DIGITAL NOTES
OF
WEB APPLICATION DEVELOPMENT
(R22A1204)
B.TECH III YEAR–I SEM
(2024-2025)

PREPARED BY

P. SWETHA
V. SUDHA RANI
Dr.A. MUMMOORTHY
T. SRINIDHI

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

MALLA REDDY COLLEGE OF ENGINEERING &TECHNOLOGY
(Autonomous Institution–UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE-Accredited by NBA & NAAC–A'Grade-ISO9001:2015Certified)

Maisammaguda,Dhulapally(PostVia.Hakimpet),Secunderabad–500100,TelanganaState,India

MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

B.TECH - III- YEAR I-SEM-CSIT **L/T/P/C**
3/-/-/3

(R22A1204) WEB APPLICATION D EVELOPMENT

COURSE OBJECTIVES:

1. To study the insights of the Web architecture and PHP Framework.
2. To gain knowledge in interfacing Java Servlet Program with JDBC Connection.
3. To be trained to dynamically generate the web pages using Java Server Pages.
4. To understand the designing applications over web using Spring Framework.
5. To get acquainted with applications over the web using 'Django' Framework.

UNIT-I:

Web Basics and Overview: Introduction to Internet, World Wide Web, Web Browsers, URL,HTTP. **PHP**: Declaring Variables, Data types, Operators, Control structures, Functions. MVC Framework and Design Pattern, Types of PHP MVC framework.

UNIT-II

Servlets: Introduction to Servlets, Benefits of Servlets, use as controller in MVC, servlet lifecycle, basic HTTP, Reading Servlet parameters, servlet session, cookies, Servlets API.
JDBC: JDBC Architecture,,JDBC API, Connecting to a Database Using JDBC.

UNIT-III

Java Server Pages: Generating Dynamic Content, Using Scripting Elements, Implicit JSP Objects. Conditional Processing – Displaying Values, setting attributes, Error Handling and Debugging.

UNIT-IV

Java Script: Introduction to Java Script, Declaring variables, Functions, Event handlers (onclick, on submit, etc.,) and Form Validation.

Spring Framework: Overview, Controllers, Handler Methods, and Developing Web Application using spring.

UNIT-V

Django: Introduction to Django, Django architecture, Django Models and Database Backends, Developing Web Application using Django.

TEXT BOOKS:

1. Hans Bergsten , Java Server Pages, O'Reilly, 2003
2. Jason Hunter, William Crawford , Java Servlet Programming, Second Edition, , O'Reilly Media

REFERENCE BOOKS:

1. Joseph J. Bambara, Paul R. Allen, Mark Ashnault, Ziyad Dean, Thomas Garben, Sherry Smith J2EE UNLEASHED — SAMS Techmedia 5 StepahnieBodoff, Dale Green, Kim Hasse, Eric Jendrock, Monica Pawlan, Beth Stearns , The J2EE Tutorial, Pearson Education , Asia.
2. Learning Django Web Development, Sanjeev Jaiswal Ratan Kumar, PACKT Publishing.
3. https://www.djangoproject.com/ spring framework (IBM).

Course Outcomes:

1. Analyze a web page and identify its elements and attributes of PHP Frame Work.
2. Installation and usage of Server Software's & Web.xml Deployment.
3. Database Connectivity to web applications.
4. Build web applications using Servlet and JSP.
5. Build web applications using Django.

INDEX

UNIT - I

Web Basics and Overview: Introduction to Internet, World Wide Web, Web Browsers, URL, HTTP. **PHP**: Declaring Variables, Data types, Operators, Control structures, Functions, MVC Framework and Design Pattern, Types of PHP MVC framework.

Introduction to Internet:

A global computer network providing a variety of information and communication facilities, consisting of interconnected networks using standardized communication protocols.

The Internet is the global system of interconnected computer networks that use the Internet protocol suite (TCP/IP) to link devices worldwide. It is a network of networks that consists of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies. The Internet carries a vast range of information resources and services.

History of Internet:

This marvelous tool has quite a history that holds its roots in the cold war scenario. A need was realized to connect the top universities of the United States so that they can share all the research data without having too much of a time lag. This attempt was a result of Advanced Research Projects Agency (ARPA) which was formed at the end of 1950s just after the Russians had climbed the space era with the launch of Sputnik. After the ARPA got success in 1969, it didn't take the experts long to understand that how much potential can this interconnection tool have. In 1971 Ray Tomlinson made a system to send electronic mail. This was a big step in the making as this opened gateways for remote computer accessing i.e. telnet.

During all this time, rigorous paper work was being done in all the elite research institutions. From giving every computer an address to setting out the rules, everything was getting penned down. 1973 saw the preparations for the vital TCP/IP and Ethernet services. At

the end of 1970s, Usenet groups had surfaced up. By the time the 80s had started, IBM came up with its PC based on Intel 8088 processor which was widely used by students and universities for it solved the purpose of easy computing. By 1982, the Defense Agencies made the TCP/IP compulsory and the term internet was coined. The domain name services arrived in the year 1984 which is also the time around which various internet based marked their debut. A worm, or a rust the computers, attacked in 1988 and disabled over 10% of the computer systems all over the world. Quite a number of computer companies became interested in dissecting the cores of the malware which resulted to the formation Computer Emergency Rescue Team (CERT). Soon after the world got over with the computer worm, World Wide Web came into existence.

World Wide Web

The World Wide Web (abbreviated WWW or the Web) is an information space where documents and other web resources are identified by Uniform Resource Locators (URLs), interlinked by hypertext links, and can be accessed via the Internet. English scientist Tim Berners-Lee invented the World Wide Web in 1989. He wrote the first web browser computer program in 1990 while employed at CERN in Switzerland. The Web browser was released outside CERN in 1991, first to other research institutions starting in January 1991 and to the general public on the Internet in August 1991.

The World Wide Web has been central to the development of the Information Age and is the primary tool billions of people use to interact on the Internet. Web pages are primarily text documents formatted and annotated with Hypertext Markup Language (HTML). In addition to formatted text, web pages may contain images, video, audio, and software components that are rendered in the user's web browser as coherent pages of multimedia content.

Embedded hyperlinks permit users to navigate between web pages. Multiple web pages with a common theme, a common domain name, or both, make up a website. Website content can largely be provided by the publisher, or interactively where users contribute content or the content depends upon the users or their actions. Websites may be mostly informative, primarily

for entertainment, or largely for commercial, governmental, or non-governmental organizational purposes.

WWW is another example of client/server computing. Each time a link is followed, the client is requesting a document (or graphic or sound file) from a server (also called a Web server) that's part of the World Wide Web that "serves" up the document. The server uses a protocol called HTTP or Hyper Text Transport Protocol. The standard for creating hypertext documents for the WWW is Hyper Text Markup Language or HTML. HTML essentially codes plain text documents so they can be viewed on the Web.

Web Browsers:

WWW Clients, or "Browser": The program you use to access the WWW is known as a browser because it "browses" the WWW and requests these hypertext documents. Browsers can be graphical, allows seeing and hearing the graphics and audio.

Text-only browsers (i.e., those with no sound or graphics capability) are also available. All of these programs understand http and other Internet protocols such as FTP, gopher, mail, and news, making the WWW a kind of "one stop shopping" for Internet users.

Year	List of Web browsers
1991	World Wide Web (Nexus)
1992	Viola WWW, Erwise, MidasWWW, MacWWW (Samba)
1993	Mosaic, Cello,[2] Lynx 2.0, Arena, AMosaic 1.0
1994	IBM WebExplorer, Netscape Navigator, SlipKnot 1.0, MacWeb, IBrowse, Agora (Argo),Minuet
1995	Internet Explorer 1, Internet Explorer 2, Netscape Navigator 2.0, OmniWeb, UdiWWW,Grail
1996	Arachne 1.0, Internet Explorer 3.0, Netscape Navigator 3.0, Opera 2.0, PowerBrowser 1.5,[4] Cyberdog, Amaya 0.9,[5] AWeb, Voyager
1997	Internet Explorer 4.0, Netscape Navigator 4.0, Netscape Communicator 4.0, Opera 3.0,[6] Amaya 1.0[5]
1998	iCab, Mozilla
1999	Amaya 2.0,[5] Mozilla M3, Internet Explorer 5.0
2000	Konqueror, Netscape 6, Opera 4,[7] Opera 5,[8] K-Meleon 0.2, Amaya 3.0,[5] Amaya 4.0[5]
2001	Internet Explorer 6, Galeon 1.0, Opera 6,[9] Amaya 5.0[5]
2002	Netscape 7, Mozilla 1.0, Phoenix 0.1, Links 2.0, Amaya 6.0,[5] Amaya 7.0[5]
2003	Opera 7,[10] Apple Safari 1.0, Epiphany 1.0, Amaya 8.0[5]
2004	Firefox 1.0, Netscape Browser, OmniWeb 5.0
2005	Opera 8,[11] Apple Safari 2.0, Netscape Browser 8.0, Epiphany 1.8, Amaya 9.0,[5] AOL Explorer 1.0, Maxthon 1.0,Shiira 1.0
2006	Mozilla Firefox 2.0, Internet Explorer 7, Opera 9,[12], SeaMonkey 1.0, K-Meleon 1.0, Galeon 2.0, Camino 1.0, Avant11, iCab 3
2007	Apple Safari 3.0, Maxthon 2.0, Netscape Navigator 9, NetSurf 1.0, Flock 1.0, Conkeror
2008	Google Chrome 1, Mozilla Firefox 3, Opera 9.5,[13], Apple Safari 3.1, Konqueror 4, Amaya10.0,[5] Flock 2, Amaya 11.0[5]
2009	Google Chrome 2–3, Mozilla Firefox 3.5, Internet Explorer 8, Opera 10,[14], Apple Safari 4, SeaMonkey 2, Camino 2,surf, Pale Moon 3.0[15]
2010	Google Chrome 4–8, Mozilla Firefox 3.6, Opera 10.50,[16], Opera 11, Apple Safari 5, K-Meleon 1.5.4,
2011	Google Chrome 9–16, Mozilla Firefox 4-9, Internet Explorer 9, Opera 11.50, Apple Safari 5.1, Maxthon 3.0, SeaMonkey 2.1–2.6

2012	Google Chrome 17–23, Mozilla Firefox 10–17, Internet Explorer 10, Opera 12, Apple Safari 6, Maxthon 4.0, SeaMonkey 2.7-2.14
2013	Google Chrome 24–31, Mozilla Firefox 18–26, Internet Explorer 11, Opera 15–18, Apple

	Safari 7, SeaMonkey 2.15-2.23
2014	Google Chrome 32–39, Mozilla Firefox 27–34, Opera 19–26, Apple Safari 8
2015	Google Chrome 40–47, Microsoft Edge, Mozilla Firefox 35–43, Opera 27–34, Vivaldi
2016	Google Chrome 48–55, Mozilla Firefox 44–50, Microsoft Edge 14, Opera 35–42, Apple Safari 10, SeaMonkey 2.24–2.30, Pale Moon 26.0.0[17], Pale Moon 27.0.0[18]
2017	Google Chrome 56–60, Microsoft Edge 15, Mozilla Firefox 51–55.0.2, Opera 43–45, OperaNeon

Uniform Resource Locators or URLs:

A Uniform Resource Locator, or URL is the address ofa document found on the WWW. Browser interprets the information in the URL in order to connect to the proper Internet server and to retrieve your desired document. Each time a click on a hyperlink in a WWW document instructs browser to find the URL that's embedded within the hyperlink.

The elements in a URL: **Protocol://server's address/filename**

Hypertext protocol: http://www.aucegypt.edu
File Transfer Protocol: ftp://ftp.dartmouth.edu
Telnet Protocol: telnet://pac.carl.org
News Protocol: news:alt.rock-n-roll.stones

Domains divide World Wide Web sites into categories based on the nature of their owner, and they form part of a site's address, or uniform resource locator (URL). Common top-level domains are:

.com—commercial enterprises	.mil—military site
org—organization site (non-profits, etc.)	int—organizations established by international treaty
.net—network	.biz—commercial and personal

.edu—educational site (universities, schools, etc.)	.info—commercial and personal
.gov—government organizations	.name—personal sites

Additional three-letter, four-letter, and longer top-level domains are frequently added. Each country linked to the Web has a two-letter top-level domain, for example .fr is France, .ie is Ireland.

Hypertext Transport Protocol(HTTP):

HTTP means Hyper Text Transfer Protocol. HTTP is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.

For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page. The other main standard that controls how the World Wide Web works is HTML, which covers how Web pages are formatted and displayed.

HTTP is called a stateless protocol because each command is executed independently, without any knowledge of the commands that came before it. This is the main reason that it is difficult to implement Web sites that react intelligently to user input.

HTTPS:

A similar abbreviation, HTTPS means Hyper Text Transfer Protocol Secure. Basically, it is the secure version of HTTP. Communications between the browser and website are encrypted by Transport Layer Security (TLS), or its predecessor, Secure Sockets Layer (SSL).

PHP:

INTRODUCTION:

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.
- PHP is a server-side scripting language that is embedded in HTML. PHP scripts are executed

on the server

- It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, Microsoft SQL Server , etc.)
- PHP is an open source software.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

Common uses of PHP:

PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.

The other uses of PHP are:

- PHP can handle forms, i.e. gather data from files, save data to a file, thru email you can send data, return data to the user.
- You add, delete, and modify elements within your database thru PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Characteristics of PHP:

- Efficiency
- Security

- Flexibility
- Familiarity

All PHP code must be included inside one of the three special markup tags are recognized by the PHP Parser.Simplicity

<?php PHP code goes here ?>

<?PHP code goes here ?>

<script language="php"> PHP code goes here </script>

Most common tag is the **<?php...?>**

SYNTAX:

<?php...?>

Short-open (SGML-style) tags Short or short-open tags look like this:

<?...?>

HTML script tags HTML script tags look like this:

<script language="PHP">...</script>

DECLARING VARIABLE:

The main way to store information in the middle of a PHP program is by using a variable. Here are the most important things to know about variables in PHP.

- A variable is used to store information.
- All variables in PHP are denoted with a leading dollar sign ($).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.

- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.

PHP variables are Perl-like.

Syntax: $var_name = value;

Eg: creating a variable containing a string, and a variable containing a number:

```
<?php
$txt="HelloWorld!";
$x=16;
?>
```

PHP is a Loosely Typed Language:

- In PHP, a variable does not need to be declared before adding a value to it.
- You do not have to tell PHP which data type the variable is
- PHP automatically converts the variable to the correct data type, depending on its value.

Naming Rules for Variables:

- A variable name must start with a letter or an underscore "_"
- A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z,0-9, and _)
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($my_string), or with capitalization/Camel notation ($myString)

PHP Variables Scope

In PHP, variables can be declared anywhere in the script. The scope of a variable is the part of the script where the variable can be referenced / used. PHP has three different variable scopes:

- **local**
- **global**
- **static**

Global and Local Scope

A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

Example

```
<?php
$x = 5; // global scope
function myTest() {
    // using x inside this function will generate an error
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();
echo "<p>Variable x outside function is: $x</p>";
?>
```

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

Example

```
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
myTest(); // using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

PHP The global Keyword

The global keyword is used to access a global variable from within a function. To do this, use the global keyword before the variables (inside the function):

Example

```
<?php
  $x = 5; $y = 10;
function myTest() {
    global $x, $y;
    $y = $x + $y;     }
myTest();
echo $y; //
?>
```

outputs 15

PHP also stores all global variables in an array called **$GLOBALS[index]**. The index holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly. The example above can be rewritten like this:

Example

```
<?php
$x = 5
$y = 10;
function myTest() {
   $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}
myTest();
echo $y; // outputs 15
?>
```

PHP The static Keyword

Normally, when a function is completed / executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

To do this, use the **static** keyword when you first declare the variable:

Example

```
<?php
function myTest() {
   static $x = 0;
   echo $x;
   $x=$x+5;
}
myTest();
myTest();
myTest();
?>
```

Output: 5 10 15

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

Note: The variable is still local to the function.

Variable Naming

Rules for naming a variable is-

- Variable names must begin with a letter or underscore character.
- A variable name can consist of numbers, letters, underscores but you cannot use characters like

+ , - , % , (,) . & , etc

There is no size limit for variables.

Data Types:

PHP has a total of **eight data types** which we use to construct our variables:

- **Integers:** are whole numbers, without a decimal point, like 4195.
- **Doubles:** are floating-point numbers, like 3.14159 or 49.1. **Scalar types**
- **Booleans:** have only two possible values either true or false.
- **Strings:** are sequences of characters, like 'PHP supports string operations.'

Arrays: are named and indexed collections of other values.

- **Objects:** are instances of programmer-defined classes. **Compound types**
- **NULL:** is a special type that only has one value: NULL.
- **Resources:** are special variables that hold references to resources external **Special types** to PHP (such as database connections)

The first four are simple types, and the next two (arrays and objects) are compound - the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

Integers:

Integers are **primitive data types.** They are **whole numbers**, without a decimal point, like 4195. They are the simplest type. They correspond to simple whole numbers, both positive andnegative {..., -2, -1, 0, 1, 2, ...}.

Integer can be in decimal (base 10), octal (base 8), and hexadecimal (base 16) format. Decimal format is the default, octal integers are specified with a leading 0, and hexadecimals have a leading 0x.

Ex: $v = 12345;
$var1 = -12345 + 12345;

notation.php

```
<?php
$var1 = 31; $var2 = 031; $var3 = 0x31;
```

Output:

```
31
25
```

```
echo "$var1\n$var2\n$var3";  ?>
```

The default notation is the **decimal**. The script shows these three numbers in decimal. In Java and C, if an integer value is bigger than the maximum value allowed, integer overflow happens. PHP works differently. In PHP, the integer becomes a float number. Floating point numbers havegreater boundaries.In 32bit system, an integer value size is four bytes. The maximum integer value is 2147483647.

boundary.php

```
<?php
$var = PHP_INT_MAX;
echovar_dump($var);
$var++;
echovar_dump($var);
 ?>
```

We assign a maximum integer value to the $var variable. We increase the variable by one. And we compare the contents.

Output:
int(2147483647)
float(2147483648)

As we have mentioned previously, internally, the number becomes a floating point value.

var_dump(): The PHP var_dump() function returns the data type and value.

DoublesorFloatingpoint numbers:

Floating point numbers represent real numbers in computing. Real numbers measure continuous quantities like weight, height or speed. Floating point numbers in PHP can be larger than integers and they can have a decimal point. The size of a float is platform dependent.

We can use various syntaxes to create floating point values.

```
<?php
 $a = 1.245;
$b = 1.2e3;
$c = 2E-10;
$d = 1264275425335735;
var_dump($a);var_dump($b); var_dump($c);var_dump($d);
 ?>
```

The **$d** variable is assigned a large number,so it is automatically converted to float type.

Output: float(1.245) float(1200) float(2.0E-10)
float(1264275425340000)

Boolean:

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true; $y = false;
```

Booleans are often used in conditional testing.

```
<?php
 $male = False;
 $r = rand(0, 1);
 $male = $r ? True: False;
if ($male) {
```

```
    echo "We will use name John\n";
    } else {

    echo "We will use name Victoria\n";
    }
?>
```

Strings:

String is a data type representing textual data in computer programs. Probably the single mostimportant data type in programming.

```
<?php
$a = "PHP ";
$b = 'PERL';
echo $a . $b; ?>
```

Output: PHP PERL

We can use single quotes and double quotes to create string literals.

The script outputs two strings to the console. The \n is a special sequence, a new line.

The escape-sequence replacements are −

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

The Concatenation Operator

There is only one string operator in PHP.

The concatenation operator **(.)**is used to put two string values together. To concatenate twostring variables together, use the concatenation operator:

```
<?php
$txt1="Hello Swetha!";
$txt2="What    a
nice  day!"; echo
$txt1   .   "   "   .
$txt2;
```

?>

O/P: Hello Swetha! What a nice day!

Search for a Specific Text within a String:

The **PHP strpos() function** searches for a specific text within a string. If a **match is found**, the function **returns the character position of the first match**. If **no match is found**, it will return

FALSE. The example below searches for the text "world" in the string "Hello world!":

Example:

```
<?php
echo strpos("Hello world!", "world");
?>
```

output: 6

Tip: The first character position in a string is 0 (not 1).

Replace Text within a String:

The PHP str_replace() function replaces some characters with some other characters in a string. The example below replaces the text "world" with "Dolly":

The strlen() function:

The **strlen()** function is used to return the length of a string. Let's find the length of a string:

Eg:

```
<?php
echo strlen("Hello world!");
?>
```

The output of the code above will be: 12

Array:

Array is a complex data type which handles a collection of elements. Each of the elements can be accessed by an index. An array stores multiple values in one single variable. In the following

example $cars is an array. The PHP var_dump() function returns the data type and value:

Example

```
<?php
$cars = array("Volvo","BMW","Toyota");
print_r($cars);
var_dump($cars);
?>
```

The **array keyword** is used to create a collection of elements. In our case we have names. The print_r function prints human readable information about a variable to the console.

O/P: Array ([0] => Volvo [1] => BMW [2] => Toyota)
array(3) { [0]=> string(5) "Volvo" [1]=> string(3) "BMW" [2]=> string(6) "Toyota" }

PHP Object

An object is a data type which stores data and information on how to process that data. In PHP, an object must be explicitly declared. First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

Example

```
<?php
class
Car {
function Car() {
$this->model = "VW";
}}
$herbie = new Car();                    // create an object
echo $herbie->model;                    // show object properties
?>
```

Output: VW

NULL:

NULL is a special data type that only has **one value: NULL**. To give a variable the NULLvalue, simply assign it like this −

Ex: $my_var = NULL;

The special constant NULL is capitalized by convention, but actually it is case

insensitive;you could just as well have typed −

$my_var = null;

A variable that has been assigned NULL has the following properties −

- It evaluates to FALSE in a Boolean context.
- It returns FALSE when tested with **IsSet()** function.

Example:

```
<?php
$x = "Hello world!";
$x = null; var_dump($x);
?>
```

PHP Resource:

The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP. A common example of using the resource data type is a database call. Resources are handlers to opened files, database connections or image canvas areas.

constant()function

As indicated by the name, this function will return the value of the constant. This is useful when you want to retrieve value of a constant, but you do not know its name, i.e. It is stored in a variable or returned by a function.constant()

Example:

```
<?php define("MINSIZE",50);
echo MINSIZE;
echo constant("MINSIZE");// same thing as the previous line
?>
```

Output: 50 50

Operators:

Arithmetic Operators	Assignment Operators
Increment/Decrement operators	Conditional (or ternary) Operators
Comparison Operators	String Operators
Logical (or Relational) Operators	Array Operators

example for expression 4 + 5 is equal to 9. **Here 4 and 5 are called operands and + is called operator.** PHP language supports following type of operators.

Arithmetic Operators:

There are following arithmetic operators supported by PHP language: Assume variable **A holds 10** And variable **B holds 20** then

Operators	Description	Example
+	Adds two operands	$A + $B will give 30
-	Subtracts second operand from the first	$A - $B will give -10
*	Multiply both operands	$A *$B will give 200
/	Divide numerator by denumerator	$B / $A will give 2
%	Modulus Operator and remainder of after an integer division	$B % $A will give 0
**	Exponentiation ($x to the $y'th power)	$A ** $B

Increment/Decrement operators:

Operator	Description	Example
++	Increment operator, increases integer value by one	$A++ - 11 / ++$A
--	Decrement operator, decreases integer value by one	$A-- will give 9 / --$A

Comparison Operators:

There are following comparison operators supported by PHP language Assume variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
==	Checks if the value of two operands are equal or not	($A==$B) is not true.
===	Identical(Returns true if $A is equal to $B, and they are of the same type)	$A === $B
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	($A != $B) is true.
<>	Returns true if $x is not equal to $y	$A <> $B
!==	Not identical (Returns true if $A is not equal to $B, or they are not of the same type)	$A !== $B
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	($A >$B) is not true.
<	Checks if the value of left operand is less (A < B) is true. Than the value of right operand, if yes then condition becomes true.	
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then returns true.	($A >= $B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition	($A <= $B) is true.

Logical Operators:

There are following logical operators supported by PHP languageAssume variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
and (or) &&	Called Logical AND operator. If both the operands are true then then condition becomes true.	($A and$B) is true. ($A &&$B) is true.
or (or) ||	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	($A or$B) is true. ($A || $B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!($A &&$B) is false.

Assignment Operators:

There are following assignment operators supported by PHP language:

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	$C = $A + $B
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	$C += $A is equivalent to $C = $C + $A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	$C -= $A is equivalent to $C = $C - $A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	$C *= $A is equivalent to $C = $C * $A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	$C /= $A is equivalent to $C = $C / $A

%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left	$C %= $A is equivalent to

Conditional Operator

There is one more operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation.

The conditional operator has this syntax:

Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

String Operators:

PHP has two operators that are specially designed for strings.

Operator	Description	Example
.	Concatenation	$txt1 . $txt2 (Concatenation of $txt1 and $txt2)
.=	Concatenation assignment	$txt1 .= $txt2 (Appends $txt2 to $txt1)

Array Operators:

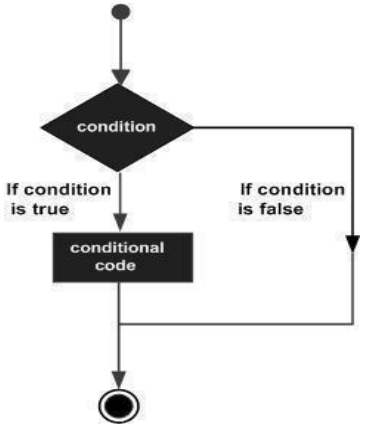
The PHP array operators are used to compare arrays.

Operator	Description	Example
+	Union	$x + $y (Union of $x and $y)
==	Equality	$x == $y (Returns true if $x and $y have the same key/value pairs)
===	Identity	$x === $y (Returns true if $x and $y have the same key/value pairs in the same order and of the same types)
!= or <>	Inequality	$x != $y or $x <> $y Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y (Returns true if $x is not identical to $y)

Control Structures:

The if, elseif ...else and switch statements are used to take decision based on the different condition. You can use conditional statements in your code to make your decisions. PHP supports following three decision making statements –

- **if...else statement** − use this statement if you want to execute a set of code when a condition is true and another if the condition is not true
- **elseif statement** − is used with the if...else statement to execute a set of code if **one** of the several condition is true
- **switch statement** − is used if you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.



TheIf...ElseStatement

Syntax	**EX:** <html> <body> <?php $d=date("D"); if($d=="Fri") echo"Have a nice weekend!"; else echo"Have a nice day!"; ?> </body></html>
if (*condition*) *code to be executed if condition is true;* else *code to be executed if condition is false;*	
Example The following example will output "Have a nice weekend!" if the current day is Friday, Otherwise, it will output "Have a nice day!":	

If you want to execute some code if a condition is true and another code if a condition is false, use the if-else statement.

The elseif Statement:

If you want to execute some code if one of the several conditions is true use the elseif statemen

Syntax	
if (*condition*) *code to be executed if condition is true;* elseif (*condition*) *code to be executed if condition is true;* else *code to be executed if condition is false;*	**EX:** <html> <body> <?php $d=date("D"); if($d=="Fri") echo"Have a nice weekend!"; elseif($d=="Sun") echo"Have a nice Sunday!"; else echo"Have a nice day!"; ?> </body> </html>
Example The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise, it will output "Have a nice day!"	

The Switch Statement

If you want to select one of many blocks of code to be executed, use the Switch statement.

The switch statement is used to avoid long blocks of if..elseif..else code.

Syntax	Example
switch (*expression*) { case *label1:* *code to be executed if expression = label1;* break; case *label2:* *code to be executed if expression = label2;* break; default: *code to be executed* *if expression is different* *from both label1 and label2;* }	The *switch* statement works in an unusual way. First it evaluates given expression then seeks a lable to match the resulting value. If a matching value is found then the code associated with the matching label will be executed or if none of the lable matches then statement will execute any specified default code.

LoopTypes:

Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.

- **for** − loops through a block of code a specified number of times.
- **while** − loops through a block of code if and as long as a specified condition is true.
- **do...while** − loops through a block of code once, and then repeats the loop as long as a special condition is true.
- **foreach** − loops through a block of code for each element in an array.

We will discuss about **continue** and **break** keywords used to control the loops execution.

Theforloopstatement

The for statement is used when you know how many times you want to execute a statement or a block of statements.

Syntax

```
for (initialization; condition; increment)
{
code to be executed;
```

The initializer is used to set the start value for the counter of the number of loop iterations. A variable may be declared here for this purpose and it is traditional to name it $i.

Example

The following example makes five iterations and changes the assigned value of two variables on each pass

`of the loop −

```
<html><body>
<?php
        $a =0;
        $b =0;
    for( $i=0; $i<5; $i++)
 $a +=10;
 $b +=5;
}
echo("At the end of the loop a=$a and b=$b");
?></body></html>
```

This will produce the following result –

At the end of the loop a=50 and b=2

The while loop statement:

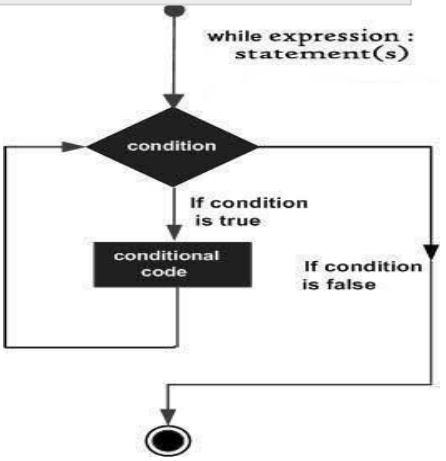
The while statement will execute a block of code if and as long as a test expression is true. If the test expression is true then the code block will be executed. After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.

Syntax

```
while (condition)
{
code to be executed;
```

Example

This example decrements a variable value on each iteration of the loop and the counter increments until it reaches 10 when the evaluation is false and the loop ends.




```
<html>
<body>
<?php
        $i =0;
        $num=50;
while( $i <10)
{
          $num--;
          $i++;
}
echo("Loop stopped at i = $i and num = $num");
?></body></html>
```

This will produce the following result –

```
Loop stopped at i = 10 and num = 40
```

The do...whileloop statement

The do...while statement will execute a block of code at least once. It then will repeat the loop as long as a condition is true.

Syntax

```
do
{
code to be executed;
```

Example

The following example will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than 10 −

```
<html>
<body>
<?php
      $i =0; $num=0;
do{
        $i++;
}while( $i <10);
echo("Loop stopped at i = $i");
?>
</body>
</html>
```

O/P: Loop stopped at i = 10

The for each loop statement

The foreach statement is used to **loop through arrays**. For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed.

Syntax

```
foreach (array as value)
{
code to be executed;




}
```

Example

Try out beside example to list out the values of an array.

```
<html>
<body>
<?php
        $array = array(1,2,3,4,5);
foreach( $array as $value )
{
          echo "Value is $value <br />";
}
?>
</body></html>
```

This will produce the following result −

```
Value is 1
Value is 2
Value is 3
Value is 4



Value is 5
```

The break statement

The PHP **break** keyword is used to terminate the execution of a loop prematurely. The **break** statement is situated inside the statement block. If gives you full control and whenever you want to exit from the loop you can come out. After coming out of a loop immediate statement to the loop will be executed.

Example

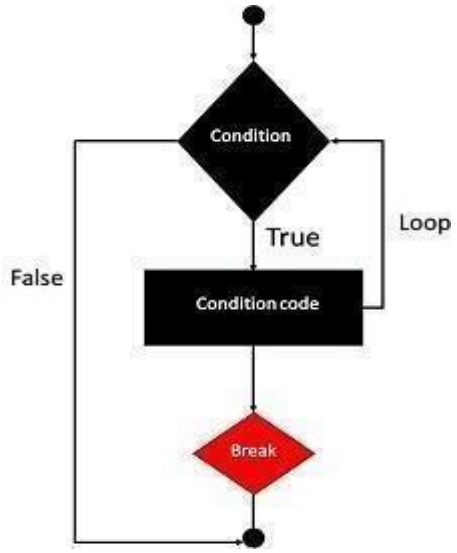
In the following example condition test becomes true when the counter value reaches 3 and loop terminates.

```
<?php
       $i =0

    while( $i <10){
          $i++;
if( $i ==3)break;}

echo("Loop stopped at i = $i");
```

?>**O/P:** Loop stopped at i=3



Thecontinue statement

The PHP **continue** keyword is used to halt the current iteration of a loop but it does not terminate the loop. Just like the **break** statement the **continue** statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering **continue** statement, rest of the loop code is skipped and next pass starts.

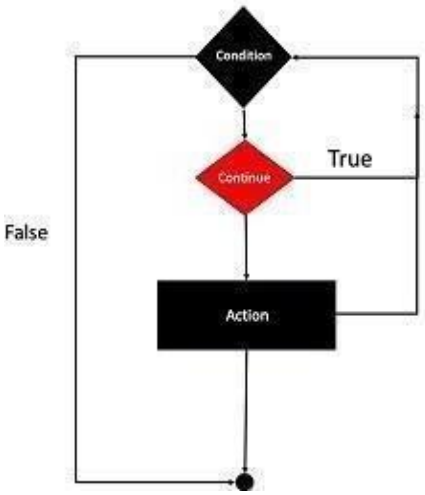
Example

In the following example loop prints the value of array but for which condition becomes true it just skip the code and next value is printed.

```
<html>
<body>
<?php
      $nos=array(1,2,3,4,5);
foreach( $nosas $value )
{
if( $value ==3)
            continue;
echo"Value is $value <br />";
}
?>
</body>
</html>
```

This will produce the following result −

```
Value is 1
Value is 2
Value is 4
```



Functions:

PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value. You already have seen many functions like **fopen()** and **fread()** etc. They are built-in functions but PHP gives you option to create your own functions as well.

There are two parts which should be clear to you –

- **Creating a PHP Function**
- **Calling a PHP Function**

In fact you hardly need to create your own PHP function because there are already more than 1000 of built-in library functions created for different area and you just need to call them according to your requirement.

Creating PHP Function

It's very easy to create your own PHP function. Suppose you want to create a PHP function which will simply write a simple message on your browser when you will call it. Following example creates a function called writeMessage() and then calls it just after creating it.

```
<html><head>
<title>Writing PHP Function</title>
</head>
<body>
<?php
/* Defining a PHP Function */
 function writeMessage()
{
echo"Have a nice time Swetha!";
}/* Calling a PHP Function */
writeMessage();
```

Output: Have a nice time Swetha!

PHP Functions with Parameters:

PHP gives you option to pass your parameters inside a function. You can pass as many as parameters you're like. These parameters work like variables inside your function. Following example takes two integer parameters and add them together and then print them.

```
<?php
function Number($firstname,$lastname)
{
echo "Employee's full name is $firstname $lastname"."<br/>";
}
Number("Alex","Anderson");
Number("John","Walker");
Number("David","Clark");
?>
```

output

Employee's full name is Alex Anderson
Employee's full name is John Walker
Employee's full name is David Clark

Passing Arguments by Reference:

It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value. Any changes made to an argument in these cases will change the value of the original variable. You can pass an argument by reference by adding an ampersand to the variable name in either the function call or the function definition.

```
<?php

function calculate(&$a1){

$a1++;

}

$a1=5;

echo "This is the value of a1 variable before the function call :: ";

echo $a1;

echo "<br>";

echo "This is the value of a1 variable after the function call :: ";

calculate($a1);

echo $a1;

?>
```

Output:

This is the value of a1 variable before the function call::5

This is the value of a1 variable after the function call::6

PHP Functions returning value:

A function can return a value using the **return** statement in conjunction with a value or object. return stops the execution of the function and sends the value back to the calling code. You can return more than one value from a function using **return array(1,2,3,4)**.

```
<?php
function Multiplication($x, $y)
{
$z = $x * $y;
return $z;
}
echo "5 * 10 = " . Multiplication(5, 10) . "<br>";
echo "7 * 13 = " . Multiplication(7, 13) . "<br>";
echo "2 * 4 = " . Multiplication(2, 4);
?>
```

Output
50
91
8

Setting Default Values for Function Parameters

You can set a parameter to have a default value if the function's caller doesn't pass it. Following function prints NULL in case use does not pass any value to this function.

```
php
function greet($name = 'Guest')
echo "Hello, $name!";
```

DynamicFunctionCalls:

It is possible to assign function names as strings to variables and then treat these variables exactly as you would the function name itself.

```
<html>
<head>
<title>Dynamic Function
Calls</title></head>
<body>
<?php
functionsayHello()
{
        echo "Hello<br />";
}
      $function_holder="sayHello";
      $function_holder();
?></body></html>
```

Output: Hello

```
<html>
<head>
<title>Dynamic Function
Calls</title></head>
<body>
<?php
function add($x,$y)
{
echo"addition=" . ($x+$y);
}
      $function_holder="add";
      $function_holder(20,30);
?></body></html>
```

Output:addition=50

PHP Default Argument Value:

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

Example

```
<?php
function setHeight($minheight = 50) {
   echo "The height is : $minheight \t";
}
setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

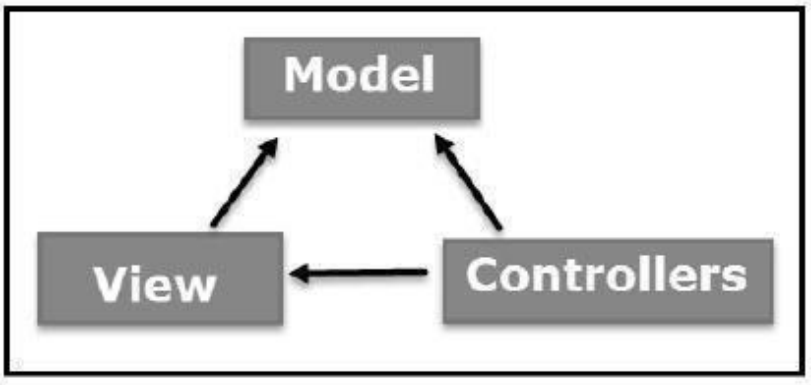
O/P: 350 50 135 80

MVC Framework:

The **Model-View-Controller (MVC)** is an architectural pattern that separates an application into three main logical components: the **model**, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most

frequently used industry-standard web development framework to create scalable and extensible projects

MVC Components:



Following are the components of MVC −

Model:

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

View:

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

Controller:

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component andinteract with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

Types of PHP MVC framework:

Selecting the best PHP framework is a challenge. You don't have to write your own framework to benefit from the advantages of MVC. You should only attempt to create your own MVC related application design for understanding how MVC frameworks work. Once you are comfortable with the way MVC frameworks work, you should move on to the mature and already tested frameworks.

The table below briefly describes some of the popular php frameworks and the features that each framework offers.

Framework	Description
CodeIgniter	It is one of the most popular PHP MVC frameworks. It's lightweight and has a short lrich set of libraries that help build websites and applications rapidly. Users with limite programming can also use it. CodeIgniter powered applications include
Kohana	It's a Hierarchical Model View Controller HMVC secure and lightweight framework. components for developing applications rapidly. Companies that use Kohana include;
CakePHP	It is modeled after Ruby on rails. It's known for concepts such as software design patt configuration, ActiveRecord etc. CakePHP powered applications include;

It is a powerful framework that is;

- Secure, reliable, fast, and scalable
- Supports Web 2.0 and creation of web services.

Zend

It features APIs from vendors like Amazon, Google, Flickr, Yahoo etc. It's ideal for deapplications. Zend powered applications include;

- Pimcore CMS,
- DotKernel.

Companies using the Zend framework include;

- BBC
- Cisco
- Webex
- Offers.com

UNIT – II

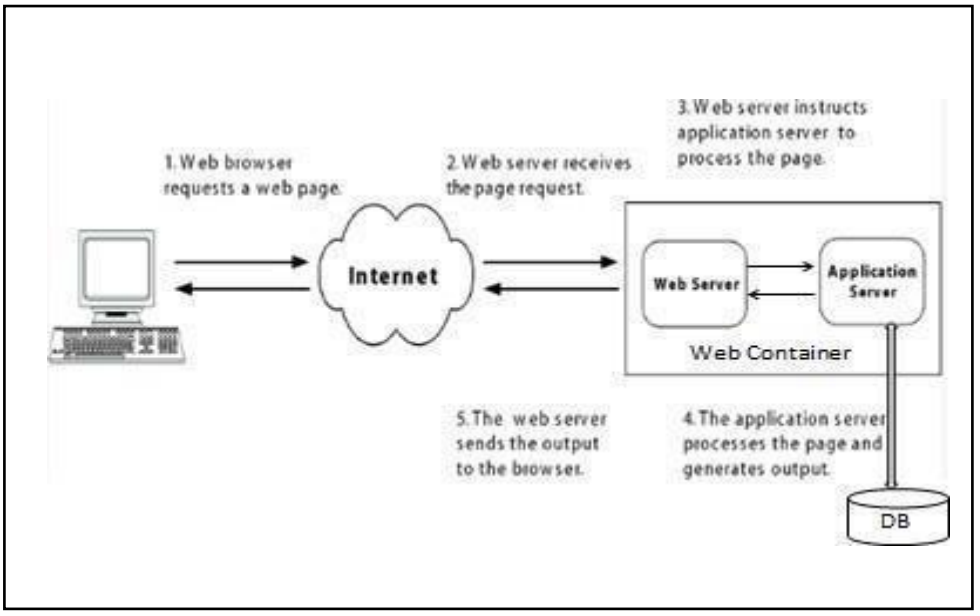
Servlets: Introduction to Servlets, Benefits of Servlets, use as controller in MVC, servlet lifecycle, basic HTTP, Reading Servlet parameters, Servlet session, Cookies, Servlets API

JDBC: JDBC Architecture, JDBC API, Connecting to a Database Using JDBC

INTRODUCTION TO SERVLETS:

Servlets:

- Servlets are server side programs that run on a Web or Application server and act as a middle layer between a requests coming from a Web browser and databases or applications on the server.
- Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.
- Servlets don't fork new process for each request, instead a new thread is created.
- Servlets are loaded and ready for each request.
- The same servlet can handle many requests simultaneously.



Benefits of Servlets:

1. **Better performance:** because it creates a thread for each request, not process.
2. **Portability:** because it uses Java language.
3. **Robust:** JVM manages Servlets, so we don't need to worry about the memory leak, garbagecollection, etc.

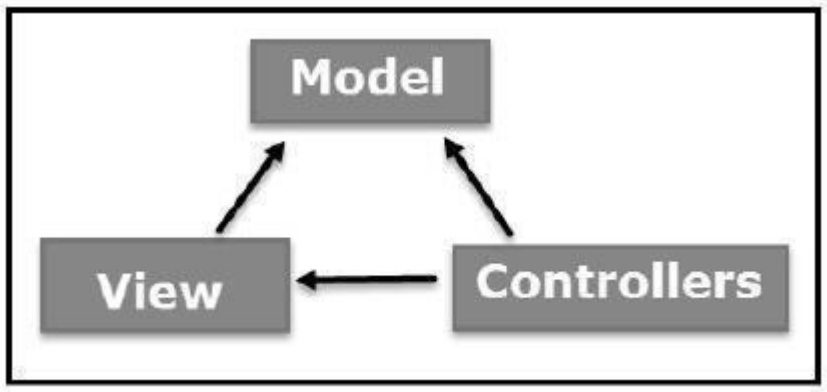
4. **Secure:** because it uses java language

Use as Controller in MVC:

The Model-View-C.ontroller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry- standard web development framework to create scalable and extensible projects

MVC Components:

Following are the components of MVC:



Model:

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

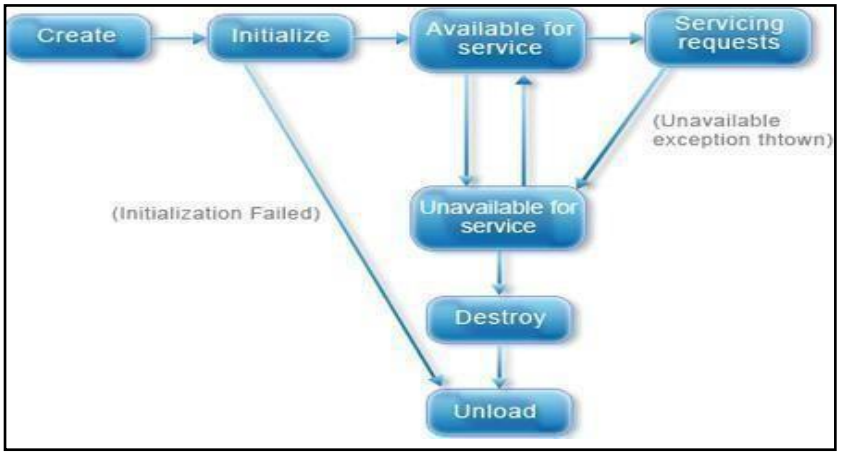
View:

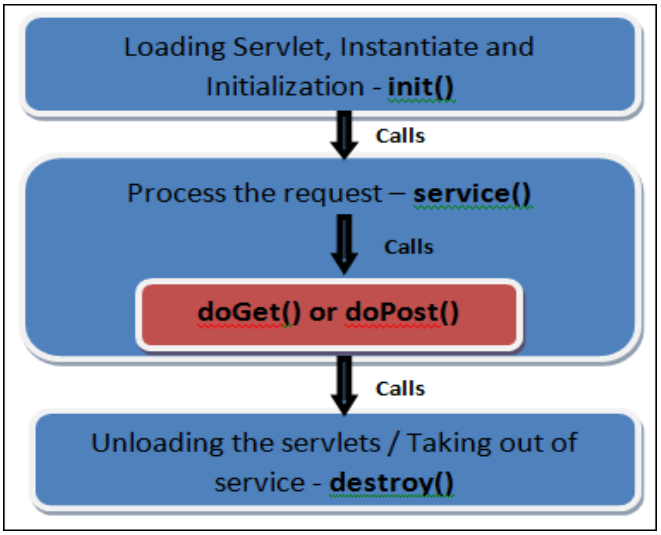
The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

Controller:

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

Life Cycle of Servlet:





Steps:

The sequence in which the Web container calls the life cycle methods of a servlet is:

1. The Web container loads the servlet class and creates one or more instances of the servlet class.
2. The Web container invokes init() method of the servlet instance during initialization of the servlet. The init() method is invoked only once in the servlet life cycle.

3. The Web container invokes the service() method to allow a servlet to process a client request.
4. The service() method processes the request and returns the response back to the Web container.
5. The Web container loads the servlet class and creates one or more instances of the servlet class.
6. The Web container invokes init() method of the servlet instance during initialization of the servlet. The init() method is invoked only once in the servlet life cycle.
7. The Web container invokes the service() method to allow a servlet to process a client request.
8. The service() method processes the request and returns the response back to the Web container.
9. The Web container loads the servlet class and creates one or more instances of the servlet class.
10. The Web container invokes init() method of the servlet instance during initialization of the servlet. The init() method is invoked only once in the servlet life cycle.
11. The Web container invokes the service() method to allow a servlet to process a client request.
12. The service() method processes the request and returns the response back to the Web container.
13. The Web container loads the servlet class and creates one or more instances of the servlet class.
14. The Web container invokes init() method of the servlet instance during initialization of the servlet. The init() method is invoked only once in the servlet life cycle.
15. The Web container invokes the service() method to allow a servlet to process a client request.
16. The service() method processes the request and returns the response back to the Web container.
17. The servlet then waits to receive and process subsequent requests as explained in steps 3 and 4.
18. The Web container calls the destroy() method before removing the servlet instance from the service. The destroy() method is also invoked only once in a servlet life cycle.

The init() method :

- The init method is designed to be called only once. It is called when the servlet is first created, and not called again for each user request.
- The servlet is normally created when a user first invokes a URL corresponding to the servlet.
- When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate.
- The init() method simply creates or loads some data that will be used throughout the life of the servlet.

```
publicvoidinit()throwsServletException{
// Initialization code...
}
```

The service()method:

- The service() method is the main method to perform the actual task.
- The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.
- Each time the server receives a request for a servlet, the server spawns a new thread and calls service.
- The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

```
publicvoid service(ServletRequest request,
ServletResponse response)
throwsServletException,IOException{
}
```

The doGet()Method

- The doGet() method processes client request, which is sent by the client, using the HTTP GET method.
- To handle client requests that are received using GET method, we need to override the doGet() method in the servlet class.
- In the doGet() method, we can retrieve the client information of the HttpServletRequest object. We can use the HttpServletResponse object to send the response back to the client.

```
publicvoiddoGet(HttpServletRequest request,
```

```
HttpServletResponse response)
throwsServletException,IOException{
// Servlet code
}
```

The doPost()Method:

- The doPost() method handles requests in a servlet, which is sent by the client, using the HTTP POST method.
- For example, if a client is entering registration data in an HTML form, the data can be sent using the POST method.

- To handle requests in a servlet that is sent using the POST method, we need to override the doPost() method. In the doPost() method, we can process the request and send the response back to the client.

```
publicvoiddoPost(HttpServletRequest request,
HttpServletResponse response)
throwsServletException,IOException{
// Servlet code
}
```

The destroy() method:

- The destroy() method is called only once at the end of the life cycle of a servlet.
- This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.
- After the destroy() method is called, the servlet object is marked for garbage collection.

```
publicvoid destroy()
{
  // Finalization code...
  }
```

```
import                    java.io.*;
importjavax.servlet.*;
importjavax.servlet.http.*;


public class HelloWorld extends HttpServlet {
private String message;
public void init() throws ServletException    {
      // Do required initialization
message = "Hello KALPANA";
  }
public void doGet(HttpServletRequestrequest,HttpServletResponse response)
throwsServletException, IOException {
      //    Set    response    content    type
response.setContentType("text/html");
      // Actual logic goes here.
```

Basic HTTP:

Reading Servlet Parameters (or) Handling HTTPRequest and HTTPResponse

- The parameters are the way in which a client or user can send information to the Http Server.

- The **HttpServletResponse** Interface provides functionality for sending response to client.
- The browser uses two methods to pass this information to web server. These methods are GET Method and POST Method.

GET method:

- The GET method sends the encoded user information appended to the page request.
- The page and the encoded information are separated by the ? character as follows:

http://www.test.com/hello?key1=value1&key2=value2

- The GET method is the defualt method to pass information from browser to web server.
- Never use the GET method if you have password or other sensitive information to pass to the server.
- The GET method has size limtation: only 1024 characters can be in a request string.
- This information is passed using QUERY_STRING header and will be accessible through QUERY_STRING environment variable.

- Servlet handles this type of requests using **doGet()** method.

POST method:

- A generally more reliable method of passing information to a backend program is the POST method.
- This message comes to the backend program in the form of the standard input which you can parse and use for your processing.
- Servlet handles this type of requests using **doPost()** method.

Session:

- HttpSession object is used to store entire session with a specific client.
- We can store, retrieve and remove attribute from HttpSession object.
- Any servlet can have access to HttpSession object throughout the getSession() method of the HttpServletRequest object.

HTTP Servlet

- HttpServlet is extends from GenericServlet and does not override init, destroy and other methods.
- It implements the service () method which is abstract method in GenericServlet.
- A subclass of HttpServlet must override at least one method, usually one of these:
    - doGet(), if the servlet supports HTTP GET requests
    - doPost(), for HTTP POST requests
    - doPut(), for HTTP PUT requests
    - doDelete(), for HTTP DELETE requests
    - Init() and destroy(), to manage resources that are held for the life of the servlet
    - getServletInfo(), which the servlet uses to provide information about itself

Cookie:

A **cookie** is a small piece of information that is persisted between the multiple client requests.

- **javax.servlet.http.Cookie** class provides the functionality of using cookies. It provides alot of useful methods for cookies.
- **public void addCookie(Cookie ck):** method of HttpServletResponse interface is used to add cookie in response object.
- **public Cookie[] getCookies():** method of HttpServletRequest interface is used to return all the cookies from the browser.

Servlet API:

Servlet API consists of two important packages that encapsulates all the important classes and interface, namely :

1. **javax.servlet**
2. **javax.servlet.http**

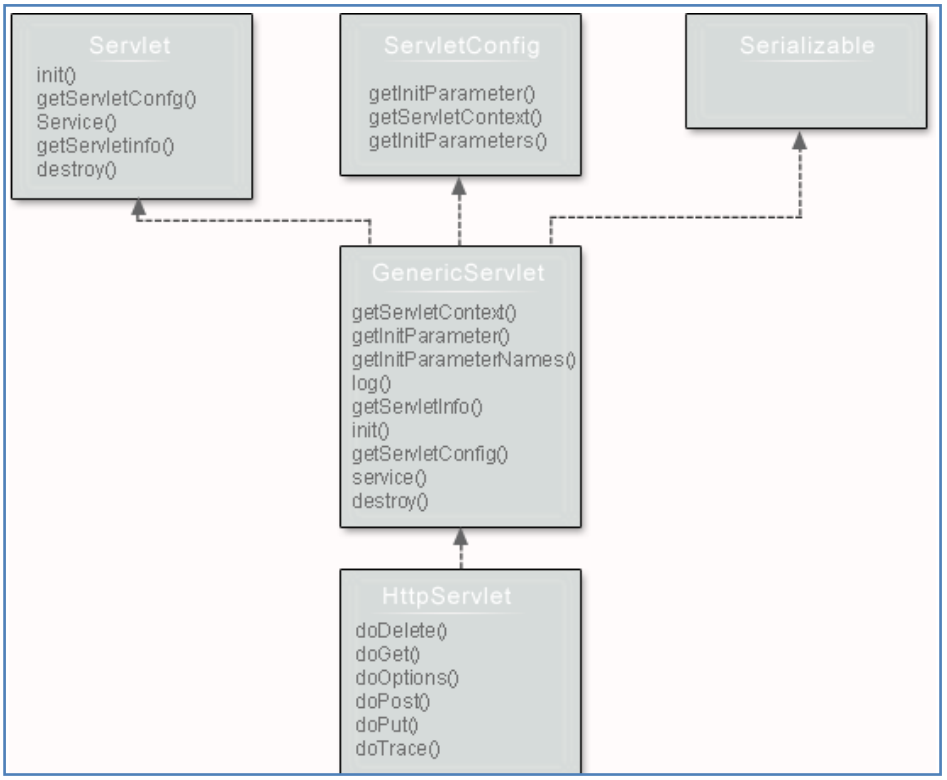
1. javax.servlet

Interfaces

1. Servlet – Declares life cycle methods for a servlet.
2. ServletConfig – To get initialization parameters
3. ServletContext- To log events and access information
4. ServletRequest- To read data from a client request
5. ServletResponse – To write data from client response

Classes

1. GenericServlet – Implements Servlet and ServletConfig
2. ServletInputStream – Provides an input stream for reading client requests.
3. ServletOutputStream - Provides an output stream for writing responses to a client.
4. ServletException – Indicates servlet error occurred.
5. UnavailableException - Indicates servlet is unavailable



Servlet Interface

- Servlet interface provides common behaviour to all the servlets.
- Servlet interface needs to be implemented for creating any servlet (either directly or indirectly).

- It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.

Method	Description
public void init(ServletConfigconfig)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequestrequest,ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfiggetServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

```
import java.io.*;
import javax.servlet.*;
public class First implements Servlet{
    ServletConfig config=null;
    public void init(ServletConfig config){
        this.config=config;
        System.out.println("servlet is initialized");
    }
    public void service(ServletRequest req,ServletResponse res)
        throws IOException,ServletException{
        res.setContentType("text/html");
        PrintWriter out=res.getWriter();
        out.print("<html><body>");
        out.print("<b>hello swetha</b>");
        out.print("</body></html>");
    }
    public void destroy(){
        System.out.println("servlet is destroyed");
    }
    public ServletConfig getServletConfig(){
        return config;
    }
    public String getServletInfo(){                    o/p hello swetha
        return "copyright 2007-1010";
    }
```

}

Servlet Config interface

- When the **Web Container** initializes a servlet, it creates a **ServletConfig** object for the servlet.
- ServletConfig object is used to pass information to a servlet during initialization by getting configuration information from **web.xml**(Deployment Descriptor).

Methods

- getInitParameter(String name): returns a String value initialized parameter
- getInitParameterNames(): returns the names of the servlet's initialization parameters as an Enumeration of String objects
- getServletContext(): returns a reference to the ServletContext
- getServletName(): returns the name of the servlet instance

ServletContext Interface

- For every **Web application** a **ServletContext** object is created by the web container.
- ServletContext object is used to get configuration information from **Deployment Descriptor**(web.xml) which will be available to any servlet.

Methods:

- getAttribute(String name) - returns the container attribute with the given name
- getInitParameter(String name) - returns parameter value for the specified parameter name
- getInitParameterNames() - returns the names of the context's initialization parameters as an Enumeration of String objects
- setAttribute(String name,Objectobj) - set an object with the given attribute name in the application scope
- removeAttribute(String name) - removes the attribute with the specified name from the application context

Servlet Request Interface

- True job of a Servlet is to handle client request.
- Servlet API provides two important interfaces **javax.servlet.ServletRequest** to encapsulate client request.
- Implementation of these interfaces provides important information about client request to a servlet.

Methods

- getAttribute(String name), removeAttribute(String name), setAttribute(String name, Object o), getAttributeName() – used to store and retrieve an attribute from request.
- getParameter(String name) - returns value of parameter by name
- getParameterNames() - returns an enumeration of all parameter names
- getParameterValues(String name) - returns an array of String objects containing all of the values the given request parameter has, or null if the parameter does not exist

Servlet Response Interface

- Servlet API provides ServletResponseto assist in sending response to client.

Methods

- getWriter()- returns a PrintWriter object that can send character text to the client.
- setContentType(String type)- sets the content type of the response being sent to the client before sending the respond.

GenericServlet class

- GenericServlet class implements **Servlet**, **ServletConfig** and **Serializable** interfaces.
- It provides the implementation of all the methods of these interfaces except the service method.
- GenericServlet class can handle any type of request so it is protocol-independent.
- You may create a generic servlet by inheriting the GenericServlet class and providing the implementation of the service method.

Methods

- **public void init(ServletConfigconfig)** is used to initialize the servlet.
- **public abstract void service(ServletRequest request, ServletResponse response)** provides service for the incoming request. It is invoked at each time when user requests for a servlet.
- **public void destroy()** is invoked only once throughout the life cycle and indicates that servlet is being destroyed.
- **publicServletConfiggetServletConfig()** returns the object of ServletConfig.
- **public String getServletInfo()** returns information about servlet such as writer, copyright, version etc.
- **public void init()** it is a convenient method for the servlet programmers, now there is noneed to call super.init(config)
- **publicServletContextgetServletContext()** returns the object of ServletContext.

- **public String getInitParameter(String name)** returns the parameter value for the given parameter name.
- **public Enumeration getInitParameterNames()** returns all the parameters defined in the
- **web.xml** file.
- **public String getServletName()** returns the name of the servlet object.
- **public void log(String msg)** writes the given message in the servlet log file.
- **public void log(String msg,Throwable t)** writes the explanatory message in the servlet log file and a stack trace.

ServletInputStream Class

- It provides stream to read binary data such as image etc. from the request object. It is an abstract class.

- The **getInputStream()** method of **ServletRequest** interface returns the instance of ServletInputStream class
- **intreadLine(byte[] b, int off, intlen)** it reads the input stream.

ServletOutputStream Class

- It provides a stream to write binary data into the response. It is an abstract class.
- The **getOutputStream()** method of **ServletResponse** interface returns the instance of ServletOutputStream class.
- ServletOutputStream class provides print() and println() methods that are overloaded.

ServletException and UnavailableException

- ServletException is a general exception that the servlet container will catch and log. The cause can be anything.
- The exception contains a root cause exception.
- Defines an exception that a servlet or filter throws to indicate that it is permanently or temporarily unavailable.
- When a servlet or filter is permanently unavailable, something is wrong with it, and it cannot handle requests until some action is taken. For example, a servlet might be configured incorrectly, or a filter's state may be corrupted.

2. javax.servlet.http

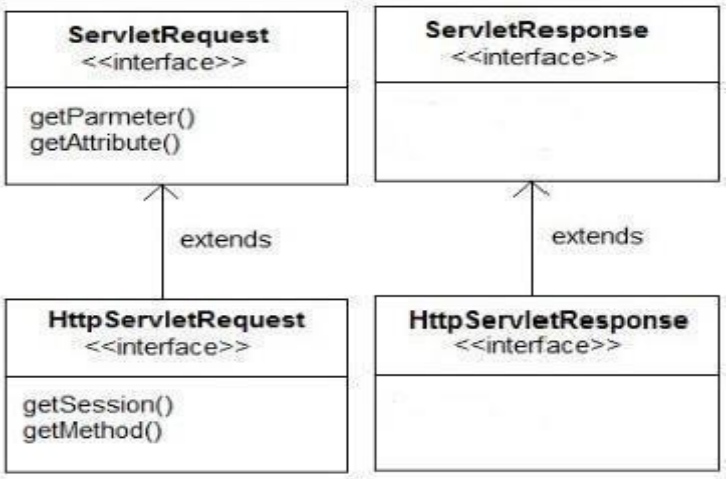
Interfaces

1. HttpServletRequest
2. HttpServletResponse
3. HttpSession

Classes

1. HttpServlet
2. Cookie

HTTPServletRequest and HTTPServletResponse



- HTTPServletRequest Extends the ServletRequest interface to provide request informationfor HTTP servlets.
- The servlet container creates an HttpServletRequest object and passes it as an argument to the servlet's service methods (doGet, doPost, etc).
- It Contains all the client's request information.
- The HttpServletRequest breaks a request down into parsed elements, such as request URI, query arguments and headers. Various get methods allow you to access different parts of the request.

1. **requestURI** – URL sent by browser

2. **Parameters** -

   The HttpServletRequest provides methods for accessing parameters of a request. The methods getParameter(), getParameterValues()and getParameterNames() are offered as

ways to access the arguments.

3. **Attributes –**

The request object defines a method called getAttribute(). The servlet interface

provides this as a way to include extra information about the request that is not covered by any of the other HttpServletRequestmethods.

4. **ServletInputStream –**

The ServletInputStream is an InputStream that allows your servlets to read all of the request's input following the headers.

- **HTTPServletResponse** Extends the ServletResponse interface and can perform these tasks

1. **Set Response Codes –**

The response code for a request is a numeric value that represents the status of the response. For example, 200 represents a successful response, 404 represents a file not found.

2. **Set Headers –**

Headers for the response can be set by calling setHeader, specifying the name and value of the header to be set.

3. **Send Redirects –**

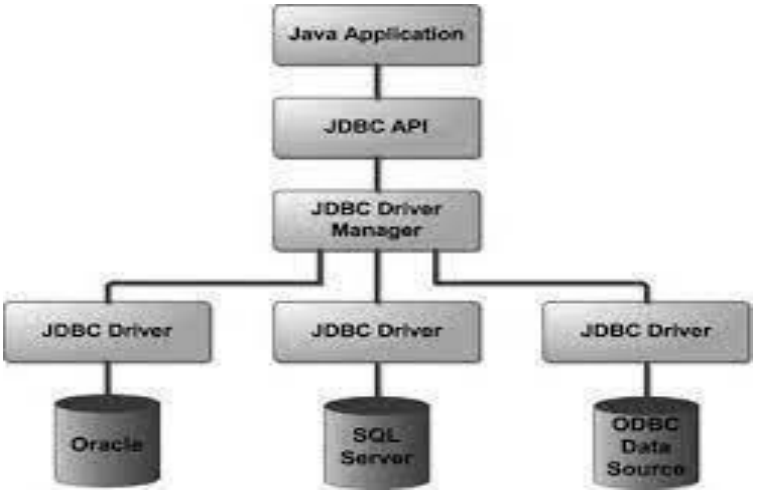
The sendRedirect method is used to issue a redirect to the browser, causing the browser to issue a request to the specified URL. The URL passed to sendRedirect must be an absolute URL—it must include protocol, machine, full path, and so on.

4. **Set ServletOutputStream –**

The ServletOutputStream is obtained by calling getOutputStream on the HttpServletResponse. It is a subclass of OutputStream that contains a number of convenient print and println methods.Data written to the ServletOutputStream goes straight back to the browser.

JDBC:

JDBC Architecture:



Description:

1. **Application:** It is a java applet or a servlet that communicates with a datasource.
2. **The JDBC API:** The JDBC API allows Java programs to execute SQLstatements and retrieve results. Some of the important classes and interfaces defined in JDBC API are as follows:
3. **DriverManager:** It plays an important role in the JDBC architecture. Ituses some database-specific drivers to effectively connect enterprise applications to databases.
4. **JDBC drivers:** To communicate with a data source through JDBC, you need a JDBC driver that intelligently communicates with the respective data source.

JDBC API:

The Java Database Connectivity (JDBC) API provides universal data access from the Java programming language. Using the JDBC API, you can access virtually any data source, from relational databases to spreadsheets and flat files. JDBC technology also provides a commonbase on which tools and alternate interfaces can be built.

The JDBC API is comprised of two packages:

- java.sql
- javax.sql

JDBC was designed to keep simple things simple. This means that the JDBC API makes everyday database tasks, like simple SELECT statements, very easy.

Import a package java.sql.* : This package provides you set of all classes that enables a network interface between the front end and back end database.

• DriverManager will create a Connection object.

• java.sql.Connection interface represents a connection with a specific database.

Methods of connection is close(),

creatStatement(),

prepareStatement(),

commit(),

close()

prepareCall()

• Statement interface used to interact with database via the execution of SQL statements. Methods of this interface are executeQuery(),

executeUpdate(),

execute()

getResultSet().

• A ResultSet is returned when you execute an SQL statement. It maintains a pointer to a row within the tablur results.

Mehods of this interface are next(),

getBoolean(),

getByte(),

getDouble(),

getString()

close()

getInt().

Connecting to a Database Using JDBC:

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

1. Register the Driver class
2. Create connection

3. Create statement
4. Execute queries
5. Close connection

1) Register the driver class

The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver class.

Syntax:

public static void forName(String className)throws ClassNotFoundException

Example to register the OracleDriver class:

Here, Java program is loading oracle driver to esteblish database connection.

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

2) Create the connection object:

The getConnection() method of DriverManager class is used to establish connection with the database.

Syntax of getConnection() method:

1) public static Connection getConnection(String url)throws SQLException
2) public static Connection getConnection(String url,String name,String password) throws SQLException

Example to establish connection with the Oracle database:

```
Connection con=DriverManager.getConnection(
    "jdbc:oracle:thin:@localhost:1521:xe","system","password");
```

3) Create the Statement object:

The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

Syntax of createStatement() method:

public Statement createStatement()throws SQLException

Example to create the statement object:

```
Statement stmt=con.createStatement();
```

4) Execute the query:

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

Syntax of executeQuery() method:

public ResultSet executeQuery(String sql)throws SQLException

Example to execute query:

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

while(rs.next()){

System.out.println(rs.getInt(1)+" "+rs.getString(2));
}
```

5) Close the connection object:

By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

Syntax of close() method:

public void close()throws SQLException

Example to close connection:

```
con.close();
```

UNIT – III

Java Server Pages: Generating Dynamic Content, Using Scripting Elements, Implicit JSP Objects. Conditional Processing – Displaying Values, setting attributes, Error Handling and Debugging

Generating Dynamic Content:

Dynamic contents means the contents that get changed based on the user inputs or states of external system or on some runtime conditions. JSP helps in handling such conditions.

There are various ways by which JSP handles the dynamic contents such as use of:

1. Directive elements 2. Java Beans 3. Scripting elements 4. Standard tag libraries 5. Standard and custom actions 6. Expression language.

1. Directive Elements:

Directive elements are used to specify the information about the page. Syntax: The directive names and attribute names are case sensitive.

Examples of some directives are: • Page • Include • Taglib • Attribute • Tag • Variable

i. Page directive:

This directive can only be used in JSP pages, not tag files. It defines page dependent attributes, such as scripting language, error page, buffer requirements.

ii. Include directive:

It includes a static file, merging its content with the including page before the combined results is converted to JSP page implementation class

iii. Taglib directive:

Declares a tag library, containing custom actions that are used in the page.

iv. Attribute directive:

This directive can only be used in tag files. It declares the attributes that tag file supports.

v. Tag directive:

This directive can only be used in tag files.

vi. Variable directive:

It is similar to variable declarations.

Using Scripting Elements:

In JSP, java code can be written inside the jsp page using the scriptlet tag. Let's see what are the scripting elements first.

JSP Scripting elements:

The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

1. scriptlet tag
2. expression tag
3. declaration tag

1. JSP scriptlet tag:

A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:

<% java source code %>

Example of JSP scriptlet tag

In this example, we are displaying a welcome message.

```
<html>
<body>
<% out.print("welcome to jsp"); %>
</body>
</html>
```

Example of JSP scriptlet tag that prints the user name:

In this example, we have created two files index.html and welcome.jsp. The index.html file gets the username from the user and the welcome.jsp file prints the username with the welcome message.

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

welcome.jsp

```
<html>
<body>
<%
String name=request.getParameter("uname");
```

```
out.print("welcome "+name);
%>
</form>

</body>
</html>
```

JSP expression tag:

The code placed within JSP expression tag is written to the output stream of the response. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

Syntax of JSP expression tag:

<%= statement %>

Example of JSP expression tag:

In this example of jsp expression tag, we are simply displaying a welcome message.

```
<html>
<body>
<%= "welcome to jsp" %>
</body>
</html>
```

JSP Declaration Tag:

The JSP declaration tag is used to declare fields and methods.

The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.

So it doesn't get memory at each request.

Syntax of JSP declaration tag:

The syntax of the declaration tag is as follows:

```
<%!  field or method declaration %>
```

Example of JSP declaration tag that declares field:

```
<html>
<body>
<%! int data=50; %>
<%= "Value of the variable is:"+data %>
</body>
</html>
```

Example of JSP declaration tag that declares method:

In this example of JSP declaration tag, we are defining the method which returns the cube of given number and calling this method from the jsp expression tag. But we can also use jsp scriptlet tag to call the declared method.

```
<html>
<body>
<%!

int cube(int n){
return n*n*n*;
}
%>
<%= "Cube of 3 is:"+cube(3) %>
</body>
</html>
```

Implicit JSP Objects:

These Objects are the Java objects that the JSP Container makes available to the developers in each page and the developer can call them directly without being explicitly declared. JSP Implicit Objects are also called pre-defined variables.

Following table lists out the nine Implicit Objects that JSP supports –

S.No.	Object & Description
1	**Request** This is the **HttpServletRequest** object associated with the request.
2	Response This is the HttpServletResponse object associated with the response to the client.
3	Out This is the PrintWriter object used to send output to the client.
4	Session This is the HttpSession object associated with the request.
5	Application This is the ServletContext object associated with the application context.
6	Config This is the ServletConfig object associated with the page.

7	pageContext This encapsulates use of server-specific features like higher performance JspWriters.
8	Page This is simply a synonym for this, and is used to call the methods defined by the translated servlet class.
9	Exception The Exception object allows the exception data to be accessed by designated JSP.

Conditional processing:

Sometimes to built the complex logic we require conditional statements. We can embed JAVA conditional statements in JSP. This task can be done by scriptlets. The syntax for scriptlets is:

<% any java code %>

Sample program for conditional processing:

```
<%@ page language=”java” ContentType=”text/html” %>
<html>
<head><title>Introduction to scriptlet</title></head>
<body>
<h1>
<% out.println(“JSP is very interesting”); %>
</h1>
</body>
</html>
```

Displaying Values Using An Expression To Set An Attribute:

The JSP expressions are used to insert java values directly into the output.

The syntax of using expression is as follows:

<%= Expression code %>

This expression gets evaluated at runtime and then the result will get displayed on the web browser.

Thus tags <%= and %> is used.

Sample program for using an expression:

Scripting.jsp file:

```
<%@ page import="java.util.Date"%>
<html>
<head><title>Scripting elements example</title></head>
<body>
<%! int count=0; %>
<% count++;
out.println("You are accessing this page for "+count+ "times");
out.println(new Date().toString());
%>
<%= count %>
</body>
</html>
```

Web.xml file:

```
<web-app>
</web-app>
```

Error Handling & Debugging:

While developing any application we may come across several errors. We may get some syntactical errors during development of JSP pages.

Errors are of two types:

1. Element syntax error
2. Expression language syntax error

1. Element syntax error:

Element syntax errors may encounter due to:

- Improperly terminate directive.
- Improperly terminate action
- Mistyped attribute
- Missing endquote in attribute values.

2. Expression language syntax error:

Expression language syntax error may occur due to:

- Missing both curly braces
- Missing end curly brace
- Misspelled property name
- Misspelled parameter name.

Sample program for error handling:

```
<%@ page language=”java” contentType=”text/html”>
<%@ page import=”java.util.*” %>
<html>
<body>Today’s date is: <%= new Date().toString() %>
</body>
</html>
```

Debugging JSP actions:

Sometimes after fixing all the syntactical errors, still the application does not work properly due to logical errors. Logical errors may not be highlighted by the tomcat explicitly. The runtime errors can be handled by using exceptions and gracefully the JSP application can be terminated.

Types:

- Logical error
- Dealing with runtime errors
- Catch exceptions

UNIT – IV

Java Script: Introduction to JavaScript, Declaring Variables, Functions, Event Handlers (onclick,onsubmit,etc…,) and Form Validation.

Spring Framework: Overview, Controllers, Handler Methods, and Developing Web Application Using Spring.

Introduction to JavaScript:

- JavaScript is the Programming Language for the Websites.
- The first scripting language was introduced by Brendan Eich in 1996
- JavaScript is an object-based scripting language which is lightweight and cross-platform.
- JavaScript can update and change both HTML and CSS.
- JavaScript can calculate, manipulate and validate data.

Features of JavaScript:

There are following features of JavaScript:

- All popular web browsers support JavaScript as they provide built-in execution environments.
- JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
- JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
- JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
- It is a light-weighted and interpreted language.
- It is a case-sensitive language.
- JavaScript is supportable in several operating systems including, Windows, macOS, etc.
- It provides good control to the users over the web browsers.

JavaScript – Syntax

JavaScript can be implemented using JavaScript statements that are placed within the <script>... </script> HTML tags in a web page.

You can place the <script> tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the <head> tags.

Example:

```
<html>
    <body>
          <h2>Welcome to JavaScript</h2>
       <script>
            document.write("Hello JavaScript by JavaScript");
       </script>
      </body>
  </html>
```

Declaring Variables:

A JavaScript variable is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

There are some rules while declaring a JavaScript variable (also known as identifiers).

Name must start with a letter (a to z or A to Z),underscore(_), or dollar($) sign.

After first letter we can use digits (0 to 9).

JavaScript variables are case sensitive, for example x and X are different variables.

```
<html>
<body>
<script>
   var x = 10;
   var y = 20;
   var z=x+y;
              document.write(z);
        </script>
     </body>
</html>
```

Variable Scope:

The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes.

Global Variables:

A global variable has global scope which means it can be defined anywhere in your JavaScript code.

Local Variables − A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

```
<script>
```

```
    function abc(){
    var x=10;//local variable
  }
</script>
```

- A JavaScript local variable is declared inside block or function. It is accessible within the function or block only.

  global variable:

  A **JavaScript global variable** is accessible from any function. A variable i.e. declared outside the function or declared with window object is known as global variable.

  For example:

```
<script>
var data=200;
//gloabal variable
function a(){
document.writeln("data from a function"data);
}
        function b(){
    document.writeln("data from b function "+data);
   }
a();//calling JavaScript function
b();
</script>
```

Functions:

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The code to be executed, by the function, is placed inside curly brackets: {}

```
    function name(parameter1, parameter2, parameter3)
     {
  // code to be executed     }
```

Example:

```
<html>
<body>
<script>
function msg()
 {
alert("hello! this is message");
}
</script>

<input type="button" onClick="msg()" value="call function"/>
</body>
</html>
```

We can call function by passing arguments.

```
<script>
    function getcube(number){
     alert(number*number*number);
    }
 </script>
<form>
    <input type="button" value="click"
      onClick="getcube(4)"/>
</form>
```

- We can call function that returns a value and use it in our program.

```
<script>
 function getInfo(){
  var str ="hello java! How r u?";
Return str;
 }
document.write(getInfo());
</script>
```

Event Handlers:

- Java script identifies an event and takes an action by executing some piece of code. The procedure of taking action is called event handling .

- the specific JavaScript that takes the action is called event handler.
- Event handlers may be interactive or non interactive . when an event take place ,java script identifies the event and execute its handler.

Java script events and their sources:

Event name	Occurs when	Applicable to
click	Mouse is clicked on an object	Button,checkbox,radio,link,reset Text,textarea, body
submitted	HTML form is submitted	Form
Select	Text is selected	Text,textarea
mouseOver	Mouse pointer is moved over an element	Link,text,button,textarea,radio
mouseOut	Mouse pointer is moved off an element	Link,text,button,textarea,radio
blur	Element loses focus	Checkbox,radio,select,text,textarea
focus	Element get focus	Select,text,textarea
change	Content of an element is changed	Select, text,textarea

- Event handlers are embedded in documents as attributes of HTML tags to which you assign JavaScript code to execute. The general syntax is

  <TAG eventHandler="JavaScript Code">

  where TAG is some HTML tag and EventHandler is the name of the event handler.

  For example, the name of the attribute for the event click is onClick.
- <input type= “button” value =“clickMe” onClick= “alert(‘you cliked me’);”>

On click:

The onclick event occurs when the user clicks on an element.

In HTML:

<element onClick="myScript"

In JavaScript:

```
object.onClick = function(){myScript};
```

In JavaScript by using the addEventListener() method

```
object.addEventListener("click", myScript);
```

```
<html>
<body>
<h1>HTML DOM Events</h1>
<h2>The onclick Event</h2>

<p>The onclick event triggers a function when an element is clicked on.</p>
<p>Click to trigger a function that will output "Hello World":</p>

<button onClick="myFunction()">Click me</button>

<p id="demo"></p>

<script>
function myFunction() {
 document.getElementById("demo").innerHTML = "Hello World";
}
</script></body>
</html>
```

OnSubmit:

Submit event occurs when HTML form is submitted

submit event is applicable for form.

```
<form name="f1" action="filename.extension" method="get/post" onSubmit="function name()">

<html>
    <head>
        <title>submit demo</title>
    </head>
<body>
```

```
<script language="JavaScript">
      Function doCheck()
     {
        if(document.frmLogin.login.value == "")
           {
              alert('login field can never be blank');
              return false;
            }
         if((document.frmLogin.password.value) == "")
          {
            alert('password field can never be blank');
             return false;
           }
        //  return true;
     }
</script>
```

Form Validation:

- It is important to validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user.
- JavaScript provides facility to validate the form on the client-side so data processing will be faster than server-side validation.
- Most of the web developers prefer JavaScript form validation.
- Through JavaScript, we can validate name, password, email, date, mobile numbers and more fields.

In this example, we are going to validate the name and password. The name can't be empty and password can't be less than 6 characters long.

Here, we are validating the form on form submit. The user will not be forwarded to the next page until given values are correct.

```
<script>
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;
  if (name==null || name==""){
```

```
 alert("Name can't be blank");
 return false;
}else if(password.length<6){
 alert("Password must be at least 6 characters long.");
 return false;
 }
} </script>
<body>
<form name="myform" method="post" action="abc.html" onSubmit="return validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
```

abc.html

```
<html>
<head>
<title>Valid User </title>
</head>
<body>
<h1>You are valid user</h1>
<p>Thanks for visiting our site</p>
</body>
</html>
```

Spring Framework Overview:

Framework:

- Frameworks are software that are developed and used by developers to build applications.
- There are many popular java frameworks including java server faces(JSF),Maven , Hibernate , Struts ,Spring etc.
- It is easy way to develop applications by using framework.

• Java Enterprise Edition(JEE):

➢ JEE is a core java that contains set of libraries and API'S.

➢ Developers develop application for business logic.

➢ EJB is used for developing business logic applications.

• Disadvantages:

➢ Implementation & usage of EJB is highly complex at initial version of JEE.

➢ It is poor performance.

➢ Due to drawback in EJB, we moved to Spring Framework.

➢ Spring Frame work is more convenient than EJB.

Controllers:

A Spring MVC is a Java framework which is used to build web applications.

➢ It follows the Model-View-Controller design pattern.

➢ A Spring MVC provides an solution to use MVC in spring framework by the help of

Dispatcher Servlet.

➢ Model-View-Controller:

Model :

A model contains the data of the application. A data can be a single object or a collection of objects.

Controller:

A controller contains the business logic of an application. Here, the @Controller annotation is used to mark the class as the controller.

View:

A view represents the provided information in a particular format.

Handler methods:

org.springframework.web.servlet.HandlerMapping is a handler mapping interface

➢ controller class is executed when users hit a URL

The work of Handler Mapping is to find out the controller bean in respect to a URL.

➢ Spring built-in Handler Mapping implementations are:

1. Bean Name Url Handler Mapping
2. Simple Url Handler Mapping
3. Controller Class Handler Mapping

1. BeanNameUrlHandlerMapping

➢ BeanNameUrlHandlerMapping is the default URL handler used by Spring MVC

➢ That means if we do not specify any URL handler then the spring will automatically load this URL handler and pass requested URls to BeanNameUrlHandlerMapping to get the controller bean to be executed.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns=https://www.springframework.org/schema/beans
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance
xmlns:context="https://www.springframework.org/schema/context"
xsi:schemaLocation="https://www.springframework.org/schema/beans
https://www.springframework.org/schema/beans/spring-beans.xsd
https://www.springframework.org/schema/context
https://www.springframework.org/schema/context/spring-context-5.0.7.xsd">
<bean name="/display.html" class="com.java.controller.HelloController"/>
<bean id="urlHandler" class="org.springframework.web.servlet.handler.BeanNameUrlHa
ndlerMapping"/>
</beans>
```

2. SimpleUrlHandlerMapping:

➢ SimpleUrlHandlerMapping is a simple way to define URL mapping using a map or property bean

➢ Mapping more than one URL to a single controller class is possible by using SimpleUrlHandlerMapping

➢ Mapping more than one URL to a single controller is not a good way to work with spring

➢ In this case we need to inject properties or map object that describes the mappings between the request URL path and Controllers

3. Controller Class Name Handler Mapping:

➢ Controller ClassName Handler Mapping is to take the short name of the controller class, remove the "Controller" suffix

Developing Web Application using Spring:

The simple steps to create the first spring application.

➢ To run this application, we are not using any IDE. We are simply using the command

prompt

➢ The simple steps to create the spring application:

1. create the class
2. create the xml file to provide the values
3. create the test class
4. Load the spring jar files
5. Run the test class

1) Create Java class

• This is the simple java bean class containing the name property only.

```
package com.javatpoint;
public class Student { private String name; public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}
public void displayInfo()
{
System.out.println("Hello: "+name);

}
}
```

2) Create the xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
<bean id="studentbean" class="com.java.Student">
<property name="name“ value=“Mounika"></property>
</bean>
```

</beans>

3) Create the test class

```
package com.java;
import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;
 public class Test {
public static void main(String[] args) {
Resource resource=new ClassPathResource("applicationContext.xml");
BeanFactory factory=new XmlBeanFactory(resource);
Student student=(Student)factory.getBean("studentbean");
student.displayInfo();
}
 }
```

4) Load the jar files required for spring framework

There are mainly three jar files required to run this application.

1. org.springframework.core-3.0.1.RELEASE-A
2. com.springsource.org.apache.commons.logging-1.1.1
3. org.springframework.beans-3.0.1.RELEASE-A

5) Run the test class : Now run the test case.

UNIT- V

Django: Introduction to Django, Django Architecture, Django Models & Database Backends, Developing web application using Django

Introduction to Django:

Django is a Python-based web framework that allows you to quickly create efficient web applications. It is also called batteries included framework because Django provides built-in features for everything including Django Admin Interface, default database – SQLlite3, etc. When you're building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc. Django gives you ready-made components to use and that too for rapid development.

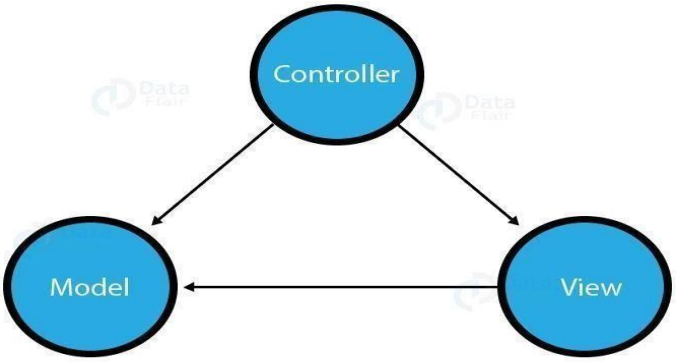
Why Django Framework ?

- Excellent documentation and high scalability.
- Used by Top MNCs and Companies, such as Instagram, Disqus, Spotify, Youtube, Bitbucket,Dropbox, etc. and the list is never-ending.
- Easiest Framework to learn, rapid development and Batteries fully included.
- The last but not least reason to learn Django is Python, Python has huge library and features such as Web Scrapping, Machine Learning, Image Processing, Scientific Computing, etc. One can integrate it all this with web application and do lots and lots of advance stuff.

Django Architecture :

In the previous article, we learned the unique features of Django. Now, we will discuss about Django architecture based on MVC pattern. We will be understanding the MVC pattern in more detail. Django MVC architecture solves lots of problems which were there in the traditional approach for web development.

We will understand the components of the MVC pattern that are Model, Views, and Controller in detail.



1. Model

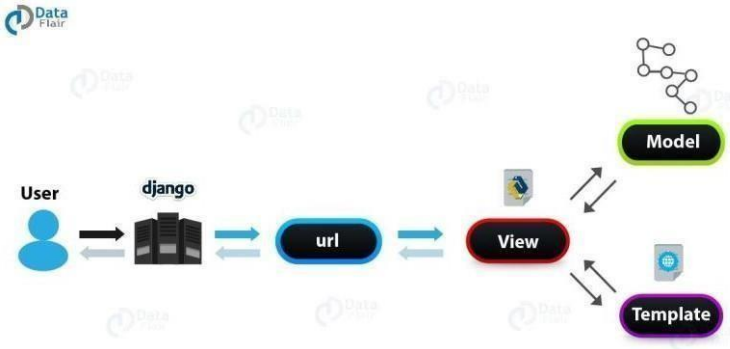
The Model is the part of the web-app which acts as a mediator between the website interface and the database. In technical terms, it is the object which implements the logic for the application's data domain. There are times when the application may only take data in a particular dataset, and directly send it to the view (UI component) without needing any database then the dataset is considered as a model.

Although today if we want any kind of website we need to have some sort of database as we must be requiring some user input even if we are creating a simple blog site.

The Model is the component which contains Business Logic in Django architecture.

For example:

When you sign up on any website you are actually sending information to the controller



which then transfers it to the models which in turn applies business logic on it and stores in the database.

2. View

This component contains the UI logic in the Django architecture.

View is actually the User Interface of the web-application and contains the parts like HTML, CSS and other frontend technologies. Generally, this UI creates from the Models component, i.e., the content comes from the Models component.

For example:

When you click on any link or interact with the website components, the new webpages that website generates is actually the specific views that stores and generates when we are interactingwith the specific components.

3. Controller

The controller as the name suggests is the main control component. What that means is, thecontroller handles the user interaction and selects a view according to the model.

The main task of the controller is to select a view component according to the user interaction andalso applying the model component.

This architecture has lots of advantages and that's why Django is also based on this architecture. Ittakes the same model to an advanced level.

For example:

When we combine the two previous examples, then we can very clearly see that the component which is actually selecting different views and transferring the data to the model's component is the controller.

Django Models & Database Backends:

Django Models:

➢ In Django, a model is a class which is used to contain essential fields and methods.

➢ Each model class maps to a single table in the database.

➢ Django Model is a subclass of django.db.models.Model

Register / Use Model:

After creating a model, register model into the INSTALLED_APPS inside settings.py.

For example: INSTALLED_APPS = [#... 'appname', #...

Django Model Fields:

➢ The fields defined inside the Model class are the columns name of the mapped table.

➢ The fields name should not be python reserve words like clean, save or delete etc

EXAMPLE:

Django Model Fields first_name = models.CharField(max_length=50)

for creating varchar column

release_date = models.DateField() # for creating date column

num_stars = models.IntegerField() # for creating integer column

Field Options:

Each field requires some arguments that are used to set column attributes.

For example:

CharField requires max_length to specify varchar database.

Django Model Example:

We created a model Student that contains the following code in models.py file.

//models.py

class Student(models.Model):

first_name = models.CharField(max_length=20)

last_name = models.CharField(max_length=30)

contact = models.IntegerField()

email = models.EmailField(max_length=50)

age = models.IntegerField()

Database Backends:

➢ The settings.py file contains all the project settings along with database connection details.

➢ By default, Django works with SQLite, database and allows configuring for other databases as well.

➢ Database connectivity requires all the connection details such as database name, user credentials, hostname drive name etc

Example:

We need to provide all connection details in the settings file.

The settings.py file of our project contains the following code for the database.

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

'HOST':'localhost', 'PORT':'3306' }

}

Developing web application using Django:

To build a simple web application that shows a simple message. First, we will create a Django project and inside that directory, we will create an application, where other operations will be performed.

Creating a project: In order to create a new Django project, run the following django-admin startproject command in the command prompt.

Now, go to the folder where the command prompt was being run from. There you can find a new project directory that looks like this.
Inside the 'website' directory (root directory), you will see another directory with name same as the root directory.

Developing Web Application using Django:

Step 1: Django Installation

1. Install Django within your virtual environment:

pip install Django

While our virtual environment is activated, install any additional packages you need for your project. For example, to install Django REST framework:

pip install djangorestframework

Verify Django Installation: After installation, we can verify that Django is installed by running:

django-admin --version

2)Create a Django Project
Project Structure

A Django Project when initialized contains basic files by default such as manage.py, view.py, urls.py etc. A simple project structure is enough to create a single-page application. Here are the major files and their explanations. Inside the folder (project folder) there will be the following files-
manage.py- This file is used to interact with your project via the command line(start the server, sync the database… etc). For getting the full list of commands that can be executed by manage.py, type the help command

python manage.py help

Let's create a project folder named test_project and change directory into it.

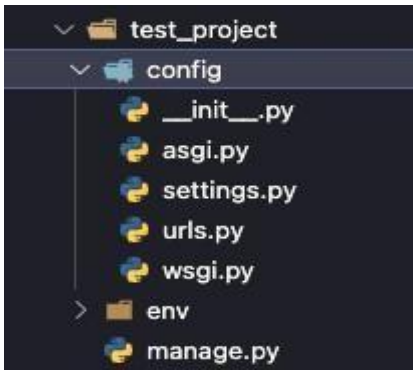
mkdir test_project

cd test_project

Now let's create a Django project named config in our current directory

django-admin startproject config .

this should create a config folder containing the following files bellow along with a mange.py file in our test_project folder.



Let's Run the Development Server
Make sure you are in the directory where they mange.py file was created in this case would be the test_project directory.

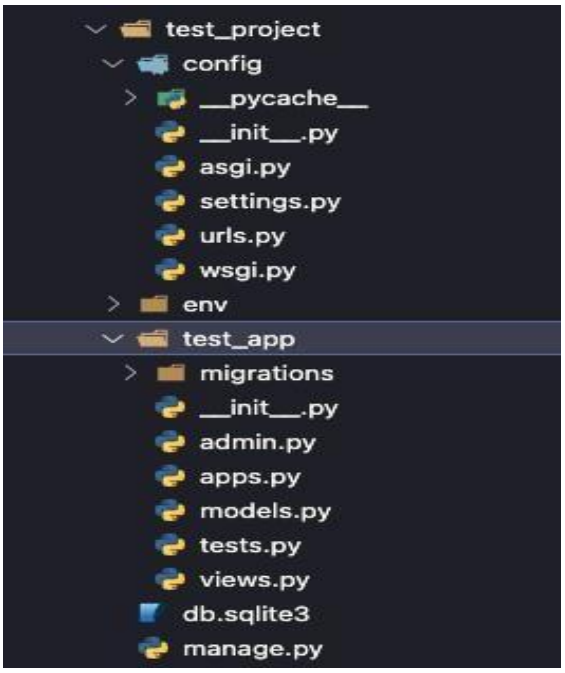
cd test_project

python manage.py runserver

Create an app in django
Before creating the app, let's make sure we change directory into our test_project folder

cd test_project

now that we are inside the test_project folder let's create create a Django app named test_app



django-admin startapp test_app

this will create a test_app directory and a db.sqlite3 file along with models and views.

Add your app definition in settings.py
Django provides some pre-installed apps for users.

To see pre-installed apps, navigate to test_project –> test_app –> settings.py. In your settings.py file, you will find INSTALLED_APPS. Apps listed in INSTALLED_APPS are provided by Django for the developer's comfort.

So, we have finally created an app but to render the app using URLs we need to include the app in our main project so that URLs redirected to that app can be rendered. Let us explore it. Go to test_project ->config-> urls.py and add below code.

from django.urls import include urlpatterns = [path("admin/", admin.site.urls),

Enter the app name in following syntax

```
    path('', include("test_app.urls")),]
```

let's create a function called index in the views.py file in our test_app folder.

```
from django.shortcuts import render

#import the django HttpResponse module from django.http import HttpResponse

 # Create your views here.

# define a function for our view with the HttpResponse function

    def index(request):

    return HttpResponse("Hello World")
```

Let's run the server again

python manage.py runserver

Go to *http://127.0.0.1:8000/* in your web browser. You should see a page similar to the following screenshot: