Accelerating Information Technology

http://aiti.mit.edu

Ghana Summer 2012
Lecture DJ02 – Models

MISTI

Models

- Suppose we want to create a web application to manage data about thousands of movies
- What information would we want to store?
  - Title
  - Rating (scale from 1 to 5)
  - Genre
  - Lead Actor
  - Supporting Actors

Models

- How would we think about storing this data?
  - Lecture 4: Dictionaries and lists are used to store data in Python
- Web applications use databases
  - Lots of options varying syntax
  - Each table represents a different model
  - Each column is a different attribute
- Django: Common interface to almost all database solutions

Example of a DB table

Movies	Title	Rating	Genre	Lead Actor	Support Actors
1	“Iron Man”	7.9	“Action”	“Robert Downey Jr.”	“Gwyneth Paltrow”

Models

- Django's database interface works with any object of type django.db.models.Model
- To create your own Model, use inheritance!

```
from django.db import models
class Movie(models.Model):
  # attributes go here
```

Models

- Models have attributes: `Fields`
- We create ‘instances’ of Model objects in a different way (no `__init__` function necessary)

```
from django.db import models
class Movie(models.Model):
   # attributes go here
   self.title = models.CharField(max_length=100)
   self.rating = models.IntegerField()
```

Models

- Some attributes indicate special relationships to other Model objects
- ForeignKey: OneToMany
- ManyToManyField: Well, it's a many-to-many field

```
from django.db import models
class Movie(models.Model):
   rating = models.IntegerField()
   title = models.CharField(max_length=100)
   genre = models.CharField()

   lead_actor = models.ForeignKey
   (Actor,related_name='lead actor')
   support_actors = models.ManyToManyField
   (Actor,related_name='support')
```

Checkpoint: Models

- Build a django `Model` class for Actor
  - What does the class inherit from?
  - What attributes should the class have?

- Build a django `Model` class for Award
  - What does the class inherit from?
  - What attributes should the class have?

Checkpoint:Models

- Actor `Model` class:

```
class Actor(models.Model):
   name = models.CharField(max_length=100)
   birth_date = models.DateField()
```

- Attributes:
  - `name`: a string – use a `CharField`
  - `birth_date`: a `datetime.date` – use a `DateField`

Checkpoint: Models

- What type of field should we use for the title of the `Award`?
  - `CharField`
- What type of field should we use to denote the winning actor?
  - `ForeignKey` (one actor, many awards)
- What type of field should we use for the `nominees` (each nominee is a `Movie`)?
  - `ManyToManyField` (each movie can be nominated for many awards, each award has many nominees)

Checkpoint: Models

```
class Award(models.Model):
    title = models.CharField(max_length=100)
    sponsor = models.CharField(max_length=100)
    year = models.DateField()
    winning_actor = models.ForeignKey(Actor)
    winning_movie = models.ForeignKey(Movie,related_name='winning
    movie')
    actor_nominees = models.ManyToMany(Actor, related_name =
    "no")
```

- Wait! How does Actor relate to the Award class that we just wrote?
  - we only need to specify relation in one of the models

Models

- So what happens when we run

```
python manage.py syncdb
```

- Database is updated!
- First time you run it, database is created

Models

- Add the application with the relevant models to the list of INSTALLED_APPS
- Specify the path to your new database
- Validate your `Model` classes
  - `django-admin.py validate`
- Create or update the models in our project
  - `python manage.py syncdb`
- Later: reset the database (clear all information)
  - `python manage.py reset app_name`

Models from yesterday

```
class Notes(models.Model):
    title   = models.CharField(max_length=255)
    author  = models.CharField(max_length=255)
    content = models.TextField()
    def __unicode__(self):
        return self.title
```

Models: Using the Admin

`admin` is a builtin Django module

Simple GUI to create and modify the entries in your database

`models.py`

```
class Actor(models.Model):
 # some code

class Award(models.Model):
 # some code

class Movie(models.Model):
 # some code

admin.site.register(Actor)
admin.site.register(Award)
admin.site.register(Movie)
```

Models from yesterday

```
class Notes(models.Model):
    title   = models.CharField(max_length=255)
    author  = models.CharField(max_length=255)
    content = models.TextField()
    def __unicode__(self):
        return self.title
```

- If we want to be able to add notes to the table using admin page, inside of `admin.py`:

```
admin.site.register(Notes)
```