django

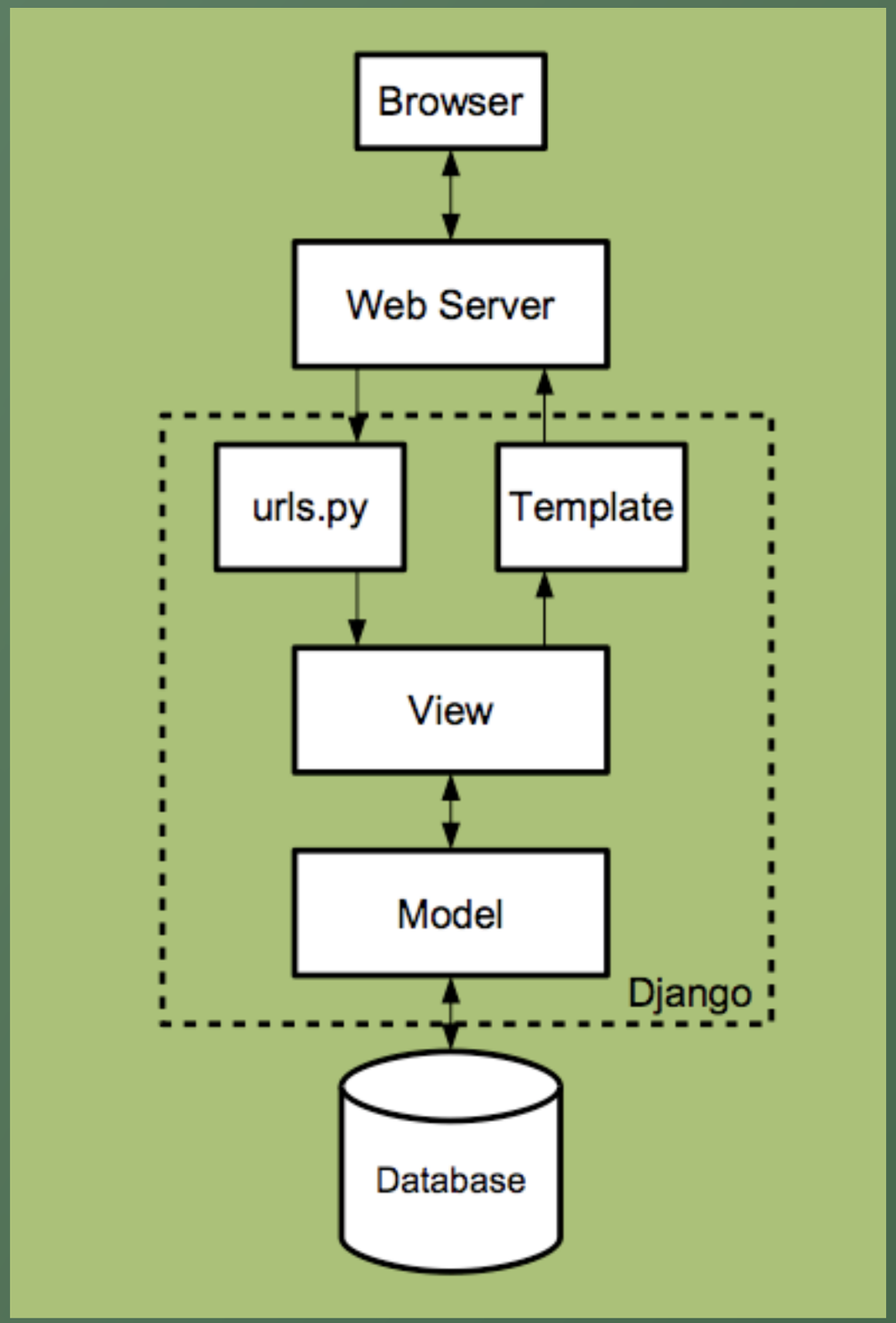
State of Pluggable Applications

Mike Cantelon
http://mikecantelon.com

Open Web Vancouver, 2008

Django in Brief

- An “MTV”–based framework (model, template, view)
- Favourite web framework of Python’s creator
- Good balance between speed and abstraction
- Little “magic” to inject the unexpected
- newforms abstracts form handling



From “A Rails/Django Comparison” by Ben Askins and Alan Green

What's New

- Trunk is kept production ready: no 1.0 release date yet
- Unicode now used throughout
- Templates now auto-escape by default
- Google AppEngine runs Django
- Oracle support added

What's in the Works

- Django Software Foundation
- newforms in admin for easier customization
- Improvements to Django's ORM
- More books about Django

New to Python?

- Culture distrusts cruft and hype
- Fond of lightweight markup languages
- Documentation often in reStructuredText
- Python Docutils allow formatting to HTML, etc.
- Play with reST at rst2a.com

Pluggable Applications

Pluggable Lingo

- Projects are made up of one or more "applications"
- Applications can be site-specific or designed for reuse
- When designed for reuse, applications are called "pluggable"

Pluggable Realm

- The Django pluggable application space is still a “Wild West”
- Plenty of room to get involved and make a difference

Pluggable Anatomy

- Pluggable apps are made up of models, templates, template tags, and views
- As loosely coupled with projects as possible

Pluggable Architecture

- Architecture may evolve
- No machine-readable metadata
- Work is being done by django-hotclub (find them in Google Groups and on #django-hotclub IRC)
- django-reusableapps is a project being developed for management of pluggable applications

Finding Pluggables

Standard Pluggables

- admin: instant CRUD of model data
- databrowse: end user interface to explore model data
- auth: user authentication framework
- syndication: RSS/Atom framework
- etc.

More Pluggables

- Most pluggable applications live at Google Code
- Popular naming convention is “django-<appname>”
- Pluggable application suites like Sphene provide apps guaranteed to integrate well

Online Registries

- http://djangoapps.org
- http://djangoapps.net
- http://djangopluggables.com
- http://code.djangoproject.com/wiki/DjangoResources

Vetting Pluggables

- Does the app try to do too much?
- Google Code lets you examine before downloading
- Is there decent documentation?
- When was the last commit?
- Is it actually pluggable?

If it contains “settings.py”
it’s probably not pluggable.

Managing Pluggables

- Maintain a reference directory for pluggables
- This allows access to sample templates, documentation, etc.

Installing Pluggables

- Many are dependent on Python modules and other pluggable apps
- Most pluggable apps and modules are installed by “python setup.py install”
- Others can be copied into your Python path or project directory
- Copy sample templates into your project’s templates directory

Configuring

- Pluggables normally hook into your project via “settings.py” and “urls.py”
- In “settings.py”, pluggables are added to INSTALLED_APPS
- In “urls.py”, add to urlpatterns

Blogging-Related

django-tagging

- Query tag intersections and unions
- Del.icio.us-style URL-based tag queries
- Template tags for display of tags and tag clouds
- Special tag widget for newforms

django-diario

- Full-featured blogging pluggable
- Leverages django-tagging
- RSS/Atom feed support
- XML sitemap support
- Pretty permalinks

Basic Configuration
django-diario

- Move to project's base directory
- Add "diario" to INSTALLED_APPS
- Issue "python manage.py syncdb"
- Add "(r'^blog/', include ('diario.urls.entries'))," to urls.py
- Copy templates from examples/templates/simple to template directory

User Management/ Feedback

django-contact-form

- Solicits feedback from users and visitors
- Comes with no sample templates
- Make sure to configure email settings in your project (see Django project documentaton)

Basic Configuration
django-contact-form

- Move to project's base directory
- Add "contact-form" to INSTALLED_APPS in "settings.py"
- Add "(r'^contact/', include ('contact_form.urls'))," to urls.py
- Set ACCOUNT_ACTIVATION_DAYS
- In your project's templates directory, three templates need to be created...

Basic Configuration
django-contact-form (cont.)

- Create contact_form/contact_form.txt
- This is the email template and should contain something like:

```
{{ name }}
{{ email }}
{{ body }}
```

Basic Configuration
django-contact-form (cont.)

- Create contact_form/contact_form.html
- This is the form template and should contain something like:

```
<form method='POST'>
{{ form.as_p }}
<input type='submit' value='Send'>
</form>
```

- Put any message in contact_form/contact_form_sent.html

django-registration

- Provides standard user registration form and email activation functionality
- Users visit accounts/register to create accounts
- Comes with no sample templates
- Again, make sure email is configured

Basic Configuration
django-registration

- Move to project's base directory
- Add "registration" to INSTALLED_APPS
- Issue "python manage.py syncdb"
- Add "(r'^accounts/', include ('registration.urls')),” to urls.py
- In templates directory, five templates need to be created

Basic Configuration
django-registration (cont.)

➜Create registration/registration_form.html:

```
<form method='POST'>
{{ form.as_p }}
<input type='submit' value='Submit'>
</form>
```

Basic Configuration
django-registration (cont.)

- registration/registration_complete.html is any form response message
- registration/activation_email_subject.txt is activation email subject
- registration/activation_email.txt is text for an email, should include “http://www.mysite.com/accounts/activate/{{ activation key }}”
- registration/activate.html is activation message

django-profiles

- The standard Django auth module provides the ability to define a profile model that will store profile data for each user
- django-profiles provides a user interface to this model
- Comes with no sample templates

Basic Configuration
django-profiles

- Move to project's base directory
- Add "profiles" to INSTALLED_APPS
- You'll need to have made a site-specific application
- Define a profile model (see Chapter 12 of The Django Book, free online)

Basic Configuration
django-profiles (cont.)

- In mysite/models.py, for example, we could put this profile model:

```
from django.contrib.auth.models import User

class MySiteProfile(models.Model):
  user = models.ForeignKey(User, unique=True)
  favorite_band = models.CharField(maxlength=100, blank=True)
  favorite_cheese = models.CharField(maxlength=100, blank=True)
  lucky_number = models.IntegerField()

  def get_absolute_url(self):
    return ('profiles_profile_detail',
    (), { 'username': self.user.username })
  get_absolute_url = models.permalink(get_absolute_url)
```

Basic Configuration
django-profiles (cont.)

- In settings, add AUTH_PROFILE_MODULE= 'mysite.MySiteProfile'
- Issue “python manage.py syncdb”
- Add “(r'^profiles/', include ('profiles.urls')),” to urls.py
- In templates directory, four templates need to be created

Basic Configuration
django-profiles (cont.)

- Create profiles/create_profile.html:

```
<form method='POST'>
{{ form.as_p }}
<input type='submit' value='Send'>
</form>
```

- Use same profile as above for profiles/edit_profile.html

Basic Configuration
django-profiles (cont.)

➜Create profiles/profile_detail.html:

```
{{ profile.favorite_band }}<br>
{{ profile.favorite_cheese }}<br>
{{ profile.lucky_number }}
```

Basic Configuration
django-profiles (cont.)

- Finally, create profiles/profile_list.html (uses Django generic list view):

```
<ul>
{% for profile in object_list %}
<li>Name: {{ profile.user }}, Number:
{{ profile.lucky_number }} </li>
{% endfor %}
</ul>
```

User-Driven Content Apps

django-forum

- SVN only, no installer
- Leverages django-registration
- Comes with sample templates

Basic Configuration
django-forum

- Move to project's base directory
- Add "forum" to INSTALLED_APPS
- In settings.py, add FORUM_BASE='/forum'
- Issue "python manage.py syncdb"
- Add "(r'^forum/', include('forum.urls'))," to urls.py
- Copy sample templates into your template directory
- Add forum(s) via the Django admin

Other Apps

django-lifestream

- Aggregates feeds from Twitter, Flickr, etc.
- Requires dateutil and Universal Feed Parser modules
- Requires Python 2.5 (easy to hack for 2.4)
- Requires cron hook for updates

django-geo

- Adds location awareness to objects
- Transparent geocoding

django-voting

- Add voting functionality (Reddit, etc.)
- Includes specialized generic views (including one for AJAX)

Poke around!

Resources

- This talk:
  http://mikecantelon.com/story/
  django_pluggable_applications
- James Bennett on creating pluggable applications:
  http://www.b-list.org/weblog/2008/mar/15/
  slides/
- The Django Book:
  http://www.djangobook.com