Accelerating Information Technology Innovation

http://aiti.mit.edu

India Summer 2012

Lecture 6 – Django: Linking Models and Views

Django:
Review of Models

Playing in the shell

Create a `City` named "Mumbai" and save it.

Playing in the shell

Find the `City` *named "Mumbai"*

Playing in the shell

Get the `id` of Mumbai.

Set the `population` of Mumbai to 12,478,447

Playing in the shell

Find the city named "Mumbai" and get its population.
Save the change to the population you made.
Then find the city again and get its population.

Playing in the shell

Get the list of all `City` *objects.*

Get all `City` *objects with population greater than 1 crore.*

Order that list by the name of the city from A to Z.

Delete all `City` *objects with population less than 1 crore.*

Some Final Notes

- Synchronize the database (syncdb) whenever you change a model
  - Not when you create or save one!
- By default, properties are *required* to save
- `blank=True` in the arg. list changes this
  - You may also need `null=True`

Linking Models

Joining Models Together

- How do `Country` and `State` relate?
  - One-to-one?
    - One capital per state (usually)
  - Many-to-many?
    - One state may have many languages
    - One language may be in many states
  - One-to-many!
    - One country has many states; a state has only one country!

Joining Models Together

- `country = ForeignKey(places.Country)`
  Builds a property to link to another model.
- `state.country` will return a `places.Country` object
- Treat it like any other property (save it!)
- Can also go backwards:
  `country.state_set`

Joining Models Together

- What about one-to-one relationships?
- `governor = OneToOneField(Person)`
- `state.governor` will return a `Person`
- `person.state` will go backwards
  - Can hide this with `related_name="+"` argument

Joining Models Together

- Many-to-many relationships?
  - Tricky! Google App Engine won't let us use `ManyToManyField`!

- Solution: A cross-join model
  - Disadvantage: We can't filter the lists easily!
  - Disadvantage: Need to write more code!

```
from django.db import models
import languages.models

class StateLanguage(models.Model):
    state = models.ForeignKey(State)
    language = models.ForeignKey(languages.models.Language)
```

Some Practice!

Making a model

Discussion:

How should `City` *link to* `State`*?*

(one-to-one? one-to-many? many-to-many?)

Making a model

Link `City` *to* `State`

Making a model

Discussion:

How could we link the capital of a `State`*?*

(one-to-one? one-to-many? many-to-many?)

Making a model

Add a capital property to `State`

Making a model

How might we link `States` *that border each other?*

Try writing the code to do so.

Views

Building a View

- Two parts:
- 1. View method
  - methods are defined in `views.py`
  - One argument (`HttpRequest`)
  - Respond with `HttpResponse` which takes string argument of response (e.g. HTML as a string)

Building a View

```
from places.models import State  # Need to import the model now!

def list_states(request):
    states = State.objects.all().order_by("name")
    html = ""
    for state in states:
        html += state.name + ", "
    html = html[:-2]  # Take the substring of html except for
                      # the last two characters.
    return HttpResponse(html)
```

Building a View

- 2. URLs:
  - URLs are defined in `urls.py`
  - `urlpatterns` **set to a** `patterns` **function**
    - All arguments except the first are:
    - `url([regular expression], [string of view])`
      - `r'[string]'` quotes regular expressions without many backslashes
      - Calls view with full package name.

Building a View

- Some hints about regular expressions
  - All text matches itself except `()\[].+?*`
  - Can match these by adding a "\" before each
    - Some "\" letter combos mean something.
      "`\d`" means numbers 0–9.
  - `^` – Matches the start of the string (at the start)
  - `$` – Matches the end of the string (at the end)
  - `(?P<var>pat)` – Matches "pat", stores in "var"

We'll cover regular expressions in more detail later!

Building a View

What do you think:

```
url(r'^states/$', 'places.views.list_states'),
```

means?

Building a View

What do you think:

```
url(r'^states/json$', 'places.views.list_states_json'),
```

means?

Building a View

What do you think:

```
url(r'^states/(?P<state_id>\d+)/$',
    'places.views.show_state'),
```

means?

Building a View

- Can pass arguments back to view
- `r'^states/(?P<state_id>\d+)/$'` to...

```
from places.models import State  # Need to import the model now!

def state_detail(request, state_id):
    state = State.objects.get(id=state_id)
    return HttpResponse(state.name)
```

Making a Simple View

Making a Simple View

Make a view method that gets the list of all `City` *objects.*

Making a Simple View

Add a URL to view the list of `City` *objects.*

Making a Simple View

Now let's view the list of `City` *objects.*

(You'll need to use the "Django" menu to run "startapp"!)

Making a Simple View

Make a view method that gets a `City` *object by its* `id`.

Making a Simple View

Make a view method that gets a `City` *object by its* `name`.

(`[A-Za-z ]+` *matches a string of letters and spaces)*

References

- Django documentation:
  - "Field Types: Relationship fields" <https://docs.djangoproject.com/en/1.3/ref/models/fields/#module-django.db.models.fields.related>
  - "Writing your first Django app: Part 3" <https://docs.djangoproject.com/en/1.3//intro/tutorial03/>
  - "Writing views" <https://docs.djangoproject.com/en/1.3//topics/http/views/>
  - "URL dispatcher" <https://docs.djangoproject.com/en/1.3//topics/http/urls/>

References

- Django documentation:
  - For more on HttpRequest and HttpResponse <https://docs.djangoproject.com/en/1.3//ref/request-response/>