MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

Recognized under 2(f) and 12 (B) of UGC ACT 1956

Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2008 Certified

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India

Department of Information Technology

WEB APPLICATION DEVELOPMENT

LECTURE NOTES

B.TECH

(III YEAR – I SEM)

(2022-2023)

MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

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

3/-/-/3

(R20A1204) WEB APPLICATION DEVELOPMENT

COURSE OBJECTIVES:

1. To study the insights of the Web architecture and how servlets works.
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, basic HTTP, servlet container, Servlets API, javax.servelet Package, Reading Servlet parameters, service method detail. HTML clients, servlet lifecycle.

Servlets with JDBC, JDBC: Architecture - JDBC API, Passing Control and Data between Pages.

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

Spring Framework Overview, Spring Web MVC Overview, Controllers, Handler Methods, Install and Configure WebServer, Developing Web Application using Spring.

UNIT V

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, SanjeevJaiswalRatanKumar,PACKT Publishing.
3. https://www.djangoproject.com/
   spring framework (IBM)

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 guide is also available on the Internet"

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. While most of the researchers regarded it as an opportunity to enhance computing as it was still in its juvenile phase, 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. Discovered by Tim Berners-Lee, World Wide Web was seen as a service to connect documents in websites using hyperlinks.

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.

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 to see and hear 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, Amaya 10.0,[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, Opera Neon

Uniform Resource Locators, or URLs: A Uniform Resource Locator, or URL is the address of a 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

What are Domains? 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.

MIME (Multi-Purpose Internet Mail Extensions):- MIME is an extension of the original Internet e-mail protocol that lets people use the protocol to exchange different kinds of

data files on the Internet: audio, video, images, application programs, and other kinds, as well as the ASCII text handled in the original protocol, the Simple Mail Transport Protocol (SMTP). In 1991, Nathan Borenstein of Bellcore proposed to the IETF that SMTP be extended so that Internet (but mainly Web) clients and servers could recognize and handle other kinds of data than ASCII text. As a result, new file types were added to "mail" as a supported Internet Protocol file type.

Servers insert the MIME header at the beginning of any Web transmission. Clients use this header to select an appropriate "player" application for the type of data the header indicates. Some of these players are built into the Web client or browser (for example, all browsers come with GIF and JPEG image players as well as the ability to handle HTML files); other players may need to be downloaded.

New MIME data types are registered with the Internet Assigned Numbers Authority (IANA).

MIME is specified in detail in Internet Request for Comments 1521 and 1522, which amend the original mail protocol specification, RFC 821 (the Simple Mail Transport Protocol) and the ASCII messaging header, RFC 822.

Hypertext Transport Protocol:

HTTP means HyperText 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 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. **RasmusLerdorf** 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:

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

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

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

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

SYNTAX OVERVIEW:

Canonical PHP tags *The most universally effective PHP tag style is:*
<?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>

PHP - VARIABLE TYPES

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.

PHP - 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.
- **Booleans:** have only two possible values either true or false.
- **Strings:** are sequences of characters, like 'PHP supports string operations.'

Scalar types

- **Arrays:** are named and indexed collections of other values.
- **Objects:** are instances of programmer-defined classes.
- **NULL:** is a special type that only has one value: NULL.

Compound types

- **Resources:** are special variables that hold references to resources external to PHP (such as database connections).

Special types

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.

PHP 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 and negative {..., -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; $var3 = 0x31;
echo "$var1\n$var2\n$var3";     ?>
```

Output:

```
31
25
49
```

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 have greater 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.

PHP Doubles or Floating point 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)
```

This is the output of beside script

PHP 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";
}        ?>
```

The script uses a **random integer** generator to simulate our case. $r = **rand**(0, 1);
The **rand( )** function returns a random number from the given integer boundaries **0 or 1**.
$male = $r? True: False;

We use the ternary operator to set a $male variable. The variable is based on the random $r value. If $r equals to **1**, the $male variable is set to **True**. If $r equals to **0**, the $male variable is set to **False**.

PHP Strings

String is a data type representing textual data in computer programs. Probably the single most important 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 two string variables together, use the concatenation operator:

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

O/P: Hello Kalpana! 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":

Example

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

Output:Hello Kalpana!

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
echostrlen("Hello world!");
?>
```

The output of the code above will be: 12

PHP 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

PHP NULL

NULL is a special data type that only has **one value: NULL**. To give a variable the NULL value, 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.

Tip: If a variable is created without a value, it is automatically assigned a value of NULL. Variables can also be emptied by setting the value to NULL:

Example1

```
<?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. We will not talk about the resource type here, since it is an advanced topic.

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

Only scalar data (boolean, integer, float and string) can be contained in constants.

PHP - Operators:

What is Operator?

Simple answer can be given using 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	**Assignment Operators**
Increment/Decrement operators	**Conditional (or ternary) Operators**
Comparison Operators	**String Operators**
Logical (or Relational) Operators	**Array Operators**

Arithmetic Operators:

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

Operator	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.

	becomes true.

Logical Operators:

There are following logical operators supported by PHP language
Assume 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

	operand	$C = $C % $A

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

PHP 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)

PHP 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)

Precedence of PHP Operators

Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator −

For example x = 7 + 3 * 2; Here x is assigned 13, not 20 because operator * has higher precedence than + so it first get multiplied with 3*2 and then adds into 7. **Ans:13**

Here operators with the highest precedence appear at the top of the table; those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Category	Operator	Associativity

Unary	! ++ --	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Relational	<<= >>=	Left to right
Equality	== !=	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=	Right to left

Ex:
```
<!DOCTYPE html>
<html>
<body>
<?php
$x = 10;
$y = 6;
echo $x + $y;
?>
</body>
</html>
```

O/P: 16

```
<!DOCTYPE html>
<html>
<body>
<?php
$x = 100;
$y = 50;
var_dump($x > $y); // returns true because $x is greater than $y
?>
```

O/P: bool(true)

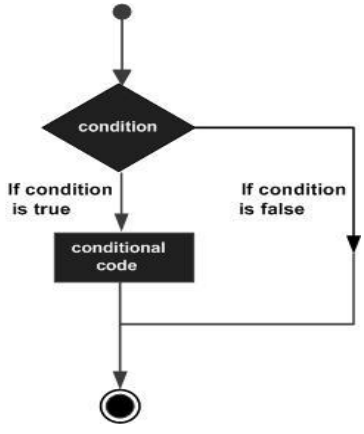
```
<!DOCTYPE html>
<html>
<body>
<?php
$x = 100;
$y = 50;
if ($x == 100 xor $y == 80) {
    echo "Hello world!";
}
?>
```

O/P: Hello world!

PHP - Decision Making

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.



The If...Else Statement

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 statement

Syntax	**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>
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;*	
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.

PHP - Loop Types

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.

The for loop statement

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;
}
```

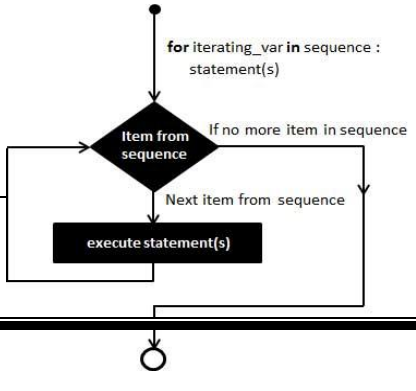
The initializer is used to set the start value for the counter of the number of loop iterations. A variable may be declared here for this purpose and it is traditional to name it $i.

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 –

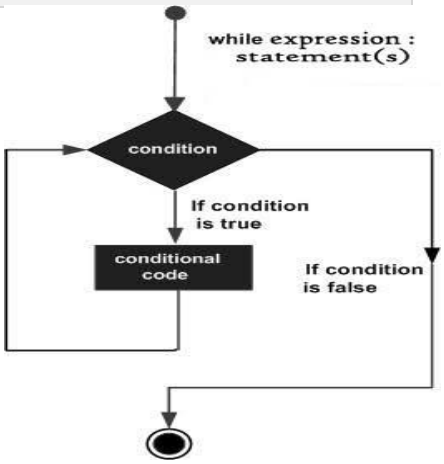
At the end of the loop a=50 and b=25

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...while loop 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;
}while (condition);
```

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 foreach 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.

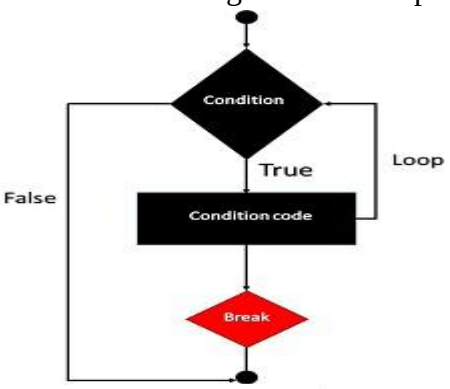
```
<html>
<body>
<?php
      $array = array(1,2,3,4,5);
foreach( $array as $value )
{
        echo "Value is $value <br />";
}
?>
</body></html>
```

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

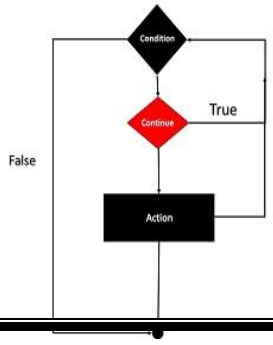
In the following example condition test becomes true when the counter value reaches 3 and loop terminates.

```
<?php
      $i =0;
while( $i <10){
        $i++;
if( $i ==3)break;}
echo("Loop stopped at i = $i");
?>
```

O/P: Loop stopped at i=3

The continue 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
Value is 5
```

PHP – 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 */
functionwriteMessage()
{
echo"Have a nice time Kalpana!";
}/* Calling a PHP Function */
writeMessage();
?>
</body>
</html>
```

Output: Have a nice time Kalpana!

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.

```
<html>
<head><title>Writing PHP Function with Parameters</title></head>
<body>
<?php
functionaddFunction($num1, $num2)
            {
          $sum = $num1 + $num2;
echo"Sum of the two numbers is : $sum";
            }
addFunction(10,20);
?></body></html>
```

Output: Sum of the two numbers is : 30

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.

```
<html>
<head>
<title>Passing Argument by Reference</title>
</head>
<body>
<?php
functionaddFive($num)
{
```

```
        $num+=5;
}
functionaddSix(&$num)
{
        $num+=6;
}
      $orignum=10;
addFive( $orignum);
echo"Original Value is $orignum<br />";
addSix( $orignum);
echo"Original Value is $orignum<br />";
?>
</body>
</html>
```

Output: Original Value is 10
Original Value is 16

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)**.

```
<html><head><title>Writing PHP Function which returns value</title></head>
<body>
<?php
functionaddFunction($num1, $num2)
{
        $sum = $num1 + $num2;
return $sum;
}
      $return_value=addFunction(10,20);
echo"Returned value from the function : $return_value";
?></body></html>
```

Output: Returned value from the function : 30

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.

```
<html><head><title>Writing PHP Function which returns value</title></head>
<body>
<?php
functionprintMe($param= NULL)
{
print $param;
}
printMe("This is test");
printMe();
?>
</body></html>
```

Output: This is test

Dynamic Function Calls

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);
?>
```

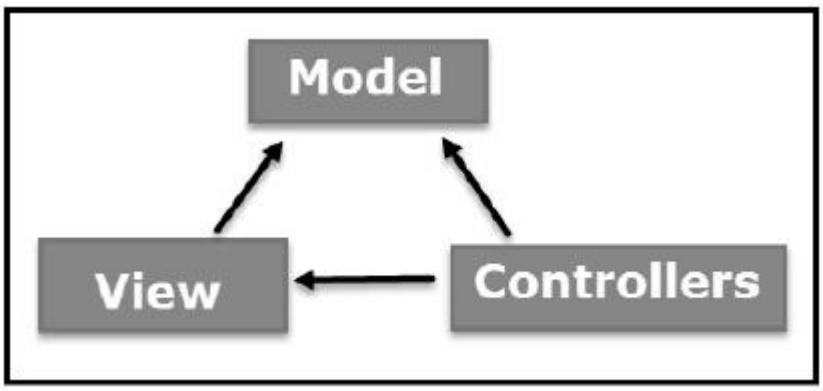
O/P: 350 50 135 80

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 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.

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 learnin rich set of libraries that help build websites and applications rapidly. Users with limited kno programming can also use it. CodeIgniter powered applications include
Kohana	It's a Hierarchical Model View Controller HMVC secure and lightweight framework. It has 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 patterns, configuration, ActiveRecord etc. CakePHP powered applications include;
Zend	It is a powerful framework that is; • Secure, reliable, fast, and scalable • Supports Web 2.0 and creation of web services. It features APIs from vendors like Amazon, Google, Flickr, Yahoo etc. It's ideal for develo applications. Zend powered applications include; • Pimcore CMS, • DotKernel. Companies using the Zend framework include; • BBC • Cisco • Webex • Offers.com

UNIT – II

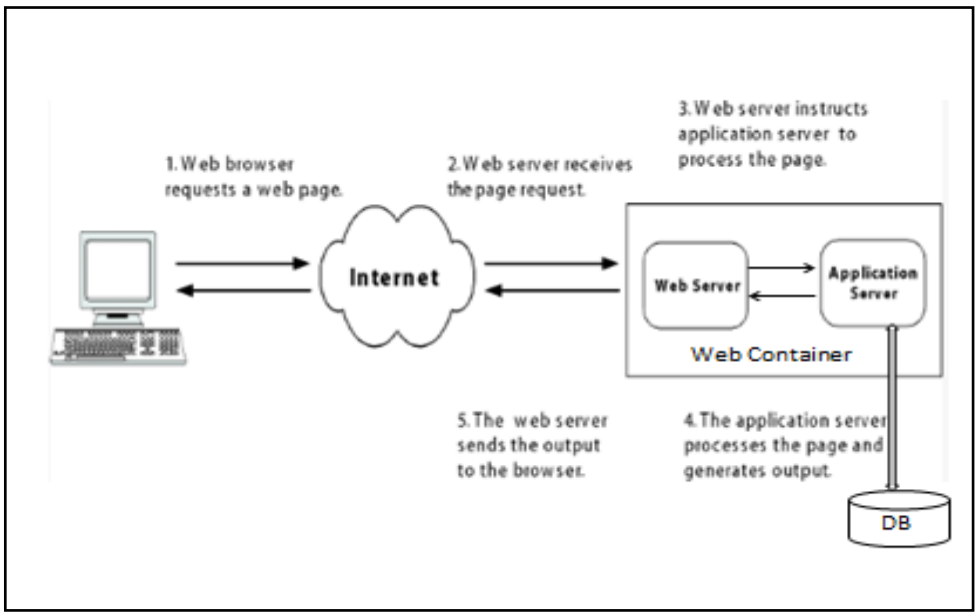
Servlets: Introduction to Servlets, Benefits of Servlets, use as controller in MVC, basic HTTP, servlet container, Servlets API, javax.servelet Package, Reading Servlet parameters, service method detail. HTML clients, servlet lifecycle.

Servlets with JDBC, JDBC: Architecture - JDBC API, Passing Control and Data between Pages.

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.



The advantages of Servlet are as follows:

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, garbage collection, etc.
4. **Secure:** because it uses java language

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**

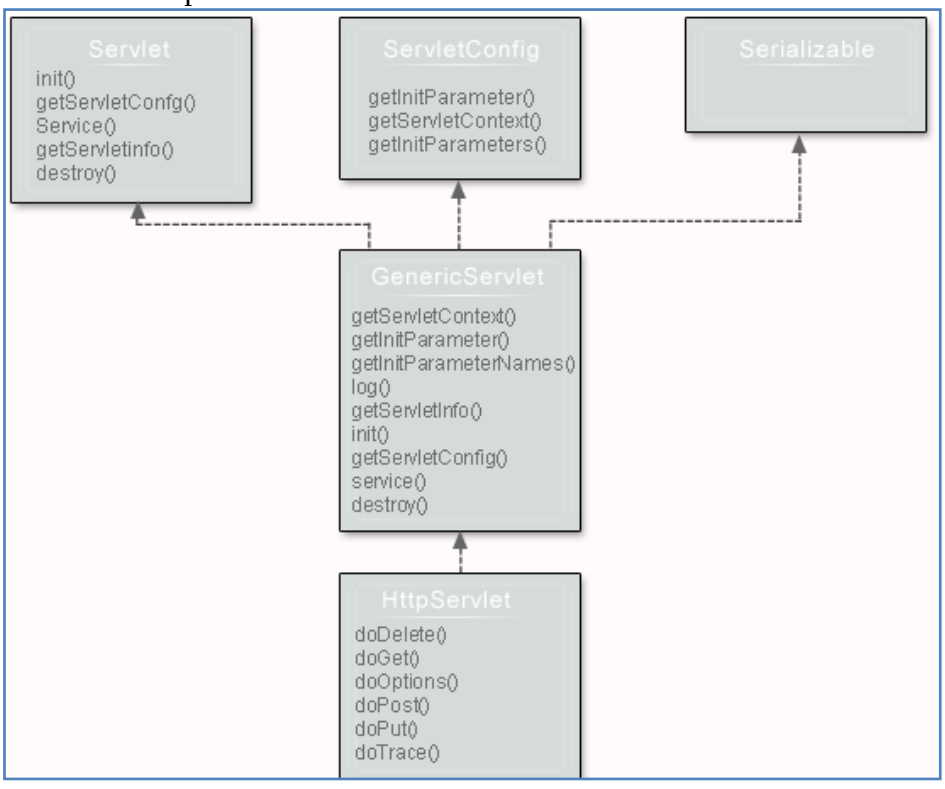
1. javax.servlet

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 KALPANA</b>");
                out.print("</body></html>");
        }
        public void destroy(){
                System.out.println("servlet is destroyed");
        }
        public ServletConfig getServletConfig(){
                return config;
        }
        public String getServletInfo(){
                return "copyright 2007-1010";
        }
}
```

ServletConfig 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 RequestInterface

- 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 ResponseInterface

- 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 no need 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

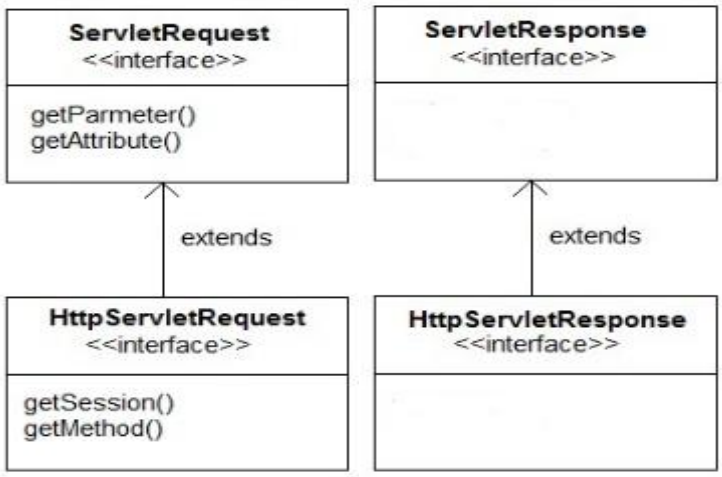
Interfaces

1. HttpServletRequest
2. HttpServletResponse
3. HttpSession

Classes

1. HttpServlet
2. Cookie

HTTPServletRequest and HTTPServletResponse



- ***HTTPServletRequest***Extends the ServletRequest interface to provide request information for 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 `HttpServletRequest` methods.

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.

HTTPSession

- **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.

HTTPServlet

- 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 a lot 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.

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 **HTTPServletRequest** interface includes methods that allow you to read the names and values of parameters that are included in a client request.
- 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:

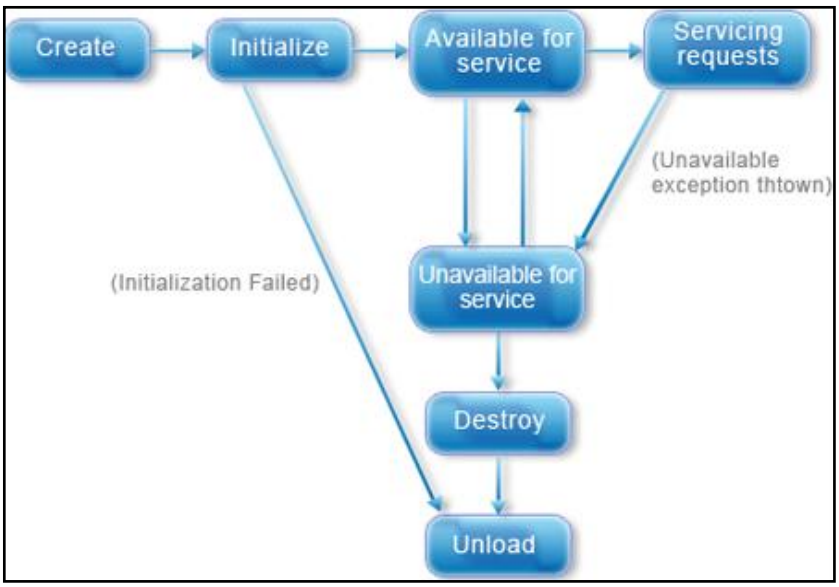
http://www.test.com/hello?key1=value1&key2=value2

- 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.

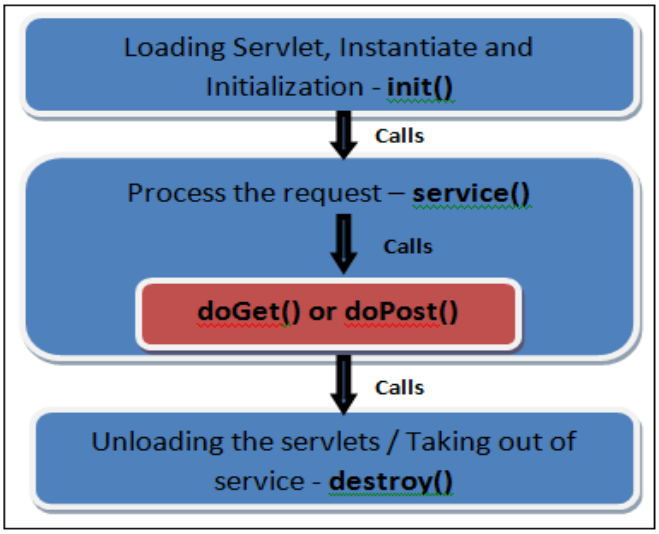
POST method:

- 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.

Life Cycle of Servlet



Life Cycle



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 servlet then waits to receive and process subsequent requests as explained in steps 3 and 4.
6. 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.
- Unlike the GET method, the POST request sends the data as part of the HTTP request body. As a result, the data sent does not appear as a part of URL.
- 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 :

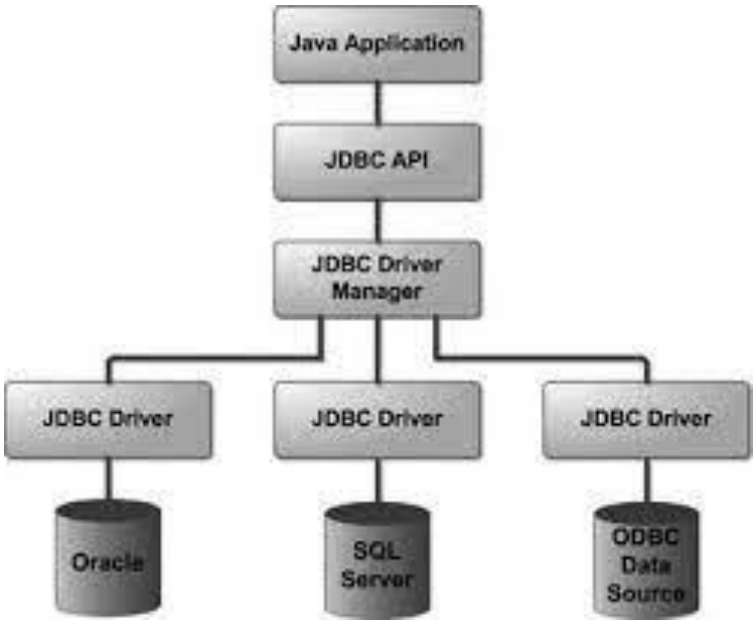
- The destroy() method is called only once at the end of the life cycle of a servlet.
- This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.
- After the destroy() method is 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.
PrintWriter out = response.getWriter();
out.println("<h1>" + message + "</h1>");
  }
public void destroy() {
      // do nothing.
  }
}
```

Servlets with JDBC

Architecture of JDBC



Description:

1. **Application:** It is a java applet or a servlet that communicates with a data source.
2. **The JDBC API:** The JDBC API allows Java programs to execute SQL statements 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. It uses 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 common base 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().

UNIT – III

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

Servlet technology and JavaServer Pages (JSP) are the two main technologies for developing java Web applications. When first introduced by Sun Microsystems in 1996, the Servlet technology was considered superior to the reigning Common Gateway Interface (CGI) because servlets stay in memory after they service the first requests. Subsequent requests for the same servlet do not require instantiation of the servlet's class therefore enabling better responsetime.

Servlets are Java classes that implement the javax.servlet.Servlet interface. They are compiled and deployed in the web server. The problem with servlets is that you embed HTML in Java code. If you want to modify the cosmetic look of the page or you want to modify the structure of the page, you have to change code. Generally speaking, this is left to the better hands (and brains) of a web page designer and not to aJava developer.

```
PrintWriter pw = response.getWriter();
pw.println("<html><head><title>Testing</title></head>"); pw.println("<body
bgcolor=\"# ffdddd\">");
```

As seen from the example above this method presents several difficulties to the web developer:

1. The code for a servlet becomes difficult to understand for theprogrammer.
2. The HTML content of such a page is difficult if not impossible for a web designer to understand ordesign.
3. This is hard to program and even small changes in the presentation, such as the page's background color, will requiretheservlettoberecompiled.AnychangesintheHTMLcontentrequiretherebuildingofthewholeservlet.
4. It's hard to take advantage of web-page development tools when designing the application interface. If such tools are used to develop the web page layout, the generated HTML must then be manually embedded into the servlet code, a process which is time consuming, error prone, and extremelyboring.
5. In many Java servlet-based applications, processing the request and generating the response are both handled by a single servletclass.
6. The servlet contains request processing and business logic (implemented by methods), and also generates the response HTML code, are embedded directly in the servletcode.

JSP solves these problems by giving a way to include java code into an HTML page using scriptlets. This way the HTML code remains intact and easily accessible to web designers, but the page can sill perform itstask.

In late 1999, Sun Microsystems added a new element to the collection of Enterprise Java tools: JavaServer Pages (JSP). JavaServer Pages are built on top of Java servlets and designed to increase the efficiency in which programmers, and even nonprogrammers, can

create webcontent.

Instead of embedding HTML in the code, you place all static HTML in a JSP page, just as in a regular web page, and add a few JSP elements to generate the dynamic parts of the page. The request processing can remain the domain of the servlet, and the business logic can be handled by JavaBeans and EJBcomponents.

A JSP page is handled differently compared to a servlet by the web server. When a servlet is deployed into a web server in compiled (bytecode) form, then a JSP page is deployed in its original, human-readable form.
When a user requests the specific page, the web server compiles the page into a servlet and from there on handles it as a standard servlet.
This accounts for a small delay, when a JSP page is first requested, but any subsequent requests benefit from the same speed effects that are associated with servlets.

The Problem with Servlet

- Servlets are difficult to code which are overcome in JSP. Other way, we can say, JSP is almost a replacement of Servlets, (by large, the better word is extension of Servlets), where coding decreases more than half.
- In Servlets, both static code and dynamic code are put together. In JSP, they are separated. For example,In Servlets:

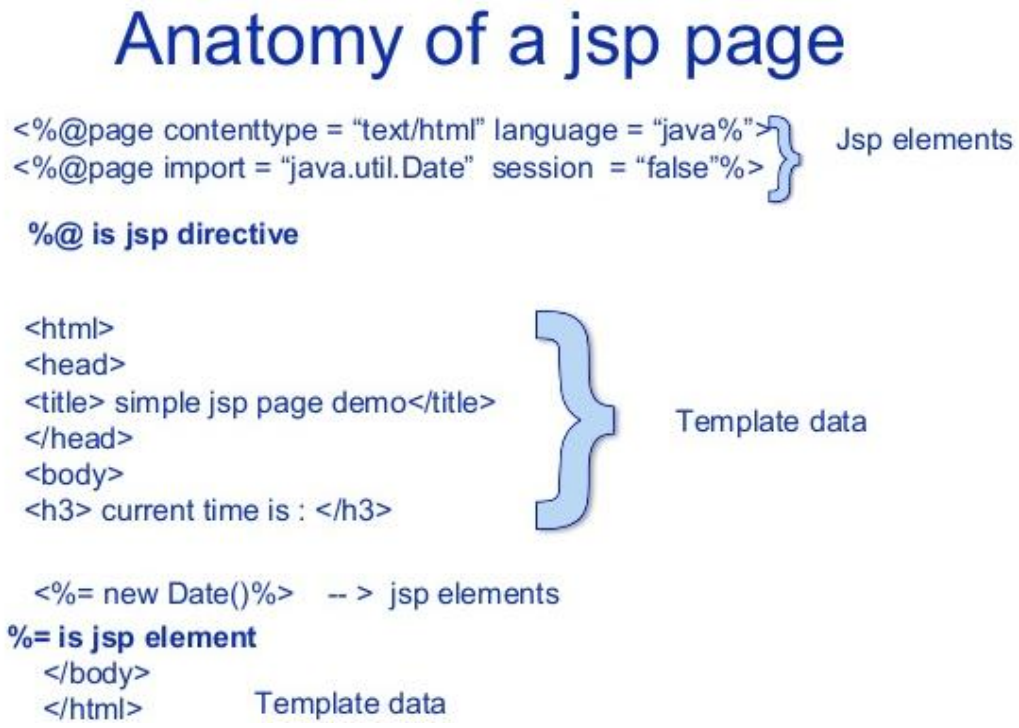
out.println("Hello Mr." + str + " you are great man");

wherestr is the name of the client which changes for each client and is known as dynamic content. The strings, "Hello Mr." and "you are great man" are static content which is the same irrespective of client. In Servlets, in println(), both are put together.

- In JSP, the static content and dynamic content is separated. Static content is written in HTML and dynamic content in JSP. As much of the response comprises of static content (nearly 70%) only, the JSP file more looks as a HTML file.
- Programmer inserts, here and there, chunks of JSP code in a running HTML developed by Designer. As much of the response delivered to cleint by server comprises of static content (nearly 70%), the JSP file more looks like a HTML file. Other way we can say, JSP is nothting but Java in HTML (servlets are HTML
- in Java); java code embedded in HTML.
- When the roles of Designer and Programmer are nicely separated, the product development becomes cleaner and fast. Cost of developing Web site becomes cheaper as Designers are much paid less than Programmers, especially should be thought in the present competitive world.
- Both presentation layer and business logic layer put together in Servlets. In JSP, they can be separated with the usage of JavaBeans.
- The objects of PrintWriter, ServletConfig, ServletContext, HttpSession and RequestDispatcher etc. are created by the Programmer in Servlets and used. But in JSP, they are builtin and are known as "implicit objects". That is, in JSP, Programmer never creates these objects and straightaway use them as they are implicitly created and given by JSP container. This decreases lot of coding.
- JSP can easily be integrated with JavaBeans.
- JSP is much used in frameworks like Sturts etc.
- With JSP, Programmer can build custom tags that can be called in JavaBeans directly. Servlets do not have this advantage. Reusability increases with tag libraries and JavaBean etc.
- Writing alias name in <url-pattern> tag of web.xml is optional in JSP but mandatory in Servlets.
- A Servlet is simply a Java class with extension .java written in normal Java code.
- A Servlet is a Java class. It is written like a normal Java. JSP is comes with some elements that are easy to write.

- JSP needs no compilation by the Programmer. Programmer deploys directly a JSP source code file in server where asincase of Servlets, the Programmer compiles manually a Servlet file and deploys a .class file in server.
- JSP is so easy even a Web Designer can put small interactive code (not knowing much of Java) in static Web pages.
- First time when JSP is called it is compiled to a Servlet. Subsequent calls to the same JSP will call the same compiled servlet (instead of converting the JSP to servlet), Ofcourse, the JSP code would have not modified. This increases performance.

Anatomy of JSP



JSP Processing

Once you have a JSP capable web-server or application server, you need to know the following information about it:

- Where to place the files
- How to access the files from your browser (with an http: prefix, not as file:)

You should be able to create a simple file, such as

<HTML>

<BODY>

Hello, world

</BODY></HTML>

Know where to place this file and how to see it in your browser with an http:// prefix.

Since this step is different for each web-server, you would need to see the web-server documentation to find out how this is done. Once you have completed this step, proceed to the next.

Your first JSP

JSP simply puts Java inside HTML pages. You can take any existing HTML page and change its extension to ".jsp" instead of ".html". In fact, this is the perfect exercise for your first JSP.

Take the HTML file you used in the previous exercise. Change its extension from ".html" to ".jsp". Now load the new file, with the ".jsp" extension, in your browser.

You will see the same output, but it will take longer! But only the first time. If you reload it again, it will load normally.

What is happening behind the scenes is that your JSP is being turned into a Java file, compiled and loaded. This compilation only happens once, so after the first load, the file doesn't take long to load anymore. (But everytime you change the JSP file, it will be re-compiled again.)

Of course, it is not very useful to just write HTML pages with a .jsp extension! We now proceed to see what makes JSP so useful

Adding dynamic content via expressions

As we saw in the previous section, any HTML file can be turned into a JSP file by changing its extension to .jsp. Of course, what makes JSP useful is the ability to embed Java. Put the following text in a file with .jsp extension (let us call it hello.jsp), place it in your JSP directory, and view it in a browser.

```
<HTML>
<BODY>
Hello! The time is now <%= new java.util.Date() %>
</BODY>
</HTML>
```

Notice that each time you reload the page in the browser, it comes up with the current time. The character sequences

<%= and %> enclose Java expressions, which are evaluated at run time.

This is what makes it possible to use JSP to generate dyamic HTML pages that change in response to user actions or vary from user to user.

Explain about JSP Elements

In this lesson we will learn about the various elements available in JSP with suitable examples. In JSP elements can be dividedinto 4 different types.

These are:

1. Expressions

We can use this tag to output any data on the generated page. These data are automatically converted to string and printed on the output stream.

Syntax of JSP Expressions are: <%="Any thing" %>

JSP Expressions start with Syntax of JSP Scriptles are with <%= and ends with %>. Between these this you can put anything and that will convert to the String and that will be displayed.

Example: <%="Hello World!" %> Above code will display 'Hello World!'

2. Scriplets

In this tag we can insert any amount of valid java code and these codes are placed in _jspService method by the JSP engine.

Syntax of JSP Scriptlesare:

<% //java codes

%>

JSP Scriptlets begins with <% and ends %> .We can embed any amount of java code in the JSP Scriptlets. JSP Engine places these code in the _jspService() method. Variables available to the JSP Scriptlets are:

a. Request: Request represents the clients request and is a subclass of HttpServletRequest. Use this variable to retrieve the data submitted along the request.

Example: <% //java codes

String userName=null; userName=request.getParameter("userName");

%>

b. Response: Response represents the server response and is a subclass of HttpServletResponse.

<% response.setContentType("text/html"); %>

c. Session: represents the HTTP session object associated with the request. Your Session ID: <%= session.getId() %>

d. Out: out is an object of output stream and is used to send any output to the client.

3. Directives

A JSP "directive" starts with <%@ characters. In the directives we can import packages, define error handling pages or the session information of the JSP page.

Syntax of JSP directives is:

<%@directive attribute="value" %>

a. page: page is used to provide the information about it. Example: <%@page language="java" %>

b. include: include is used to include a file in the JSP page. Example: <%@ include file="/header.jsp" %>

c. taglib:taglib is used to use the custom tags in the JSP pages (custom tags allows us to defined our own tags). Example: <%@ tagliburi="tlds/taglib.tld" prefix="mytag" %>

Page tag attributes are:

a. language="java"

This tells the server that the page is using the java language. Current JSP specification supports only java language. Example: <%@page language="java" %>

b. extends="mypackage.myclass"

This attribute is used when we want to extend any class. We can use comma(,) to import more than one packages. Example: %@page language="java" import="java.sql.*" %

c. session="true"

When this value is true session data is available to the JSP page otherwise not. By default this value is true.

Example: <%@page language="java" session="true" %>

d. errorPage="error.jsp"

errorPage is used to handle the un-handled exceptions in the page. Example: <%@page session="true" errorPage="error.jsp" %>

e. contentType="text/html;charset=ISO-8859-1"

Use this attribute to set the mime type and character set of the JSP. Example: <%@page contentType="text/html;charset=ISO-8859-1" %>

4. Declarations

This tag is used for defining the functions and variables to be used in theJSP. Syntax of JSP Declaratives are:

<%!

//java codes

%>

JSP Declaratives begins with <%! and ends %> with .We can embed any amount of java code in the JSP Declaratives. Variables and functions defined in the declaratives are class level and can be used anywhere in the JSP page.

Example

```
<%@ page import="java.util.*" %>
<HTML>
<BODY>
<%!
Date theDate = new Date(); Date getDate()
{
System.out.println( "In getDate() method" ); return theDate;
}
%>
Hello! The time is now <%= getDate() %>
</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.

The request Object

The request object is an instance of a **javax.servlet.http.HttpServletRequest** object. Each time a client requests a page the JSP engine creates a new object to represent that request.

The request object provides methods to get the HTTP header information including form data, cookies, HTTP methods etc.

The response Object

The response object is an instance of a **javax.servlet.http.HttpServletResponse** object. Just as the server creates the request object, it also creates an object to represent the response to the client.

The response object also defines the interfaces that deal with creating new HTTP headers. Through this object the JSP programmer can add new cookies or date stamps, HTTP status codes, etc.

The out Object

The out implicit object is an instance of a **javax.servlet.jsp.JspWriter** object and is used to send content in a response.

The initial JspWriter object is instantiated differently depending on whether the page is buffered or not. Buffering can be easily turned off by using the **buffered = 'false'** attribute of the page directive.

The JspWriter object contains most of the same methods as the **java.io.PrintWriter** class. However, JspWriter has some additional methods designed to deal with buffering. Unlike the PrintWriter object, JspWriter throws **IOExceptions**.

Following table lists out the important methods that we will use to write **boolean char, int, double, object, String**, etc.

S.No.	Method & Description
1	**out.print(dataTypedt)** Print a data type value
2	**out.println(dataTypedt)** Print a data type value then terminate the line with new line character.
3	**out.flush()** Flush the stream.

The session Object

The session object is an instance of **javax.servlet.http.HttpSession** and behaves exactly the same way that session objects behave under Java Servlets.

The session object is used to track client session between client requests..

The application Object

The application object is direct wrapper around the **ServletContext** object for the generated Servlet and in reality an instance of a **javax.servlet.ServletContext** object.

.This object is a representation of the JSP page through its entire lifecycle. This object is created when the JSP page is initialized and will be removed when the JSP page is removed by the **jspDestroy()** method.

By adding an attribute to application, you can ensure that all JSP files that make up your web application have access to it

The config Object

The config object is an instantiation of **javax.servlet.ServletConfig** and is a direct wrapper around the **ServletConfig** object for the generated servlet.

This object allows the JSP programmer access to the Servlet or JSP engine initialization parameters such as the paths or file locations etc.

The following **config** method is the only one you might ever use, and its usage is trivial −

config.getServletName();

This returns the servlet name, which is the string contained in the **<servlet-name>** element defined in the **WEB-INF\web.xml** file.

The pageContext Object

The pageContext object is an instance of a **javax.servlet.jsp.PageContext** object. The pageContext object is used to represent the entire JSP page.

This object is intended as a means to access information about the page while avoiding most of the implementation details.

This object stores references to the request and response objects for each request. The **application, config, session**, and out objects are derived by accessing attributes of this object.

The pageContext object also contains information about the directives issued to the JSP page, including the buffering information, the errorPageURL, and page scope.

The PageContext class defines several fields, including **PAGE_SCOPE, REQUEST_SCOPE, SESSION_SCOPE,** and **APPLICATION_SCOPE**, which identify the four scopes. It also supports more than 40 methods, about half of which are inherited from the **javax.servlet.jsp.JspContext class**.

One of the important methods is **removeAttribute**. This method accepts either one or two arguments. For example, **pageContext.removeAttribute ("attrName")** removes the attribute from all scopes, while the following code only removes it from the page scope – pageContext.removeAttribute("attrName", PAGE_SCOPE);

The page Object

This object is an actual reference to the instance of the page. It can be thought of as an object that represents the entire JSP page.

The page object is really a direct synonym for the **this** object.

The exception Object

The exception object is a wrapper containing the exception thrown from the previous page. It is typically used to generate an appropriate response to the error condition.

UNIT – IV

Spring Framework Overview

Spring makes it easy to create Java enterprise applications. It provides everything you need to embrace the Java language in an enterprise environment, with support for Groovy and Kotlin as alternative languages on the JVM, and with the flexibility to create many kinds of architectures depending on an application's needs. As of Spring Framework 5.1, Spring requires JDK 8+ (Java SE 8+) and provides out-of-the-box support for JDK 11 LTS. Java SE 8 update 60 is suggested as the minimum patch release for Java 8, but it is generally recommended to use a recent patch release.

Spring supports a wide range of application scenarios. In a large enterprise, applications often exist for a long time and have to run on a JDK and application server whose upgrade cycle is beyond developer control. Others may run as a single jar with the server embedded, possibly in a cloud environment. Yet others may be standalone applications (such as batch or integration workloads) that do not need a server.

Spring is open source. It has a large and active community that provides continuous feedback based on a diverse range of real-world use cases. This has helped Spring to successfully evolve over a very long time.

1. What We Mean by "Spring"

The term "Spring" means different things in different contexts. It can be used to refer to the Spring Framework project itself, which is where it all started. Over time, other Spring projects have been built on top of the Spring Framework. Most often, when people say "Spring", they mean the entire family of projects. This reference documentation focuses on the foundation: the Spring Framework itself.

The Spring Framework is divided into modules. Applications can choose which modules they need. At the heart are the modules of the core container, including a configuration model and a dependency injection mechanism. Beyond that, the Spring Framework provides foundational support for different application architectures, including messaging, transactional data and persistence, and web. It also includes the Servlet-based Spring MVC web framework and, in parallel, the Spring WebFlux reactive web framework.

A note about modules: Spring's framework jars allow for deployment to JDK 9's module path ("Jigsaw"). For use in Jigsaw-enabled applications, the Spring Framework 5 jars come with "Automatic-Module-Name" manifest entries which define stable language-level module names ("spring.core", "spring.context", etc.) independent from jar artifact names (the jars follow the same naming pattern with "-" instead of ".", e.g. "spring-core" and "spring-context"). Of course, Spring's framework jars keep working fine on the classpath on both JDK 8 and 9+.

2. History of Spring and the Spring Framework

Spring came into being in 2003 as a response to the complexity of the early J2EE specifications. While some consider Java EE and Spring to be in competition, Spring is, in fact, complementary to Java EE. The Spring programming model does not embrace the Java EE platform specification; rather, it integrates with carefully selected individual specifications from the EE umbrella:

- Servlet API (JSR 340)
- WebSocket API (JSR 356)
- Concurrency Utilities (JSR 236)
- JSON Binding API (JSR 367)
- Bean Validation (JSR 303)
- JPA (JSR 338)
- JMS (JSR 914)
- as well as JTA/JCA setups for transaction coordination, if necessary.

The Spring Framework also supports the Dependency Injection (JSR 330) and Common Annotations (JSR 250) specifications, which application developers may choose to use instead of the Spring-specific mechanisms provided by the Spring Framework.

As of Spring Framework 5.0, Spring requires the Java EE 7 level (e.g. Servlet 3.1+, JPA 2.1+) as a minimum - while at the same time providing out-of-the-box integration with newer APIs at the Java EE 8 level (e.g. Servlet 4.0, JSON Binding API) when encountered at runtime. This keeps Spring fully compatible with e.g. Tomcat 8 and 9, WebSphere 9, and JBoss EAP 7.

Over time, the role of Java EE in application development has evolved. In the early days of Java EE and Spring, applications were created to be deployed to an application server. Today, with the help of Spring Boot, applications are created in a devops- and cloud-friendly way, with the Servlet container embedded and trivial to change. As of Spring Framework 5, a WebFlux application does not even use the Servlet API directly and can run on servers (such as Netty) that are not Servlet containers.

Spring continues to innovate and to evolve. Beyond the Spring Framework, there are other projects, such as Spring Boot, Spring Security, Spring Data, Spring Cloud, Spring Batch, among others. It's important to remember that each project has its own source code repository, issue tracker, and release cadence. See spring.io/projects for the complete list of Spring projects.

3. Design Philosophy

When you learn about a framework, it's important to know not only what it does but what principles it follows. Here are the guiding principles of the Spring Framework:

- Provide choice at every level. Spring lets you defer design decisions as late as possible. For example, you can switch persistence providers through configuration without changing your code. The same is true for many other infrastructure concerns and integration with third-party APIs.

- Accommodate diverse perspectives. Spring embraces flexibility and is not opinionated about how things should be done. It supports a wide range of application needs with different perspectives.

- Maintain strong backward compatibility. Spring's evolution has been carefully managed to force few breaking changes between versions. Spring supports a carefully chosen range of JDK versions and third-party libraries to facilitate maintenance of applications and libraries that depend on Spring.

- Care about API design. The Spring team puts a lot of thought and time into making APIs that are intuitive and that hold up across many versions and many years.

- Set high standards for code quality. The Spring Framework puts a strong emphasis on meaningful, current, and accurate javadoc. It is one of very few projects that can claim clean code structure with no circular dependencies between packages.

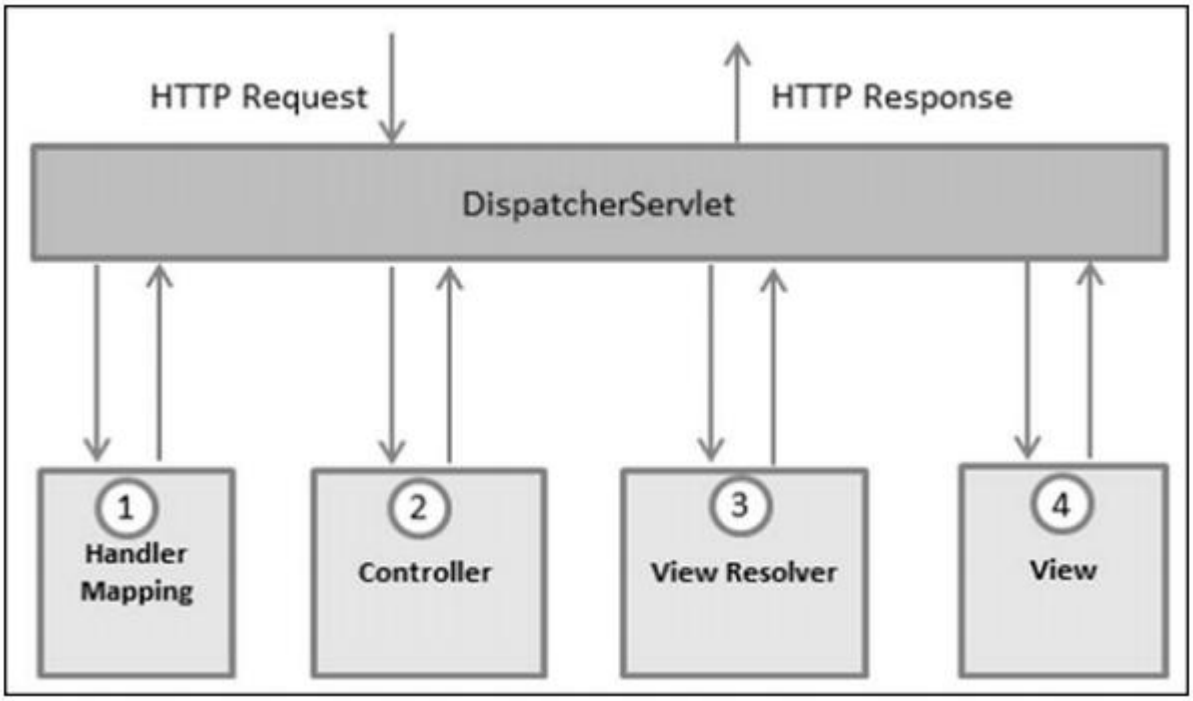
Spring - MVC Framework

The Spring Web MVC framework provides a model-view-controller architecture and ready components that can be used to develop flexible and loosely coupled web applications. The MVC pattern results in separating the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements.

- The Model encapsulates the application data and in general, they will consist of POJO.
- The View is responsible for rendering the model data and in general, it generates HTML output that the client's browser can interpret.
- The Controller is responsible for processing User Requests and Building Appropriate Model and passes it to the view for rendering.

The Dispatcher Servlet

The Spring Web model-view-controller (MVC) framework is designed around a DispatcherServlet that handles all the HTTP requests and responses. The request processing workflow of the Spring Web MVC DispatcherServlet is shown in the following illustration.



Following is the sequence of events corresponding to an incoming HTTP request to DispatcherServlet −

- After receiving an HTTP request, DispatcherServlet consults the **HandlerMapping** to call the appropriate Controller.
- The Controller takes the request and calls the appropriate service methods based on used **GET** or **POST method**. The service method will set model data based on defined business logic and returns view name to the DispatcherServlet.
- The DispatcherServlet will take help from **ViewResolver** to pick up the defined view for the request.
- Once view is finalized, The DispatcherServlet passes the model data to the view, which is finally rendered, on the browsers.

All the above-mentioned components, i.e. HandlerMapping, Controller and ViewResolver are parts of **WebApplicationContext**, which is an extension of the plain **ApplicationContext** with some extra features necessary for web applications.

Required Configuration

We need to map requests that you want the DispatcherServlet to handle, by using a URL mapping in the **web.xml** file. The following is an example to show declaration and mapping for **HelloWeb** DispatcherServlet −

```
<web-app id = "WebApp_ID" version = "2.4"
  xmlns = "http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "http://java.sun.com/xml/ns/j2ee
  http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

  <display-name>Spring MVC Application</display-name>

  <servlet>
    <servlet-name>HelloWeb</servlet-name>
```

```
    <servlet-class>
      org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>HelloWeb</servlet-name>
    <url-pattern>*.jsp</url-pattern>
  </servlet-mapping>
</web-app>
```

The **web.xml** file will be kept in the **WebContent/WEB-INF** directory of your web application. Upon initialization of the **HelloWeb** DispatcherServlet, the framework will try to load the application context from a file named **[servlet-name]-servlet.xml** located in the application's WebContent/WEB-INF directory. In this case, our file will be **HelloWeb-servlet.xml**.

Next, the **<servlet-mapping>** tag indicates which URLs will be handled by which DispatcherServlet. Here, all the HTTP requests ending with .jsp will be handled by the **HelloWeb** DispatcherServlet.

If you do not want to go with the default filename as **[servlet-name]-servlet.xml** and default location as WebContent/WEB-INF, you can customize this file name and location by adding the servlet listener **ContextLoaderListener** in your web.xml file as follows −

```
<web-app...>

  <!-------- DispatcherServlet definition goes here----->
  ....
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/HelloWeb-servlet.xml</param-value>
  </context-param>

  <listener>
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>
</web-app>
```

Now, let us check the required configuration for **HelloWeb-servlet.xml** file, placed in your web application's WebContent/WEB-INF directory.

```
<beans xmlns = "http://www.springframework.org/schema/beans"
  xmlns:context = "http://www.springframework.org/schema/context"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
  http://www.springframework.org/schema/context
  http://www.springframework.org/schema/context/spring-context-3.0.xsd">

  <context:component-scan base-package = "com.tutorialspoint" />

  <bean class = "org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name = "prefix" value = "/WEB-INF/jsp/" />
    <property name = "suffix" value = ".jsp" />
  </bean>

</beans>
```

Following are some important points about **HelloWeb-servlet.xml** file −

- The **[servlet-name]-servlet.xml** file will be used to create the beans defined, overriding the definitions of any beans defined with the same name in the global scope.

- The **<context:component-scan...>** tag will be used to activate the Spring MVC annotation scanning capability, which allows to make use of annotations like **@Controller** and **@RequestMapping**, etc.
- The **InternalResourceViewResolver** will have rules defined to resolve the view names. As per the above-defined rule, a logical view named **hello** is delegated to a view implementation located at **/WEB-INF/jsp/hello.jsp**.

Let us now understand how to create the actual components i.e., Controller, Model and View.

Defining a Controller

The DispatcherServlet delegates the request to the controllers to execute the functionality specific to it. The **@Controller** annotation indicates that a particular class serves the role of a controller. The **@RequestMapping** annotation is used to map a URL to either an entire class or a particular handler method.

```
@Controller
@RequestMapping("/hello")
public class HelloController{

  @RequestMapping(method = RequestMethod.GET)
  public String printHello(ModelMap model) {
    model.addAttribute("message", "Hello Spring MVC Framework!");
    return "hello";
  }

}
```

The **@Controller** annotation defines the class as a Spring MVC controller. Here, the first usage of **@RequestMapping** indicates that all handling methods on this controller are relative to the **/hello** path.

The next annotation **@RequestMapping (method = RequestMethod.GET)** is used to declare the **printHello()** method as the controller's default service method to handle HTTP GET request. We can define another method to handle any POST request at the same URL.

We can also write the above controller in another form, where we can add additional attributes in the @RequestMapping as follows −

```
@Controller
public class HelloController{

  @RequestMapping(value = "/hello", method = RequestMethod.GET)
  public String printHello(ModelMap model) {
    model.addAttribute("message", "Hello Spring MVC Framework!");
    return "hello";
  }

}
```

The **value** attribute indicates the URL to which the handler method is mapped and the **method** attribute defines the service method to handle the HTTP GET request.

Following are some important points to be noted regarding the controller defined above −

- You will define the required business logic inside a service method. You can call another method inside this method as per the requirement.
- Based on the business logic defined, you will create a model within this method. You can set different model attributes and these attributes will be accessed by the view to present the result. This example creates a model with its attribute "message".
- A defined service method can return a String, which contains the name of the **view** to be used to render the model. This example returns "hello" as the logical view name.

Creating JSP Views

Spring MVC supports many types of views for different presentation technologies. These include - **JSPs, HTML, PDF, Excel Worksheets, XML, Velocity Templates, XSLT, JSON, Atom** and **RSS** feeds, **JasperReports**, etc. However, the most common ones are the JSP templates written with JSTL. So, let us write a simple hello view in /WEB-INF/hello/hello.jsp −

```
<html>
  <head>
    <title>Hello Spring MVC</title>
  </head>
  <body>
    <h2>${message}</h2>
  </body>
</html>
```

CONTROLLERS

The following example shows how to use the Controller Class Name Handler Mapping using the Spring Web MVC framework. The **ControllerClassNameHandlerMapping** class is the convention-based handler mapping class, which maps the URL request(s) to the name of the controllers mentioned in the configuration. This class takes the Controller names and converts them to lower case with a leading "/".

For example − HelloController maps to "/hello*" URL.

```
<beans>
  <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name = "prefix" value = "/WEB-INF/jsp/"/>
    <property name = "suffix" value = ".jsp"/>
  </bean>

  <bean class = "org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping"/>

  <bean class = "com.tutorialspoint.HelloController" />

  <bean class = "com.tutorialspoint.WelcomeController"/>
</beans>
```

For example, using above configuration, if URI

- /helloWorld.htm or /hello{any letter}.htm is requested, DispatcherServlet will forward the request to the **HelloController**.
- /welcome.htm is requested, DispatcherServlet will forward the request to the **WelcomeController**.
- /Welcome.htm is requested where W is capital cased, DispatcherServlet will not find any controller and the server will throw 404 status error.

To start with it, let us have a working Eclipse IDE in place and follow the subsequent steps to develop a Dynamic Form based Web Application using Spring Web Framework.

Step	Description
1	Create a project with a name TestWeb under a package com.tutorialspoint as explained in the Spring MVC - Hello World chapter.
2	Create Java classes HelloController and WelcomeController under the com.tutorialspoint package.
3	Create view files hello.jsp, welcome.jsp under the jsp sub-folder.
4	The final step is to create the content of the source and configuration files and export the application as explained below.

HelloController.java

```
package com.tutorialspoint;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.AbstractController;

public class HelloController extends AbstractController{

   @Override
   protected ModelAndView handleRequestInternal(HttpServletRequest request,
      HttpServletResponse response) throws Exception {
      ModelAndView model = new ModelAndView("hello");
      model.addObject("message", "Hello World!");
      return model;
   }
}
```

WelcomeController.java

```
package com.tutorialspoint;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.AbstractController;

public class WelcomeController extends AbstractController{

   @Override
   protected ModelAndView handleRequestInternal(HttpServletRequest request,
      HttpServletResponse response) throws Exception {
      ModelAndView model = new ModelAndView("welcome");
      model.addObject("message", "Welcome!");
      return model;
   }
}
```

TestWeb-servlet.xml

```
<beans xmlns = "http://www.springframework.org/schema/beans"
   xmlns:context = "http://www.springframework.org/schema/context"
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation = "
   http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
   http://www.springframework.org/schema/context
   http://www.springframework.org/schema/context/spring-context-3.0.xsd">

   <bean class = "org.springframework.web.servlet.view.InternalResourceViewResolver">
      <property name = "prefix" value = "/WEB-INF/jsp/"/>
      <property name = "suffix" value = ".jsp"/>
   </bean>

   <bean class = "org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping"/>

   <bean class = "com.tutorialspoint.HelloController" />

   <bean class = "com.tutorialspoint.WelcomeController"/>
</beans>
```

hello.jsp

```
<%@ page contentType="text/html; charset = UTF-8" %>
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <h2>${message}</h2>
  </body>
</html>
```

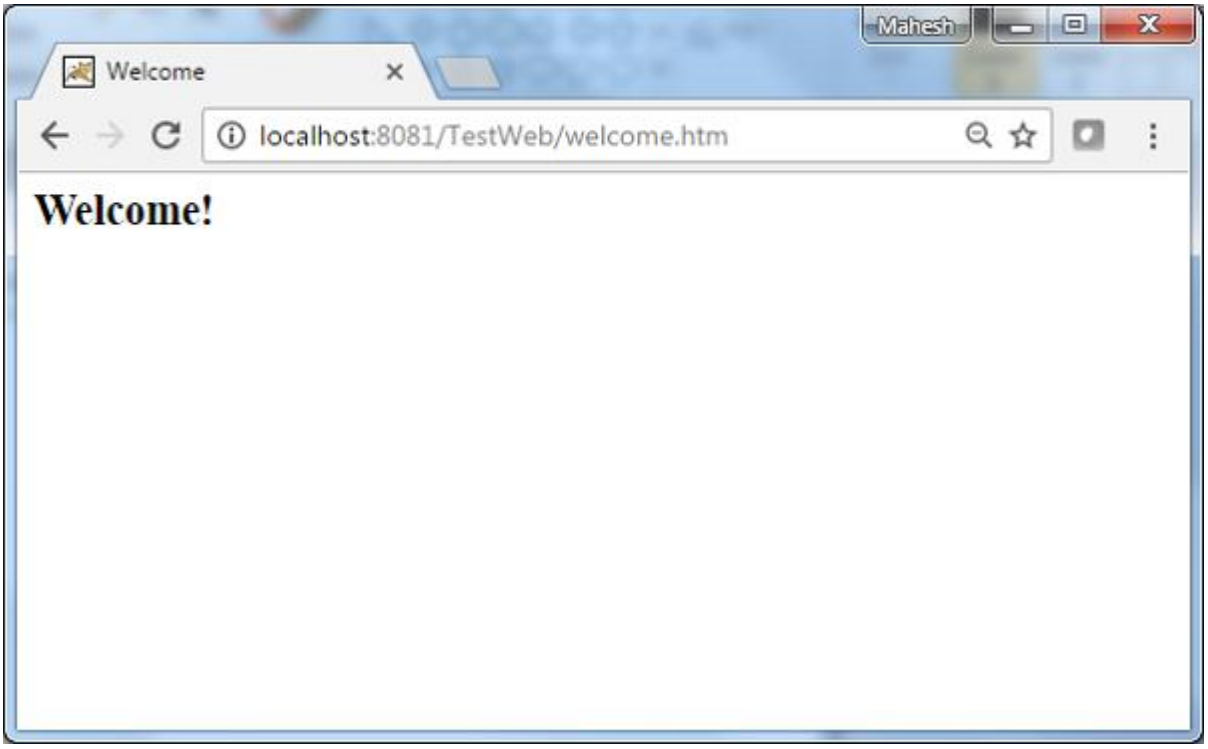
welcome.jsp

```
<%@ page contentType = "text/html; charset=UTF-8" %>
<html>
  <head>
    <title>Welcome</title>
  </head>
  <body>
    <h2>${message}</h2>
  </body>
</html>
```

Once you are done with creating source and configuration files, export your application. Right click on the application, use the **Export → WAR File** option and save the **TestWeb.war** file in Tomcat's webapps folder.

Now, start your Tomcat server and make sure you are able to access other webpages from the webapps folder using a standard browser. Try a URL **− http://localhost:8080/TestWeb/helloWorld.htm** and we will see the following screen, if everything is fine with the Spring Web Application.

Try a URL **http://localhost:8080/TestWeb/hello.htm** and we will see the following screen, if everything is fine with the Spring Web Application.



Try a URL **http://localhost:8080/TestWeb/Welcome.htm** and we will see the following screen, if everything is fine with the Spring Web Application.

Spring - Exception Handling Example

The following example shows how to write a simple web-based application using Spring MVC Framework, which can handle one or more exceptions raised inside its controllers. To start with, let us have a working Eclipse IDE in place and take the following steps to develop a Dynamic Form-based Web Application using Spring Web Framework −

Step	Description
1	Create a *Dynamic Web Project* with a name *HelloWeb* and create a package *com.tutorialspoint* under the *src* folder in the created project
2	Drag and drop below mentioned Spring and other libraries into the folder *WebContent/WEB-INF/lib*.
3	Create a Java classes *Student*, *StudentController* and *SpringException* under the *com.tutorialspoint* package.
4	Create Spring configuration files *Web.xml* and *HelloWeb-servlet.xml* under the *WebContent/WEB-INF* folder.
5	Create a sub-folder with a name *jsp* under the *WebContent/WEB-INF* folder. Create a view files *student.jsp*, *result.jsp*, *error.jsp*, and *ExceptionPage.jsp* under *jsp* sub-folder.
6	The final step is to create the content of all the source and configuration files and export the application as explained below.

Following is the content of **Student.java** file

```
package com.tutorialspoint;

public class Student {
  private Integer age;
  private String name;
  private Integer id;

  public void setAge(Integer age) {
```

```
    this.age = age;
  }
  public Integer getAge() {
    return age;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getName() {
    return name;
  }
  public void setId(Integer id) {
    this.id = id;
  }
  public Integer getId() {
    return id;
  }
}
```

Following is the content of **SpringException.java** file

```
package com.tutorialspoint;

public class SpringException extends RuntimeException{
  private String exceptionMsg;

  public SpringException(String exceptionMsg) {
    this.exceptionMsg = exceptionMsg;
  }
  public String getExceptionMsg(){
    return this.exceptionMsg;
  }
  public void setExceptionMsg(String exceptionMsg) {
    this.exceptionMsg = exceptionMsg;
  }
}
```

Following is the content of **StudentController.java** file. Here, you need to annotate a service method using *@ExceptionHandler* where you can specify one or more exceptions to be handled. If you are specifying more than one exception then you can use comma separated values.

```
package com.tutorialspoint;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.ui.ModelMap;

@Controller
public class StudentController {
  @RequestMapping(value = "/student", method = RequestMethod.GET)
  public ModelAndView student() {
    return new ModelAndView("student", "command", new Student());
  }
  @RequestMapping(value = "/addStudent", method = RequestMethod.POST)
  @ExceptionHandler({SpringException.class})
  public String addStudent( @ModelAttribute("HelloWeb")Student student,
    ModelMap model) {

    if(student.getName().length() < 5 ){
      throw new SpringException("Given name is too short");
    } else {
```

```
      model.addAttribute("name", student.getName());
    }

    if( student.getAge() < 10 ){
      throw new SpringException("Given age is too low");
    } else {
      model.addAttribute("age", student.getAge());
    }
    model.addAttribute("id", student.getId());
    return "result";
  }
}
```

Following is the content of Spring Web configuration file **web.xml**

```
<web-app id = "WebApp_ID" version = "2.4"
  xmlns = "http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "http://java.sun.com/xml/ns/j2ee
  http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

  <display-name>Spring Exception Handling</display-name>

  <servlet>
    <servlet-name>HelloWeb</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>HelloWeb</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

</web-app>
```

Following is the content of another Spring Web configuration file **HelloWeb-servlet.xml**

```
<beans xmlns = "http://www.springframework.org/schema/beans"
  xmlns:context = "http://www.springframework.org/schema/context"
  axmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
  http://www.springframework.org/schema/context
  http://www.springframework.org/schema/context/spring-context-3.0.xsd">

  <context:component-scan base-package = "com.tutorialspoint" />

  <bean class = "org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name = "prefix" value = "/WEB-INF/jsp/" />
    <property name = "suffix" value = ".jsp" />
  </bean>

  <bean class = "org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
    <property name = "exceptionMappings">
      <props>
        <prop key = "com.tutorialspoint.SpringException">
          ExceptionPage
        </prop>
      </props>
    </property>
    <property name = "defaultErrorView" value = "error"/>
  </bean>

</beans>
```

Here you specified ExceptionPage as an exception view in case SpringException occurs, if there is any other type of exception then a generic view error will take place.

Following is the content of Spring view file **student.jsp**

```
<%@taglib uri = "http://www.springframework.org/tags/form" prefix = "form"%>
<html>
  <head>
    <title>Spring MVC Exception Handling</title>
  </head>

  <body>
    <h2>Student Information</h2>
    <form:form method = "POST" action = "/HelloWeb/addStudent">
      <table>
        <tr>
          <td><form:label path = "name">Name</form:label></td>
          <td><form:input path = "name" /></td>
        </tr>
        <tr>
          <td><form:label path = "age">Age</form:label></td>
          <td><form:input path = "age" /></td>
        </tr>
        <tr>
          <td><form:label path = "id">id</form:label></td>
          <td><form:input path = "id" /></td>
        </tr>
        <tr>
          <td colspan = "2"><input type = "submit" value = "Submit"/></td>
        </tr>
      </table>
    </form:form>
  </body>

</html>
```

Following is the content of Spring view file **error.jsp**

```
<html>
  <head>
    <title>Spring Error Page</title>
  </head>

  <body>
    <p>An error occured, please contact webmaster.</p>
  </body>
</html>
```

Following is the content of Spring view file **ExceptionPage.jsp**. Here you will access the exception instance via ${exception}.

```
<%@taglib uri = "http://www.springframework.org/tags/form" prefix = "form"%>
<html>
  <head>
    <title>Spring MVC Exception Handling</title>
  </head>

  <body>
    <h2>Spring MVC Exception Handling</h2>
    <h3>${exception.exceptionMsg}</h3>
  </body>
</html>
```

Following is the content of Spring view file **result.jsp**

```
<%@taglib uri = "http://www.springframework.org/tags/form" prefix = "form"%>
```

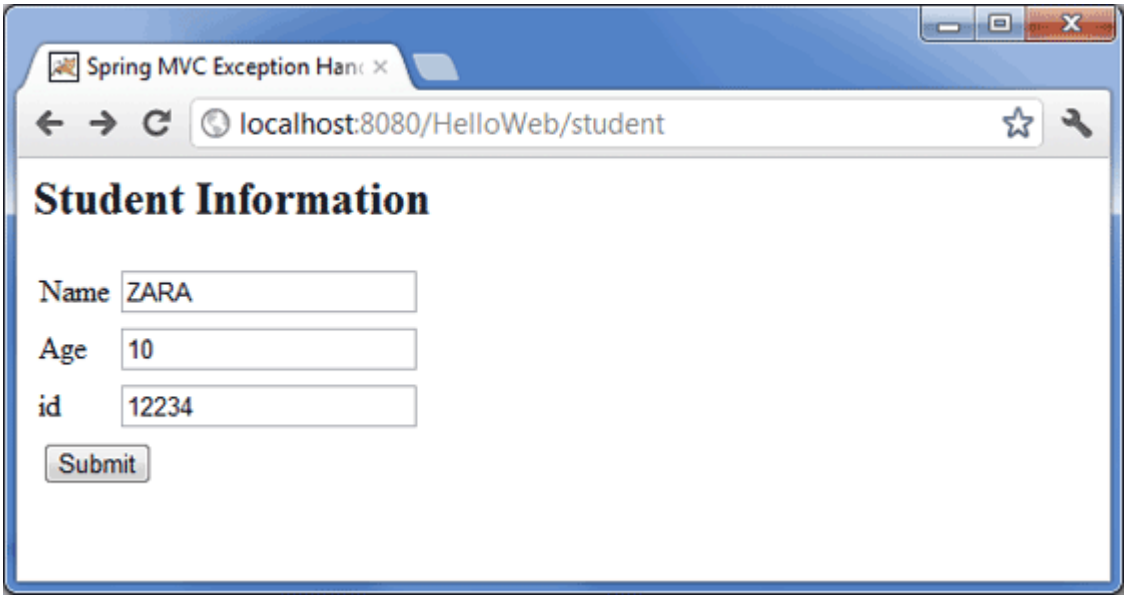
```
<html>
  <head>
    <title>Spring MVC Form Handling</title>
  </head>

  <body>
    <h2>Submitted Student Information</h2>

    <table>
      <tr>
        <td>Name</td>
        <td>${name}</td>
      </tr>
      <tr>
        <td>Age</td>
        <td>${age}</td>
      </tr>
      <tr>
        <td>ID</td>
        <td>${id}</td>
      </tr>
    </table>
  </body>

</html>
```

Finally, following is the list of Spring and other libraries to be included in your web application. You simply drag these files and drop them in WebContent/WEB-INF/lib folder.

- commons-logging-x.y.z.jar
- org.springframework.asm-x.y.z.jar
- org.springframework.beans-x.y.z.jar
- org.springframework.context-x.y.z.jar
- org.springframework.core-x.y.z.jar
- org.springframework.expression-x.y.z.jar
- org.springframework.web.servlet-x.y.z.jar
- org.springframework.web-x.y.z.jar
- spring-web.jar

Once you are done creating the source and configuration files, export your application. Right-click on your application and use the Export > WAR File option and save yourHelloWeb.war file in Tomcat's webapps folder.

Now start your Tomcat server and make sure you are able to access other web pages from webapps folder using a standard browser. Now try to access the URL http://localhost:8080/HelloWeb/student. If everything is fine with your Spring Web Applicationand, you should see the following result.



Enter the values as shown above and click the Submit buttom. If everything is fine with your Spring Web Application, you should see the following result.

INSTALL AND CONFIGURE WEBSERVER

Windows In this section, we will cover the installation process in Windows 7. The installation of these software packages is commonly known as WAMP (Windows Apache MySQL PHP). Let's get started.

STEP 1

Go to http://www.wampserver.com/en/download.php and download the version most applicable to your system. If you are running a 64-bit PC, then you will want the 64-bit installer. If you are unsure, stick with the 32-bit installer, which should work on both 64-bit and 32-bit PCs. After you have downloaded the installer, start it up, and start completing the steps, as shown in Figure A-1.

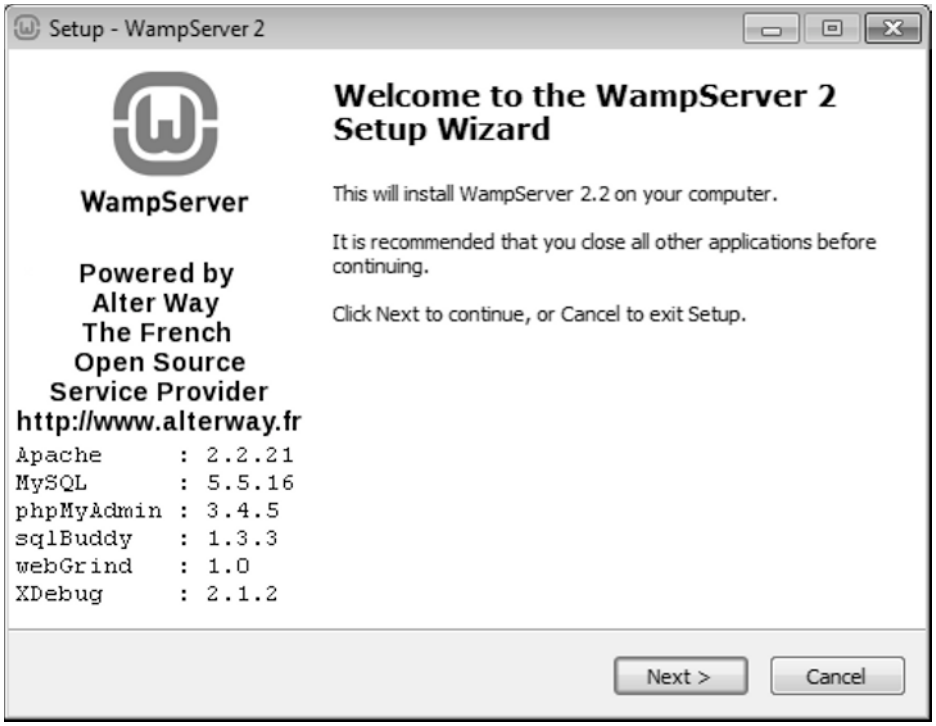


Figure A-1. *Setting up with the WAMP installer*

You can install WAMP to the default c:\wamp directory, which it suggests. You can also use the default SMTP and e-mail settings it suggests, which you can change later if you need to. At some point, it will ask you to choose

your default browser. Since I have Google Chrome installed, I want to use that as my default. The quickest way to find that executable is to right-click the Google Chrome shortcut icon and select Properties ° Shortcut. Then copy the whole Target value and paste it into the File name box. This is demonstrated in Figures A-2 and A-3.

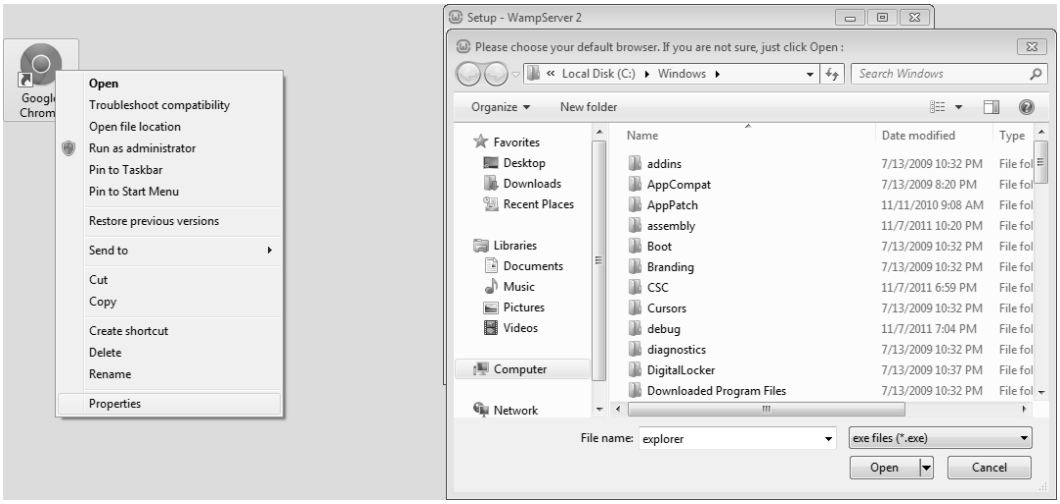


Figure A-2. *Choosing the default browser*

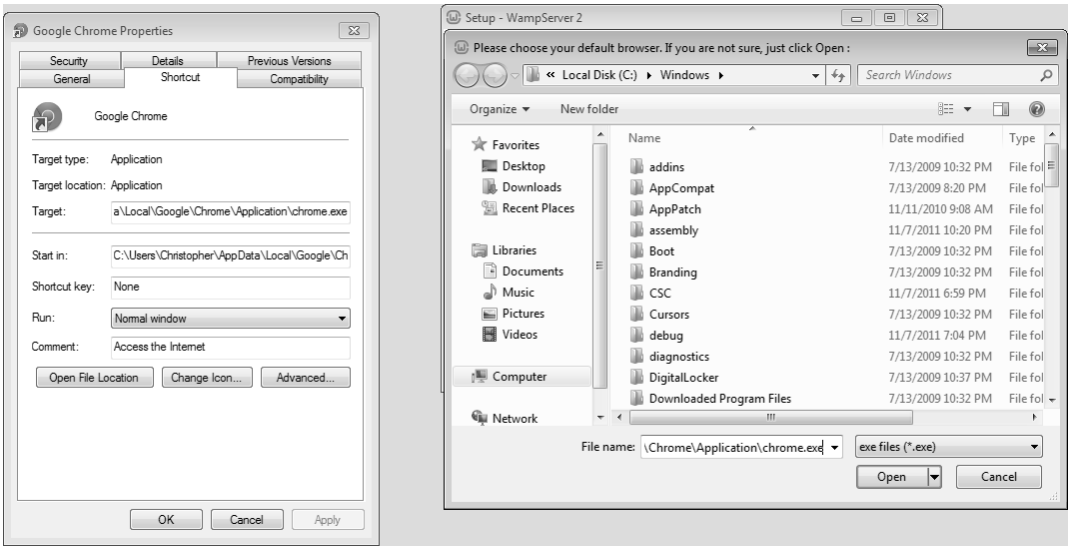


Figure A-2. *Choosing the default browser*

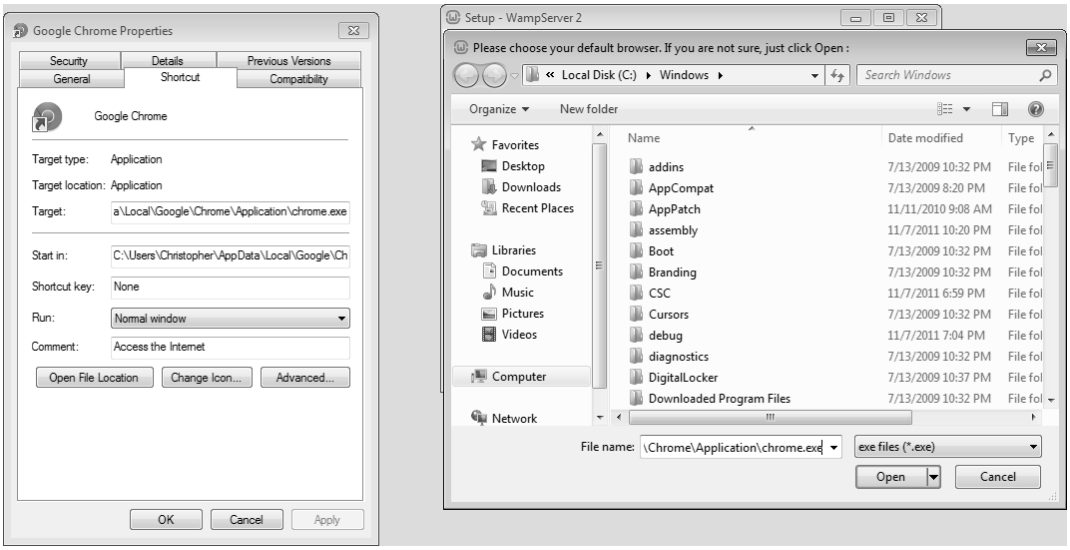


Figure A-3. *Getting the path to Google Chrome*

After you have completed the installation of WAMP, you will see a new icon in your system tray. If you left- click on this icon, you will see a pop-up menu. Click localhost.

Developing Web Application using Spring.

Web Applications

Spring makes building web applications fast and hassle-free. By removing much of the boilerplate code and configuration associated with web development, you get a modern web programming model that streamlines the development of server-side HTML applications, REST APIs, and bidirectional, event-based systems.

Developer productivity

Spring Boot is the starting point of your developer experience, whatever you're building. Spring Boot is designed to get you up and running as quickly as possible, with minimal upfront configuration. With its embedded application servers, you can be serving in seconds.

Spring's out-of-the-box, production-ready features (like tracing, metrics, and health status) provide developers with deep insight into their applications.

Finally, Spring supports multiple JVM languages: Java, Kotlin, and Groovy.

Battle-tested security

When it's time to secure your web application, Spring Security supports many industry-standard authentication protocols, including SAML, OAuth, and LDAP.

Get protection from top OWASP attacks, such as session fixation, clickjacking, cross-site request forgery, and more.

Data access made easy

Spring helps developers connect their web applications to a number of data stores. It supports relational and non-relational databases, map-reduce frameworks, and cloud-based data services.

Get Started with JPA or Get Started with MongoDB

Unit- V

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.

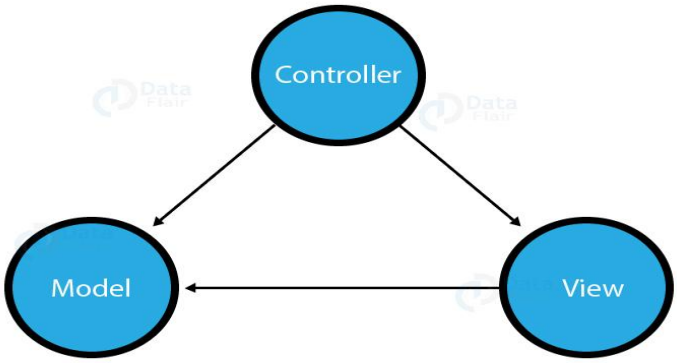
Why Django Framework ?

- 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 – 3 Major Components of MVC Pattern

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

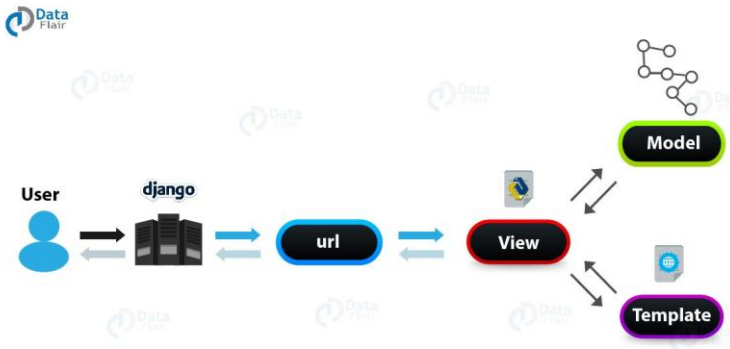
The Model is the part of the web-app which acts as a mediator between the website interface and the database. In technical terms, it is the object which implements the logic for the application's data domain. There are times when the application may only take data in a particular dataset, and directly 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.



Explore how to create a project in Django.

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 interacting with the specific components.

3. Controller

The controller as the name suggests is the main control component. What that means is, the controller 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 and also applying the model component.

This architecture has lots of advantages and that's why Django is also based on this architecture. It takes 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 and Database Backends

Databases

Django officially supports the following databases:

- PostgreSQL
- MariaDB
- MySQL
- Oracle
- SQLite

PostgreSQL notes

Django supports PostgreSQL 10 and higher. psycopg2 2.8.4 or higher is required, though the latest release is recommended.

PostgreSQL connection settings

See HOST for details.

To connect using a service name from the connection service file and a password from the password file, you must specify them in the OPTIONS part of your database configuration in DATABASES:

settings.py¶

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.postgresql',
      'OPTIONS': {
         'service': 'my_service',
         'passfile': '.my_pgpass',
      },
   }
}
```

MariaDB notes

Django supports MariaDB 10.2 and higher.

To use MariaDB, use the MySQL backend, which is shared between the two. See the MySQL notes for more details.

MySQL notes

Version support

Django supports MySQL 5.7 and higher.

Django's inspectdb feature uses the information_schema database, which contains detailed data on all database schemas.Django expects the database to support Unicode (UTF-8 encoding) and delegates to it the task of enforcing transactions and referential integrity. It is important to be aware of the fact that the two latter ones aren't actually enforced by MySQL when using the MyISAM storage engine, see the next section.

MySQL notes

Version support

Django supports MySQL 5.7 and higher.Django's inspectdb feature uses the information_schema database, which contains detailed data on all database schemas.Django expects the database to support Unicode (UTF-8 encoding) and delegates to it the task of enforcing transactions and referential integrity. It is important to be aware of the fact that the two latter ones aren't actually enforced by MySQL when using the MyISAM storage engine, see the next section.