Tutorial 3:
Database Application Development

Vo Duy Tin

Outline

- Part I: Database access via programming language
  - Use programming language (Python) to access a DBMS (MySQL).
- Part II: Database application development
  - Construct a simple Web app to search from DBMS.

Part I

1. Create a database on MySQL
2. Connect to the database and create tables
3. Insert
4. Query

Prerequisites

This tutorial is implemented in Ubuntu 14.04

- MySQL-server:

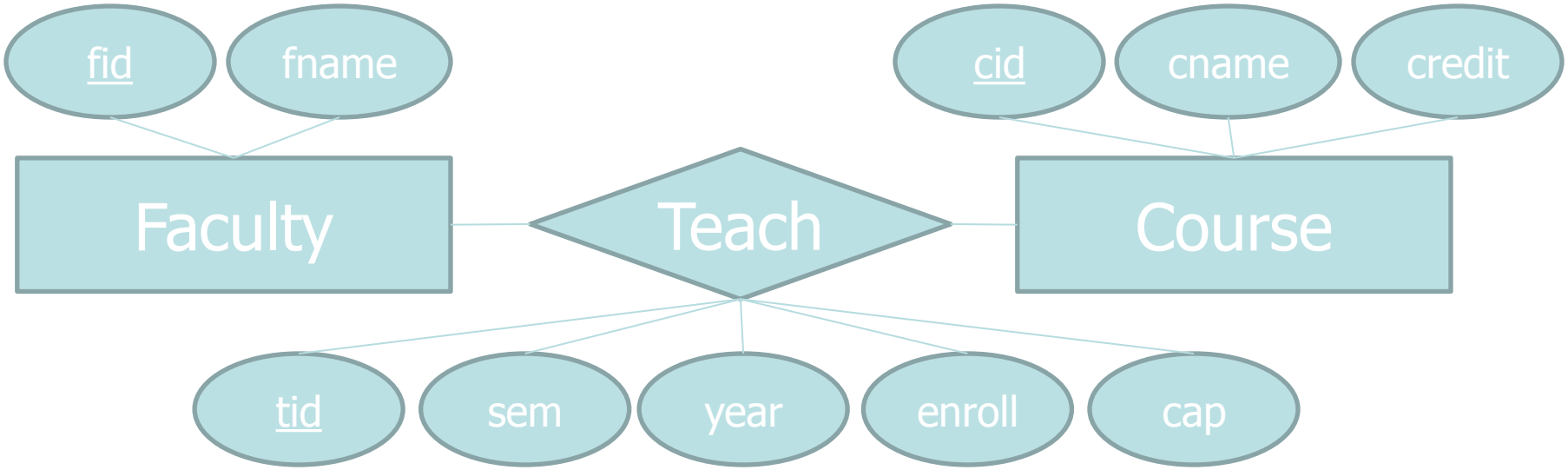
  sudo apt-get install mysql-server

- MySQLdb python package:

  sudo apt-get build-dep python-mysqldb

  pip install MySQL-python

An Example



- **faculty**(fid, fname)
- **course**(cid, cname, credits)
- **teach**(tid, cid, fid, semester, year, enrollment, capacity)

Create a database on MySQL

- Create a database called “tutorial”

  create database tutorial;

```
mysql> create database tutorial;
Query OK, 1 row affected (0.00 sec)

mysql> use tutorial;
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql>
```

Connect to the database

- Use MySQLdb package in python to connect to MySQL
- Create three tables as in the example by creating "part1/create_table.py" and running it

```
mysql> show tables;
+--------------------+
| Tables_in_tutorial |
+--------------------+
| course             |
| faculty            |
| teach              |
+--------------------+
3 rows in set (0.00 sec)

mysql> select * from faculty;
Empty set (0.00 sec)

mysql> select * from course;
Empty set (0.00 sec)

mysql> select * from teach;
Empty set (0.00 sec)
```

```
import MySQLdb as mdb

con = mdb.connect(host = "localhost",user = "root",passwd = "",db = "tutorial")

with con:
    cur = con.cursor()
    cur.execute("CREATE TABLE faculty(fid varchar(9) primary key, fname varchar(60))")
    cur.execute("CREATE TABLE course(cid varchar(9) primary key,cname varchar(60), credits int);")
    cur.execute("CREATE TABLE teach(tid varchar(9) primary key, cid varchar(9) references Course(cid), \
                fid varchar(9) references Faculty(fid), semester varchar(6), year int, enrollment int, capacity int)")
```

Insert

- Connect to MySQL and insert tuples into tables by running "part1/insert_table.py"

```
import MySQLdb as mdb

con = mdb.connect(host = "localhost",user = "root",passwd = "",db = "tutorial")

with con:
    cur = con.cursor()
    cur.execute("INSERT INTO faculty VALUES('f01','Meihui Zhang')")
    cur.execute("INSERT INTO course VALUES('c50.008','Database',12)")
    cur.execute("INSERT INTO teach VALUES('t01','c50.008','f01','Fall',2015, 53, 60)")
```

```
mysql> select * from faculty;
+-----+--------------+
| fid | fname        |
+-----+--------------+
| f01 | Meihui Zhang |
+-----+--------------+
1 row in set (0.01 sec)

mysql> select * from course;
+---------+----------+---------+
| cid     | cname    | credits |
+---------+----------+---------+
| c50.008 | Database |      12 |
+---------+----------+---------+
1 row in set (0.00 sec)

mysql> select * from teach;
+-----+---------+------+----------+------+------------+----------+
| tid | cid     | fid  | semester | year | enrollment | capacity |
+-----+---------+------+----------+------+------------+----------+
| t01 | c50.008 | f01  | Fall     | 2015 |         53 |       60 |
+-----+---------+------+----------+------+------------+----------+
1 row in set (0.00 sec)
```

Query

- Connect to MySQL and perform a query by running “part1/query_table.py”

```
import MySQLdb as mdb

def querycourse(cid):
    con = mdb.connect(host = "localhost",user = "root",passwd = "",db = "tutorial")
    with con:
        cur = con.cursor()
        query="select cname,credits from course where cid = '%s'"%(cid)
        cur.execute (query)
        if cur.rowcount==0:
            print "Course does not exist"
        else:
            row=cur.fetchone()
            print row
```

```
In [11]: querycourse('c50.008')
('Database', 12L)

In [12]: querycourse('c50.010')
Course does not exist
```

References

How to access MySQL on Python using MySQLdb:

- http://zetcode.com/db/mysqlpython/
- http://www.tutorialspoint.com/python/python_database_access.htm
- http://mysql-python.sourceforge.net/MySQLdb.html
- http://www.cs.columbia.edu/~hgs/teaching/ap/examples/scripts_python_dbapi.pdf

Part II

1. Create a project
2. Build a plug-in app
3. Develop an admin page
4. Design the app

Prerequisites

This tutorial is implemented in Ubuntu 14.04

- Django:

pip install Django

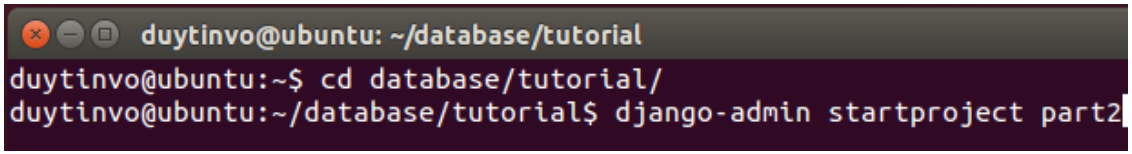
"With Django, you can take Web applications from concept to launch in a matter of hours. Django takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source."

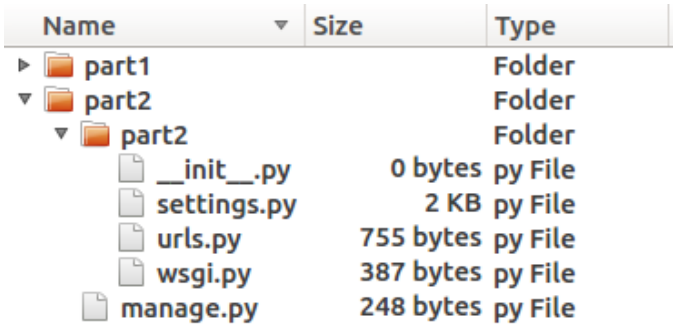
Source: https://www.djangoproject.com/start/overview/

Create a project

- Create a project name part2

  django-admin startproject part2





- Change file "part2/setting.py" to connect to MySQL

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'tutorial',
        'USER': 'root',
        'PASSWORD':'',
    }
}
```

Create a project

- Apply changes on MySQL by changing current directory to "part2" (folder containing "manage.py") and running:

python manage.py migrate

```
duytinvo@ubuntu:~/database/tutorial$ ls
part1
duytinvo@ubuntu:~/database/tutorial$ django-admin startproject part2
duytinvo@ubuntu:~/database/tutorial$ ls
part1  part2
duytinvo@ubuntu:~/database/tutorial$ cd part2
duytinvo@ubuntu:~/database/tutorial/part2$ ls
manage.py  part2
duytinvo@ubuntu:~/database/tutorial/part2$ python manage.py migrate
Operations to perform:
  Synchronize unmigrated apps: staticfiles, messages
  Apply all migrations: admin, contenttypes, auth, sessions
Synchronizing apps without migrations:
  Creating tables...
    Running deferred SQL...
  Installing custom SQL...
Running migrations:
  Rendering model states... DONE
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying sessions.0001_initial... OK
duytinvo@ubuntu:~/database/tutorial/part2$
```

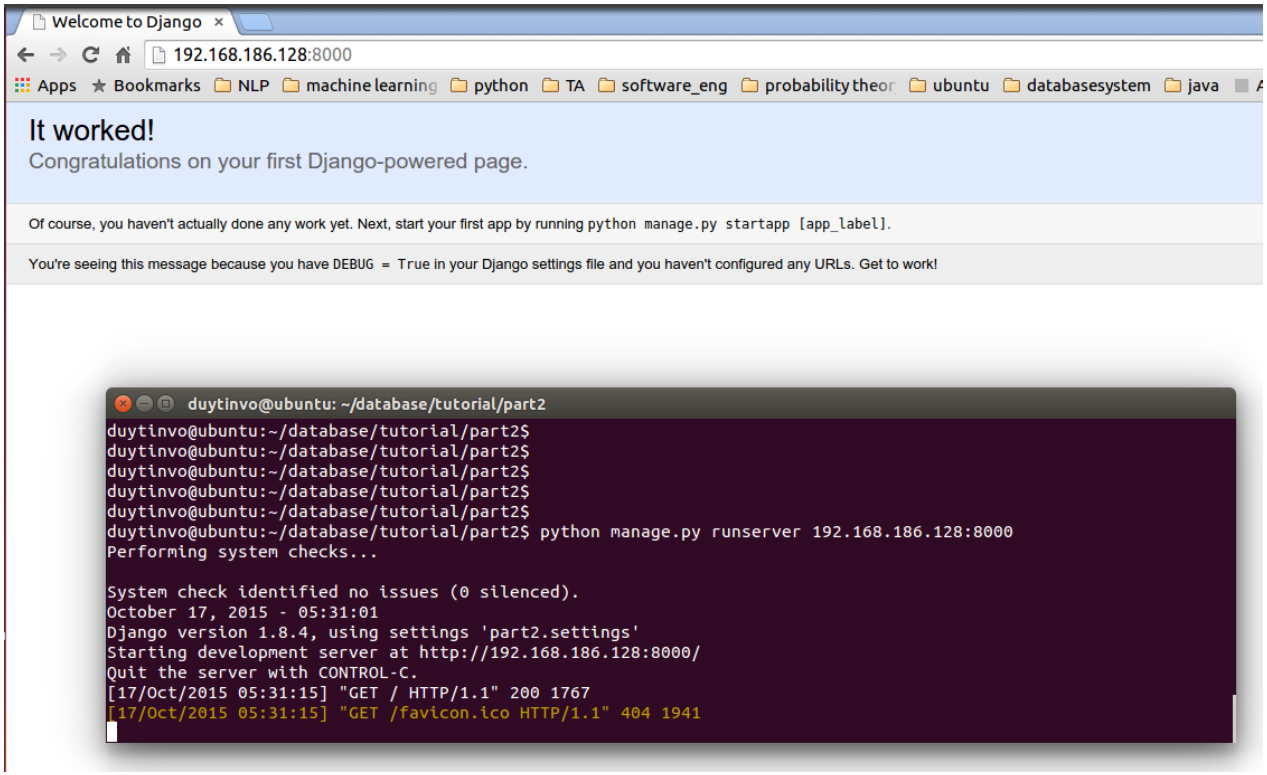
```
mysql> show tables;
+--------------------+
| Tables_in_tutorial |
+--------------------+
| course             |
| faculty            |
| teach              |
+--------------------+
3 rows in set (0.00 sec)

mysql> show tables;
+----------------------------+
| Tables_in_tutorial         |
+----------------------------+
| auth_group                 |
| auth_group_permissions     |
| auth_permission            |
| auth_user                  |
| auth_user_groups           |
| auth_user_user_permissions |
| course                     |
| django_admin_log           |
| django_content_type        |
| django_migrations          |
| django_session             |
| faculty                    |
| teach                      |
+----------------------------+
13 rows in set (0.00 sec)
```

Create a project

- Run a server

  python manage.py runserver 192.168.186.128:8000



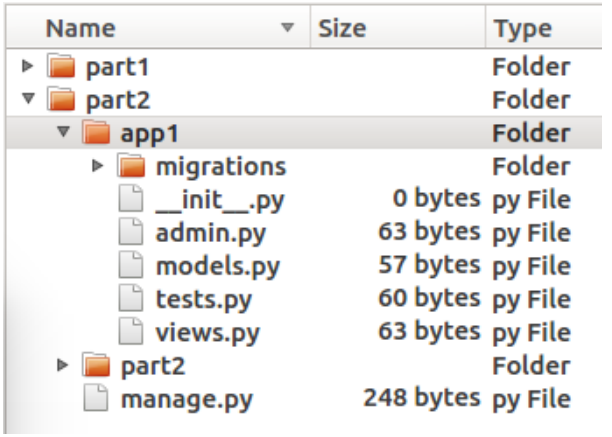
Build a plug-in app

- Type "Ctrl+C" to exit running server
- Create an app named "app1"

  python manage.py startapp app1

```
duytinvo@ubuntu:~/database/tutorial/part2$ python manage.py runserver 192.168.186.128:8000
Performing system checks...

System check identified no issues (0 silenced).
October 17, 2015 - 05:31:01
Django version 1.8.4, using settings 'part2.settings'
Starting development server at http://192.168.186.128:8000/
Quit the server with CONTROL-C.
[17/Oct/2015 05:31:15] "GET / HTTP/1.1" 200 1767
[17/Oct/2015 05:31:15] "GET /favicon.ico HTTP/1.1" 404 1941
^Cduytinvo@ubuntu:~/database/tutorial/part2$
duytinvo@ubuntu:~/database/tutorial/part2$
duytinvo@ubuntu:~/database/tutorial/part2$
duytinvo@ubuntu:~/database/tutorial/part2$
duytinvo@ubuntu:~/database/tutorial/part2$ python manage.py startapp app1
duytinvo@ubuntu:~/database/tutorial/part2$
```



Build a plug-in app

- Create three tables in the example above in app1/models.py
- Plug app1 into the project by modifying 'part2/setting.py'

models.py* | settings.py

```python
from django.db import models

# Create your models here.

class faculty(models.Model):
    fid = models.CharField(max_length=9, primary_key=True)
    fname = models.CharField(max_length=60)
    def __unicode__(self):
        return u'%s' % (self.fname)

class course(models.Model):
    cid = models.CharField(max_length=9, primary_key=True)
    cname = models.CharField(max_length=60)
    credits = models.IntegerField()
    def __unicode__(self):
        return u'%s' % (self.cname)

class teach(models.Model):
    tid = models.CharField(max_length=9, primary_key=True)
    cid = models.ForeignKey(course, db_column='cid')
    fid = models.ForeignKey(faculty, db_column='fid')
    semester = models.CharField(max_length=6)
    year = models.IntegerField()
    enrollment = models.IntegerField()
    capacity = models.IntegerField()
    def __unicode__(self):
        return u'%s, %s' % (self.tid, self.enrollment)

```

```python
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'app1',
)
```

Build a plug-in app

- Compile the app

  python manage.py makemigrations app1

- Apply changes on MySQL

  python manage.py migrate

```
duytinvo@ubuntu:~/database/tutorial/part2$ python manage.py makemigrations app1
Migrations for 'app1':
  0001_initial.py:
    - Create model course
    - Create model faculty
    - Create model teach
duytinvo@ubuntu:~/database/tutorial/part2$ python manage.py migrate
Operations to perform:
  Synchronize unmigrated apps: staticfiles, messages
  Apply all migrations: admin, contenttypes, auth, app1, sessions
Synchronizing apps without migrations:
  Creating tables...
    Running deferred SQL...
  Installing custom SQL...
Running migrations:
  Rendering model states... DONE
  Applying app1.0001_initial... OK
duytinvo@ubuntu:~/database/tutorial/part2$
```

```
mysql> show tables;
+----------------------------+
| Tables_in_tutorial         |
+----------------------------+
| app1_course                |
| app1_faculty               |
| app1_teach                 |
| auth_group                 |
| auth_group_permissions     |
| auth_permission            |
| auth_user                  |
| auth_user_groups           |
| auth_user_user_permissions |
| course                     |
| django_admin_log           |
| django_content_type        |
| django_migrations          |
| django_session             |
| faculty                    |
| teach                      |
+----------------------------+
16 rows in set (0.00 sec)
```

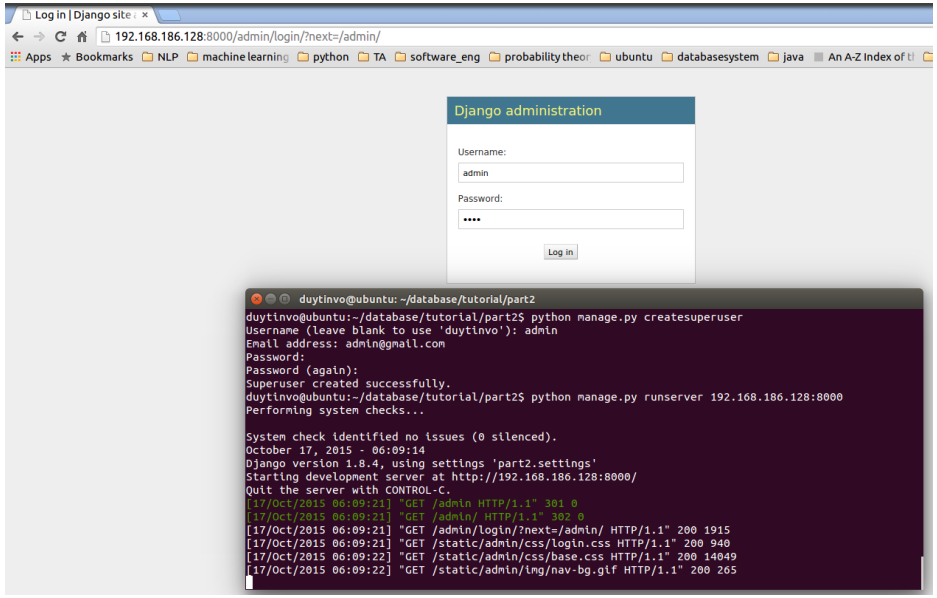
Develop an admin page

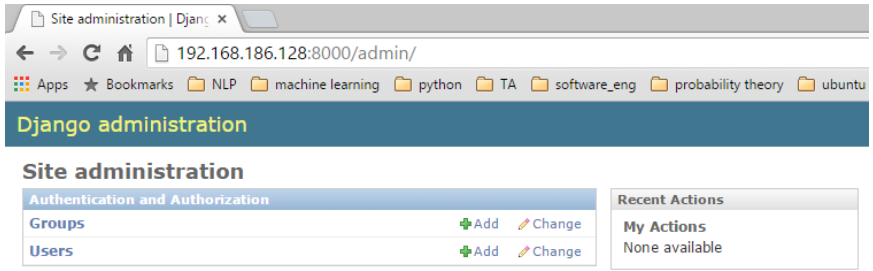
- Create an admin account

  python manage.py createsuperuser

- Start the server

  python manage.py runserver 192.168.186.128:8000



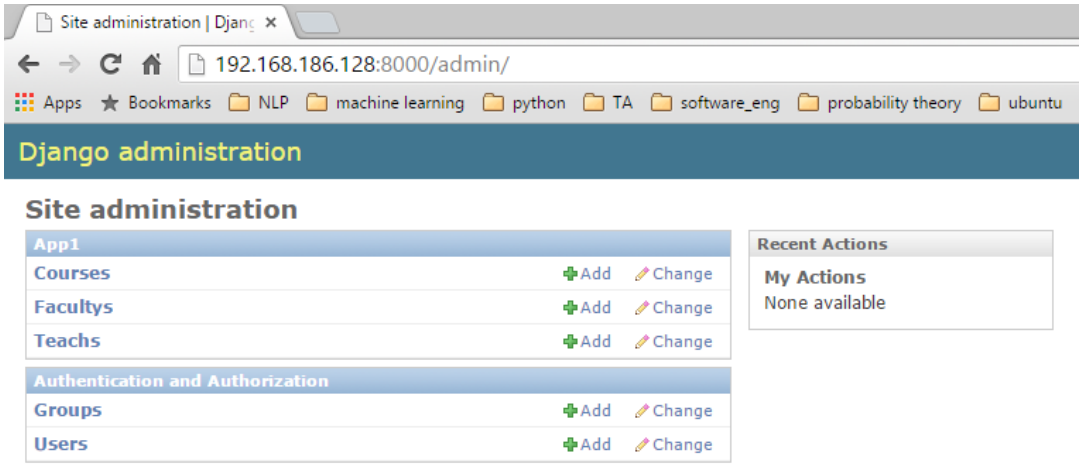


Develop an admin page

- Add three created tables into admin page by modifying file “app1/admin.py”
- Reload WebBrowser to see changes

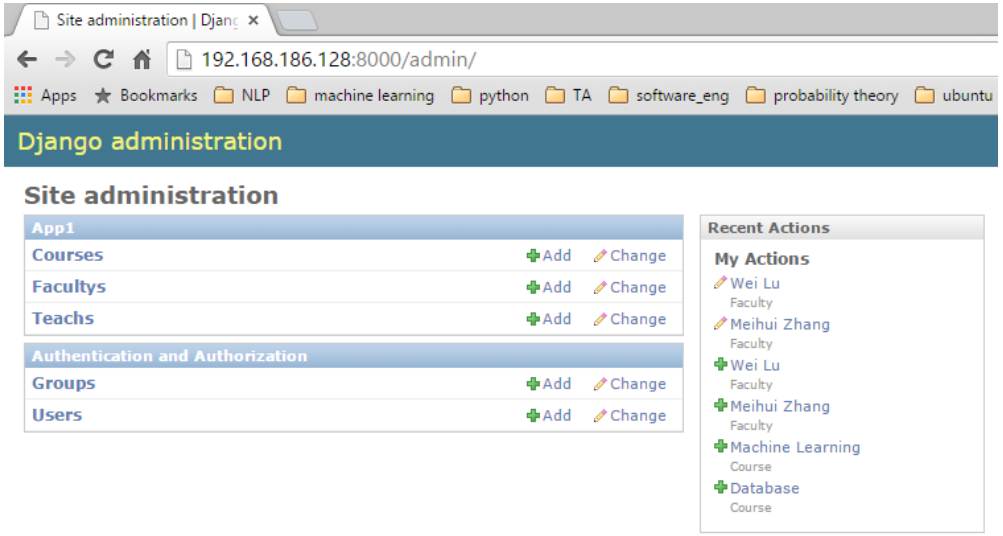
admin.py

```
1 from django.contrib import admin
2
3 # Register your models here.
4 from .models import faculty,teach,course
5
6 class teachinline(admin.StackedInline):
7     model = teach
8     extra=1
9
10 class facultyadmin(admin.ModelAdmin):
11     inlines=[teachinline]
12     list_display = ('fid', 'fname')
13
14 class courseadmin(admin.ModelAdmin):
15     inlines=[teachinline]
16     list_display = ('cid', 'cname')
17
18 admin.site.register(faculty,facultyadmin)
19 admin.site.register(teach)
20 admin.site.register(course,courseadmin)
21
```



Develop an admin page

- Now, we can insert tuples into tables by one of three following methods
  - MySQL (lecture)
  - Python (part1)
  - Admin page
- Choose one of them to insert some tuples



Design the app

- Write the first view by opening file "app1/views.py" and inserting following Python codes

admin.py | urls.py | views.py | urls.py

```
1 from django.shortcuts import render
2
3 # Create your views here.
4 from django.http import HttpResponse
5
6 def hello(request):
7     return HttpResponse("Hello, world. This is tutorial 3!!!")
8
```

- To call the view, we need to map it to a URL - and for this we need a URLconf. To create a URLconf in the app1 directory, create a file called "app1/urls.py"

admin.py | urls.py | views.py

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Oct 17 14:43:09 2015
4
5 @author: duytinvo
6 """
7
8 from django.conf.urls import url
9
10 from . import views
11
12 urlpatterns = [
13     url(r'^$', views.hello, name='hello'),
14 ]
15
```

Design the app

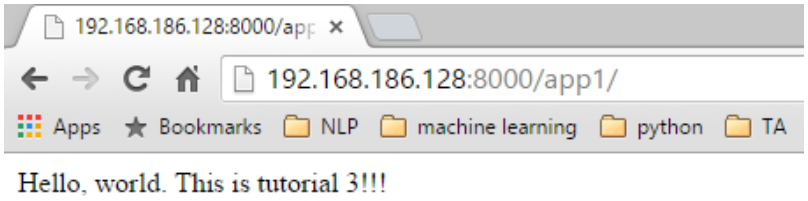
- The next step is to point the root URLconf at the app1.urls module by inserting in "part2/urls.py" this module.
- Now, open the webBrowser and enter the address: *http://192.168.186.128:8000/app1/*

admin.py | urls.py | views.py | urls.py

```
"""part2 URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/1.8/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  url(r'^$', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  url(r'^$', Home.as_view(), name='home')
Including another URLconf
    1. Add an import:  from blog import urls as blog_urls
    2. Add a URL to urlpatterns:  url(r'^blog/', include(blog_urls))
"""
from django.conf.urls import include, url
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', include(admin.site.urls)),
    url(r'^app1/', include('app1.urls')),
]
```



Summary

Steps to build an app

- Create a project
- Create an app
- Compile the app
- Apply changes into DBMS
- Run a server

Steps to develop the app

- Write functions in "app1/views.py"
- Map functions into "app1/urls.py"

References

How to develop an app by Django:

- https://docs.djangoproject.com/en/1.8/intro/ (part1-part6)
- http://slav0nic.org.ua/static/books/python/Packt.Publishing.Learning.Website.Development.with.Django.Mar.2008.pdf
- http://gsl.mit.edu/media/programs/south-africa-summer-2015/materials/djangobook.pdf

NOW, LET DESIGN THE APP THAT ACTUALLY DO SOMETHING!!!

Search a faculty given fid

- Write function “searchfaculty” in “app1/views.py”

```
1 from django.shortcuts import render
2
3 # Create your views here.
4 from django.http import HttpResponse, Http404
5 import MySQLdb as mdb
6
7 def hello(request):
8     return HttpResponse("Hello, world. This is tutorial 3!!!")
9
10 def searchfaculty(request, fid):
11     conn = mdb.connect (host = "localhost",user = "root",passwd = "",db = "tutorial")
12     with conn:
13         cursor = conn.cursor ()
14         cursor.execute ("select fname from app1_faculty where fid = '%s'"%(fid))
15         if cursor.rowcount==0:
16             raise Http404("Facuty does not exist")
17         else:
18             row=cursor.fetchone()
19             html="You're searching Prof. '%s' with id '%s'." % (row[0],fid)
20     return HttpResponse(html)
```

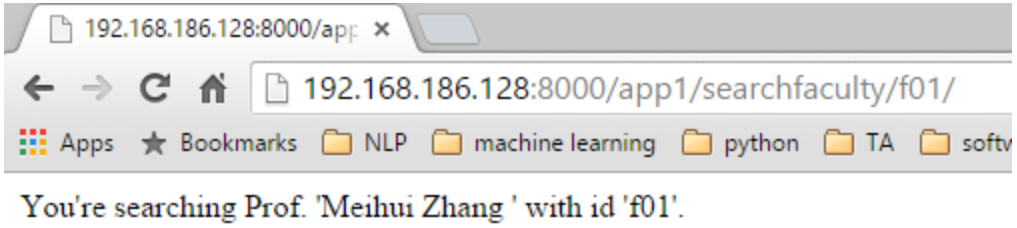
- Map the function into “app1/urls.py”

```
7
8 from django.conf.urls import url
9
10 from . import views
11
12 urlpatterns = [
13     url(r'^$', views.hello, name='hello'),
14     url(r'^searchfaculty/(.{3,9})/$',views.searchfaculty,name='searchfaculty'),
15 ]
16
```

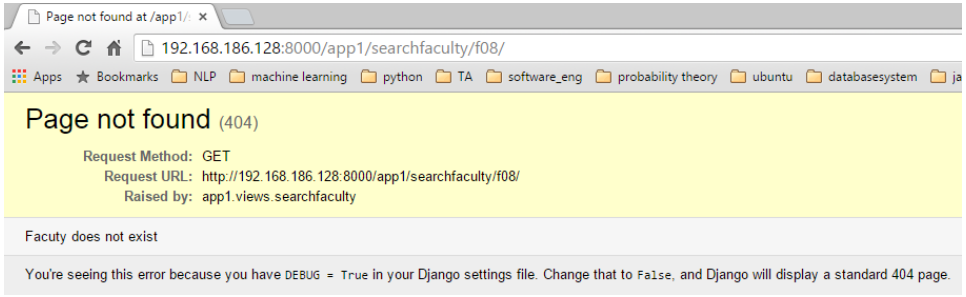
Search a faculty given fid

- Perform a query by entering:

http://192.168.186.128:8000/app1/searchfaculty/f01/

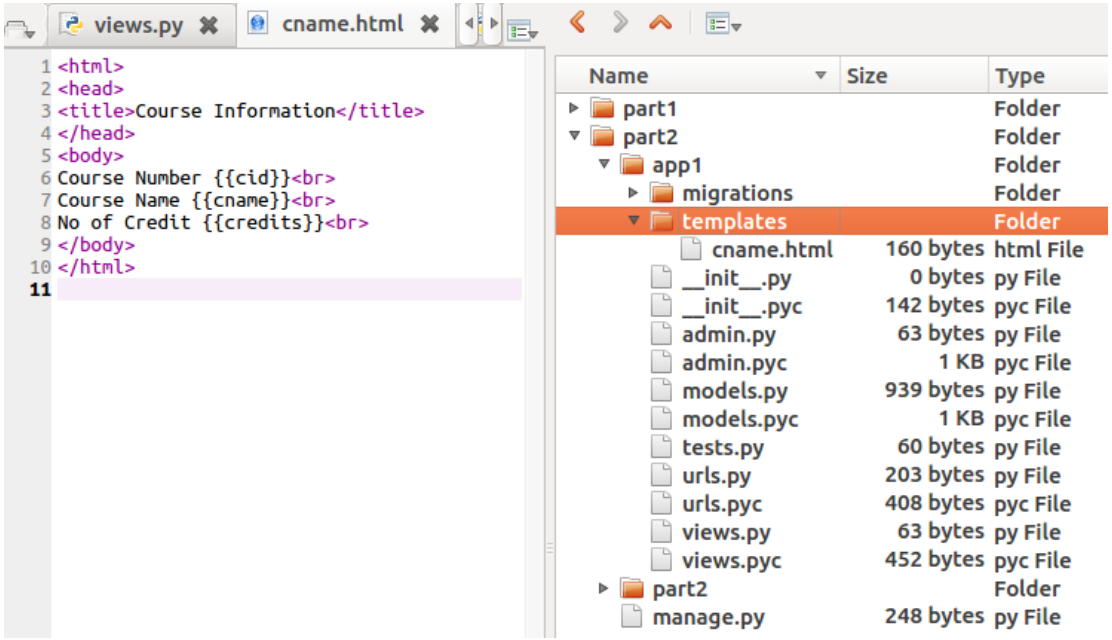


http://192.168.186.128:8000/app1/searchfaculty/f08/



Search a course given cid

- Create a folder named “app1/templates”
- Write cname.html file to display the content of searched page



Search a course given cid

- Write function “searchcourse” in “app1/views.py”, which loads “cname.html”

```
22 def searchcourse(request, cid):
23     conn = mdb.connect (host = "localhost",user = "root",passwd = "",db = "tutorial")
24     with conn:
25         cursor = conn.cursor ()
26         cursor.execute ("select cname,credits from app1_course where cid = '%s'"%(cid))
27         if cursor.rowcount==0:
28             raise Http404("Course does not exist")
29         else:
30             row=cursor.fetchone()
31             context={'cid':cid,'cname':row[0],'credits':row[1]}
32     return render(request,'cname.html',context)
33
34
```

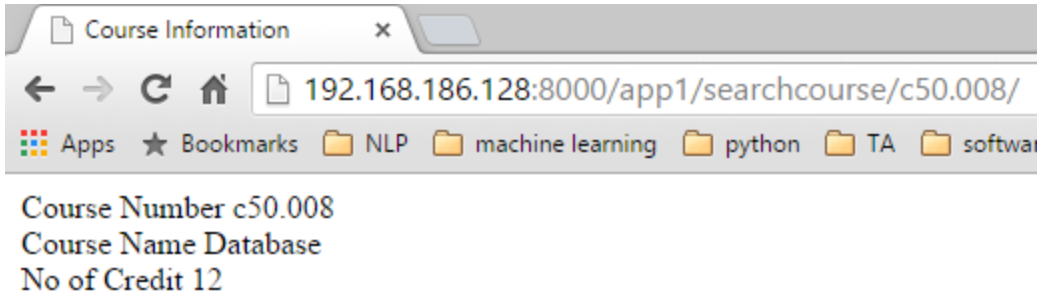
- Map functions into “app1/urls.py”

```
 8 from django.conf.urls import url
 9
10 from . import views
11
12 urlpatterns = [
13     url(r'^$', views.hello, name='hello'),
14     url(r'^searchfaculty/(.{3,9})/$',views.searchfaculty,name='searchfaculty'),
15     url(r'^searchcourse/(.{3,9})/$',views.searchcourse,name='searchcourse'),
16 ]
17
```

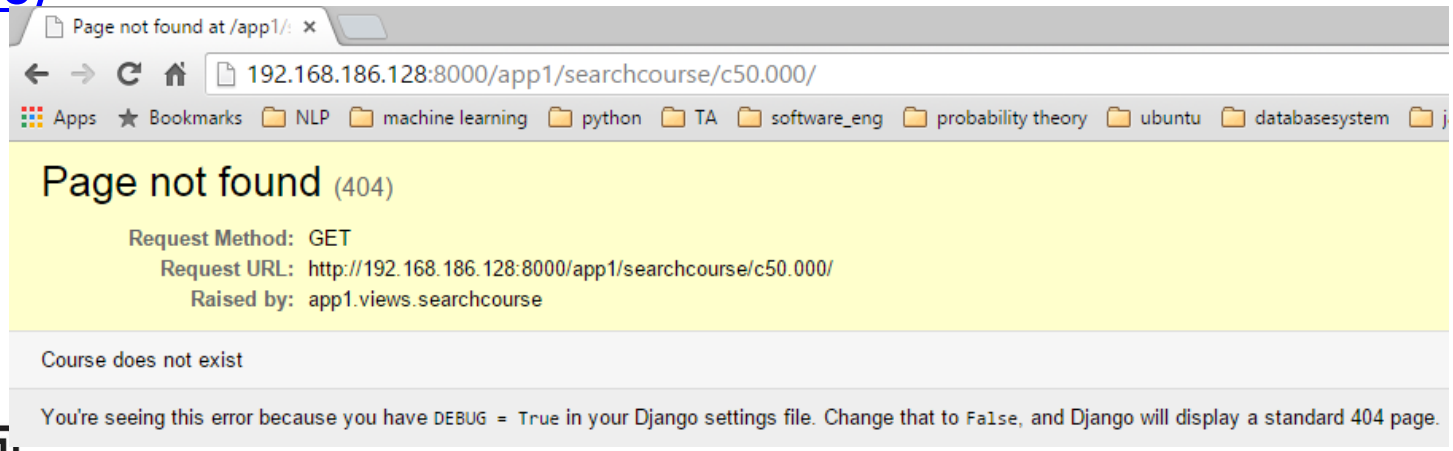
Search a course given cid

- Perform a query by entering:

http://192.168.186.128:8000/app1/searchcourse/c50.008/



http://192.168.186.128:8000/app1/searchcourse/c50.000/



Search open terms given cid

- Create an interface by writing "search_form.html"

urls.py | views.py | search_form.html

```
<html>
<head>
<title>Search Open Sections</title>
</head>
<body>
{% if error %}
<p style="color: red;">Please submit a course number.</p>
{% endif %}
<form action="" method="get">
<input type="text" name="cnum">
<input type="submit" value="Search">
</form>
</body>
</html>
```

- Write "search_result.html" to return the result

.py | views.py | search_form.html | search_result.html

```
<html>
<head>
<title>Open Sections</title>
</head>
<body>
<p>You searched for the course: <strong>{{ query }}</strong></p>
{% if sections %}
<p>Found {{ sections|length }} course{{ sections|pluralize }}.</p>
<ul>
{% for section in sections %}
<li>Teach id: {{ section.tid}}; Course name: {{ section.cid}};
    Faculty: {{ section.fid}}; Term: {{ section.semester}}{{ section.year};
    No of Students: {{ section.enrollment}}; Capacity: {{ section.capacity}}</li>
{% endfor %}
</ul>
{% else %}
<p>No open sections found.</p>
{% endif %}
</body>
</html>
```

Name

- part1
- part2
  - app1
    - migrations
    - templates
      - cname.html
      - search_form.html
      - search_result.html
    - __init__.py
    - __init__.pyc
    - admin.py
    - admin.pyc
    - models.py
    - models.pyc
    - tests.py
    - urls.py
    - urls.pyc
    - views.py
    - views.pyc
  - part2
  - manage.py

Search open terms given cid

- Write function “searchteach” in “app1/views.py”, which calls either “search_form.html” or “search result.html”

```
36 def searchteach(request):
37     error = False
38     cid=None
39     if 'cnum' in request.GET:
40         cid = request.GET['cnum']
41     if not cid:
42         error = True
43     else:
44         sections = teach.objects.filter(cid__exact=cid).filter(enrollment__lt=F('capacity'))
45         return render_to_response('search_result.html',{'sections': sections, 'query': cid})
46     return render_to_response('search_form.html', {'error': error})
47
```

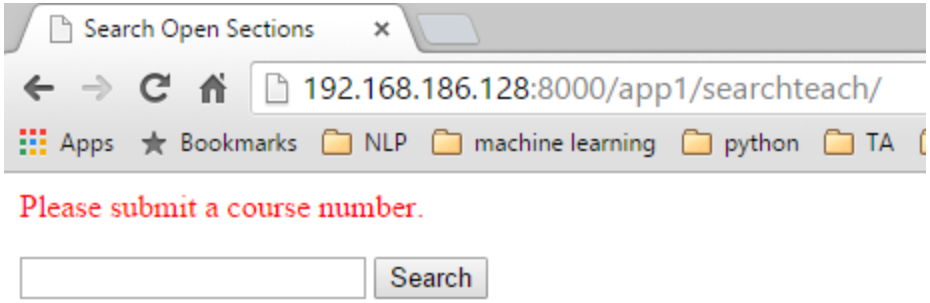
- Map functions into “app1/urls.py”

```
 8 from django.conf.urls import url
 9
10 from . import views
11
12 urlpatterns = [
13     url(r'^$', views.hello, name='hello'),
14     url(r'^searchfaculty/(.{3,9})/$',views.searchfaculty,name='searchfaculty'),
15     url(r'^searchcourse/(.{3,9})/$',views.searchcourse,name='searchcourse'),
16     url(r'^searchteach/$',views.searchteach,name='searchteach'),
17 ]
18
```

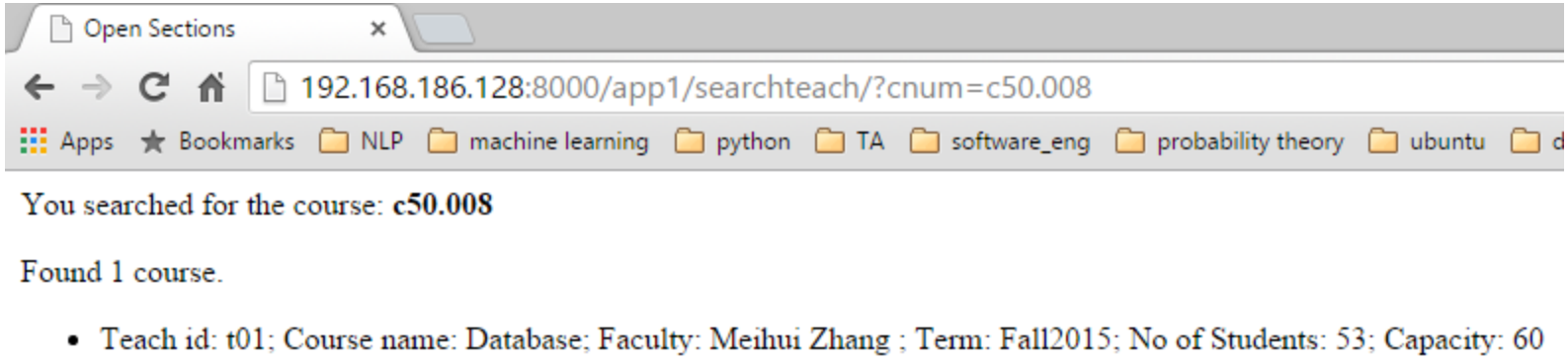
Search open terms given cid

- Perform searching:

  http://192.168.186.128:8000/app1/searchteach/



- Enter "c50.008" into the web interface



References

How to write an HTML file:

- http://csis.pace.edu/~wolf/HTML/htmlnotepad.htm
- https://developer.mozilla.org/en-US/Learn/HTML/Write_a_simple_page_in_HTML

References

Build an app by other programming languages

- http://www3.ntu.edu.sg/home/ehchua/programming/#Cpp
- Tutorial 3 of last semester (build an app using Java)