Lecture 5

S3 - Summary of Server Side/Web Development in JEE

presentation

DAD – Distributed Applications Development

Cristian Toma

D.I.C.E/D.E.I.C – Department of Economic Informatics & Cybernetics
www.dice.ase.ro

Cristian Toma – Business Card

Agenda for Lecture 5

DAD Section 3 - Summary of Web Development in JEE, JSP Architecture, Lifecycle, Samples

JSP - Java Server Page Intro & Recap

1. Java Server Pages – JSP Technology

What is JSP – Java Server Pages?

- JSP Architecture
- JSP Life-cycle
- JSP Syntax + Directive + DEMO
- JSP Predefined Variables
- JSP Predefined Directives + Actions
- JSP Beans
- DEMO JSP & JSPX

JSP Technology Necessity

With Java Servlet technology is easy to:

- Get the data from HTML form
- Get the headers values from HTTP request
- Set the error status code and header from HTTP response
- Use of “cookies” & “session tracking” – “state-full” HTTP
- Sharing the data between Java servlets

With Java Servlet technology is NOT easy to:

- Generate HTML tags using the method “println” from ‘PrintWriter’ class
- Maintenance of HTML generated code

JSP Approach

Ideas:

– Use of standard HTML code in most of the pages sections

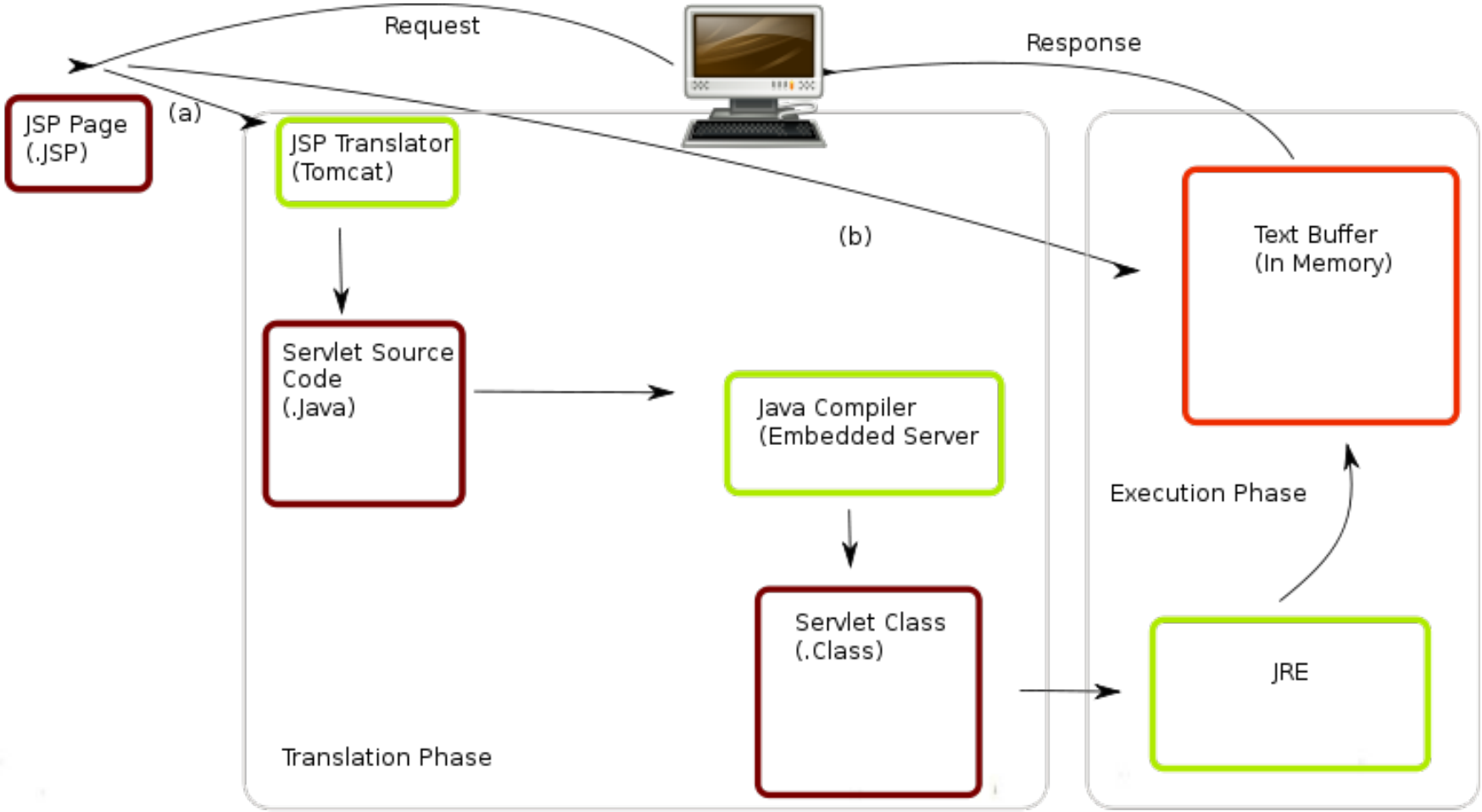
– The entire JSP page is “translated” into Java servlet, and the servlet’s methods are called for each HTTP request

- **Sample:**

```
<!DOCTYPE …>
<HTML>
<HEAD>
 <TITLE>Order Confirmation</TITLE>
 <LINK REL=STYLESHEET HREF="JSP-Styles.css“ TYPE="text/css">
</HEAD>
<BODY>
 <H2>Order Confirmation</H2>
 Thanks for ordering <I><%= request.getParameter("title") %></I>
</BODY>
</HTML>
```

1. Java Server Pages – JSP Technology

JSP Translation Process



JSP Container
(a) Translation occurs at this point if JSP has been changed or is new
(b) if not, translation is skipped.

1. Java Server Pages – JSP Technology

JSP Lifecycle

	Page first written	Request #1	Request #2	Server restarted	Request #3	Request #4	Page modified	Request #5	Request #6
JSP page translated into servlet		Yes	No		No	No		Yes	No
Servlet compiled		Yes	No		No	No		Yes	No
Servlet instantiated and loaded into server's memory		Yes	No		Yes	No		Yes	No
init (or equivalent) called		Yes	No		Yes	No		Yes	No
doGet (or equivalent) called		Yes	Yes		Yes	Yes		Yes	Yes

1. Java Server Pages – JSP Technology

DEMO Sample – test01.jsp:

Although in Java Servlet the may do as much as in JSP (because JSP is Java Servlet), it is recommended to use JSP for:

- Writing easy HTML code
- Debugging and maintaining the HTML code
- There is nothing about environment variables, just simple copy the JSP files in a web server directory

test.jsp

```
<HTML>
<HEAD>
 <TITLE>Test jsp</TITLE>
</HEAD>
<BODY>
<H2>JSP Expressions</H2>
<UL>
<LI>Current time: <%= new java.util.Date() %>
<LI>Server: <%= application.getServerInfo() %>
<LI>Session ID: <%= session.getId() %>
<LI>The <CODE>testParam</CODE> form parameter: <%= request.getParameter("testParam") %>
</UL>
</BODY>
</HTML>
```

1. Java Server Pages – JSP Technology

JSP API

JSP Element	Syntax	Interpretation	Notes
JSP Expression	`<%= expression %>`	Expression is evaluated and placed in output.	XML equivalent is `<jsp:expression>` `expression` `</jsp:expression>`. Predefined variables are `request`, `response`, `out`, `session`, `application`, `config`, and `pageContext` (available in scriptlets also).
JSP Scriptlet	`<% code %>`	Code is inserted in `service` method.	XML equivalent is `<jsp:scriptlet>` `code` `</jsp:scriptlet>`.
JSP Declaration	`<%! code %>`	Code is inserted in body of servlet class, outside of `service` method.	XML equivalent is `<jsp:declaration>` `code` `</jsp:declaration>`.

1. Java Server Pages – JSP Technology

JSP API

JSP Element	Syntax	Interpretation	Notes
JSP page Directive	`<%@ page att="val" %>`	Directions to the servlet engine about general setup.	XML equivalent is `<jsp:directive.page att="val"\>`. Legal attributes, with default values in bold, are: • import="*package.class*" • contentType="*MIME-Type*" • isThreadSafe="**true**|false" • session="**true**|false" • buffer="*size*kb|none" • autoflush="**true**|false" • extends="*package.class*" • info="*message*" • errorPage="*url*" • isErrorPage="true|**false**" • language="java"
JSP `include` Directive	`<%@ include file="url" %>`	A file on the local system to be included when the JSP page is translated into a servlet.	XML equivalent is `<jsp:directive.include file="url"\>`. The URL must be a relative one. Use the `jsp:include` action to include a file at request time instead of translation time.

1. Java Server Pages – JSP Technology

JSP API

JSP Element	Syntax	Interpretation	Notes
JSP Comment	`<%-- comment --%>`	Comment; ignored when JSP page is translated into servlet.	If you want a comment in the resultant HTML, use regular HTML comment syntax of `<-- comment -->`.
The `jsp:include` Action	`<jsp:include page="relative URL" flush="true"/>`	Includes a file at the time the page is requested.	If you want to include the file at the time the page is translated, use the `page` directive with the `include` attribute instead. **Warning**: on some servers, the included file must be an HTML file or JSP file, as determined by the server (usually based on the file extension).

1. Java Server Pages – JSP Technology

JSP API

JSP Element	Syntax	Interpretation	Notes

The `jsp:useBean` Action	`<jsp:useBean att=val*/>` or
	`<jsp:useBean att=val*>`
	`...`
	`</jsp:useBean>`
		Find or build a Java Bean.	Possible attributes are:
			• id="*name*"
			• scope="**page**|request|session|application"
			• class="*package.class*"
			• type="*package.class*"
			• beanName="*package.class*"

The `jsp:setProperty` Action	`<jsp:setProperty att=val*/>`	Set bean properties, either explicitly or by designating that value comes from a request parameter.	Legal attributes are • name="*beanName*" • property="*propertyName*|*" • param="*parameterName*" • value="*val*"

The `jsp:getProperty` Action	`<jsp:getProperty`
	`name="propertyName"`
	`value="val"/>`
		Retrieve and output bean properties.	

1. Java Server Pages – JSP Technology

JSP API

JSP Element	Syntax	Interpretation	Notes
The `jsp:forward` Action	`<jsp:forward page="relative URL"/>`	Forwards request to another page.	
The `jsp:plugin` Action	`<jsp:plugin attribute="value"*> ... </jsp:plugin>`	Generates `OBJECT` or `EMBED` tags, as appropriate to the browser type, asking that an applet be run using the Java Plugin.	

JSP API

Predefined Objects in JSP

In order to simplify the Java source code in the scriptlets and expressions, in JSP there are 8 predefined objects.

1 request

This is the HttpServletRequest associated with the request, and lets you look at the request parameters (via getParameter), the request type (GET, POST, HEAD, etc.), and the incoming HTTP headers (cookies, Referer, etc.). Strictly speaking, request is allowed to be a subclass of ServletRequest other than HttpServletRequest, if the protocol in the request is something other than HTTP. This is almost never done in practice.

2 response

This is the HttpServletResponse associated with the response to the client. Note that, since the output stream (see out below) is buffered, it is legal to set HTTP status codes and response headers, even though this is not permitted in regular servlets once any output has been sent to the client.

JSP API

3 out

This is the PrintWriter used to send output to the client. However, in order to make the response object (see the previous section) useful, this is a buffered version of PrintWriter called JspWriter. Note that you can adjust the buffer size, or even turn buffering off, through use of the buffer attribute of the page directive. This was discussed in Section 5. Also note that out is used almost exclusively in scriptlets, since JSP expressions automatically get placed in the output stream, and thus rarely need to refer to out explicitly.

4 session

This is the HttpSession object associated with the request. Recall that sessions are created automatically, so this variable is bound even if there was no incoming session reference. The one exception is if you use the session attribute of the page directive to turn sessions off, in which case attempts to reference the session variable cause errors at the time the JSP page is translated into a servlet.

JSP API

5 application

This is the ServletContext as obtained via getServletConfig().getContext().

6 config

This is the ServletConfig object for this page.

7 pageContext

JSP introduced a new class called PageContext to encapsulate use of server-specific features like higher performance JspWriters. The idea is that, if you access them through this class rather than directly, your code will still run on "regular" servlet/JSP engines. It is used also for Java Bean synchronization and session tracking info storage for not allocating more beans for one page session.

8 page

This is simply a synonym for this, and is not very useful in Java. It was created as a placeholder for the time when the scripting language could be something other than Java.

JSP API

ACTIONS in JSP

The actions in JSP uses XML for controlling the behavior of the web server Java Servlet container. Using the JSP actions, the programmer can dynamically insert files, reuse Java Beans components, redirect the web browser to other web resources or generate HTML with JavaScript or Java applets.

- jsp:include – Include a file in the HTTP request time.
- jsp:useBean – Use or instantiate a JavaBean component.
- jsp:setProperty – Sets the JavaBean field value using a property.
- jsp:getProperty – Gets the JavaBean field value using a property.
- jsp:forward – Redirect an HTTP request to a new page/web resource.
- jsp:plugin - Generate the specific web browser code which contains OBJECT or EMBED HTML tag for Java Applets that are running at the client web browser.

Section Conclusion

Fact: **DAD needs Web Programming**

In few **samples** it is simple to remember: Java Server Pages Programming with HTTP protocol analysis in real time for request headers, responses' codes and headers, session tracking – generates standards HTML pages as entering gate for distributed computing and systems.

DAD Section 3 - Summary of Web Development in JEE, JSP Tag libs, JSP MVC Concept, Web Server Clusters

Java Server Page Tech Advanced Topics

2. Advanced Java Server Page – JSP Tech Topics

JSP DEMO Tag-lib

http://jakarta.apache.org/taglibs/doc/standard-doc/intro.html

Using JSTL: JSTL includes a wide variety of tags that fit into discrete functional areas. To reflect this, as well as to give each area its own namespace, JSTL is exposed as multiple tag libraries.
The URIs for the libraries are as follows:

Core: http://java.sun.com/jsp/jstl/core

XML: http://java.sun.com/jsp/jstl/xml

Internationalization: http://java.sun.com/jsp/jstl/fmt

SQL: http://java.sun.com/jsp/jstl/sql

Functions: http://java.sun.com/jsp/jstl/functions

2. Advanced Java Server Page – JSP Tech Topics

JSP DEMO Tag-lib

Area	Subfunction	Prefix
Core	Variable support	c
	Flow control	
	URL management	
	Miscellaneous	
XML	Core	x
	Flow control	
	Transformation	
I18N	Locale	fmt
	Message formatting	
	Number and date formatting	
Database	SQL	sql
Functions	Collection length	fn
	String manipulation	

Area	Function	Tags	Prefix
Core	Variable support	remove set	c
	Flow control	choose when otherwise forEach forTokens if	
	URL management	import param redirect param url param	
	Miscellaneous	catch out	

2. Advanced Java Server Page – JSP Tech Topics

JSP DEMO Tag-lib

Area	Function	Tags	Prefix
I18N	Setting Locale	setLocale requestEncoding	fmt
	Messaging	bundle message param setBundle	
	Number and Date Formatting	formatNumber formatDate parseDate parseNumber setTimeZone timeZone	

Area	Function	Tags	Prefix
XML	Core	out parse set	x
	Flow control	choose when otherwise forEach if	
	Transformation	transform param	

2. Advanced Java Server Page – JSP Tech Topics

JSP DEMO Tag-lib

Area	Function	Tags	Prefix
Database	Setting the data source	`setDataSource`	`sql`

	SQL	`query`
		`  dateParam`
		`  param`
		`transaction`
		`update`
		`  dateParam`
		`  param`	

2. Advanced Java Server Page – JSP Tech Topics

JSP MVC – Model View Controller – Model 1 – Architecture Design Pattern

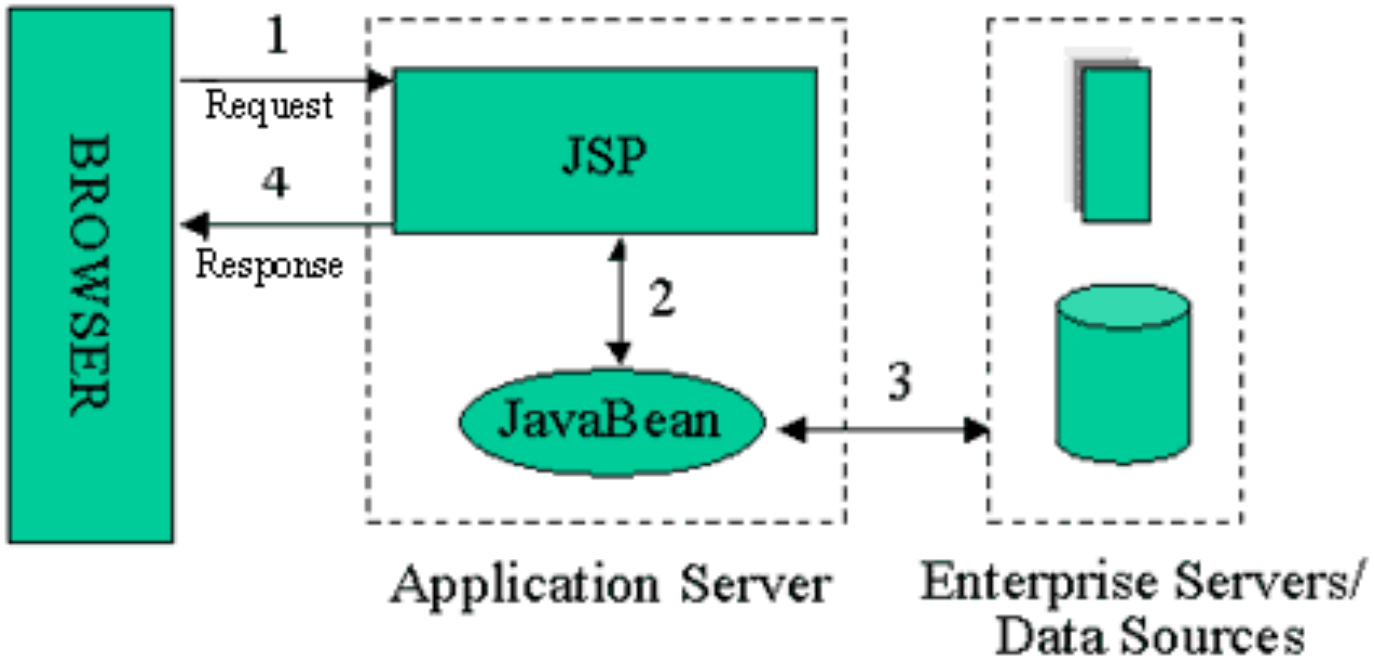
The early JSP specifications advocated two philosophical approaches for building applications using JSP technology. These approaches, termed the JSP Model 1 and Model 2 architectures, differ essentially in the location at which the bulk of the request processing was performed.

In the Model 1 architecture, shown in Figure of Model 1, the JSP page alone is responsible for processing the incoming request and replying back to the client. There is still separation of presentation from content, because all data access is performed using beans. Although the Model 1 architecture should be perfectly suitable for simple applications, it may not be desirable for complex implementations. Indiscriminate usage of this architecture usually leads to a significant amount of scriptlets or Java code embedded within the JSP page, especially if there is a significant amount of request processing to be performed. While this may not seem to be much of a problem for Java developers, it is certainly an issue if your JSP pages are created and maintained by designers -- which is usually the norm on large projects.

Ultimately, it may even lead to an unclear definition of roles and allocation of responsibilities, causing easily avoidable project-management headaches.

2. Advanced Java Server Page – JSP Tech Topics

JSP MVC – Model View Controller – Model 1 – Architecture Design Pattern



2. Advanced Java Server Page – JSP Tech Topics

JSP MVC – Model View Controller – Model 2 – Architecture Design Pattern

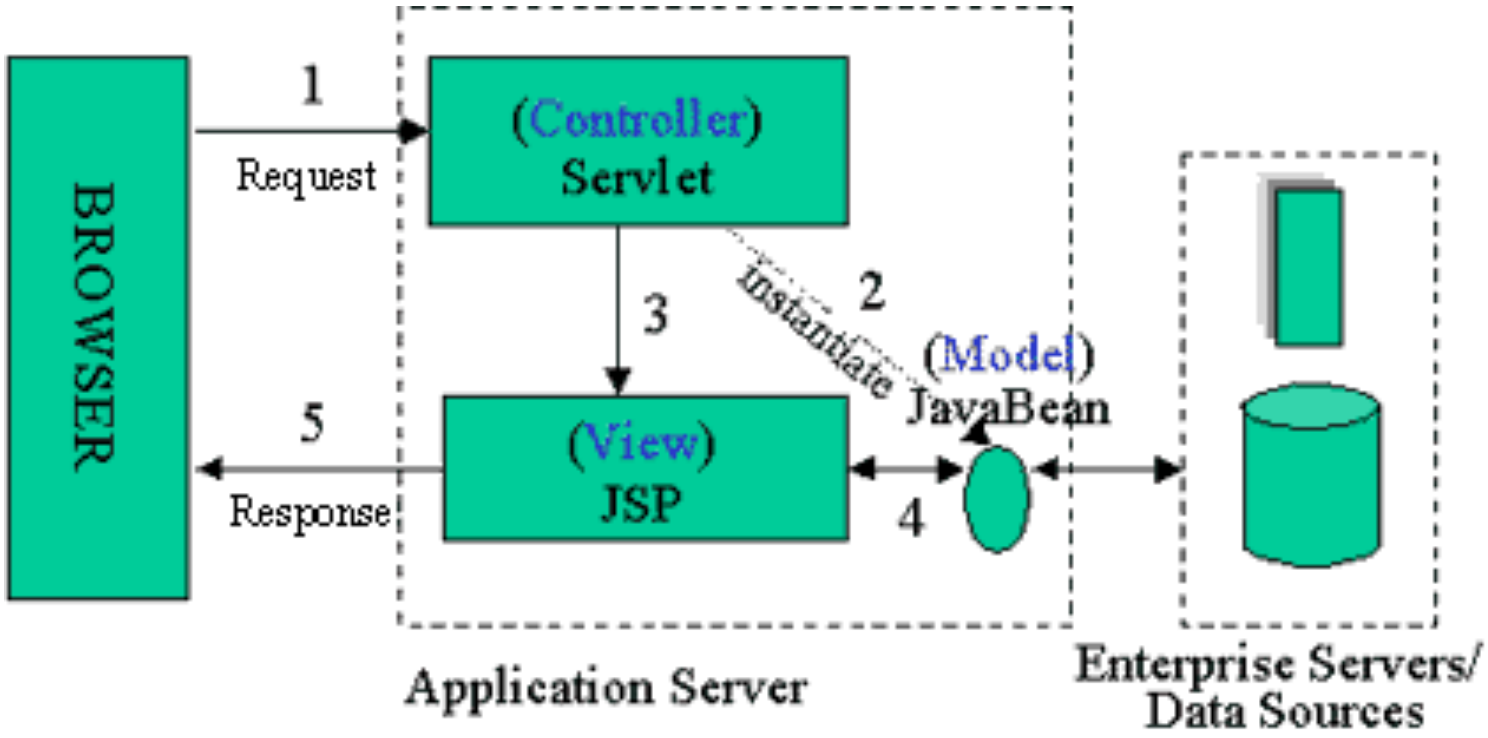
The Model 2 architecture, shown in Figure of Model 2, is a hybrid approach for serving dynamic content, since it combines the use of both servlets and JSP. It takes advantage of the predominant strengths of both technologies, using JSP to generate the presentation layer and servlets to perform process-intensive tasks.

Here, the servlet acts as the *controller* and is in charge of the request processing and the creation of any beans or objects used by the JSP, as well as deciding, depending on the user's actions, which JSP page to forward the request to. Note particularly that there is no processing logic within the JSP page itself; it is simply responsible for retrieving any objects or beans that may have been previously created by the servlet, and extracting the dynamic content from that servlet for insertion within static templates.

This approach typically results in the cleanest separation of presentation from content, leading to clear delineation of the roles and responsibilities of the developers and page designers on your programming team. In fact, the more complex your application, the greater the benefits of using the Model 2 architecture should be.

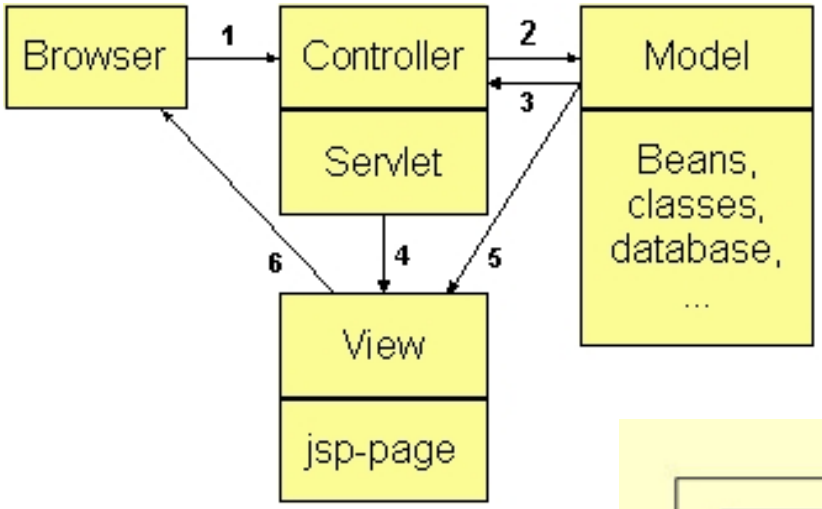
2. Advanced Java Server Page – JSP Tech Topics

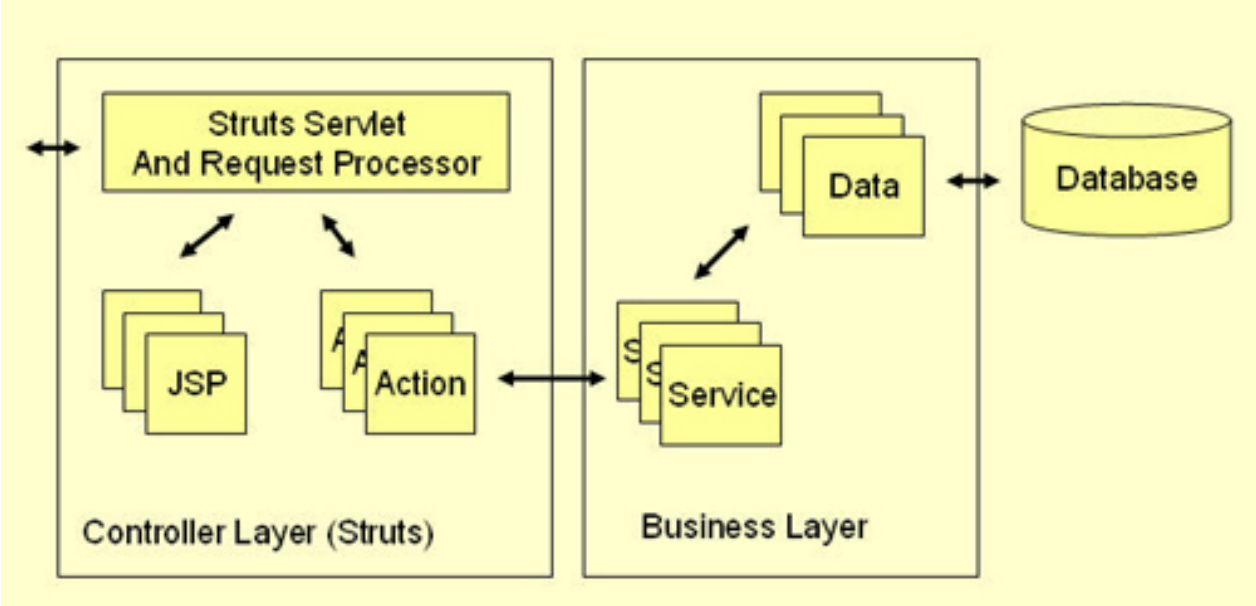
JSP MVC – Model View Controller – Model 2 – Architecture Design Pattern



2. Advanced Java Server Page – JSP Tech Topics

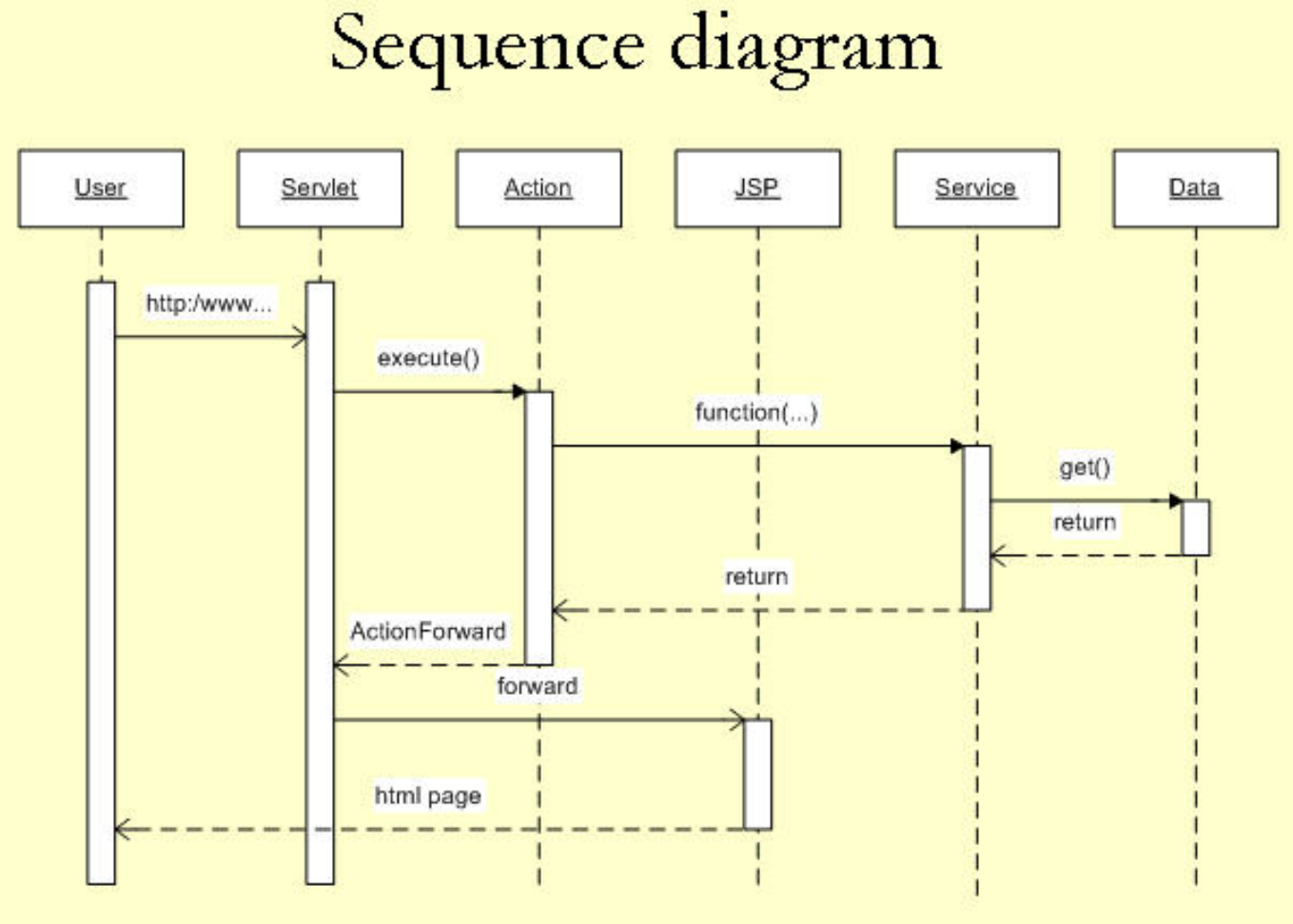
JSP MVC – Model View Controller – Model 2 – Architecture Design Pattern





2. Advanced Java Server Page – JSP Tech Topics

JSP MVC – Model View Controller – Model 2 – Architecture Design Pattern



2. Advanced Java Server Page – JSP Tech Topics

JSP MVC – Model View Controller – Model 2 – Architecture Design Pattern

Java MVC – Struts Intro

1. User clicks on a link in an HTML page.

2. Servlet controller receives the request, looks up mapping information in struts-config.xml, and routes to an action.

3. Action makes a call to a Model layer service.

4. Service makes a call to the Data layer (database) and the requested data is returned.

5. Service returns to the action.

6. Action forwards to a View resource (JSP page)

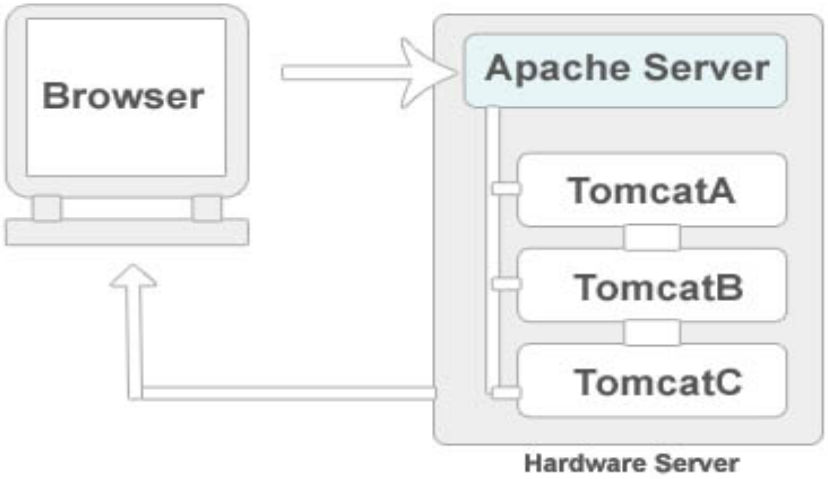
7. Servlet looks up the mapping for the requested resource and forwards to the appropriate JSP page.

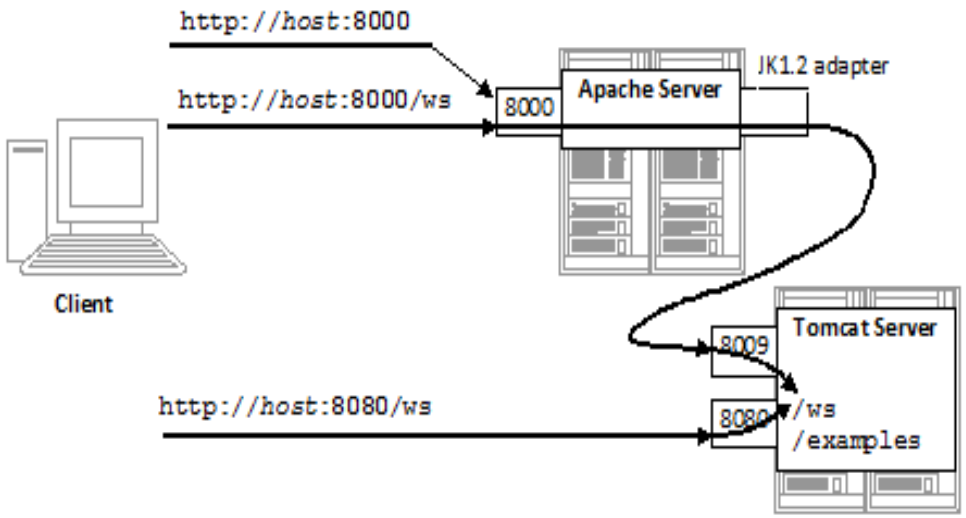
8. JSP file is invoked and sent to the browser as HTML.
9. User is presented with a new HTML page in a web browser.

2. Advanced Java Server Page – JSP Tech Topics

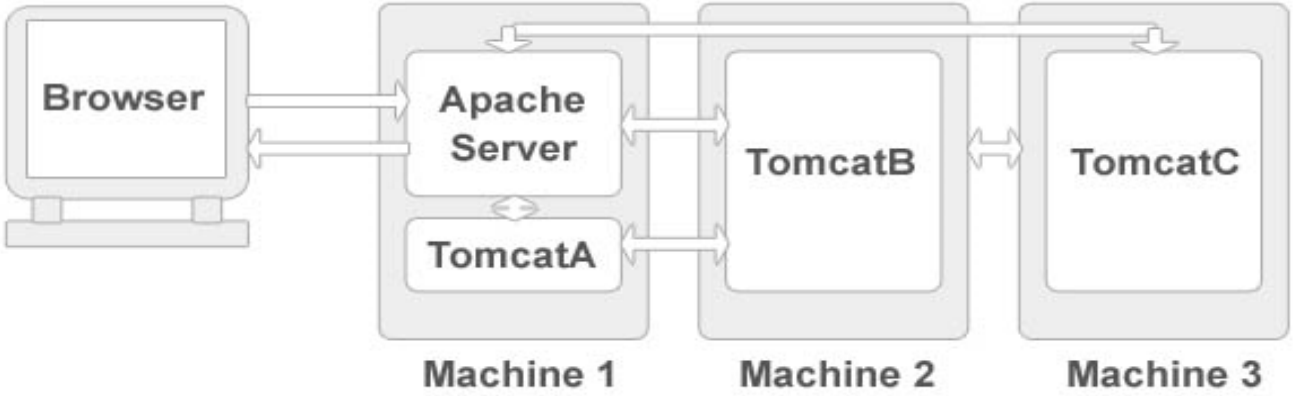
Web Server Cluster

Horizontal Clustering



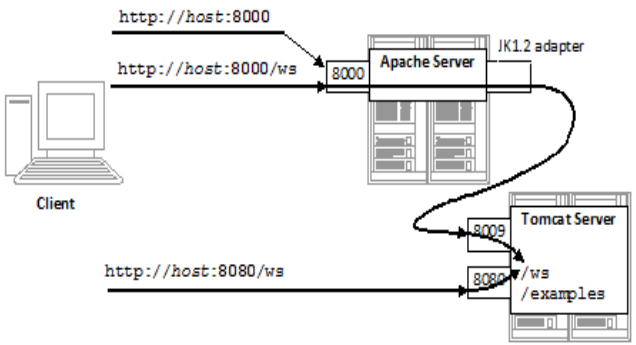


Vertical Clustering



2. Advanced Java Server Page – JSP Tech Topics

Web Server Cluster




```
<?xml version="1.0" encoding="ISO-8859-1"?>

<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee htt
  version="2.5">
  <distributable />

</web-app>
```

1. <Server port="8005" shutdown="SHUTDOWN">

2. <Connector port="8080" protocol="HTTP/1.1"
   connectionTimeout="20000"
   redirectPort="8443" />

3. <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />

4. Add jvmRoute
<Engine name="Catalina" defaultHost="localhost" jvmRoute="tomcatC">

5. Uncommented clustering tag
<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>

2. Advanced Java Server Page – JSP Tech Topics

Web Server Cluster

DEMO – httpd.conf

for horizontal tomcat clustering

```
workers.tomcat_home=/tomcatA
workers.java_home=$JAVA_HOME
ps=/
worker.list=tomcatA,tomcatB,tomcatC,loadbalancer


worker.tomcatA.port=8009
worker.tomcatA.host=192.168.1.1
worker.tomcatA.type=ajp13
worker.tomcatA.lbfactor=1


worker.tomcatB.port=8009
worker.tomcatB.host=192.168.1.2
worker.tomcatB.type=ajp13
worker.tomcatB.lbfactor=1

worker.tomcatC.port=8009
worker.tomcatC.host=192.168.1.3
worker.tomcatC.type=ajp13
worker.tomcatC.lbfactor=1

worker.loadbalancer.type=lb
worker.loadbalancer.balanced_workers=tomcatA,tomcatB,tomcatC
worker.loadbalancer.sticky_session=1
```

```
LoadModule jk_module modules/mod_jk-apache-2.2.4.so
JkWorkersFile "C:\cluster\Apache\conf\workers.properties"
JkLogFile "logs/mod_jk.log"
JkLogLevel error
JkMount /cluster loadbalancer
JkMount /cluster/* loadbalancer
```

lbfactor properties define for load balancing factor, restrict number of request to send particular tomcat instance
e.g
worker.tomcatC.lbfactor=100
increase and decrease request to this tomcatC instance

Java Server Page Programming
for easy sharing

Section Conclusions

Java Servlet & JSP Programming uses HTTP knowledge.

Any JSP file is translated into a Java Servlet by the Java web app server

Java Servlets migrates the boilerplate configuration code from web.xml to Java source code though annotations

Most important predefined objects in JSP are: request, response, out, session, application, config, pageContext, page

Most important actions in JSP are: jsp:include; jsp:useBean, (jsp:setProperty, jsp:getProperty); jsp:forward, jsp:plugin

JSP "directives" are different that "actions" + tag-libs/filters are useful for a modular and reliable solution

MVC – Model View Controller is a design pattern implemented with success by various frameworks in Java

For HA – High Availability and Scalability, the one may use Web Servers Clusters or Cloud Solutions - such as *Google App Engine* – Cloud PaaS Platform for Java, Ruby, Python or GO

Cloud Computing Technology Intro – IaaS, PaaS, SaaS

Communicate & Exchange Ideas

Questions & Answers!

But wait…

There's More!

Thanks!

DAD – Distributed Application Development
End of Lecture 5