Servlet

INTRODUCTION

FortuneCloud
We Understand Technology

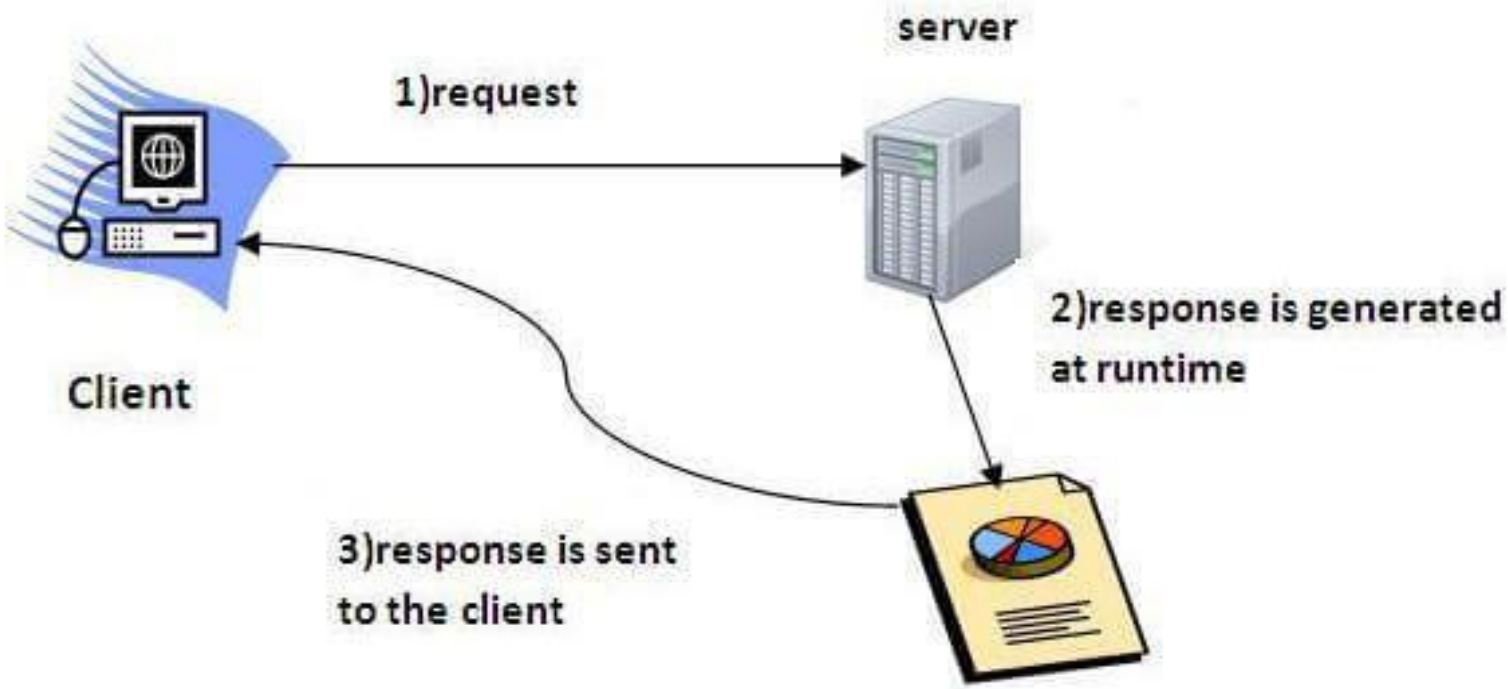
- **Servlet** technology is used to create a web application.
- resides at server side and generates a dynamic web page.
- Servlet is an API that provides many interfaces and classes.
- **Servlet** technology is robust and scalable because of java language.

What is a web application?

FortuneCloud
We Understand Technology

- A web application is an application accessible from the web.
- The web components typically execute in Web Server and respond to the HTTP request.
- A web application is composed of web components like Servlet, JSP, Filter, etc.

Working of Servlet



Servlet Terminology	Description
Website: static vs dynamic	web pages that may contain text, images, audio and video.
HTTP	used to establish communication between client and server.
HTTP Requests	It is the request send by the computer to a web server.
Get vs Post	GET and POST request.
Container	used in java for dynamically generating the web pages on the server side.
Server: Web vs Application	It is used to manage the network resources and for running the program or software that provides services.
Content Type	It is HTTP header that provides the description about what are you sending to the browser.

Website

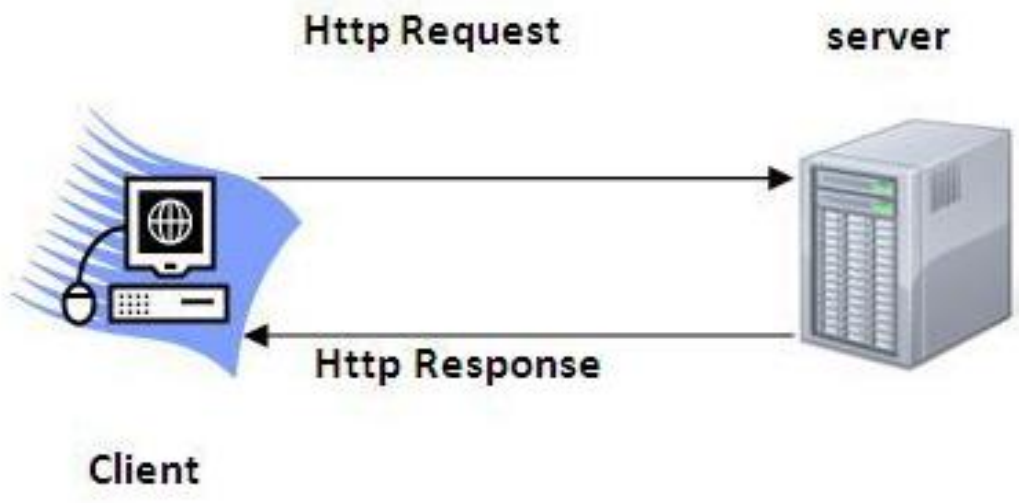
- Website is a collection of related web pages that may contain text, images, audio and video.
- Each website has specific internet address (URL) that you need to enter in your browser to access a website.
- A website is managed by its owner that can be an individual, company or an organization.
- Website is hosted on one or more servers.

Static

Dynamic

HTTP(Hyper Text Transfer Protocol)

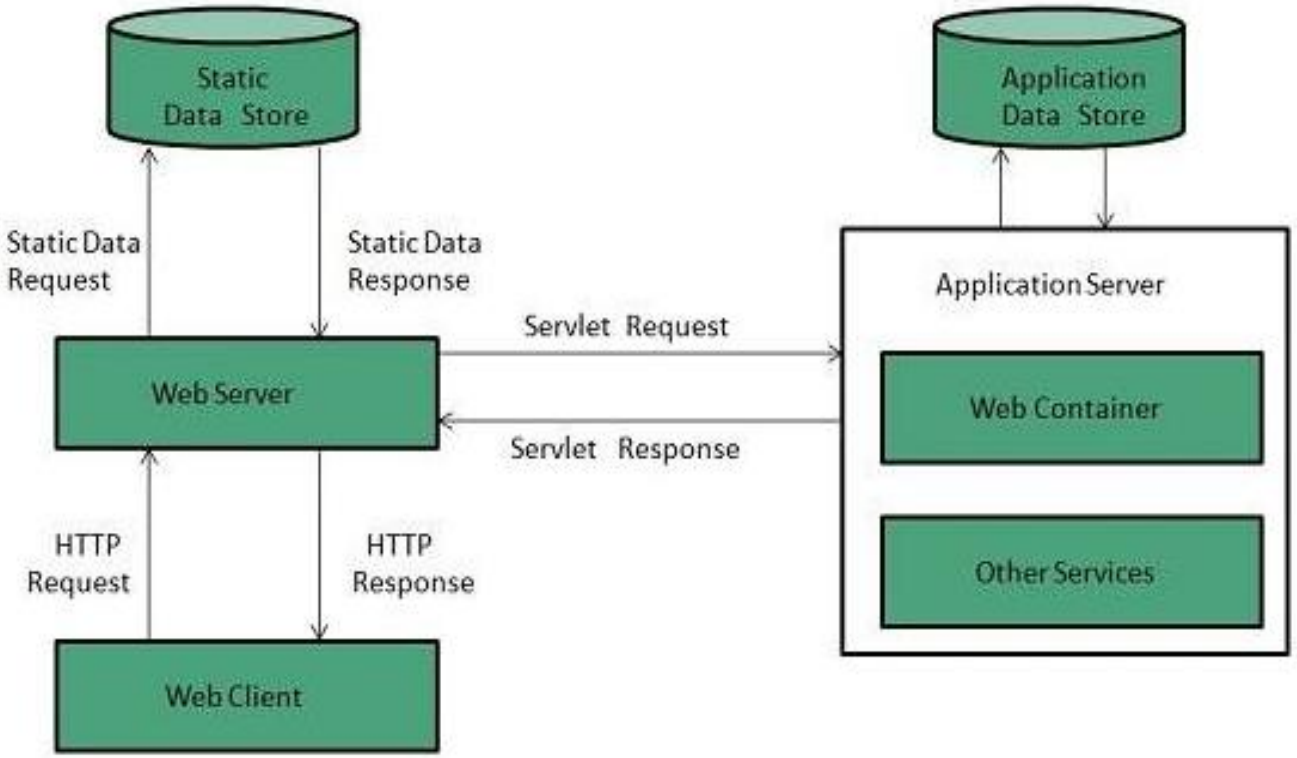
- It is the data communication protocol used to establish communication between client and server.
- HTTP is TCP/IP based communication protocol.
- used to deliver the data on the World Wide Web (WWW).

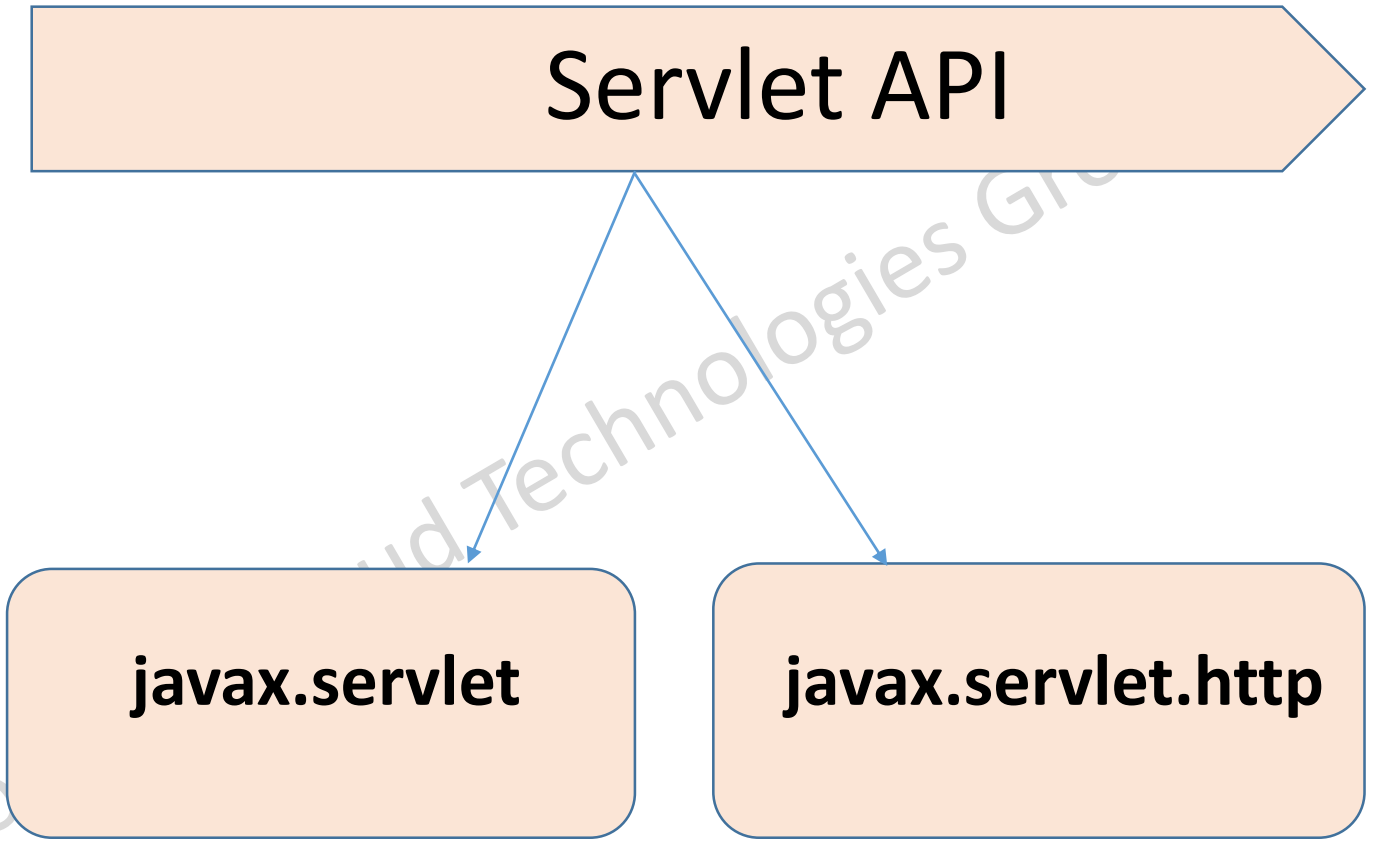


Get vs. Post

GET	POST
1) In case of Get request, only **limited amount of data**	**large amount of data** can be sent.
2) Get request is **not secured** because data is exposed in URL bar.	Post request is **secured** because data is not exposed in URL bar.
3) Get request **can be bookmarked.**	Post request **cannot be bookmarked.**
4) Get request is **more efficient** and used more than Post.	Post request is **less efficient** and used less than get.

Web Server Working

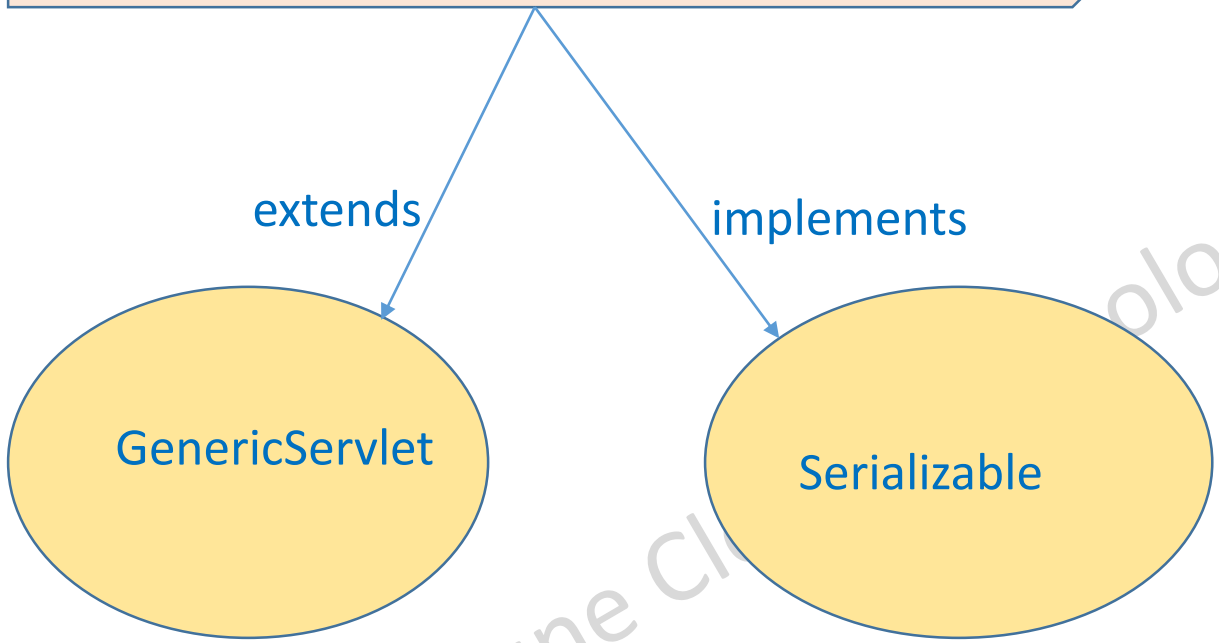




Servlet Interface

Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request,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 ServletConfig getServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

HttpServlet



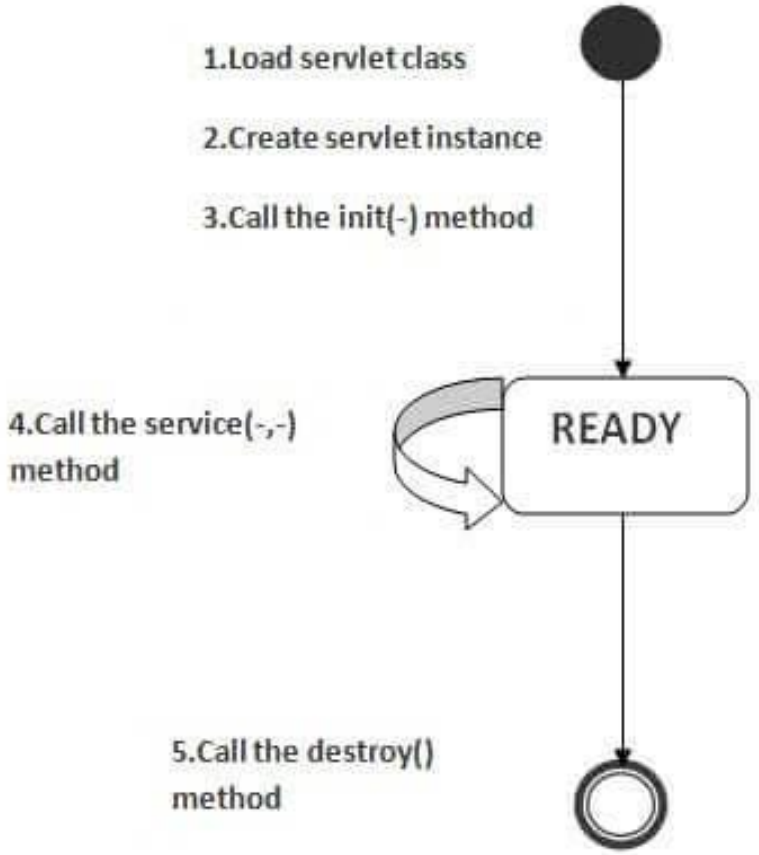
Methods of HttpServlet class:-

- doGet
- doPost
- doHead
- doTrace

Servlet Life Cycle

STEP:-

- ❖Servlet class is loaded.
- ❖Servlet instance is created.
- ❖init method is invoked.
- ❖service method is invoked.
- ❖destroy method is invoked.



Servlet In Eclipse

FortuneCloud
We Understand Technology

1) Create the dynamic web project:

click on File Menu -> New -> Project..-> Web -> dynamic web project -> write your project name e.g. first -> Finish.

2) Create the servlet in eclipse IDE:

explore the project by clicking the + icon -> explore the Java Resources -> right click on src -> New -> servlet -> write your servlet name e.g. Hello -> uncheck all the checkboxes except doGet() -> next -> Finish.

3) add jar file in eclipse IDE:

right click on your project -> Build Path -> Configure Build Path -> click on Libraries tab in Java Build Path -> click on Add External JARs button -> select the servlet-api.jar file under tomcat/lib -> ok.

4) Start the server and deploy the project:

Right click on your project -> Run As -> Run on Server -> choose tomcat server -> next -> addAll -> finish.

ServletRequest Interface

FortuneCloud
We Understand Technology

- An object of ServletRequest is used to provide the client request information to a servlet such as
- content type, content length,
- parameter names and values,
- header informations,
- attributes etc.

Methods of ServletRequest interface:-

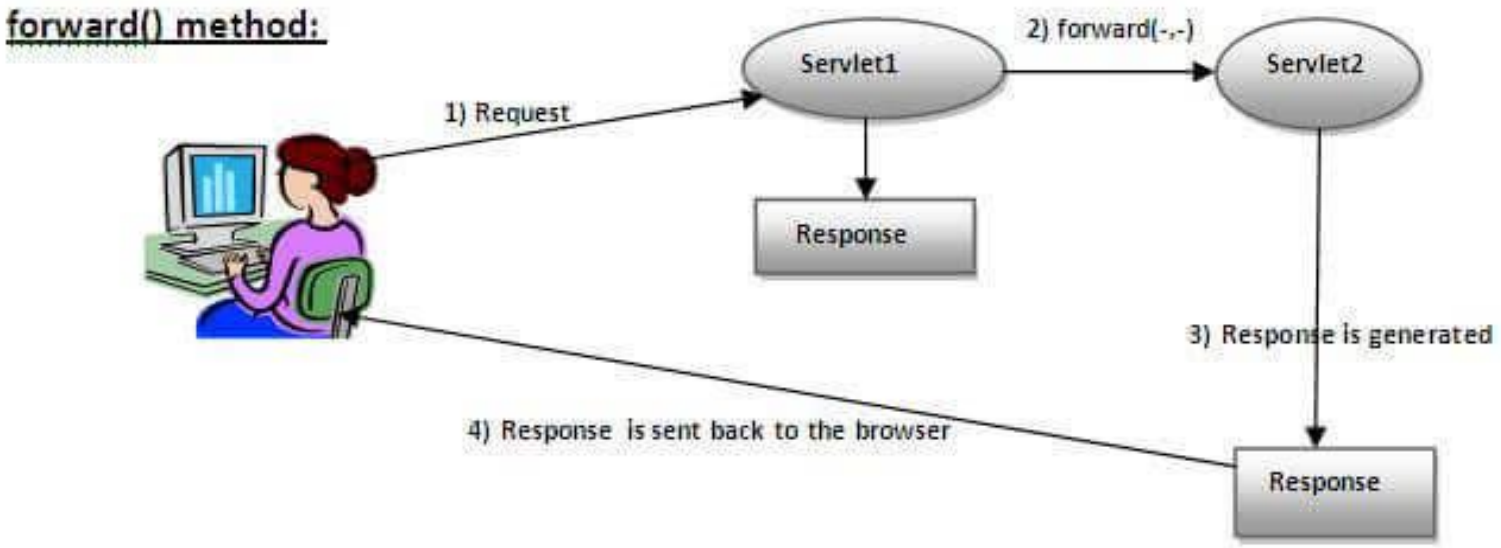
- **public String getParameter(String name)**
- **public int getContentLength()**
- **public String getContentType()**

RequestDispatcher

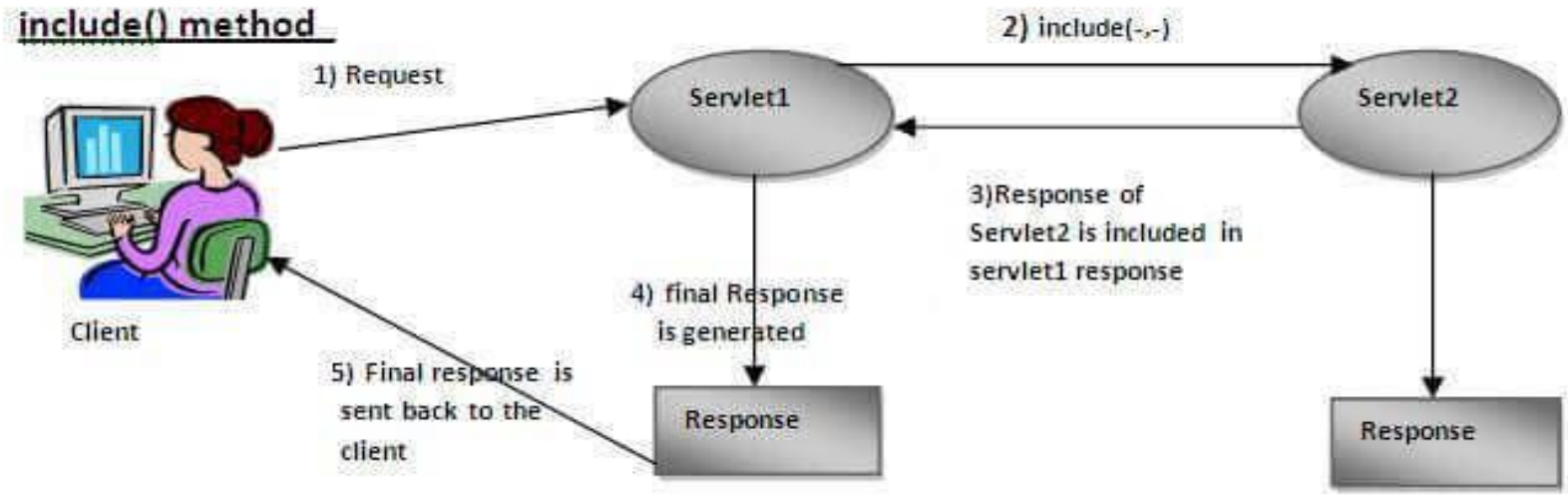
- The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp.

Include()

Forward()



Include method



SendRedirect

- used to redirect response to another resource.
- The sendRedirect() method works at client side.
- It always sends a new request.
- used within and outside the server.

- Example

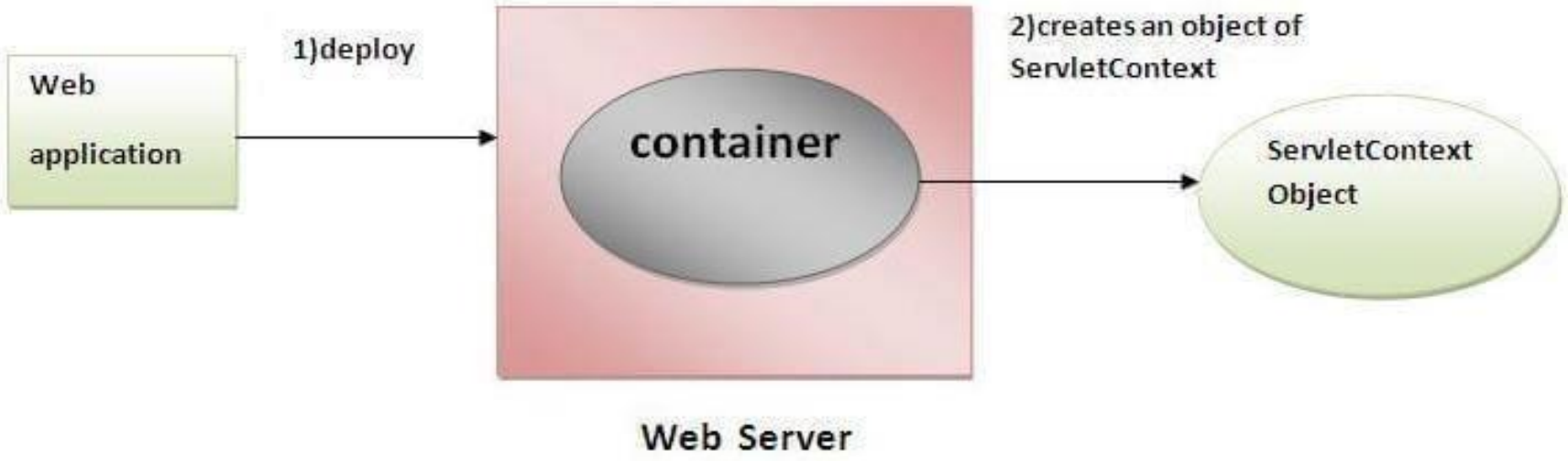
```
response.sendRedirect("http://www.javatpoint.com");
```

forward() vs sendRedirect()

forward() method	SendRedirect() method
The forward() method works at server side.	The sendRedirect() method works at client side.
It sends the same request and response objects to another servlet.	It always sends a new request.
It can work within the server only.	It can be used within and outside the server.
Example: request.getRequestDispacher("servlet2").forward(request,response);	Example: response.sendRedirect("servlet2");

ServletContext Interface

- An object of ServletContext is created by the web container at time of deploying the project.
- provides an interface between the container and servlet.
- used to set, get or remove attribute from the web.xml file.



Attribute in Servlet

FortuneCloud
We Understand Technology

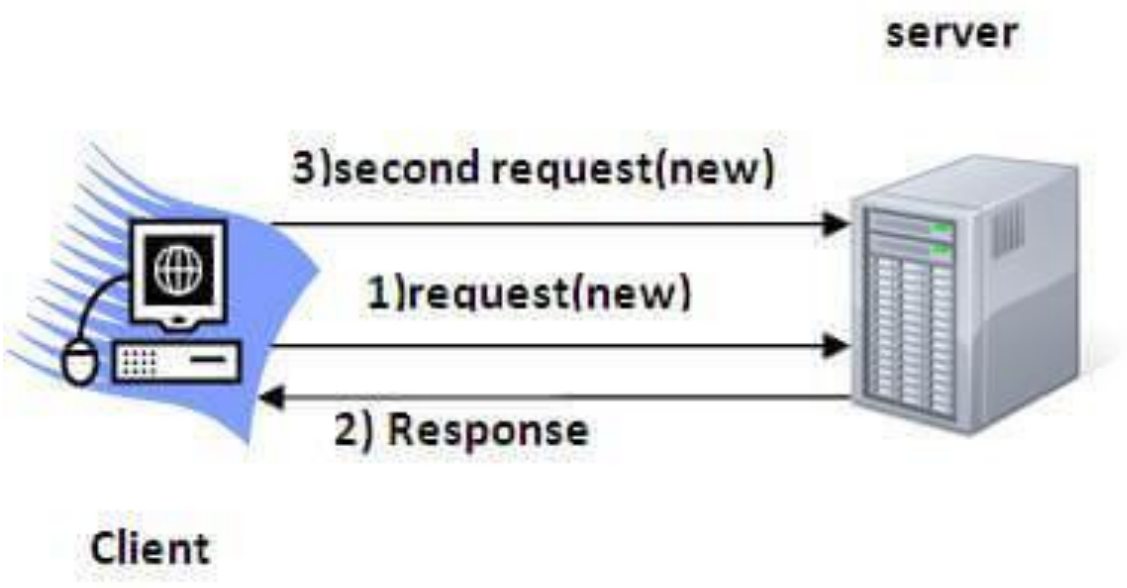
- An **attribute in servlet** is an object that can be set, get or removed from one of the following scopes:-
- request scope
- session scope
- application scope

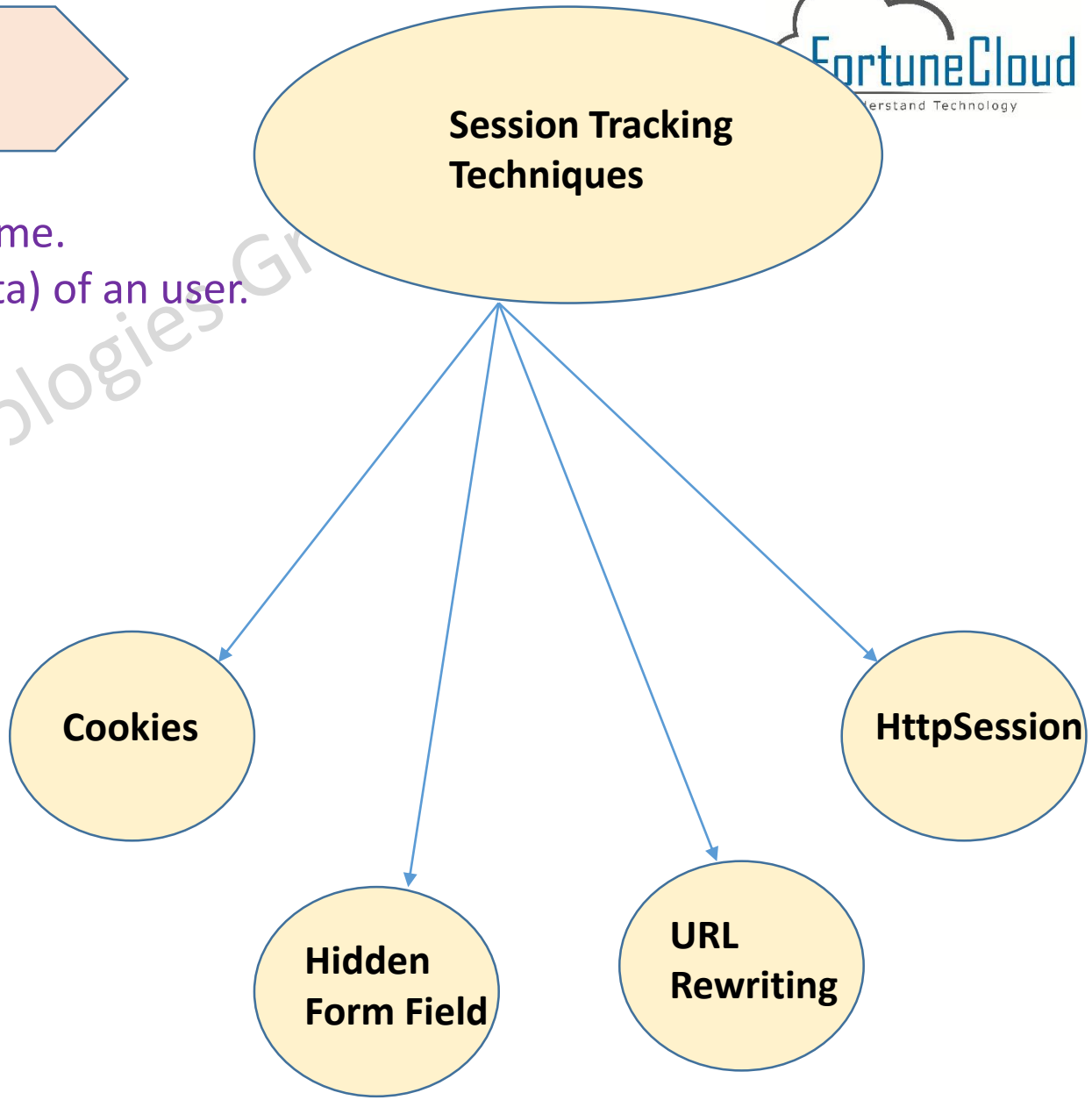
- 4 attribute specific methods:-

- **public void setAttribute(String name,Object object)**
- **public Object getAttribute(String name)**
- **public Enumeration getInitParameterNames()**
- **public void removeAttribute(String name)**

Session Tracking

- **Session** simply means a particular interval of time.
- **Session Tracking** is a way to maintain state (data) of an user.



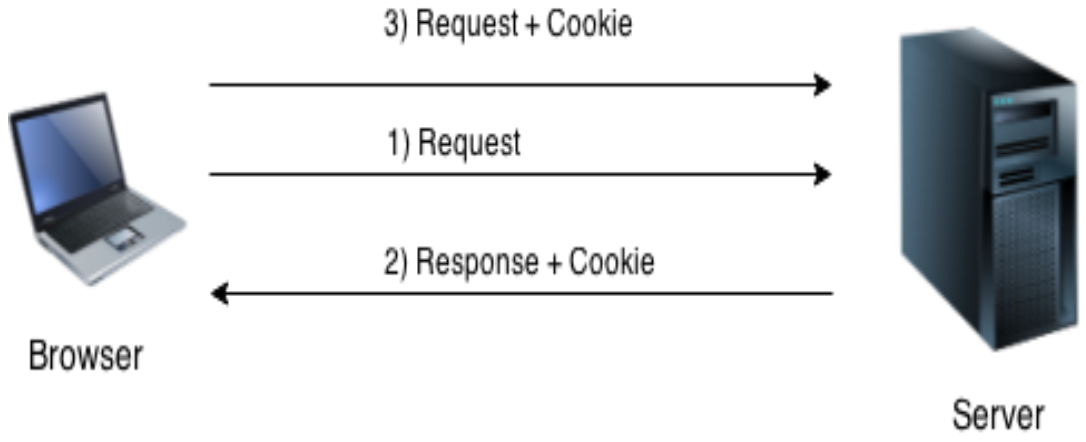


1.Cookies

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

How Cookie works ?

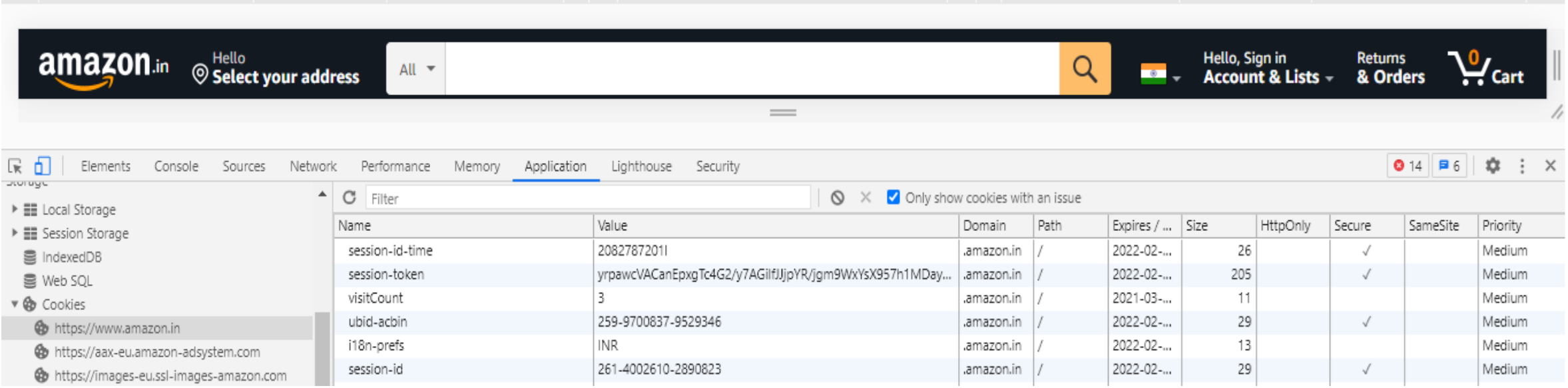
- cookie is **stored in the cache of the browser**.
- After that if request is sent by the user, cookie is added with request by default.
- Thus, we **recognize the user as the old user**.



Methods:-

- **public void addCookie(Cookie ck)**
- **public Cookie[] getCookies()**

Example Cookies

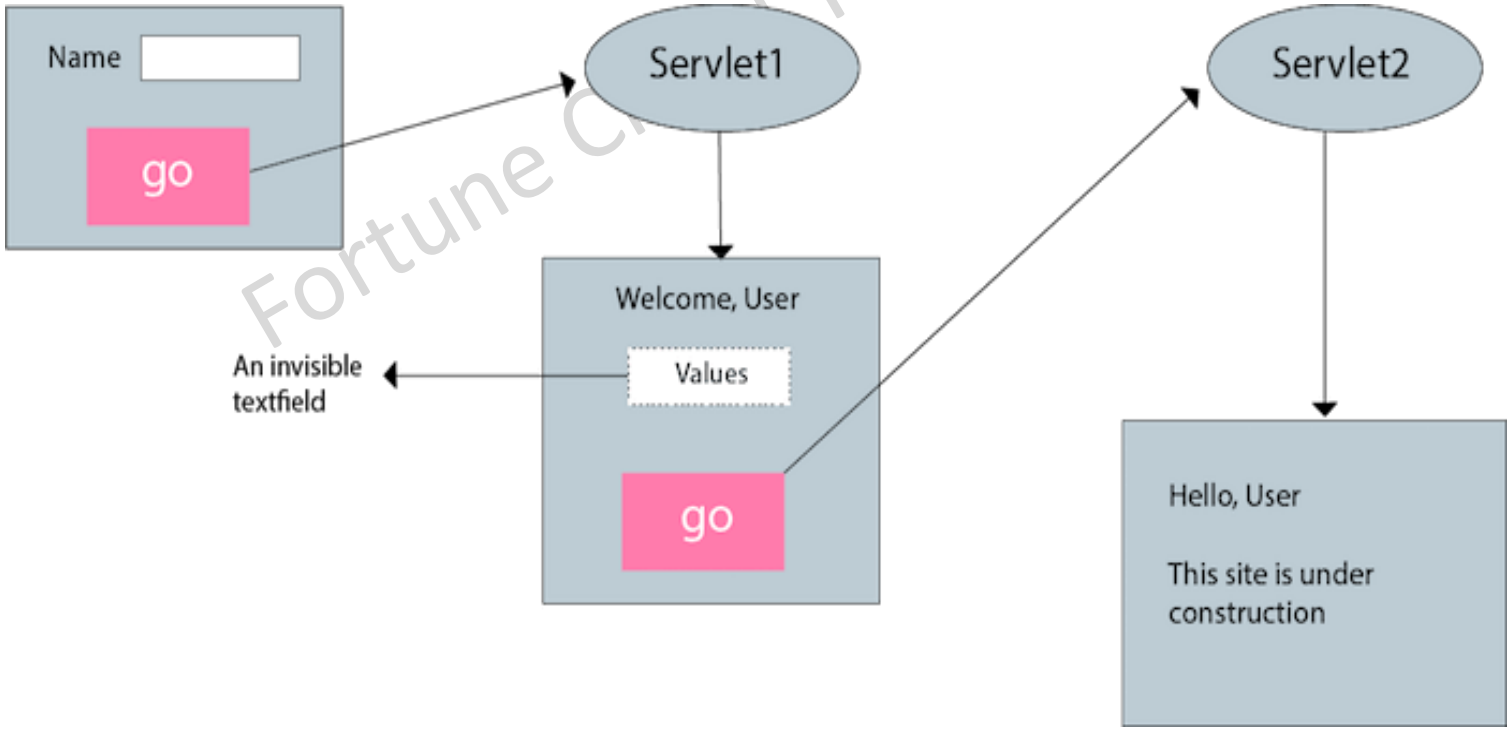


Name	Value	Domain	Path	Expires / ...	Size	HttpOnly	Secure	SameSite	Priority
session-id-time	2082787201l	.amazon.in	/	2022-02-...	26		✓		Medium
session-token	yrpawcVACanEpxgTc4G2/y7AGilfJJjpYR/jgm9WxYsX957h1MDay...	.amazon.in	/	2022-02-...	205		✓		Medium
visitCount	3	.amazon.in	/	2021-03-...	11				Medium
ubid-acbin	259-9700837-9529346	.amazon.in	/	2022-02-...	29		✓		Medium
i18n-prefs	INR	.amazon.in	/	2022-02-...	13				Medium
session-id	261-4002610-2890823	.amazon.in	/	2022-02-...	29		✓		Medium

2.Hidden Form Field

- Hidden Form Field **a hidden (invisible) textfield** is used for maintaining the state of an user.
- we store the information in the hidden field and get it from another servlet.

```
<input type="hidden" name="user" value="JonMichel">
```

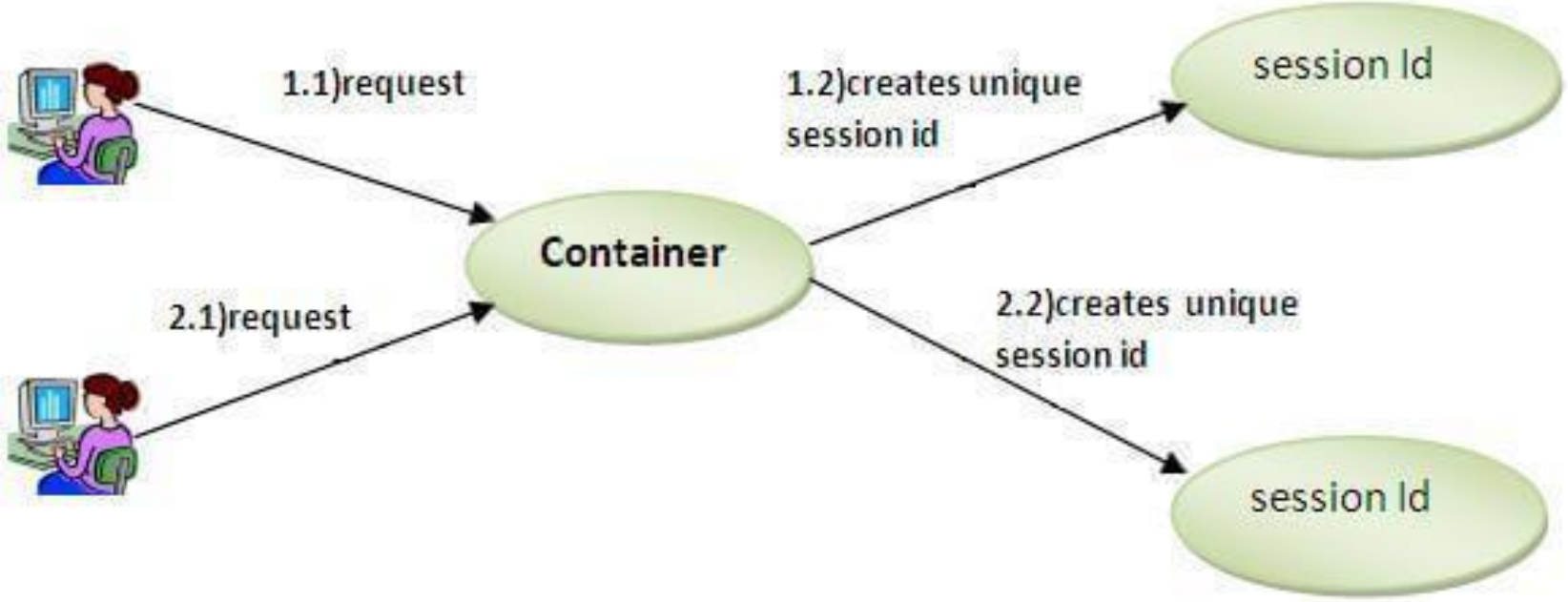


URL Rewriting

- we append a token or identifier to the URL of the next Servlet.
- When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server.
- url?name1=value1&name2=value2&??

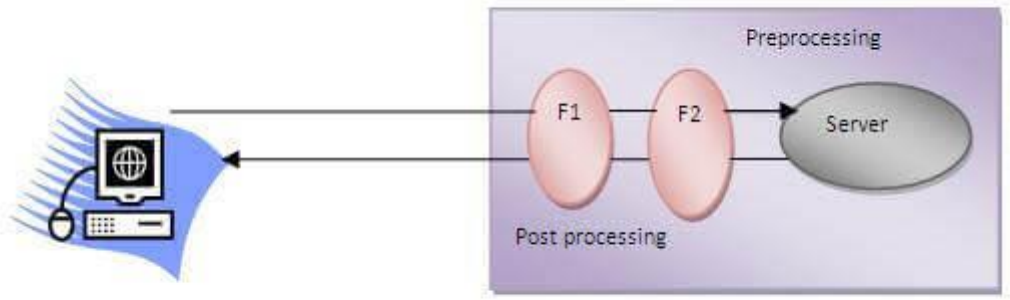
HttpSession interface

- container creates **a session id for each user**.
- uses this id to **identify** the particular user.

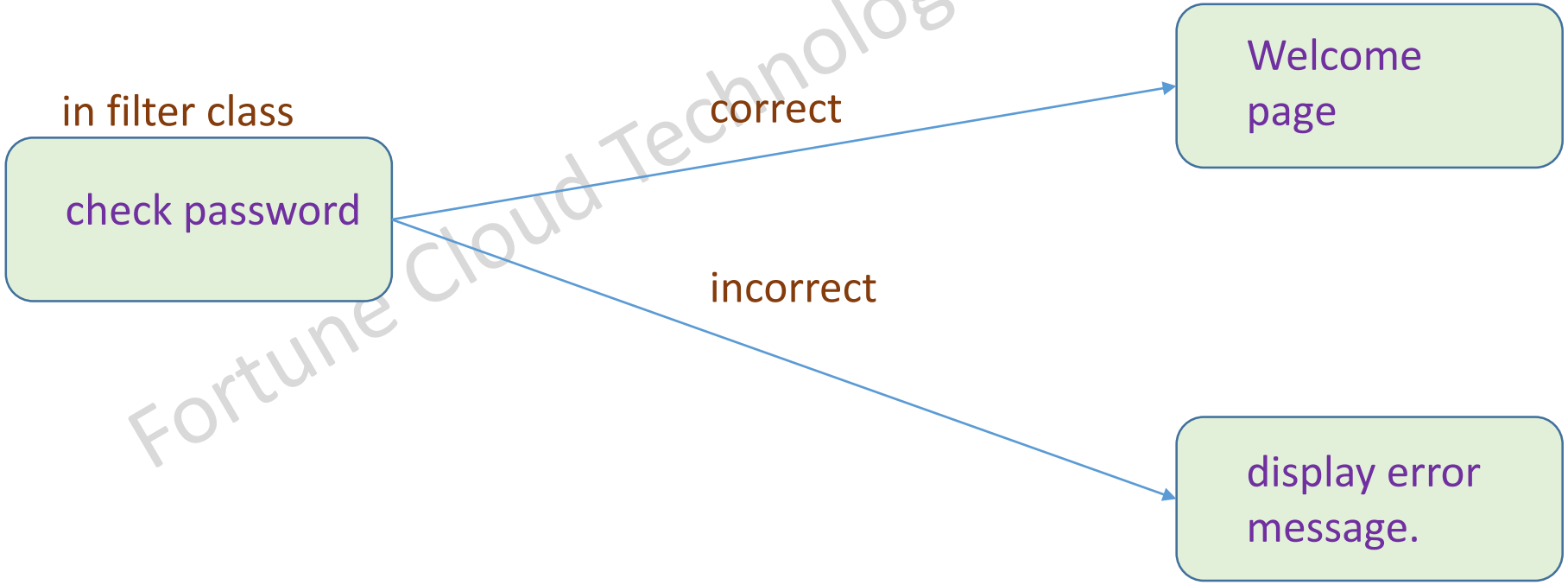


Servlet Filter

- A **filter** is an object that is invoked at the **preprocessing** and **postprocessing** of a request.
- It is mainly used to perform filtering tasks such as **conversion**, **logging**, **compression**, **encryption and decryption, input validation.**
- **if we remove the entry of filter from the web.xml file**, filter will be removed automatically and **we don't need to change the servlet.**



Authentication Filter



Servlet with JDBC

FortuneCloud
We Understand Technology

STEPS:-

- Create table in database
- Create a form in HTML
- In java class dopost method and write JDBC Connection.
- Web.xml inside <url-pattern> path of html page

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

PreparedStatement ps=con.prepareStatement(
"insert into registeruser values(?,?,?,?)");
 ps.setString(1,n);
```

FortuneCloud
We Understand Technology

Thank you

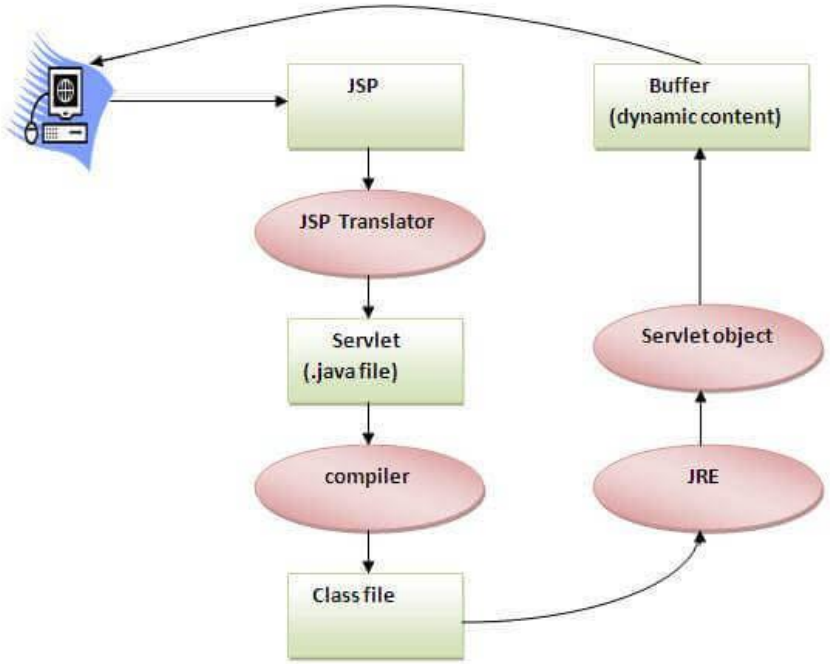
JSP

- JSP technology is used to create web application just like Servlet technology.
- It provides more functionality than servlet such as expression language, JSTL, Custom Tags,etc.
- A JSP page consists of HTML tags and JSP tags.
- We can use all the features of the Servlet in JSP.
- JSP can be easily managed because we can easily separate our business logic with presentation logic.

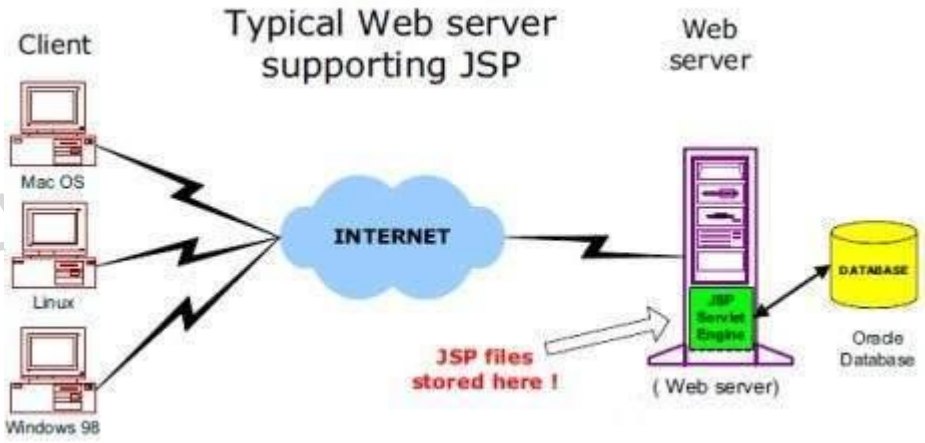
LIFECYCLE JSP

- Translation of JSP Page
- Compilation of JSP Page
- Classloading (the classloader loads class file)
- Instantiation (Object of the Generated Servlet is created).
- Initialization (the container invokes jspInit() method).
- Request processing (the container invokes _jspService() method).
- Destroy (the container invokes jspDestroy() method).

FLOW DIAGRAM



JSP - Architecture



Creating JSP in Eclipse IDE with Tomcat server

- Create the dynamic web project.
- For creating a dynamic web project click on File Menu -> New -> dynamic web project -> write your project name e.g. first -> Finish.
- Create the JSP file in eclipse IDE
- For creating a jsp file explore the project by clicking the + icon -> right click on WebContent -> New -> jsp -> write your jsp file name e.g. index -> next -> Finish.
- Start the server and deploy the project:
- For starting the server and deploying the project in one step Right click on your project -> Run As -> Run on Server -> choose tomcat server -> next -> addAll -> finish.
- Yes, Let's see JSP is successfully running now.

JSP scripting elements

Three Type

1.Scriptlet tag

2.Expression tag

3.Declaration tag

1.JSP scriptlet tag

- A scriptlet tag is used to execute java source code in JSP.
- In this example, we are displaying a welcome message.


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

2.JSP expression tag

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

Example of JSP expression tag

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

3.JSP Declaration Tag

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

Example of JSP declaration tag that declares field & method

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

9 Implicit Objects

Object	Type
1.Out	JspWriter
2.Request	HttpServletRequest
3.Response	HttpServletResponse
4.Config	ServletConfig
5.Application	ServletContext
6.Session	HttpSession
7.pageContext	PageContext
8.Page	Object
9.exception	Throwable

1) JSP out implicit object

- JSP provides an implicit object named out.
- It is the object of JspWriter.

- Example of out implicit object

Index.jsp

```
<html>
<body>
<% out.print("hello") %>
</body>
</html>
```

2. Request object

- object of type HttpServletRequest i.e. created for each jsp request by the web container.
- used to set, get and remove attributes from the jsp request scope.

Example

index.html

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

welcome.jsp

```
<%
String name=request.getParameter
("uname");
out.print("welcome "+name);
%>
```

Output:

```
Welcome Ramesh
```

3.Response implicit object

- Object of type HttpServletResponse.
- The instance of HttpServletResponse is created by the web container for each jsp request.
- It can be used to add or manipulate response such as redirect response to another resource.

Index.html

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

Welcome.jsp

```
<%
response.sendRedirect("http://www.go
ogle.com");
%>
```

4.Config implicit object

- In JSP, config is an implicit object of type *ServletConfig*.
- it is used to get initialization parameter from the web.xml file.
- The config object is created by the web container for each jsp page.

Web.xml

```
<init-param>
<param-name>dname</param-name>
<param-value>Hello Tom</param-value>
</init-param>
```

Welcome.jsp

```
String s= config.getInitParameter
("dname");
out.print("driver name is="+s);
```

output

```
Driver name=Hello Tom
```

5.Application implicit object

- application is an implicit object of type *ServletContext*.
- This object can be used to get initialization parameter from configuaration file (web.xml).
- It can also be used to get, set or remove attribute from the application scope.

Web.xml

```
<context-param>
<param-name>dname</param-name>
<param-value>Hello Tom</param-value>
</context-param>
```

Welcome.jsp

```
<% String s= config.getInitParameter
("dname");
out.print("driver name is="+s);
%>
```

output

```
Driver name=Hello Tom
```

6) session implicit object

- session is an implicit object of type HttpSession.
- The Java developer can use this object to set,get or remove attribute or to get session information.

Index.html

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

Session.jsp

```
<%
 session.setAttribute("user",name);

String name=(String)session.getAttribute("user")
;
out.print("Hello "+name);
%>
```

Output:-

```
Hello Raj
```

7. pageContext

- pageContext is an implicit object of type PageContext class.
- The pageContext object can be used to set,get or remove attribute.

Index.html

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

Session.jsp

```
<%
 pageContext.setAttribute("user",name,PageCon
text.SESSION_SCOPE);

String name=(String)pageContext.getAttribute("
user",PageContext.SESSION_SCOPE);
out.print("Hello "+name);
%>
```

Output:-

```
Hello Raj
```

8. page

Example:- Object page=this;

- page is an implicit object of type Object class.
- This object is assigned to the reference of auto generated servlet class.

For using this object it must be cast to Servlet type.For example:

```
<% (HttpServlet)page.log("message"); %>
```

For using this object it must be cast to Servlet type.For example:

```
<% this.log("message"); %>
```

9.Exception

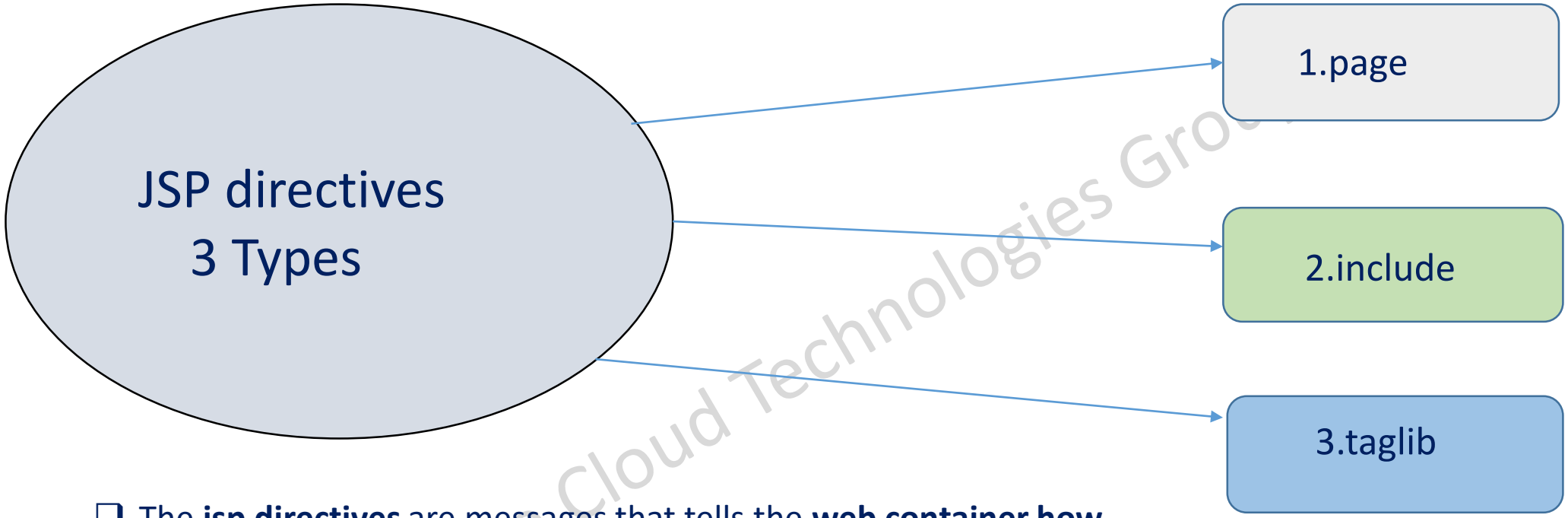
- exception is an implicit object of type java.lang.Throwable class.
- This object can be used to print the exception. But it can only be used in error pages.

Error.jsp

```
<%@ page isErrorPage="true" %>
<html>
<body>

Sorry following exception occured:
<%= exception %>

</body>
</html>
```



- The **jsp directives** are messages that tells the **web container how to translate a JSP page into the corresponding servlet.**

Syntax:-

```
<%@ directive attribute="value" %>
```

1.page

- The page directive defines attributes that apply to an entire JSP page.

Syntax:-

```
<%@ page attribute="value" %>
```

- Attributes of JSP page directive:-

1.import	2.contentType	3.extends
4.info	5.buffer	6.language
7.isELIgnored	8.isThreadSafe	9.errorPage
10.isErrorPage	11.session	12. autoFlush

2. Include Directive

- It is used to include the contents of any resource it may be jsp file, html file or text file.
- The jsp page is translated only once so it will be better to include static resource.

- Syntax of include directive:-

```
<%@ include file="resourceName" %>
```

3. Taglib Directive

- The JSP taglib directive is used to define a tag library that defines many tags.
- We use the TLD (Tag Library Descriptor) file to define the tags.

- Example JSP Taglib directive:-

```
<%@ taglib uri="http://www.javatpoint.com/tags" prefix="mytag" %>
```

Exception Handling in JSP

- The exception is normally an object that is thrown at runtime. Exception Handling is the process to handle the runtime errors.
- In JSP, there are two ways to perform exception handling:
- By **errorPage** and **isErrorPage** attributes of page directive
- By **<error-page>** element in web.xml file

index.jsp

```
<form action="process.jsp">
No1:<input type="text" name="n1" />
No1:<input type="text" name="n2" />
<input type="submit" value="divide"/>
</form>
```

error.jsp

```
<%@ page isErrorPage="true" %>

<h3>Sorry an exception occured!</h3>

Exception is: <%= exception %>
```

process.jsp

```
<%@ page errorPage="error.jsp" %>
<%

String num1=request.getParameter("n1");
String num2=request.getParameter("n2");

int a=Integer.parseInt(num1);
int b=Integer.parseInt(num2);
int c=a/b;
out.print("division of numbers is: "+c); %>
```

JSP Action Tags

JSP Action Tags

JSP Action Tags	Description
jsp:forward	forwards the request and response to another resource.
jsp:include	includes another resource.
jsp:useBean	creates or locates bean object.
jsp:setProperty	sets the value of property in bean object.
jsp:getProperty	prints the value of property of the bean.
jsp:plugin	embeds another components such as applet.
jsp:param	sets the parameter value. It is used in forward and include mostly.
jsp:fallback	can be used to print the message if plugin is working. It is used in jsp:plugin.

Expression Language (EL) in JSP

- The **Expression Language** (EL) simplifies the accessibility **of data stored in the Java Bean component, and other objects like request, session, application etc.**

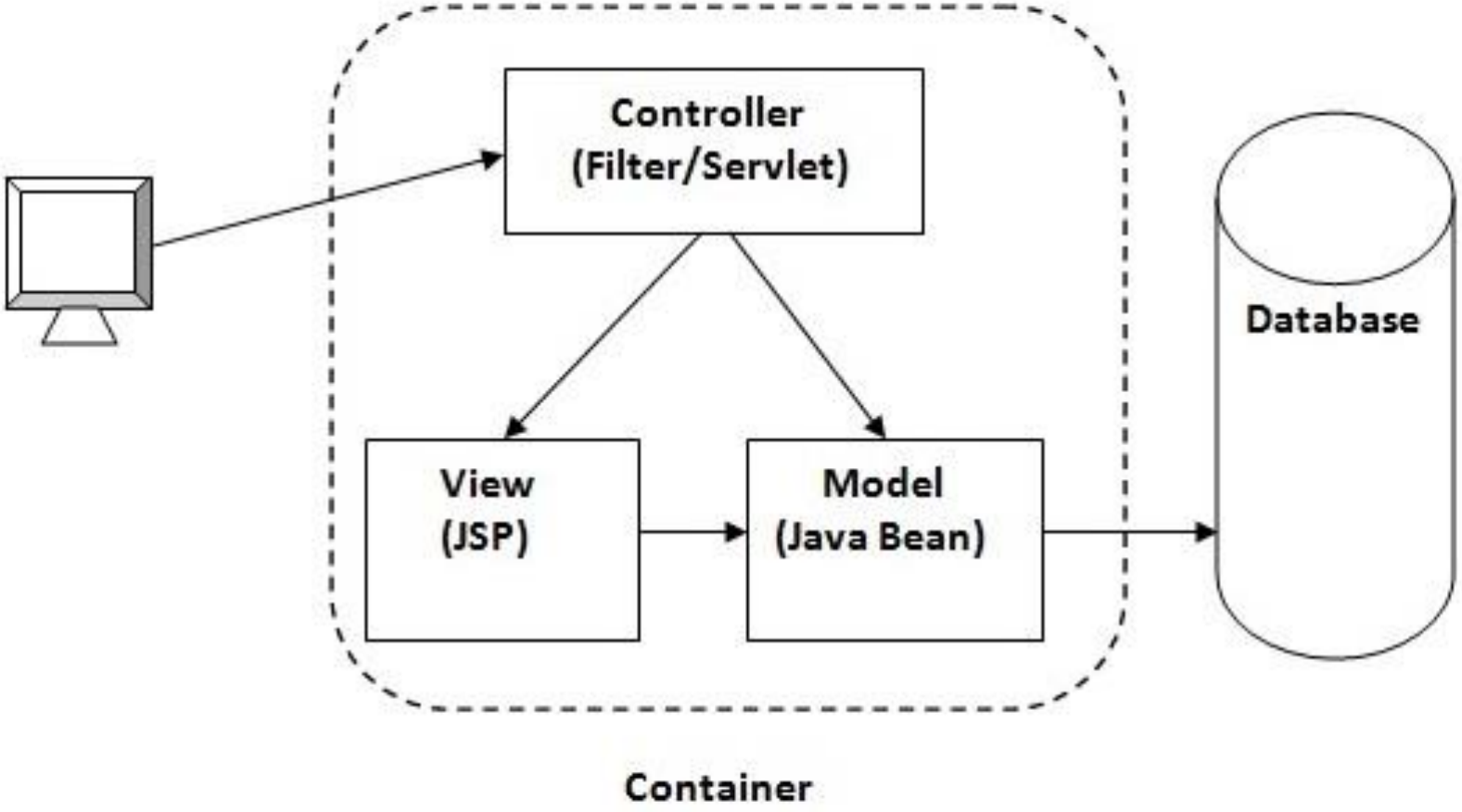
- Syntax

```
${ expression }
```

MVC IN JSP

- **MVC** stands for Model View and Controller. It is a **design pattern** that separates the business logic, presentation logic and data.
- **Controller** acts as an interface between View and Model. Controller intercepts all the incoming requests.
- **Model** represents the state of the application i.e. data. It can also have business logic.
- **View** represents the presentation i.e. UI(User Interface).

MVC ARCHITECTURE



FortuneCloud
We Understand Technology

Thank you

SPRING

Spring Introduction

FortuneCloud
We Understand Technology

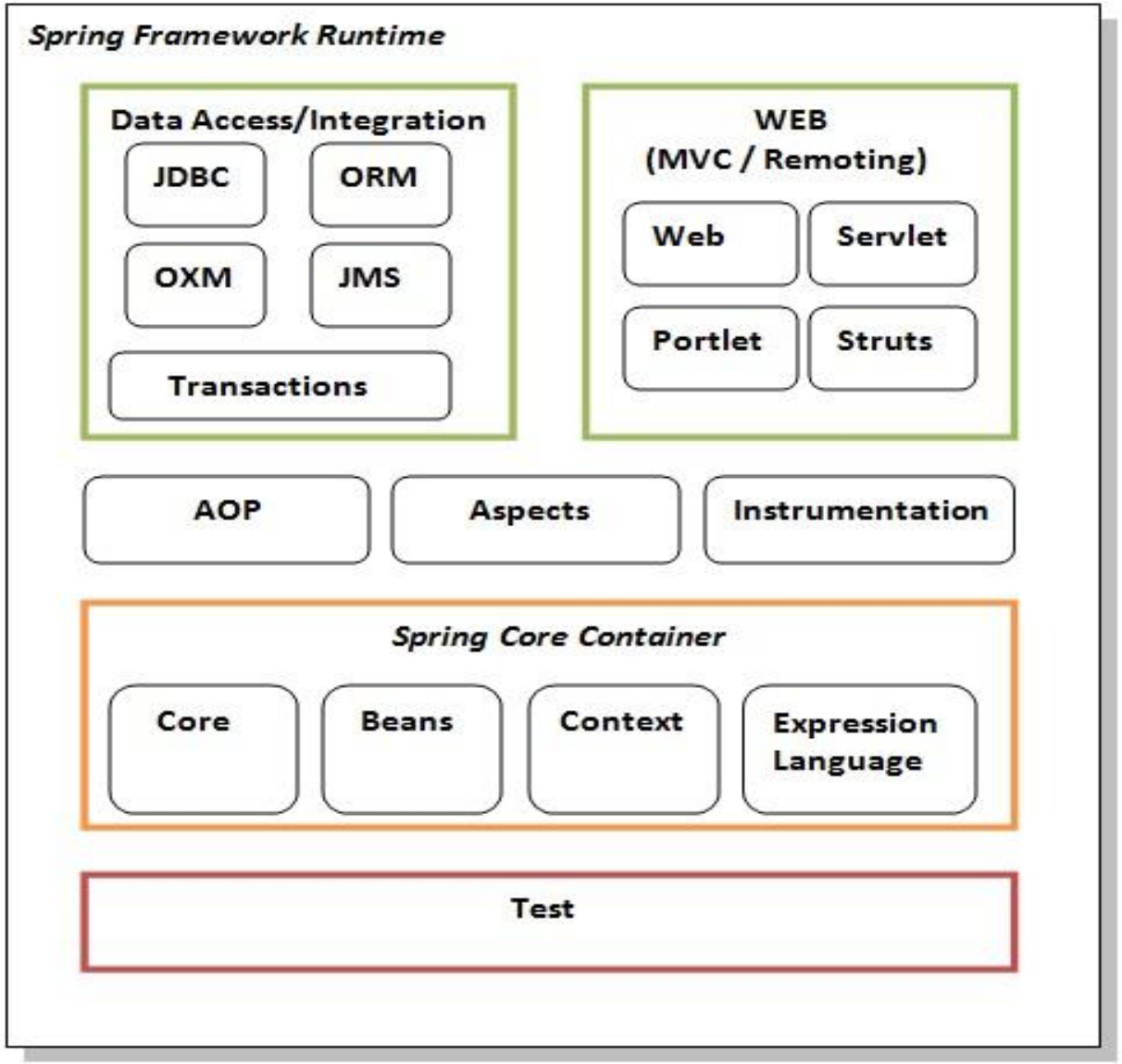
- Spring framework is an open source Java platform
- Rod Johnson and was first released under the Apache 2.0 license in June 2003.
- Spring framework makes the easy development of JavaEE application.
- Spring is the most popular application development framework for enterprise Java.
- Millions of developers around the world use Spring Framework to create high performing, easily testable, and reusable code.

WHY SPRING?

FortuneCloud
We Understand Technology

- Modern Java-based enterprise applications.
- Spring's web framework is a well-designed web MVC framework.
- Spring delivers delightful experiences to millions of end-users
- Spring also has contributions from all the big names in tech, including **Alibaba**, **Amazon**, **Google**, **Microsoft**, and more.
- **Spring Security** makes it easier for you to integrate with industry-standard security schemes

Spring Modules



Creating spring application in Eclipse IDE

- 1) Create the Java Project.
- 2) Add spring jar files
- 3) Create Java class
- 4) Create the xml file
- 5) Create the Main method class

IoC Container

- The IoC container is responsible to **instantiate**, **configure** and **assemble** the objects.
- The IoC container gets information's from the XML file and works accordingly.
- There are two types of IoC containers.

  - **BeanFactory**
  - **ApplicationContext**

❑ BeanFactory

```
Resource resource=new
ClassPathResource("applicationContext.xml");
 BeanFactory factory=new XmlBeanFactory(resource);
 student s1 = (student) factory.getBean("demo");
s1.displayInfo();
```

❑ ApplicationContext

```
ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

student s1 = (student) context.getBean("demo");
s1.displayInfo();
```

Dependency Injection

- Dependency Injection (DI) is a design pattern that removes the dependency from the programming code so that it can be easy to manage and test the application.
- Dependency Injection makes our programming code loosely coupled.

  - Two ways to perform Dependency Injection
    - By Constructor
    - By Setter method

Example

```
<!-- BY DI CONSTRUCTOR -->
  <!-- <bean id="demo" class="com.student">
  <constructor-arg value="Tom" type="String"></constructor-arg>
  <constructor-arg value="101" type="int"></constructor-arg>
  </bean>
   -->
```

```
<!-- BY DI setter method -->
  <bean id="demo" class="com.student">
  <property name="id"> <value>188</value>  </property>
   <property name="name"> <value>TOMMY</value>
</property>
  </bean>
```

Setter Injection with Collection

FortuneCloud
We Understand Technology

- We can inject collection values by **setter** method in spring framework.
- There can be used three elements inside the **property** element.

Type:-

- **list**
- **set**
- **map**

Autowiring in Spring

FortuneCloud
We Understand Technology

- Autowiring feature of spring framework enables you to inject the object dependency implicitly.
- It internally uses setter or constructor injection.
- Spring provides a way to automatically detect the relationships between various beans.
- This can be done by declaring all the bean dependencies in Spring configuration file.

Autowiring Modes

No.	Mode	Description
1)	no	It is the default autowiring mode.
2)	byName	In such case, property name and bean name must be same. It internally calls setter method.
3)	byType	So property name and bean name can be different. It internally calls setter method.
4)	constructor	The constructor mode injects the dependency by calling the constructor of the class.

@Autowired Annotation

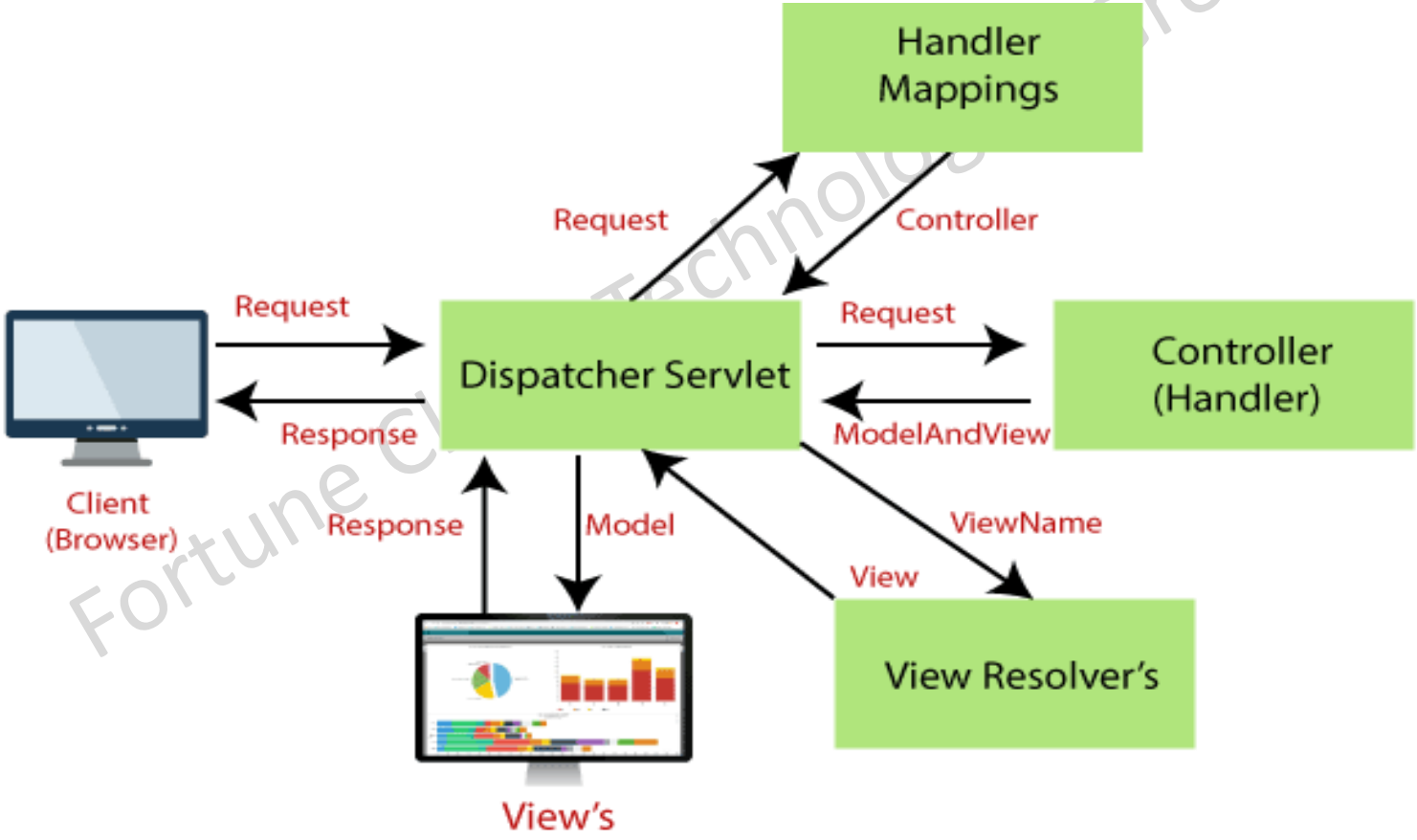
FortuneCloud
We Understand Technology

- @Autowired annotation to auto wire bean on the setter method, constructor or a field
- We must first enable the annotation using below configuration in configuration file. <context:annotation-config />

MVC

- A Spring MVC is a Java framework which is used to build web applications.
- It follows the Model-View-Controller design pattern.
- **DispatcherServlet** is a class that receives the incoming request and maps it to the right resource such as controllers, models, and views.
- The Spring MVC facilitates fast and parallel development.

Flow Diagram



RequestParam Annotation

- **@RequestParam** annotation is used to read the form data and bind it automatically to the parameter present in the provided method.
- The @RequestParam is used to read the HTML form data provided by a user and bind it to the request parameter.
- The Model contains the request data and provides it to view page.

JdbcTemplate

- Spring **JdbcTemplate** is a powerful mechanism to connect to the database and execute SQL queries
- JdbcTemplate class such as insertion, updating, deletion and retrieval of the data from the database.


```
public class EmployeeDao {
private JdbcTemplate jdbcTemplate;

public void setJdbcTemplate(JdbcTemplate jdbcTem
plate) {
   this.jdbcTemplate = jdbcTemplate;
}
```

Methods

No.	Method	Description
1)	public int update(String query)	is used to insert, update and delete records.
2)	public int update(String query,Object... args)	is used to insert, update and delete records using PreparedStatement using given arguments.
3)	public void execute(String query)	is used to execute DDL query.
4)	public T execute(String sql, PreparedStatementCallback action)	executes the query by using PreparedStatement callback.
5)	public T query(String sql, ResultSetExtractor rse)	is used to fetch records using ResultSetExtractor.
6)	public List query(String sql, RowMapper rse)	is used to fetch records using RowMapper.

FortuneCloud
We Understand Technology

Thank you