django

Presentation to Muug
June 10th, 2008
by
Bill reid

django

A Web framework is a collection of packages or modules which allow developers to write Web applications.

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

Why I was interested in a Web Framework

I had developed a MS Access application using ODBC to a MySQL database. I wanted to move it to a Web frontend. I was keen of Ruby on Rails but was a Python user so looked for a comparable framework.

Other Web Frameworks

- Ruby on Rails
  - Getting a lot of press. Very popular.
- Other Python frameworks
  - Pylon
  - Turbogears
  - Zope
- Lots of Java and PHP choices

django

- History
  - Initially developed for a newspaper web site World Online in Lawrence, Kansas
  - Started in 2003 by Adrian Holovaty and Simon Willison.
  - 2005 open sourced the software. Named after Django Reinhardt, a famous Blegium jazz guitarist.
  - Still pre-release 1.0 but ... a lot of production sites.
  - Not ready to maintain backwards compatibility

MVC framework?

- **M**odel (database), **V**iew (presentation), **C**ontroller(logic connecting input to view)
  - Ruby on rails
- Django is more like **M**odel(database), **T**emplate(presentation), **V**iew(logic)
  - Connecting input to view is handled by the django framework (URLconf)

Design Philosophies

- Loose coupling
- Less code
- Quick Development
- Don't repeat yourself (DRY)
- Explicit is better than implicit (minimum magic)
  - A core Python principle (import this)
- Consistency

django Framework

- Object relation Mapper (ORM)
  - Define data model entirely in Python
- Automatic admin interface
- Elegant URL design
  - No page.php script.cgi?pageid=12 0,123,2.htm
- Template system
  - Separate design, content and Python code
- Cache system
- Internationalization

Why django?

- Uses Python, a dynamic, object oriented programming language.
- Modular design – loosely coupled
  - Encourages pluggable applications
- Well documented
  - 4 part tutorial
  - Online book
- Very stable
- Active community

Models

- Explicit better than implicit
  - Field names should not imply behavour
- Include all relevant domain logic
- Database API
  - SQL effciency
    - MySQL, PostgreSQL, SQLite, Oracle
  - Terse powerful syntax
    - Joins automatic
  - Drop into raw SQL if required

Let's start

- Easy install instructions
- I use the SVN trunk
- For testing you can use SQLite and the builtin Web server
- Comes with a bash completion script
- First create a project, then an application.

Templates

- Separate logic from presentation
- Discourage redundancy
  - Template inheritance
- Decoupled from HTML
  - Should be able to output other text formats
- Treat white space obviously
- Not a programming language
- Safety and security
- Extensibility

```
<html>
<head><title>Ordering notice</title></head>
<body>

<p>Dear {{ person_name }},</p>
<p>Thanks for placing an order from {{ company }}. It's scheduled to
ship on {{ ship_date|date:"F j, Y" }}.</p>

<p>Here are the items you've ordered:</p>
<ul>
{% for item in item_list %}
<li>{{ item }}</li>
{% endfor %}
</ul>
{% if ordered_warranty %}
<p>Your warranty information will be included in the packaging.</p>
{% endif %}
<p>Sincerely,<br />{{ company }}</p>

</body>
</html>
```

Other stuff

- Form Processing
- Generic views
- Sessions, Users, and Registration
- Middleware (i.e hooks)
- Deployment (mod_python)
- Ajax – not in the admin interface.
  - Dojo and jQuery are often used
- Community contributed modules