Intro to Web Apps
What is a Web App?

An application where the data (and ways to edit or view it) are available via the web.
Why Web Apps?

- Not everyone has a smartphone
- But many more have web on their phone
- Can also provide desktop access
How Can I Use Web Apps?

- Store/share persistent data off the phone
  - Restaurant reviews, maps, phone numbers, etc.
- Allow access from desktops and phones with web access
- Show off your non-web app (advertise!)
What is Google App Engine?

• A cloud-based web app provider
  ○ Reliable
  ○ Scalable
  ○ You don’t need to do system administration

• Free to start development
Intro to Django
What is Django?

- A platform for rapid web development
- Python-based
- Worry about content, not display
How does Django work?

- Three basic components:
  - *Model* – An object that is stored in a database
    - e.g. Restaurant, Hotel, Person
  - *Template* – How models are displayed/rendered
    - Usually HTML, but also JSON/XML for web services
    - e.g. Restaurant list, Hotel details, edit form for a Person
  - *View* – Select/modify models for a template
How does Django work?

• Components belong to an *application*
  ○ A set of models and views [actions] that work together as a single component
    ▪ Four different applications: polls, posts, users, comments on a blog
    ▪ Or perhaps: a developer blog, restaurants, and locations with Zomato
A Basic Django App:
TravellIndia

A web app for Indian travelers looking for good hotels, places to see, restaurants, etc.
A Simple Model – State
A Simple Model – State

• What does it represent?
A Simple Model – \textit{State}

- What does it represent?
  - A single (Indian) state
A Simple Model – State

• What does it represent?
  ○ A single (Indian) state

• What might its properties be?
A Simple Model – State

• What does it represent?
  ○ A single (Indian) state

• What might its properties be?
  ○ Name, Capital, Population, Area...
A Simple Model – State

• What does it represent?
  ○ A single (Indian) state

• What might its properties be?
  ○ Name, Capital, Population, Area...

• May correspond with a State class (and its instances) in an Android app!
A Simple Model – State

• Let’s consider 4 possible properties. What data type (integer, string, etc.) should they be?
  ○ Name
  ○ Capital
  ○ Population
  ○ Area
from django.db import models

class State(models.Model):
    name = models.CharField(max_length=255)
    capital = models.CharField(max_length=255)
    population = models.BigIntegerField()
    area = models.DecimalField(max_digits=11, decimal_places=2)
Some Model Fields

- **CharField** – A (short) string
  - **TextField** – Longer strings (e.g. comment text)

- **IntegerField** – An integer
  - **BigIntegerField** – 64-bit integer

- **DecimalField** – A decimal
  - As opposed to a **FloatField**, which contains floating-point numbers
Some More Model Fields

- **DateTimeField** – A date and time
- **BooleanField** – A true/false value
- **EmailField** – An e-mail address
- **URLField** – A URL
- **SlugField** – A short string with letters, numbers, underscores, and hyphens
More about Models

• Can have methods of their own.
  ○ `def population_density(self):`
    `return self.population / self.area`

• Two common methods:
  ○ `__unicode__(self)` –
    Return the object’s string representation
  ○ `get_absolute_url(self)` –
    Return the object’s default URL (for display)
Let’s make a Model!
Making a model

Create a new model named City.
Making a model

**Discussion:**
What properties do you think it should have?
Making a model

**Discussion:**

*What properties do you think it should have?*

*(We’ll talk about how to relate them to states tomorrow!)*
Making a model

Discussion:
What types do you think those properties should have?
Making a model

Add the properties to the City model.
Making a model

Add a `__unicode__` method to City.
Playing with Models using the shell
Accessing Models

- **State.objects** – Access saved States
  - **State.objects.all()** – Get all States
  - **State.objects.get()** – Get a single State
    - `get(id=my_id)` – Get by id
    - `get(name=my_name)` – Get by name
  - **State.objects.filter()** – Get several States (like `get()`)
Accessing Models

• `get(id=my_id) and get(name=my_name)`
  lookup by field

• We can do more advanced filtering:
  ○ `filter(count__gt=10)` – Get all where the `count` field > 10
  ○ `filter(name__iexact='my name')` – The `name` field contains “my name” in any case
Accessing Models

• More field lookups:
  ○ `filter(name__contains='Name')`—Get all where the `name` field contains “Name”
  ○ `filter(count__in=[1, 2, 3])`—The `count` field is either 1, 2, or 3
  ○ `filter(name__istartswith='na')`—The `name` field starts with “na” in any case
Accessing Models

- `State.objects.all()`
  - `.order_by()` – Order the list by properties
    - `'name'` – Sort by name ascending (A→Z)
    - `'-name'` – Sort by name descending (Z→A)
  - `.reverse()` – Reverse the order
  - `.count()` – Count the number of items
Changing Models

- Create an instance
  \(\text{my\_state} = \text{State}(\text{name}='\text{name}')\), or:

- Change the property
  \(\text{my\_state}.\text{name} = '\text{name}'\), then:

  - \text{my\_state}.\text{save}() – Save the changes
  - \text{my\_state}.\text{delete}() – Delete the object
Let’s play with the City model
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.
References

• The Django site itself has great documentation:
  ○ Tutorial: “Writing your first Django app”
    <https://docs.djangoproject.com/en/1.3/intro/tutorial01/>
  ○ “Models”
    <https://docs.djangoproject.com/en/1.3/topics/db/models/>
  ○ “Field Types”
    <https://docs.djangoproject.com/en/1.3/ref/models/fields/>
  ○ “QuerySets”
    <https://docs.djangoproject.com/en/1.3/ref/models/querysets/>
References

• More on the Django site:
  ○ “Making Queries”  
    <https://docs.djangoproject.com/en/1.3/topics/db/queries/>

• Introduction to Google App Engine:  
  <https://developers.google.com/appengine/docs/whatisgoogleappengine>