Dharmsinh Desai University, Nadiad
Faculty of technology, Department of Computer Engineering
Subject : Software Project
Lab Manual

Lab7 : Developing application to retrieve data from database and display in html page.

Requirements : Python, Django,

Prerequisit : Understanding of django framework , working with django projects and apps, creating views, urls, models and templates using django.

Step 1 : Setup the project

1.1 Create the app,

```
$python3.6 manage.py startapp firstdbtest
```

1.2
Go to settings.py and check the following,

- Base directory field should be as follows,

```
BASE_DIR=os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
```

- In template check 'DIRS' field (If it is different from what is shown below, dont change it for now. You may later modify it if you observe template location related error)

```
TEMPLATES = [
            {

'BACKEND':'django.template.backends.django.DjangoTemplates',
                 'DIRS': [os.path.join(BASE_DIR, 'templates')],
                 'APP_DIRS': True,
                 'OPTIONS': {
                     'context_processors': [
                         'django.template.context_processors.debug',
                         'django.template.context_processors.request',
                         'django.contrib.auth.context_processors.auth',
             'django.contrib.messages.context_processors.messages',
                     ],
                 },
            },
      ]
```

1.3 Go to app folder and create app's *urls.py* and *templates* directory.

```
firstdbtest$mkdir templates
firstdbtest$touch urls.py //This creates empty file.
```

include app's ***urls.py*** in projects ***urls.py***. Go to project directory and open ***urls.py***. And update it as shown below.

```
from django.contrib import admin
from django.conf.urls import url,include
urlpatterns = [
     url(r'^admin/', admin.site.urls),
     url('firstdbtest/', include('firstdbtest.urls')),
]
```

Step 2 : Create views and update urls.py

Now let us first create views in (app's ***views.py***)

We need to import the following,

(i) 'HttpResponseRedirect' : to read and right from html,

(ii) 'render_to_response' : to render the resulting web page.

(iii) 'generic' : for Listview

(iv) 'csrf' : a middleware template which protects from 'cross site request forgery' (we don't need to worry about this attach much. However, curious souls may visit, https://docs.djangoproject.com/en/2.0/ref/csrf/)

(v) apart from above 4, lets also add usual TemplateView.

```
from django.shortcuts import render_to_response
from django.views.generic import TemplateView
from django.http import HttpResponseRedirect
from django.views import generic
from django.template.context_processors import csrf
```

(look at the last import csrf. The context_processors path should be same as what we saw in project's settings.py (step 1). Modify the path (here, in views) if it is not the same.)

Now let's start defining views,

(1) getstudentinfo view : It creates a csrf token and calls login.html (defined in templates)

```
def getstudentinfo(request):
     c = {}
     c.update(csrf(request))
     return render_to_response('addstudentinfo.html', c)
```

(2) addstudentinfo view: It reads data from addstudentinfo.html, store it into database table named student and redirect to addsuccess view.

```
def addstudentinfo(request):
     sname = request.POST.get('studentname', '')
     sdate = request.POST.get('birthdate', '')
     s = Student(student_name = sname, student_dob=sdate)
     s.save()
     return HttpResponseRedirect('/firstdb/addsuccess/')
```

(3) addrecord view: This view simply renders addrecord.html.

```
def addsuccess(request):
    return render_to_response('addrecord.html')
```

(4) StudentList view : This view retrieve data from database and assign it to generic List. (Here we need to assign table name to model.

```
class StudentListView(generic.ListView):
    model = Student
```

Now, update app's urls.py with views informations.

```
from django.urls import path
from django.conf.urls import url
from . import views

urlpatterns = [
     # path('', views.index, name='index'),
    url(r'^addstudentinfo/$', views.addstudentinfo),
    url(r'^getstudentinfo/$', views.getstudentinfo),
    url(r'^addsuccess/$', views.addsuccess),
    url('students/', views.StudentListView.as_view(), name =
     'students'),
]
```

Step 3 : go to applications templates directory and create html templates.

(i) base template : base template are place holders where other htmls can put content. Let's first create a base.html as follows,

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Student Portal</title>
    </head>
    <body>
         <h1>Student Record System</h1>
        {% block content %}
        {% endblock %}
    </body>
</html>
```

- Here, {% block content %}   {% endblock %} is the part where other htmls can put their content. Apart from that base.html is not doing much here. You may beutify the base html further and glorify your project. :)

(2) addstudentinfo.html : Write addstudentinfo.html as follows,

```
 {% extends 'base.html' %}
{% block content %}
      {% if form.errors %}

            <p class="error"> Invalid Student info </p>
      {% endif %}
      <form action="/firstdb/addstudentinfo/" method = "post">{% csrf_token %}
            <label for="studentname">Student Name : </label>
            <input type="text" name = "studentname" value = "" id="studentname">
            <label for="birthdate"> Birth Date:</label>
            <input type="date" name="birthdate" value="" id="birthdate">
            <input type="submit" value="Enter"/>
      </form>
      <p> Click <a href="/firstdb/students/"</a> to view student records. </p>
{% endblock %}
```

- Notice that, the login.html extends base.html and puts content between {%block content%} and {% endblock %}. Apart from that the html has two text fields to enter student name, date_of_birth and a submit button.

(3) addrecord.html : This view display successful message and provide the link to getstudentinfo.html

```
  {%extends "base.html" %}
  {% block content %}
      <h2> Student record added successfully </h2>
      <p> Click <a href="/firstdb/getstudentinfo/"</a> to add another  record. </p>
 {% endblock %}
```

(4) student_list.html: Here in file name before underscore sign, we must provide name same as table name and after underscore sign 'list' must be provided.

```
{% extends "base.html" %}
{% block content %}
    <h1>Student List</h1>
    {% if student_list %}
    <ul>
      {% for student in student_list %}
      <li>
        <a href="{{ student.get_absolute_url }}">{{ student.student_name }}</a>
({{student.student_dob}})
      </li>
      {% endfor %}
    </ul>
    {% else %}
      <p>There are no students registered.</p>
    {% endif %}
    <p> Click <a href="/firstdb/getstudentinfo/"</a> to add new record. </p>
{% endblock %}
```

Step 4 : update app's model.py file

create a class for student table as below

```
from django.db import models
# Create your models here.
class Student(models.Model):
     student_name = models.CharField(max_length=100)
     student_dob = models.DateTimeField('date published')
```

Step 5 : Testing the app

(1) Run usual makemigration and migrate commands from project’s root directory. (where manage.py is located)

```
$ python3.6 manage.py makemigrations
$ pthon3.6 manage.py migrate
```

(2) Now run server: `$python manage.py runserver`

(3) go to ***127.0.0.1:8000/firstdbtest/addstudentinfo*** and test the app.