Accelerating Information Technology Innovation

http://aiti.mit.edu

Nigeria Summer 2012
Lecture DJ02 – Django Models
Setting up your Django DB

• We will be using sqlite3 … it is bundled with Django, no installations required
• In your settings.py, modify
  Engine : django.db.backends.sqlite3
  Name : C:\pyprojects\mysite\db
• Run python manage.py syncdb to create db required by your imported libraries.
• More on

https://docs.djangoproject.com/en/1.4/intro/tutorial01/
Creating a Django App within a Project

• An app is a Web application that does something -- e.g., a Weblog system, a database of public records or a simple poll app. A project is a collection of configuration and apps for a particular Web site. A project can contain multiple apps. An app can be in multiple projects.

• Always add the app name to settings.py to inform it about the apps existence
What is a model?

- A class describing data in your application
- Basically, a class with attributes for each data field that you care about
- The schema for your data
Django models

- Avoid direct work with the database
- No need to handle database connections, timeouts, etc. Let Django do it for you.
- Class that extends models.Model
Django fields

• All you do is define a field type
  – Ex: active = models.BooleanField()

• Django handles the rest:
  – Bit value in sql database
  – Represented as a checkbox on a webpage
  – Validation of values
Django Model Syntax

class Musician(models.Model):
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    instrument = models.CharField(max_length=100)
    def __unicode__(self):
        return last_name + ", " + first_name

class Album(models.Model):
    artist = models.ForeignKey(Musician)
    name = models.CharField(max_length=100)
    release_date = models.DateField()
    num_stars = models.IntegerField()
    def __unicode__(self):
        return name
Django Model Syntax

- class Album(models.Model): artist = models.ForeignKey(Musician) name = models.CharField(max_length=100) release_date = models.DateField() num_stars = models.IntegerField()
Important Django field types

- **BooleanField**
  - Checkbox

- **CharField**(max_length)
  - Single-line textbox

- **DateField**
  - Javascript calendar

- **DateTimeField**
  - Javascript calendar, time picker
Important Django field types

- **DecimalField**(max_digits, decimal_places)
  - Decimal numbers
- **EmailField**
  - Charfield that validates email address
- **FileField**
  - File upload, stores path in database
- **FloatField**
  - Floating point numbers
Important Django field types

- **ImageField** ***Don’t use***
  - Stores images
- **IntegerField**
  - Integer textbox
- **PositiveIntegerField**
  - Integer textboxes for positive integers
- **TextField**
  - Multi-line textbox
Important Django Field types

- TimeField
  - Time picker
- URLField
  - Textbox for URLs
- Anything you create
Field options

• Null
• Blank
• Choices:
  – List or tuple of 2-tuples to use as field choices
  – Django will represent it with a drop-down instead of a textbox
• Default
• Help text
More field options

- Primary key
- unique
- Verbose field name
DateField and DateTimeField options

- **Auto_now**
  - Any time the object is saved, the field will be updated with the current time.

- **Auto_now_add**
  - The time will always be equal to the creation date of the object.
Model Methods

• `__unicode__()`:  
  – Equivalent of `toString` – used for auto-generated admin pages

• `Get_absolute_url()`  
  – Used for deciding URLs that reference a specific object
Django Relationship Fields

- **ForeignKey** (foreign class)
  - Many-to-one

- **ManyToManyField** (foreign class)
  - Uses a temporary table to join tables together

- **OneToOneField** (foreign class)
  - Enforces uniqueness
Rules of Django Models

1. When you update a model, ALWAYS RUN python manage.py syncdb
2. Keep code clean
3. Always create a __unicode__() method
4. Name your variables well
5. Don’t think too much about the database
Commands

• View sql for models in a webapp
  `python manage.py sql appname`

• Create the tables in database
  `python manage.py syncdb`
Add an App to Admin Interface

- Create admin.py in your appname directory

```python
from appname.models import Tablename
from django.contrib import admin
admin.site.register(Tablenames)
```