django As A Second Language

Eric Snyder
Chariot Solutions

Who Is Django?

- Superb jazz guitarist
- 1910 – 1953
- Member of The Quintet Of The Hot Club Of France

What Is Django?

- Python based web framework. A dynamically typed, interpreted language (like Ruby).
- MTV (Model Template View) – really MVC (like Rails).
- 'Fat model' model (like Rails).
- Originated from a 'real world' application (like Rails).

What Is Rails?

- Ruby based web framework. A dynamically typed, interpreted language (like Python).
- MVC (like Django).
- 'Fat model' model (like Django).
- Originated from a 'real world' application (like Django).
- Emphasizes 'Convention Over Configuration'.

Python Philosophy

- Rejects arcane language features.
- Priority on readability over expressiveness. Whitespace is significant.
- Explicit is better than implicit.
- There should be only one obvious way to do anything.

Ruby Philosophy

- Has many arcane features. See the splat * for an example or referencing the eigenclass.
- Not as much emphasis on readability (IMHO).
- Implicit is ok. There are many examples in Rails.
- There can be many ways to do the same thing. This is a philosophy inherited from Perl.

Riddle Me This – A Ruby Example

```
class << self; self; end.send(:define_method, :do_magic)
    do |a|
    puts "Umm, yeah..."
    p a
end
```

Applications & Projects

- Rails: An application consists of models, controllers, routing information, templates and configuration information.
- Django:
  - A project consists of configuration information, some routing info. and one or more applications.
  - An application consists of routing info., models, views (controllers) and templates.

Routing - Rails

Starting w. 1.2 we have RESTful routes

map.resources :teams

teams_path	GET	/teams	index
teams_path(id)	GET	/teams/1	show
new_team_path	GET	/teams/new	new
teams_path	POST	/teams	create
edit_team_path	GET	/teams/1;edit	edit
team_path(id)	PUT	/teams/1	update
team_path(id)	DELETE	/teams/1	destroy

Routing - Django

Map requests to views with regex.

```
urlpatterns = patterns('',
    (r'', include('timetobrew.brewtools.urls')),
    (r'^accounts/login/$', 'django.contrib.auth.views.login'),
    (r'^admin/', include('django.contrib.admin.urls')),
    (r'^brewtools/', include('timetobrew.brewtools.urls')),
)
```

Django Models

- Explicit declaration of everything, no introspection.
- Field types can extend beyond the underlying database datatypes. E.g. URLField, XMLField, FilePathField.
- Rich set of attributes can be set on a field. E.g. editable, help_text, unique_for_date.

Rails Models

- Database introspection for attribute metadata.
- By default attribute names are assumed to be the same as database column names.
- Relationships explicitly specified.
- Rich and robust model level validation.

Django Views

- Controllers and actions. Deal with request and response.
- Form handling.
  - Form level validation.
  - Form 'cleaning' of data (conversion).
- Exchange data w. templates via a Context dictionary.

Django Generic Views

- Common controller patterns abstracted.
  - Master/detail.
  - Date based drill down.
  - Create/Update/Delete.
- No controller code necessary.
- Templates still required.

Rails Controllers

- Controllers and actions. Deal with request and response.
- Injects controller instance variables into the template.
- Implicit rendering of like named templates.
- No built-in concept of forms.
  - No conversion phase (JSF).
  - No form level validation. This complicates model level validation.

Django Templates

- Block inheritance.
- Simple set of template 'tags'.
- Restricted to simple logic. This is a good thing.
- Forms can be rendered by simply printing them. Validation messages are rendered as well.

Rails eRb Templates

- Mix ruby code and HTML.
- Can use view helpers which can function like tags.
- Easily abused.
- Partials – partial pages used to reduce clutter and enable AJAX requests to replace DOM elements.

AJAX Support

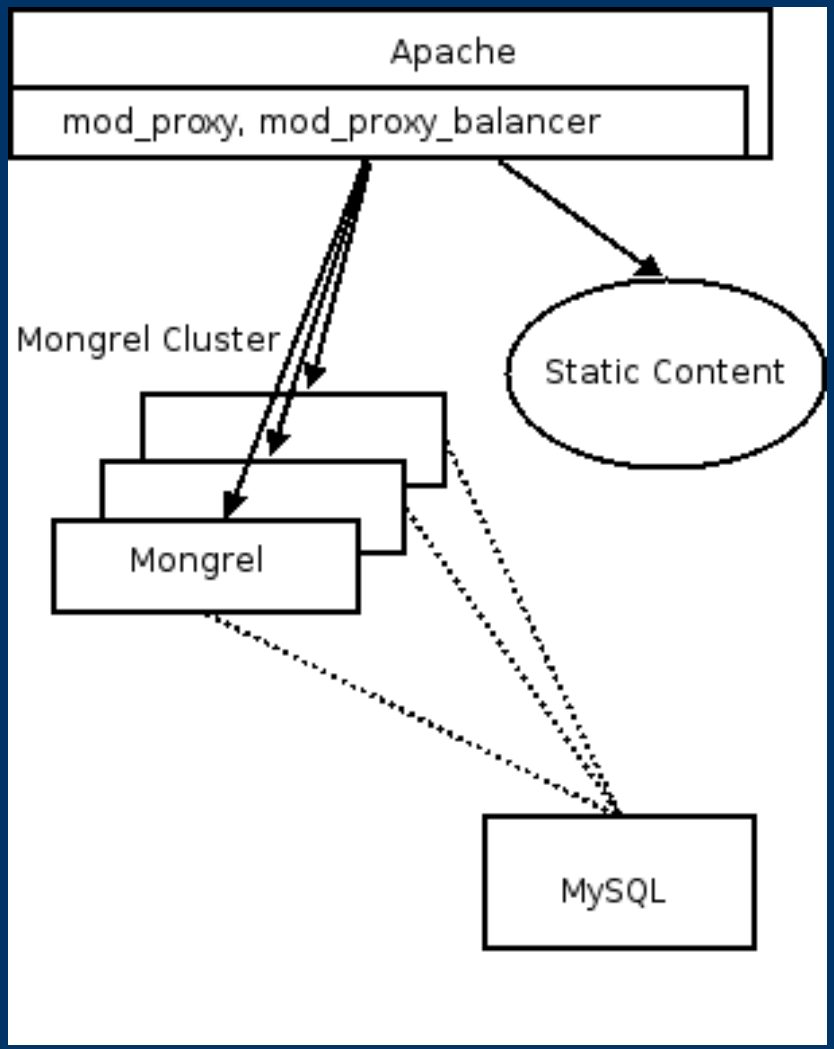
- Rails: Directly integrated with prototype and scriptaculous.
  - Rails view template helpers generate calls to prototype functions.
  - Can call prototype and scriptaculous functions directly in Rails RJS templates .
- Django: No AJAX integration (yet).
  - Some discussion of Dojo integration.
  - Contrary to Django philosophy.

Managing The Schema

- Rails ActiveRecord Migrations
  - Manage the evolution of a schema.
  - DDL
  - DML
  - Database agnostic
- Django
  - manage.py custom command - execute SQL for a model, once only
  - fixtures – load data from flat files, format is JSON or XML

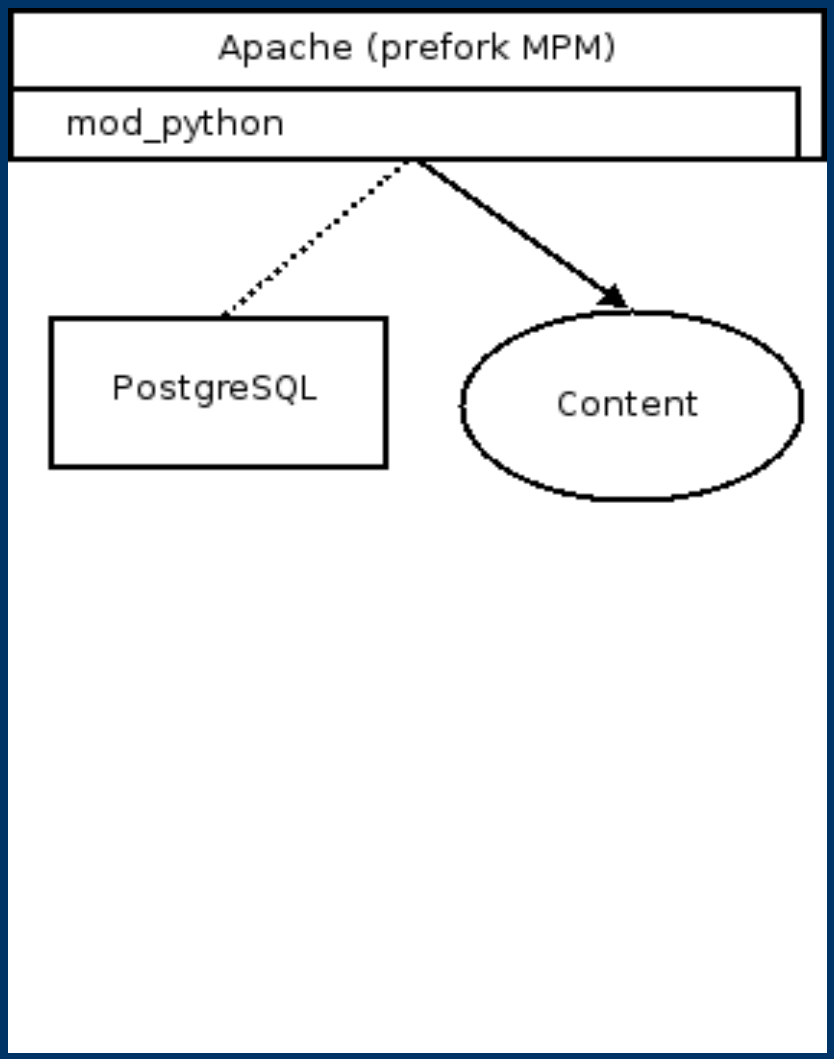
Rails Deployment With Mongrel

- Mongrel
  - HTTP Server
  - Written In Ruby
- Rails is not thread safe. Each request gets its own process.
- Could be simpler but mod_ruby is broken.
- FastCGI is another alternative.



Django Deployment With mod_python

- Django is not thread safe (?) but prefork MPM means each request gets its own Apache process.
- FastCGI is another alternative.



My Observations

- Django is generally on par with Rails in terms of how productive you can be.
- Django's philosophy of favoring the explicit over the implicit is appealing.
- Django is more flexible and less opinionated.
- Django's reluctance to embrace AJAX is a weakness.
- Python's richer set of libraries gives Django an advantage, especially in the enterprise.
- Django's deployment model is simpler and well proven.