Django by Kamal Sir

Index Page:

Topic No:	Topic Name:	Page No:
1	Django Basics	1
2	Django Project and Apps	2
3	Django MVT	3
4	HTML – Hyper Text Markup Language	5
5	CSS – Cascading Style Sheet	12
6	DTL – Django Template Language	16
7	Form Validation - HTML	18
8	Form Validation – JavaScript	19

The book is to be used with the points discussed in the lectures for understanding Django with Kamal Sir.

Topic 1: Django Basics

Framework:

- Framework is a set of conceptual structure and guidelines, used to build something useful.
- Framework may include predefined classes and functions which would allow the developer to focus on building their project faster rather than writing code from scratch.
- Framework Advantages:
  - Save Time
  - Improve Productivity
  - Clean code
  - Reusable Code

Web Framework:

- Web Framework help to build Web Applications.
- Web Framework provide tools and libraries to simplify web development operations like:
  - Templating
  - Database Operations
  - Authentications
- Some popular Web Framework are:
  - Java → Spring
  - Php → Laravel, CodeIgniter
  - Python → Django, Flask

Django:

- Django is **Free, Open-Source Python based Web Framework** for rapid development of web apps quickly and with less code.
- It is also called batteries included framework because it provides:
  - Development WebServer
  - Forms
  - Database – SQLite3
  - Models for database management
  - Admin panel for administrative activities
  - User Authentication

Django Insights:

- It was created in 2003 by Web Programmers, Adrian Holovaty and Simon Willison while working at the Lawrence Journal-World newspaper.
- It is named after jazz guitarist Django Reinhardt.
- Django is maintained by Django Software Foundation (DSF) from 2008.

Topic 2: Django Project and Apps

Django Commands:

- **To install:**
  - > pip install django
- **To uninstall:**
  - >pip uninstall django
- **To check version:**
  - >django-admin --version

Django Project:

- Django Project may contain multiple Project Application.
- It means group of Application and Files is called as Django Project.

Understanding Project Directory:

- **manage.py:** it helps to create applications in the project and also to run the server.
- **settings.py:** it contains information about project settings like:
  - Installed Apps
  - Templates
  - Database Configurations.
- **urls.py:** it contains the information of url attached with the application

Run Server:

- Django provides built-in web server for developing and testing purpose.
- This server is pre-configured to work with Django and it restarts whenever you modify the code.

Django Apps:

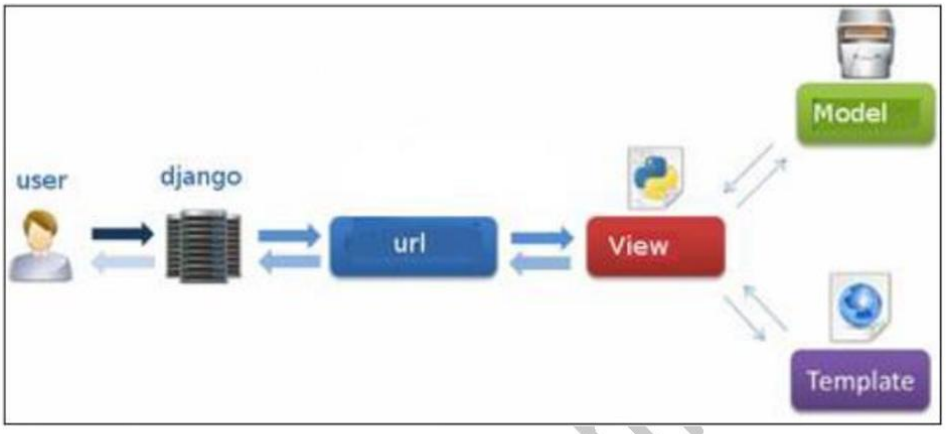
- Django project contains one or more applications.

Understanding App Directory:

- **admin.py:** it is used to register sql tables so that we can perform CRUD operation from Admin Application.
- **apps.py:** it is used to config app
- **models.py:** it is used to create model class which would be later converted into database table by Django.
- **tests.py:** it is used to create tests.
- **views.py:** it is the place where we write our logic.
- **migrations folder:** it will contain the files which are created after running makemigration command.

Topic 3: Django MVT

Django MVT Pattern:



Explanation:

- Django itself takes care of the Controller part (Software Code that controls the interactions between the Model and View), leaving us with the template.
- The developer provides the Model, the view and the template then just maps it to a URL and Django does the magic to serve it to the user.
- The template is a HTML file mixed with Django Template Language (DTL).

URL Dispatcher:

- urls.py contains mapping between URL path expressions and view functions.
- Syntax:
- **path(route, view, name=None)**
  - Where:
  - **route** – contains URL pattern
  - **view** – is the name of the view function defined in views.py.
  - **name** – it is used for URL reversing.
- Note:
  - we can also specify **dictionary to pass arguments** to the view function.
  - We can define multiple url for same view function.

Function Based Views:

- Function Based View, is a Python function that takes a web request and returns a web response.
- Response can be:
  - Text, Html contents, Redirect, Data from Database, Etc.
- Syntax:
- **def function_name(request):**
  return HttpResponse('text/html')

Template:

- Template is a text file which can generate any text based format(eg HTML, XML).
- Template is used by view function to represent the data to the user.
- When we create templates, we separate business logic and presentation
  - **Business logic:** **views.py**
  - **Presentation :** **template file**
- Template can contain
  - Variables: which get replaced with values when template is evaluated and
  - tags : which control the logic of the template.

render():

- views.py is responsible to process the template files for which we use render() function in views.py
- Syntax:
- **render(request, template_name, context=dict_name)**
  - **request:** request object used to generate this response.
  - **template_name:** full template name to be used.
  - **context:** dictionary of values to add to the template context.

Why URL pattern inside app:

- In all practicals until now we have defined url pattern at project level.
- This increases the dependency of apps in project, meaning if we want to use a particular application for another project we may face issues.
- Benefits of defining url inside app:
  - Reduce the dependency of app on project.
  - Reusability of app becomes easy.

include():

- It is a function that takes a full Python import path to another URL module that should be included in this place.
- Syntax is :
  - **include(module)**

Topic 4: HTML – Hyper Text Markup Language

Introduction:

- HTML stands for Hyper Text Markup Language, most widely used to develop web pages.
- HTML was created by Berners-Lee in 1991 and HTML-5.x is the latest version.
- Hypertext refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.
- Markup Language means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

HTML Page Structure:

```
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

HTML Tags:

- HTML uses various tags to format the content.
- HTML tags are not case sensitive, but its recommended to write in lowercase.
- These tags are enclosed with angle braces <Tag Name>.
- Mostly tags has opening tag and closing tag.
- <html> is opening tag and </html> is closing tag.
- Never skip the closing tag else we can get unexpected results.
- The purpose of web browser is to read HTML documents and display them correctly. Browser does not display the HTML tags, but uses them to determine how to display the document.

HTML Element:

- An HTML element is defined by:
  - Start tag
  - Some content
  - End tag
- **Format:**
- <tagname> Content </tagname>
- **Eg:**
  - <h1>This is a heading</h1>
  - Hello World!

HTML Comments:

- HTML comments are not displayed in the browser, but they can help document your HTML source code.
- Syntax is:
- <!-- Write your comments here -->
- Comments are also great for debugging HTML, because you can comment out HTML lines of code, one at a time, to search for errors.

Heading Tags:

- Any document starts with a heading.
- HTML has six levels of headings, <h1> most important heading to <h6> least important heading.
- While displaying any heading, browser adds margin before and after the heading.

Paragraph Tag:

- The tag offers a way to structure your text into different paragraphs.
- Each paragraph of text should go in between opening and closing .
- Paragraph always starts on a new line and browser will add margin before and after a paragraph.
- The browser will automatically remove any extra spaces and lines when the page is displayed.

Line Break Tag:

- Whenever we use
 , anything following it starts from the next line.

Horizontal Lines:

- They are used to visually break-up sections of document.
- <hr> tag creates a line in the document.

Centering Content:

- We can use tag to put any content in the center of the page.

HTML Forms:

- An HTML form is used to collect user input. The user input is most often sent to a server for processing.
- The <form> element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc.
- Syntax is:

```
<form>
.
form elements
.
</form>
```

HTML Forms Attributes:

action:

- The action attribute defines the action to be performed when the form is submitted.
- Usually, the form data is sent to a file on the server when the user clicks on the submit button.

target:

- The target attribute specifies where to display the response that is received after submitting the form.
- _blank
  - The response is displayed in a new window or tab
- _self
  - The response is displayed in the current window

method:

- ❑ The method attribute specifies the HTTP method to use used when submitting the form data.
- ❑ The form-data can be sent as URL variables (with method="get") or as HTTP post transaction (with method="post").
- ❑ The default HTTP method when submitting form data is GET.

Eg:

1) <form method="get">
2) <form method="post">

GET:

- ✓ Appends the form data to the URL, in name/value pairs
- ✓ GET is good for non-secure data, like query strings in Google
- ✓ The length of a URL is limited (2048 characters)
- ✓ Useful for form submissions where a user wants to bookmark the result

Note: NEVER use GET to send sensitive data! (the submitted form data is visible in the URL!)

POST:

- ✓ Appends the form data inside the body of the HTTP request (the submitted form data is not shown in the URL)
- ✓ Use POST if the form data contains sensitive or personal information!
- ✓ POST has no size limitations, and can be used to send large amounts of data.
- ✓ Form submissions with POST cannot be bookmarked

HTML input type:

text:

- <input type="text"> defines a single-line text input field

password:

- <input type="password"> defines a password field

number:

- The <input type="number"> defines a numeric input field.
- You can also set restrictions on what numbers are accepted.
- For float: step
  - Specifies the legal number intervals for an input field.
  - step="0.01" for currencies step="any" for everything else.

email:

- The <input type="email"> is used for input fields that should contain an e-mail address.

tel:

- The <input type="tel"> is used for input fields that should contain telephone number.

submit:

- <input type="submit"> defines a button for submitting form data to a form-handler.
- The form-handler is typically a server page with a script for processing input data.

reset:

- The <input type="reset"> defines a reset button which resets all form values to its initial values.

radio:

- <input type="radio"> defines a radio button.
- Radio buttons let a user select ONLY ONE of a limited number of choices:

checkbox:

- <input type="checkbox"> defines a checkbox.
- Checkboxes let a user select ZERO or MORE options of a limited number of choices.

date:

- The <input type="date"> is used for input fields that should contain a date.
- Depending on browser support, a date picker can show up in the input field.

time:

- The <input type="time"> allows the user to select a time (no time zone).
- Depending on browser support, a time picker can show up in the input field.

HTML input attributes:

placeholder:

- The input placeholder attribute specifies short a hint that describes the expected value of an input field (a sample value or a short description of the expected format).
- The short hint is displayed in the input field before the user enters a value.
- The placeholder attribute works with the following input types: text, search, url, tel, email, and password.

value:

- The input value attribute specifies an initial value for an input field

required:

- The input required attribute specifies that an input field must be filled out before submitting the form.
- The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

autofocus:

- The input autofocus attribute specifies that an input field should automatically get focus when the page loads.

size:

- The input size attribute specifies the visible width, in characters, of an input field.
- The default value for size is 20.
- **Note:** The size attribute works with the following input types: text, search, tel, url, email, and password.

maxlength:

- The input maxlength attribute specifies the maximum number of characters allowed in an input field.
- **Note:** When a maxlength is set, the input field will not accept more than the specified number of characters. However, this attribute does not provide any feedback. So, if you want to alert the user, you must write JavaScript code.

min and max:

- The input min and max attributes specify the minimum and maximum values for an input field.
- The min and max attributes work with the following input types: number, range, date, datetime-local, month, time and week.
- **Tip:** Use the max and min attributes together to create a range of legal values.

pattern:

- The input pattern attribute specifies a regular expression that the input field's value is checked against, when the form is submitted.
- The pattern attribute works with the following input types: text, date, search, url, tel, email, and password.

autocomplete:

- The input autocomplete attribute specifies whether a form or an input field should have autocomplete on or off.
- Autocomplete allows the browser to predict the value. When a user starts to type in a field, the browser should display options to fill in the field, based on earlier typed values.
- The autocomplete attribute works with <form> and the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.

Topic 5: CSS - Cascading Style Sheets

Introduction:

- CSS is used to format the layout of a webpage.
- HTML is a structural language and CSS is presentational language for web pages.
- With CSS we can control:
  - color, font, text size, etc
- CSS can be added to the HTML documents in 3 ways:
  - **Inline:** by using the <style> attribute inside HTML elements.
  - **Internal:** by using a <style> element in the <head> section.
  - **External:** by using a <link> element to link to an external CSS file.

Precedence of style rules:

- The type of style being considered more important than other is known as precedence of style.
- Order of precedence is as follow:
  1. Inline style
  2. Internal style
  3. External style
  4. Browser default style

Inline CSS:

- An inline CSS is used to apply a unique style to a single HTML element.
- An inline CSS uses the style attribute of an HTML element.

Limitations of Inline CSS:

- If we have many tags and all of them have same/common styles, then we have to copy paste the entire declaration list to every other tag which would increase the webpage size.

Internal CSS:

- An internal CSS is used to define a style for single HTML page.
- An internal CSS is defined in the <head> section of an HTML page within <style> element.

Limitations of Internal CSS:

- If we have many webpages and all of them have same/common styles, then we have to copy paste the entire declaration list to every other page which would increase the website size.

External CSS:

- An external CSS is used to define the style for many HTML pages.
- In this we define css file with extension .css
- To use an external CSS, add a link to the <head> section of each HTML page.

Selector types:

- To apply the style to an element or more than one element we use selectors.
- Few types of selectors are: tag selector, class selector , id selector etc

Tag Selector:

- To select tags by their name and apply styles to them we use tag selector.
- Syntax is:

```
tagname {
                declaration_list;
}
```

id Selector:

- To select tags by their id attribute value and apply styles to them we use id selector.
- Syntax is:

```
#idattributevalue {
                declaration_list;
}
```

class Selector:

- To select tags by their class attribute value and apply styles to them we use class selector.
- Syntax is:

```
.classattributevalue {
                declaration_list;
}
```

- We can specify the same class attribute value to more than one tag.
- An html element can have list of class attribute values separated by white space.

HTML style attribute:

- HTML style attribute is used to add styles to an element such as color, font, size and more.
- Syntax is:
  - **<tagname style="property:value;">**
  - property is CSS property and value is CSS value.

background-color

- defines the background color for an HTML element.

color

- defines the text color for an HTML element

border

- defines the border color for an HTML element. (width style color)
- border-width: px, pt, cm, etc
- border-style: dotted, dashed, solid, double etc

HTML Colors:

- HTML colors are specified with predefined color names, or with RGB, HEX, HSL, RGBA, or HSLA values.
- All browsers support 140 color names.
- Value can be colorname, #hexcode, rgb(0-255, 0-255, 0-255), rgba(0-255, 0-255, 0-255, 0.1)

font-family:

- Web safe fonts are fonts that are universally installed across all browsers and devices.
- You should add a list of similar "backup fonts" in the font-family property. If the first font does not work, the browser will try the next one, and the next one, and so on. Always end the list with a generic font family name.
- Eg: font-family: Tahoma, Verdana, sans-serif;

font-size:

- The CSS font-size property defines the text size for an HTML element.
- Values supported are: xx-small , x-small, smaller, small, medium, large, largers, x-large, xx-large, length or %.

font-weight:

- The CSS font-weight property defines the boldness of the font.
- Values supported are: lighter, normal, bold, bolder, 100, 200, … 900

font-style:

- The CSS font-style property defines the style of the text to be displayed.
- Values supported are: normal, italic, oblique

text-align:

- The CSS text-align property defines the horizontal text alignment for an HTML element.
- Values supported are: left, right, center, justify

text-decoration:

- The CSS text-decoration property specifies the line formatting of the text to be displayed.
- Values supported are: none, underline, line-through, overline

Topic 6: DTL - Django Template Language

Introduction:

Why DTL?

- Django makes it possible to separate Python(logic and code) and HTML(design)
- Python code goes in views and HTML goes in templates.
- To link the Python and HTML, Django uses
  - render function and
  - Django Template Language - DTL

---

render function:

- **render()** takes as input three parameters:
  1. **request:** the initial request
  2. **path to template:** path to the html files (project level | app level)
  3. **dictionary of parameters:** it is a dict that contains all the variables needed in the template.

What is DTL?

- Django's template engine provides a mini-language DTL to pass information from Python to HTML.
- DTL consists of:
  1) Template variable
  2) Template tag
- Template variable is surrounded by {{ var_name }}
- Template tag is surrounded by {% %}

Template Tag:

- Template tag can be used for:
- Display logic:
  - **{% if %} ... {% endif %}**
- Loop Control :
  - **{% for d in data %} ... {% endfor %}**
- URL:
  - **{% url 'some_url_name'%}**
- To link static files:
  - **{% load static %}**
  - **<img src="{% static 'images/hi.jpg' %}" alt="Hi!">**
  - **<link rel="stylesheet" href="{% static user_stylesheet %}" type="text/css" >**
- Security:
  - **{% csrf_token %}**

if Tag:

- {% if variable %} tag evaluates a variable and if that variable exists / true the content of the block are output.
- The syntax is:

```
{% if variable %}
        // statements
{% else %}
        // statements
{% endif %}
```

- In if condition we can use operators too:
  - Relational operators: ==, !=, <, <=, >, >=
  - Boolean operators: and, or and not.

for Tag:

- for tag can be used to loop over each item in the given list.
- Anything enclosed between for tag would be repeated the number of times the loop is run.
- Syntax is:

```
{% for i in list %}
        {{ i }}
{% endfor %}
```

CSRF token:

- CSRF stands for Cross-site Request Forgery and is a type of security threat to websites taking form data.
- Django provides you built-in support for this, by making it necessary to use this token.
- The CSRF field has some value which is randomly generated by your server and that value shall match with your server when the form is submitted.
- That particular value is also not acceptable by other servers. Thus, web data remains where it should be.

Topic 7: Form Validations - HTML

Form Validation:

- Form Validation means the data entered would be checked by the browser and/or the server to see that
  - The data is in the correct format.
  - Data is within the constraints set by the application.
- Validation done in the browser is called **client-side validation** and validation done on the server is called **server-side validation**.

Client side validation:

- Client-side validation is an initial check and an important feature for catching invalid data on the client side so the user can fix it right away.
- if invalid data gets to the server and is then rejected, a noticeable delay is caused by a round trip to the server and then back to the client side to tell the user to fix the data.

Different types of client-side validation:

- There are two types of client-side validation:
- **1 Built-in form validation:**
  - it uses HTML5 form validation features.
  - But it is not as customizable as JavaScript validation.
- **2) JavaScript validation:**
  - It uses JavaScript for validation.
  - This validation is completely customizable.

Built-in form validation:

- HTML5 form controls can validate most user data without relying on JavaScript.
- This can be done by using following validation attributes on form elements:
  - **required:**
    specifies whether a form field needs to be filled in before the form can be submitted.
  - **minlength and maxlength:**
    specifies the min and max length of textual data(strings).
  - **min and max:**
    specifies the min and max values of numerical input types.
  - **type:**
    specifies whether the data needs to be a number, an email address, etc.
  - **pattern:**
    specifies a regular expression that defines a pattern the entered data needs to follow.

Topic 8: Form Validations - JavaScript

JavaScript HTML DOM:

- When a web page is loaded, the browser creates a Document Object Model of the page.
- HTML DOM is a standard for how to get and change HTML elements.
- **Finding HTML element:**
  - **document.getElementById(id)**
    Find an element by element id
  - **document.getElementsByTagName(name)**
    Find elements by tag name
  - **document.getElementsByClassName(name)**
    Find elements by class name

- **Changing HTML element:**
  - **element.attribute = new value**
    Change the attribute value of an HTML element
  - **element.setAttribute(attribute, value)**
    Change the attribute value of an HTML element

- **Alert Box: alert():**
  When an alert box pops up, the user will have to click OK to proceed.
- **focus():**
  This method is used to give focus to an element.
- **onSubmit():**
  It is the attribute that fires when a form is submitted.