Django

# Reference links

1. <https://www.youtube.com/channel/UCotLFNcTlh1jS-rVUFbu8hg>
2. <https://www.tangowithdjango.com/book17/>
3. educba – [bhadrinathve@gmail.com](mailto:bhadrinathve@gmail.com) – a3952f
4. Model reference - <https://docs.djangoproject.com/en/2.2/ref/models/fields/>
5. <https://wsvincent.com/>
6. <https://www.youtube.com/watch?v=dBctY3-Z5hY>

# Installing Python

1. Download python - <https://www.python.org/downloads/>

Windows x84-64 executable installer

1. To check python installation

Command line –

C:\Users\BHADRINATH>python

Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> ^Z

To close – Ctrl +Z

To check script path

C:\Users\BHADRINATH>pip

# Installing Eclipse

1. Install Eclipse - <https://www.eclipse.org/downloads/>

# Installing PyDev

1. Help
2. Eclipse Marketplace
3. Find: PyDev
4. Install
5. Confirm
6. Accept licence

# Create new python project

1. File -> New -> Other
2. PyDev -> PyDev Project
3. Select the interpreter
4. Next
5. Finish

To chec k interpreter

Window -> Preferences -> PyDev -> Interpreters -> Python Interpreters

# Install Django

1. pip install django –user
2. To check install

python

>>> import django

>>> django.VERSION

(2, 2, 4, 'final', 0)

# Creating Django project from command line

D:\Workspace\Django Training\test>django-admin startproject demo

# Creating Djanjo project in Eclipse

1. File -> New -> Other
2. PyDev -> PyDev Django Project
3. Select the interpreter
4. Next
5. Finish

# Creating basic websitw in Django

1. Create a PyDev package for the application (avtechpassport).
2. In application package, create the modules urls.py and views.py
3. In project package, settings.py

# Application definition

INSTALLED\_APPS = [

*'django.contrib.admin'*,

*'django.contrib.auth'*,

*'django.contrib.contenttypes'*,

*'django.contrib.sessions'*,

*'django.contrib.messages'*,

*'django.contrib.staticfiles'*,

*'Django'*, #Add application package name

]

1. To redirect, add url in urlpatterns in urls.py of project package

from django.contrib import admin

from django.urls import path,include

urlpatterns = [

path(*'admin/'*, admin.site.urls),

path(*'djangotest/'*, include(*'Django.urls'*)),

1. Redirect the url to application module in application package

from django.conf.urls import url

from Django import views

urlpatterns = [

url(*r'^$'*,views.index, name =*'index'*),

]

1. In views.py, add index module for package

from django.http import HttpResponse, request

def **index**(request):

return HttpResponse(*"Hello World"*)

# Creating HTML pages

1. Create a templates folder, at the same level as project and application packages.
2. Create a folder with application name inside templates folder.
3. In settings.py of project package, add template path

TEMPLATE\_PATH = os.path.join(BASE\_DIR, *'templates'*)

TEMPLATES = [

{

*'BACKEND'*: *'django.template.backends.django.DjangoTemplates'*,

*'DIRS'*: [

TEMPLATE\_PATH,

],

*'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. Create index.html file inside application folder of templates

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Rango</title>

</head>

<body>

<h1>Rango says...</h1>

hello world!{{name}} <strong> {{ boldmessage }} </strong> <br />

<a href=*"/rango/about/"*> About </a> <br />

</body>

</html>

1. In views.py of application package, define the view

from django.shortcuts import render

# Create your views here.

from django.http import HttpResponse

def **index**(request):

# Construct a dictionary to pass to the template engine as its context.

# Note the key boldmessage is the same as {{ boldmessage }} in the template!

context\_dict = {*'boldmessage'*: *"I am bold font from the context"*,*'name'*: *"Bhadrinath"*}

# Return a rendered response to send to the client.

# We make use of the shortcut function to make our lives easier.

# Note that the first parameter is the template we wish to use.

return render(request, *'rango/index.html'*, context\_dict)

def **about**(request):

return HttpResponse(*'Rango Says: Here is the about page.<br><a href="/rango/">Back to home page</a>'*)

# Adding static media

1. Create a static folder at the same level as project and application packages.
2. Inside the static folder, create sub folders as

* images
* css
* js

1. Add image to the image folder
2. In index.html file, add image

<img src=*"{% static "*images/logo.jpg" %}" alt=*"Picture of Rango"* />

1. In settings.py file, add static file path

# Static files (CSS, JavaScript, Images)

# <https://docs.djangoproject.com/en/1.10/howto/static-files/>

STATIC\_PATH = os.path.join(BASE\_DIR,*'static'*)

STATIC\_URL = *'/static/'* # You may find this is already defined as such.

STATICFILES\_DIRS = (

STATIC\_PATH,

)

1. In index.html file, include staticfiles

{% load staticfiles %}{% load staticfiles %}

# MVC

Model View Control

# Creating models

category

name =string(max\_length=128, unique=True)

description = string(max\_length=256, blank=True)

views = IntegerField(default=0) **How many times viewed**

slug = (populate\_from=’name’, unique =True)

status = BooleanField(default=True)

created\_at = DateTimeField(null=True, auto\_now\_add=True) Time stamp

updated\_at = DateTimeField(null=True, auto\_now=True)

course

category = ForeignKey(Category) **In table -> category id**

title = CharField(max\_length = 128)

url= URLField()

summary = CharField(max\_length =256, blank = True)

phone = CharField(max\_length=15)

startingdate = DateField(auto\_now=False, auto\_now\_add=False)

hascertification = BooleanField(default=False)

status = BooleanField(default=True)

views = IntegerField(default=0)

Django also provides simple mechanism that allows us to relate models/ database tables together. These mechanism are encapsulated in three further field types, and are listed below.

ForeignKey: a field type that allows us to create a one-to-many relationship.

OneToOneField: a field type that allows us to define a strict one-to-one relationship.

ManyToManyField: a field type which allows us to define a many-to-many relationship.

In modules.py

from django.db import models

from autoslug import AutoSlugField **pip install django-autoslug**

# Create your models here.

class **Category**(models.Model):

name = models.CharField(max\_length=128, unique=True)

description = models.CharField(max\_length=256, blank=True)

views = models.IntegerField(default=0)

slug = AutoSlugField(populate\_from=*'name'*, unique =True)

**AutoSlugField convert English phase into legal acceptable URL**

status = models.BooleanField(default=True)

created\_at = models.DateTimeField(null=True, auto\_now\_add=True)

updated\_at = models.DateTimeField(null=True, auto\_now=True)

def **\_\_str\_\_**(*self*):

return *self*.name

class **Course**(models.Model):

category = models.ForeignKey(Category)

title = models.CharField(max\_length = 128)

url= models.URLField()

summary = models.CharField(max\_length =256, blank = True)

phone = models.CharField(max\_length=15)

startingdate = models.DateField(auto\_now=False, auto\_now\_add=False)

hascertification = models.BooleanField(default=False)

status = models.BooleanField(default=True)

views = models.IntegerField(default=0)

def **\_\_str\_\_**(*self*):

return *self*.title

# Database connectivity

## Java

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.Statement;

**public** **class** dbcheck

{

**public** **static** **void** main(String args[])

{

**try**{

//step1 load the driver class

Class.*forName*("oracle.jdbc.driver.OracleDriver");

//step2 create the connection object

Connection con=DriverManager.*getConnection*(

"jdbc:oracle:thin:@localhost:1521/xe","system","root");

//step3 create the statement object

Statement stmt=con.createStatement();

//step4 execute query

ResultSet rs=stmt.executeQuery("select \* from emp");

**while**(rs.next())

System.***out***.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));

//step5 close the connection object

con.close();

}**catch**(Exception e){ System.***out***.println(e);}

}

}

## Python

# Creating and making migration

1. In settings.py, add

DATABASES = {

*'default'*: {

*'ENGINE'*: *'django.db.backends.sqlite3'*,

*'NAME'*: os.path.join(BASE\_DIR, *'db.sqlite3'*),

}

}

1. In command prompt, go to the project path

python manage.py makemigrations

Migrations for 'rango':

rango\migrations\0001\_initial.py:

- Create model Category

- Create model Course

1. python manage.py migrate
2. To create superuser account for <http://127.0.0.1:8000/admin>

python manage.py createsuperuser

**admin, bhadri1007**

1. Register category and course in admin model

In admin.py of application

1. Update category and course in django admin

# Using templates and views to display data in database

Steps to get the records in the database to the web pages

(Assuming you already have the model created):

1. Update the ‘index’ function in the application (rango)’s ‘views.py’ module

The function will retrieve the records from the database

Then pass the retrieved records to the designated template by use a Python dictionary

1. Update the template to use the Django template variable that was passed via ‘index’

Add the Category page to the site

* Modify the ‘index.html’ byuse the ‘slug’ attribute in category object the to constructor while iterating through each object in the dictionary.
* While we get the <href>, the link goes nowhere.
* Check the error message. It is giving us the hint.
* Need to add a new URl switch in the ‘rango’ application ‘urls.py’ file.
* Here it is:

url(r’^category/(?P<category\_name\_slug>[\w\-]+)$’, views.category. name=’category’)

Please note the “(?P<category\_name\_slug>[\w\-]+)”. This is a python regular expression.

It says: only accept a string that contains a-z, A-Z, 0-9. Once it passed validation then assign it to ‘category\_name\_slug’. The state also says: call views.category with the variable ‘categiry\_name\_slug’ as the parameter.

* With the ‘switch’ statement in place, we need to add a function ‘category’.

This is to tell URL switch what function to call when the link is clicked.

In views.py of application package

category\_list = Category.objects.order\_by(‘-views’)[:5]

**Order by views**

**-views -> (-) Descending**

**[:5] -> First five records**

context\_dict = {‘categories’: category\_list}

In index.html page

<!DOCTYPE html>

{% load staticfiles %}{% load staticfiles %}

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Rango</title>

</head>

<body>

<h1>Rango says...</h1>

{% if categories %}

<ul>

{% for category in categories %}

<li><a href=*"/rango/category/{{ category.slug }}"*>{{category.name}}</a></li>

{% endfor %}

</ul>

{% else %}

<strong>There are no categories present.</strong>

{% endif %}

<a href=*"/rango/about/"*> About </a> <br />

<img src=*"{% static "*images/logo1.jpg" %}" alt=*"Picture of Rango"* />

</body>

</html>

# Forms to database

The basic steps involved in creating a form and allowing users to enter data via the form is as follows,

1. If you haven’t already get one, create a forms.py file within your Django application’s directory. This is to store form-related classes.
2. Create a model form class for each model that you wish to represent as a form
3. Customise the forms as you desire.
4. Create or update a view to handle the form including displaying the form, saving the form data, and flagging us errors which may occur when the user enters incorrect data (or no data at all) in the form.
5. Create or update a template to display the form.
6. Add a url pattern to map to the view (if you created a new one).

# Notes:

## To Cover

* JQuery
* Bootstrap – Data table
* Node
* Spring boot
* REST

## Oracle details

* url : 127.0.0.1:8080/
* user name : system
* password : root

## Course

503149