References

http://www.djangoproject.com/documentation/

Django Tutorial part 1 http://www.djangoproject.com/documentation/tutorial1/


Database API, http://www.djangoproject.com/documentation/db_api/

Reading

Installing Django

http://www.djangoproject.com/documentation/install/

Python 2.3 or higher

Don't need Apache and mod_python

Need database
  Sqlite, Postgres, or MySql

Database driver

Use official Django version (0.95)
SQLite

Zero-configuration

No separate program to run

3.3.7 latest version

http://www.sqlite.org/
  info & downloads
pysqlite

http://initd.org/tracker/pysqlite

Python driver for Sqlite

2.3.2 latest version
Django & rohan

Django, SQLite & psycopg2 installed on rohan
Running Django on rohan

python manage.py runserver 0.0.0.0:8765

0.0.0.0
  Allows remote access to Django web server

8765
  Pick some port number to use
  Only one person can use a port at a time
  1024 - 65535 available ports

http://rohan.sdsu.edu:8765/etc
Django Documentation

http://www.djangoproject.com/documentation/
Basic Parts

Models and Database access
Templates
Forms
URLs
Sessions
Authentication
Assumptions for Examples

Using existing cs683 project

Django web server started using:
    manage.py runserver

    using port 8000

In cs683 project create polls app:

    python manage.py startapp polls
Models and Database Access
from django.db import models

class Poll(models.Model):
    question = models.CharField(maxlength=200)
    pub_date = models.DateTimeField('date published')

    def __str__(self):
        return self.question

class Admin:
    pass

class Choice(models.Model):
    poll = models.ForeignKey(Poll)
    choice = models.CharField(maxlength=200)
    votes = models.IntegerField()

    def __str__(self):
        return self.choice

class Admin:
    pass
INSTALLED_APPS = (  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.sites',  
    'cs683.simpleExamples',  
    'cs683.polls',  
    'django.contrib.admin',  
)
Generating some sql

>python manage.py sqlall polls
BEGIN;
CREATE TABLE "polls_poll" (    
    "id" integer NOT NULL PRIMARY KEY,
    "question" varchar(200) NOT NULL,
    "pub_date" datetime NOT NULL
);
CREATE TABLE "polls_choice" (    
    "id" integer NOT NULL PRIMARY KEY,
    "poll_id" integer NOT NULL REFERENCES "polls_poll" ("id"),
    "choice" varchar(200) NOT NULL,
    "votes" integer NOT NULL
);
CREATE INDEX polls_choice_poll_id ON "polls_choice" ("poll_id");
COMMIT;
# Tables with some data

## polls_poll

<table>
<thead>
<tr>
<th>id</th>
<th>question</th>
<th>pub_date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Do you like Django Reinhardt's music?</td>
<td>2006-09-13 18:45</td>
</tr>
<tr>
<td>2</td>
<td>Do you like Python?</td>
<td>2006-09-14 11:22</td>
</tr>
</tbody>
</table>

## polls_choice

<table>
<thead>
<tr>
<th>id</th>
<th>poll_id</th>
<th>choice</th>
<th>votes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Yes</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>No</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Who is Django Reinhardt?</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Yes</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>No</td>
<td>0</td>
</tr>
</tbody>
</table>
Do you like Django Reinhardt's music?

question: Do you like Django Reinhardt's music?

pub_date: 2006-09-13 18:45

choices: Yes 0 No 0

Who is Django Reinhardt?

0
Creating the tables

> python manage.py syncdb
Creating table polls_poll
Creating table polls_choice
Adding permission 'poll | Can add poll'
Adding permission 'poll | Can change poll'
Adding permission 'poll | Can delete poll'
Adding permission 'choice | Can add choice'
Adding permission 'choice | Can change choice'
Adding permission 'choice | Can delete choice'
Useful commands

python manage.py

sql [appname appname ...]
sqlall [appname appname ...]
sqlclear [appname appname ...]
sqlindexes [appname appname ...]
sqlinitialdata [appname appname ...]
sqlreset [appname appname ...]
syncdb

http://www.djangoproject.com/documentation/django_admin/
Admin Pages

http://127.0.0.1:8000/admin/

class Choice(models.Model):
    ...
    class Admin:
        pass

class Poll(models.Model):
    ...
    class Admin:
        pass
Adding to Database

Django administration

Add choice

Poll:  
Choice:  
Votes:  

Save and add another  Save and continue editing  Save
Adding Models

If you add a new
    app
    model
    attribute to existing model

Update your database with:

    python manage.py syncdb
Changing Existing Attributes

syncdb will not modify existing columns

Modify column by hand

use python manage.py reset appname to get SQL

Delete database and rebuild
Accessing SQLite Databases

sqlite3
  command-line tool
  part of SQLite distribution

Third party SQLite clients
  Google sqllite or sqlite manager
Initializing data
project/app/sql/modename.sql

cs683/polls/sql/poll.sql
INSERT INTO polls_poll (question, pub_date) VALUES ('What's up doc?', '2006-09-14 13:00:00');
INSERT INTO polls_poll (question, pub_date) VALUES ('Where is the beef?', '2006-09-14 13:00:00');

syncdb will run project/app/sql/modename.sql files to initialize tables
shell access to database

>manage.py shell

Python 2.3.5 (#1, Mar 20 2005, 20:38:20)
[ GCC 3.3 20030304 (Apple Computer, Inc. build 1809) ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)

>>> from cs683.polls.models import Poll, Choice

>>> Poll.objects.all();
>>>
Some Queries

>>> Poll.objects.all();

>>> Poll.objects.get(id=1)
<Poll: Do you like Django Reinhardt's music?>

>>> Poll.objects.count()
3

>>> Poll.objects.filter(question__contains='Django')
[<Poll: Do you like Django Reinhardt's music?>]

>>> Poll.objects.filter(pub_date__year=2006)
[<Poll: Do you like Django Reinhardt's music?>, <Poll: Do you like Python?>]

>>> Poll.objects.all().order_by('pub_date')
QuerySets

```python
>>> Poll.objects.get(id=1).__class__
<class 'cs683.polls.models.Poll'>

>>> Poll.objects.filter(question__contains='Django').__class__
<class 'django.db.models.query.QuerySet'>
```

http://code.djangoproject.com/browser/django/trunk/django/db/models/query.py
>>> a = Choice.objects.get(id=1)
>>> a.poll
<Poll: Do you like Django Reinhardt's music?>
>>> a.votes
0
>>> a.poll.question
"Do you like Django Reinhardt's music?"

>>> poll = Poll.objects.get(id=1)
>>> poll.choice_set.all()
**Queries**

Returns QuerySet

- filter(**kwargs)
- exclude(**kwargs)
- order_by(*fields)
- distinct()
- values(*fields)
- dates(field, kind, order='ASC')
- select_related()
- extra(select=None, where=None, params=None, tables=None)

Does not return QuerySet

- get(**kwargs)
- create(**kwargs)
- get_or_create(**kwargs)
- count()
- in_bulk(id_list)
- latest(field_name=None)

http://www.djangoproject.com/documentation/db_api/
Lazy Evaluation

```python
>>> a = Choice.objects.get(id=1)
>>> a.poll
<Poll: Do you like Django Reinhardt's music?>
>>> a.votes
0

>>> a.poll.question  #requires database lookup
"Do you like Django Reinhardt's music?"

>>> a.poll.question  #uses cached value
```
Pre-fetching Related Data

```python
>>> a = Choice.objects.select_related().get(id=1)

>>> a.poll  #no database access
<Poll: Do you like Django Reinhardt's music?>
```