Java Capability

Java Server Page – JSP

AGENDA

- Java Server Pages (JSP) is a server-side programming technology that enables in creating dynamic web page for building java based Web applications.

PREREQUISITES

- Mandatory knowledge on Java
- HTML , CSS , Java Script , XML
- Knowledge on web applications
- Web server / Application server

INTRODUCTION

- Java Server Pages (JSP) is a server-side programming technology that enables for building dynamic Web-based applications.
- JSP has Java inside HTML pages.
- ".jsp“ would be the file extension
- Java Server page uses special JSP tags most of which start with <% and end with %>.
- A Java Server Pages component is a type of Java servlet.
- It can be thought of as an extension to servlet
- A JSP page consists of HTML tags and JSP tags

GETTING FAMILIAR WITH YOUR JSP SERVER

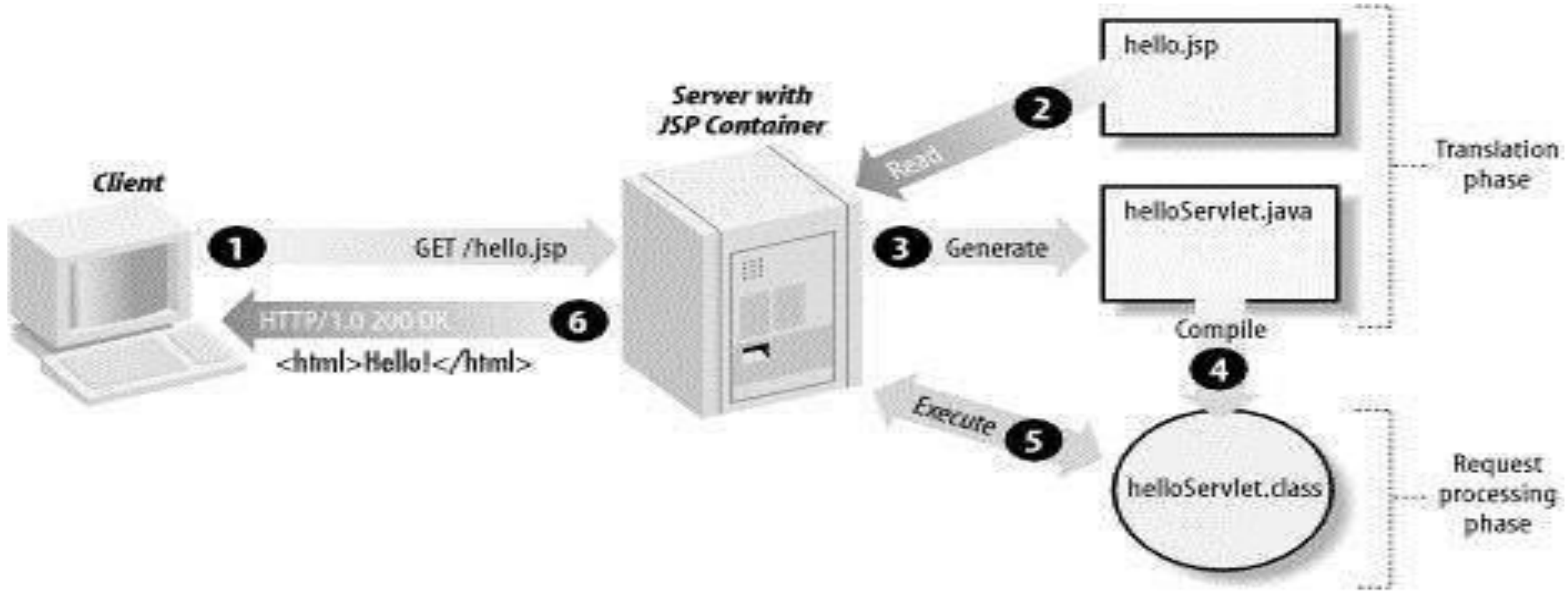
- JSP capable web-server or application server.
  - Apache tomcat
  - Oracle WebLogic
  - IBM WebSphere
  - JBose
  - Any JEE web server / Application server

JSP - ENVIRONMENT SETUP

- JDK
  - set the PATH and JAVA_HOME environment variables
- Latest version of Tomcat from https://tomcat.apache.org/
- Latest version of JEE eclipse http://www.eclipse.org/downloads/eclipse-packages/

JSP - ARCHITECTURE

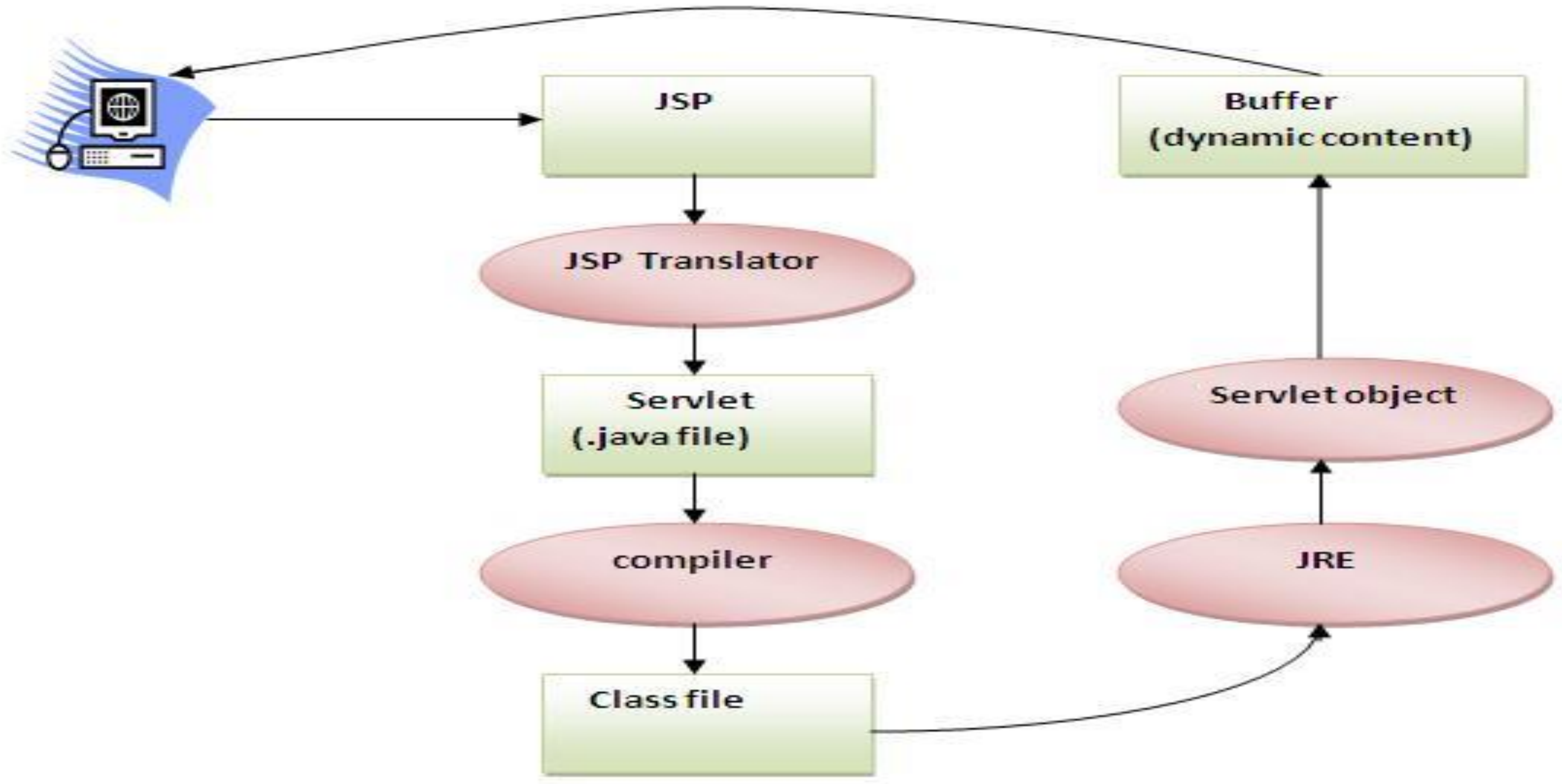
- The web server needs a JSP engine, i.e., a container to process JSP pages.
- The JSP container is responsible for intercepting requests for JSP pages.
- A JSP container works with the Web server to provide the runtime environment and other services a JSP needs



JSP PROCESSING

- Browser/client sends an HTTP request to the web server.(http://.......)
- Web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine.
- The JSP engine loads the JSP page and converts it into a servlet content.
- The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
- Web server calls the servlet engine loads the Servlet class and executes it as an output in HTML format.
- The web server forwards the HTTP response to your browser in terms of static HTML content.

LIFE CYCLE OF A JSP PAGE



JSP PROCESSING CONTD..

- Typically, the JSP engine checks to see whether a servlet for a JSP file already exists and whether the modification date on the JSP is older than the servlet.
- If the JSP is older than its generated servlet, the JSP container assumes that the JSP hasn't changed and that the generated servlet still matches the JSP's contents.

LIFE CYCLE OF A JSP PAGE

- Translation of JSP Page
- Compilation of JSP Page
- Class loading (class file is loaded by the classloader)
- Instantiation (Object of the Generated Servlet is created).
- Initialization (jspInit() method is invoked by the container).
- Request processing (_jspService() method is invoked by the container).
- Destroy (jspDestroy() method is invoked by the container).

Note :

jspInit(), _jspService() and jspDestroy() are the life cycle methods of JSP. The underscore _ signifies that you cannot override this method.

CREATING A SIMPLE JSP PAGE

- <Filename>.jsp

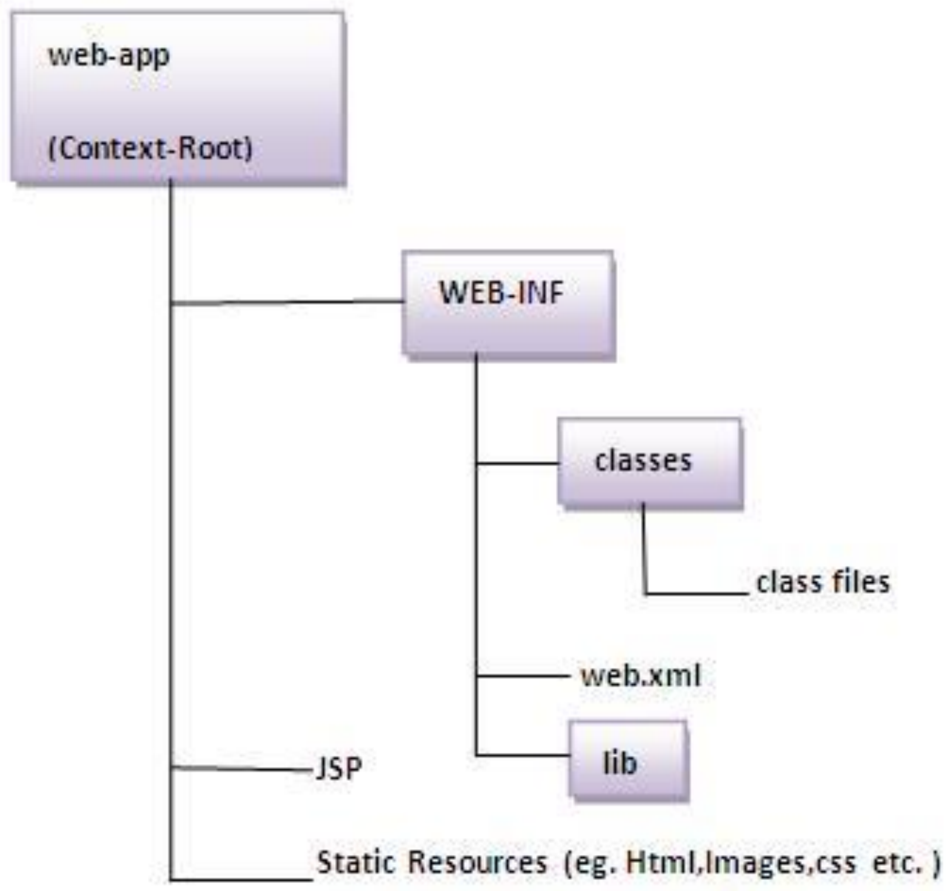
```
    <html>
      <body>
          <% out.print("Welcome to JSP "); %>   <br>
          The time is now <%= new java.util.Date( ) %>
      </body>
  </html>
```

- Start the server
- Place the jsp file in respective folder and deploy on the server
- visit the browser by the url http://localhost:portno/contextRoot/jspfile

  e.g. http://localhost:8080/myapplication/index.jsp

DIRECTORY STRUCTURE OF JSP

- Since JEE is a specification we need to follow the directory structure



JSP IMPLICIT OBJECTS

- These objects are created by the web container that are available to all the jsp pages.

Object	Type
out	JspWriter
request	HttpServletRequest
response	HttpServletResponse
config	ServletConfig
application	ServletContext
session	HttpSession
pageContext	PageContext
page	Object
exception	Throwable

ELEMENTS OF JSP

- JSP Scripting elements
  - scriptlet tag
  - expression tag
  - declaration tag
- JSP Directives
  - page directive
  - include directive
  - taglib directive
- JSP Actions
  - jsp:include jsp:useBean jsp:setProperty
  - jsp:forward jsp:plugin jsp:getProperty
  - jsp:param

JSP SCRIPTING ELEMENTS

- The scripting elements provides the ability to insert java code inside the jsp.
- There are three types of scripting elements:
  - scriptlet tag
  - expression tag
  - declaration tag

<u>JSP scriptlet tag</u>

- A scriptlet tag is used to execute java source code in JSP. Syntax

```
<%  java source code %>        equivalent to XML      <jsp:scriptlet>
```

Ex :

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

JSP SCRIPTING ELEMENTS CONTD..

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.

Syntax : **<%=** statement %**>** equivalent to XML < jsp:declaration>

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

Do not end your statement with semicolon (;) in case of expression tag

JSP SCRIPTING ELEMENTS CONTD..

JSP Declaration Tag

- The JSP declaration tag is used to declare fields and methods.
- The code written inside the jsp declaration tag is placed outside the service( ) method of auto generated servlet.
- it doesn't get memory at each request.
- Syntax <%! field or method declaration %> equivalent to XML <jsp:expression>

Ex :
```
<html>
    <body>
        <%! int data=500; %>
         <%= "Value of the variable is:“ + data %>
     </body>
  </html>
```

JSP COMMENTS

- JSP comment marks text or statements that the JSP container should ignore.

Syntax

```
<%-- This is JSP comment --%>
```

Note :

`<!-- comment -->` An HTML comment. Ignored by the browser.

JSP DIRECTIVES

- JSP directive affects the overall structure of the servlet class

1. <%@ page ... %>
   - Defines page-dependent attributes, such as scripting language, error page, and buffering requirements.
2. <%@ include ... %>
   - Includes a file during the translation phase.
3. <%@ taglib ... %>
   - Declares a tag library, containing custom actions, used in the page

JSP PAGE DIRECTIVE

• Import	contentType	extends
• Info	buffer	language
• *isELIgnored	isThreadSafe	autoFlush
• Session	pageEncoding	errorPage
• isErrorPage		

* Expression Language (EL)

JSP PAGE DIRECTIVE

- myerrorpage.jsp

```
<html>
  <body>
    <%@ page isErrorPage="true" %>
       Sorry an exception occurred!  <%= exception %>
   </body>
</html>
```

- index.jsp

```
<html>
  <body>
    <%@ page errorPage="myerrorpage.jsp" %>
        <%= 100/0 %>
   </body>
</html>
```

JSP INCLUDE DIRECTIVE

- The include directive is used to include the contents of any resource it may be jsp file, html file or text file.
- The include directive includes the original content of the included resource at page translation time (the jsp page is translated only once so it will be better to include static resource).

Syntax :   <%@ include file="resourceName" %>

Ex   <%@ include file="header.html" %>

JSP TAGLIB DIRECTIVE

- JSP taglib directive is used to define custom tags
- Syntax <%@ taglib uri="uriOfTheTagLibrary" prefix="prefixOfTagLibrary" %>
- We use the TLD (Tag Library Descriptor) file to define the tags. (jstl-1.2.jar to include)

Ex :

```
<html>
  <body>
     <%@ taglib uri="http://www.hcltraining.com/tags" prefix="mytag" %>
      <mytag:currentDate/>           // Defined in TLD file
   </body>
</html>
```

JSP ACTIONS

- The action tags are used to control the flow between pages and to use Java Bean.

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.

DIFFERENCE BETWEEN JSP INCLUDE DIRECTIVE AND INCLUDE ACTION

JSP include directive	JSP include action
includes resource at translation time.	includes resource at request time.
better for static pages.	better for dynamic pages.
includes the original content in the generated servlet.	calls the include method.
<%@ include file="resourceName" %>	<jsp:include> </jsp:include>

JAVA BEAN

- A Java Bean is a java class that should follow following conventions:
  - It should have a no-arg constructor.
  - It should be Serializable.
  - It should provide methods to set and get the values of the properties, known as getter and setter methods.
- Ex :

```
public class Employee implements java.io.Serializable{
    private int id;
    private String name;
    public Employee( ){ }
    // generate getter and setter
```

JSP:USEBEAN ACTION TAG

- The jsp:useBean action tag is used to locate or instantiate a bean class.
- If bean object of the Bean class is already created, it doesn't create the bean depending on the scope. But if object of bean is not created, it instantiates the bean.

```
<jsp:useBean id= "instanceName" scope= "page | request | session | application"
     class= "packageName.className" type= "packageName.className"
      beanName="packageName.className | <%= expression >" >
</jsp:useBean>
```

JSP:SETPROPERTY AND JSP:GETPROPERTY ACTION TAGS

- The setProperty and getProperty action tags are used for developing web application with Java Bean.
- The jsp:setProperty action tag sets a property value or values in a bean using the setter method.

```
<jsp:setProperty name="instanceOfBean" property= "*"   |
     property="propertyName" param="parameterName"  |
    property="propertyName" value="{ string | <%= expression %>}"
/>
```

Ex :<jsp:setProperty name=“employee" property=“name" value=“HCL" />

JSP:GETPROPERTY ACTION TAG

- The jsp:getProperty action tag returns the value of the property.

Syntax :

```
<jsp:getProperty name="instanceOfBean" property="propertyName" />
```

Ex :

```
<jsp:getProperty name=“employee" property="name" />
```

THE <JSP:FORWARD> ACTION

- The forward action terminates the action of the current page and forwards the request to another resource such as a static page, another JSP page, or a Java Servlet.

Syntax : <jsp:forward page = "Relative URL" />

Note : page Should consist of a relative URL of another resource such as a static page, another JSP page, or a Java Servlet.

CLASSIFICATION OF THE JSTL TAGS

The JSTL tags can be classified, according to their functions, into the following JSTL tag library groups that can be used when creating a JSP page −

- **Core Tags -** most used JSTL tags
- **Formatting tags -** to format and display text, the date, the time, and numbers for internationalized Websites
- **SQL tags** for interacting with relational databases
- **XML tags -** creating and manipulating the XML documents
- **JSTL Functions -** are common string manipulation functions

JSP <-> SERVLET <-> HTML

- SERVLET to JSP
- JSP to JSP
- JSP to SERVLET
- SERVLET to SERVLET

Flow from JSP to servlet or viceversa or JSP to JSP or servlet to servlet

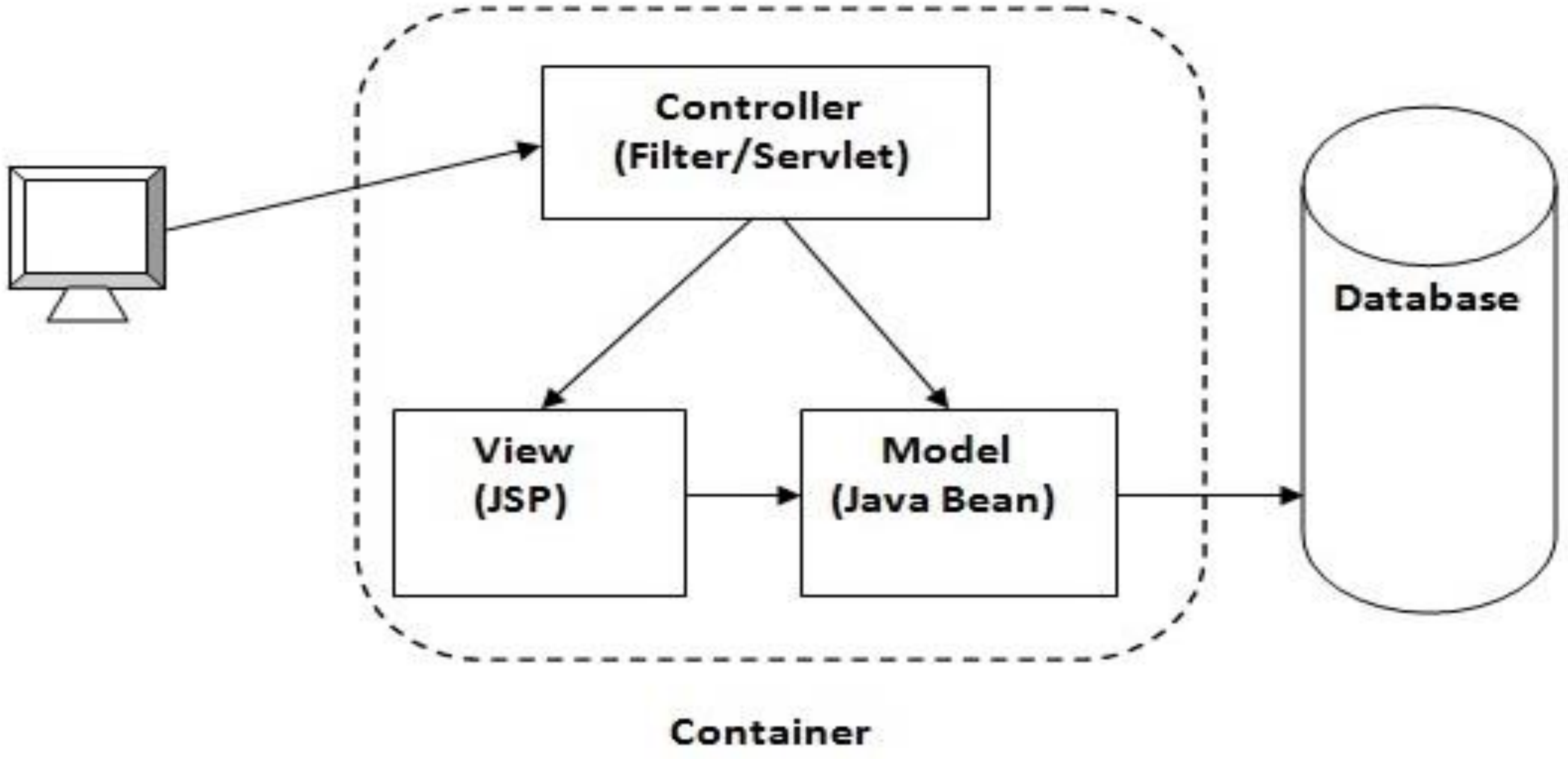
- <a href="servletname/jsp"> - GET method
- <button onclick="location.href=servletname/jsp"> - GET method
- <input type="button" onclick="location.href=servletname/jsp">
- <form action="servletname/jsp"> - POST/PUT/GET/DELETE

- request.setParameter or request.setAttribute or methods
- request.getParameter or request.getAttribute methods
- ${} or <% %>

MVC

- MVC stands for **M**odel **V**iew and **C**ontroller.
- 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


Controller
(Filter/Servlet)
View
(JSP)
Model
(Java Bean)
Database
Container

MVC DEMO

- Step 1 : Create a user interface (UI) or View “login.jsp”

```
<form action=“./ControllerServlet" method="post">
        Name      :   <input type="text" name="name">                <br>
        Password :   <input type="password" name="password"> <br>
                      <input type="submit" value="login">
</form>
```

- Step 2 : Create the model “User.java”

```
        public class User {
                private String name,
                private String password;
                // generate getter and setter methods
                public boolean validate(){
                if(password.equals("admin")){
                 return true;
                }
                else{
                return false;
            }
        }
```

MVC DEMO CONTD..

- Step 3 : Create the controller “ControllerServle.java”

```
public class ControllerServlet extends HttpServlet {
   protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
     String name=request.getParameter("name");
     String password=request.getParameter("password");
     User user = new User( );
     user.setName(name);
     user.setPassword (password);
     request.setAttribute ("bean", user);
     RequestDispatcher rd = null;
     if(user.validate()){
         RequestDispatcher rd=request.getRequestDispatcher("login-success.jsp");
     } else{
         RequestDispatcher rd=request.getRequestDispatcher("login-error.jsp");
      }
      rd.forward(request, response);
    }
```

MVC DEMO CONTD..

- Step 4 :configure in web.xml

```
<servlet>
  <servlet-name>ApplicationServlet</servlet-name>
  <servlet-class>com.controller.ControllerServlet</servlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name>ApplicationServlet</servlet-name>
  <url-pattern>/ControllerServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

MVC DEMO CONTD..

- Step 5 : create view page “login-success.jsp ”

```
<%@page import="com.model.User"%>
  <p>You are successfully logged in!</p>
<%  User bean=(User)request.getAttribute("bean");
       out.print("Welcome, “ + bean.getName());
%>
```

- Step 6 : Create view page “login-error.jsp”

```
<p>Sorry! username or password error</p>
<%@ include file=“login.jsp" %>
```

JAVA BEAN VS JAVA POJO

IMPORTANT TO NOTE FOR WEB APPLICATION

1] Should works as per recommended browser(s) - look for compatible version as well

2] Use attribute for any validation, then go for Javascript, then server validation.

3] Use Controller as much as possible to control your flow.

4] Re-usability (no duplication), loose coupled (divide logically for methods/classes), java coding standards, Clean code, etc.,

5] U&I should be very good

5a] appropriate spaces, tabs, line breaks, header, footer

5b] User friendly msg/inputs

5c] horizontal scroll bar must be avoided

5d] Display the logged user in the header/appropriate place thro' all the page till he logs out. (exception login/sign-in)

5e] Confine your UI colors 2-4 colors. (For 4 colors - 12 combination example blue --> light, normal, dark)

5f) Number (right aligned), String left aligned, SHORT Form text center aligned.

IMPORTANT TO NOTE FOR WEB APPLICATION

6] Explore multiple third party at UI end. (datatable, jquery/ajax).. (taglib)

7] Use appropriate method (get/put/delete/post)

8] "Reset" , "Cancel" button (appropriate)

9] copyright problem will occur in case you use internet (code/image/audio/video/files)

10] response time should be quick.

11] Minified (JS, CSS) version

12] Use relative path not the full path of the web application URL

13] Visit as many as web site and absorb the best layout/flow/type of controls/colors+graphics/etc.,

DEMO

Console to Web Application

Only one main method entry, where each page is entry point based upon the request method (GET/POST/DELETE/PUT)

REFERENCES & FURTHER STUDIES

- https://docs.oracle.com/javaee/5/tutorial/doc/bnagx.html
- https://www.tutorialspoint.com/jsp/index.htm
- https://www.javatpoint.com/jsp-tutorial

HCL

Relationship™

BEYOND THE CONTRACT

$6.8 BILLION | **109,000** EMPLOYEES | **31** COUNTRIES

WATCH THE **FILM**