Lecture 7
S3 - Summary of Web Development in JEE
Agenda for Lecture 7

1. JSP – Java Server Page Intro
2. JSP Tag-lib, MVC Intro & Web Servers Cluster
3. Cloud Computing Intro + Exchange Ideas
DAD Section 3 - Summary of Web Development in JEE, JSP Architecture, Lifecycle, Samples

JSP - Java Server Page Intro & Recap
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:

```html
<!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 Page (.JSP) -> JSP Translator (Tomcat) -> Servlet Source Code (.Java) -> Java Compiler (Embedded Server) -> Servlet Class (.Class) -> JRE

Request

Text Buffer (In Memory)

Response

Execution Phase

(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

<table>
<thead>
<tr>
<th></th>
<th>Request #1</th>
<th>Request #2</th>
<th>Request #3</th>
<th>Request #4</th>
<th>Request #5</th>
<th>Request #6</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSP page translated into servlet</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Servlet compiled</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Servlet instantiated and loaded into server’s memory</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>init (or equivalent) called</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>doGet (or equivalent) called</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
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

```html
<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

<table>
<thead>
<tr>
<th>JSP Element</th>
<th>Syntax</th>
<th>Interpretation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSP Expression</td>
<td>&lt;%= expression %&gt;</td>
<td>Expression is evaluated and placed in output.</td>
<td>XML equivalent is <code>&lt;jsp:expression&gt; expression &lt;/jsp:expression&gt;</code>. Predefined variables are request, response, out, session, application, config, and pageContext (available in scriptlets also).</td>
</tr>
<tr>
<td>JSP Scriptlet</td>
<td>&lt;% code %&gt;</td>
<td>Code is inserted in service method.</td>
<td>XML equivalent is <code>&lt;jsp:scriptlet&gt; code &lt;/jsp:scriptlet&gt;</code>.</td>
</tr>
<tr>
<td>JSP Declaration</td>
<td>&lt;%! code %&gt;</td>
<td>Code is inserted in body of servlet class, outside of service method.</td>
<td>XML equivalent is <code>&lt;jsp:declaration&gt; code &lt;/jsp:declaration&gt;</code>.</td>
</tr>
</tbody>
</table>
## 1. Java Server Pages – JSP Technology

### JSP API

<table>
<thead>
<tr>
<th>JSP Element</th>
<th>Syntax</th>
<th>Interpretation</th>
<th>Notes</th>
</tr>
</thead>
</table>
| 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="sizekb|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

<table>
<thead>
<tr>
<th>JSP Element</th>
<th>Syntax</th>
<th>Interpretation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSP Comment</td>
<td><code>&lt;%-- comment --&gt;%&gt;</code></td>
<td>Comment, ignored when JSP page is translated into servlet.</td>
<td>If you want a comment in the resultant HTML, use regular HTML comment syntax of <code>&lt;!-- comment --&gt;</code>.</td>
</tr>
<tr>
<td>The jsp:include Action</td>
<td><code>&lt;jsp:include page=&quot;relative URL&quot; flush=&quot;true&quot;/&gt;</code></td>
<td>Includes a file at the time the page is requested.</td>
<td>If you want to include the file at the time the page is translated, use the page directive with the include attribute instead. <strong>Warning:</strong> 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).</td>
</tr>
</tbody>
</table>
## 1. Java Server Pages – JSP Technology

### JSP API

<table>
<thead>
<tr>
<th>JSP Element</th>
<th>Syntax</th>
<th>Interpretation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <code>jsp:useBean</code> Action</td>
<td><code>&lt;jsp:useBean att=val*/&gt;</code> or <code>&lt;jsp:useBean att=val*&gt; ... &lt;/jsp:useBean&gt;</code></td>
<td>Find or build a Java Bean.</td>
<td>Possible attributes are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• id=&quot;name&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• scope=&quot;page</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• class=&quot;package.class&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• type=&quot;package.class&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• beanName=&quot;package.class&quot;</td>
</tr>
<tr>
<td>The <code>jsp:setProperty</code> Action</td>
<td><code>&lt;jsp:setProperty att=val*/&gt;</code></td>
<td>Set bean properties, either explicitly or by designating that value comes from a request parameter.</td>
<td>Legal attributes are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• name=&quot;beanName&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• property=&quot;propertyName '*'&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• param=&quot;parameterName&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• value=&quot;val&quot;</td>
</tr>
<tr>
<td>The <code>jsp:getProperty</code> Action</td>
<td><code>&lt;jsp:getProperty name=&quot;propertyName&quot; value=&quot;val&quot;/&gt;</code></td>
<td>Retrieve and output bean properties.</td>
<td></td>
</tr>
</tbody>
</table>
## JSP API

<table>
<thead>
<tr>
<th>JSP Element</th>
<th>Syntax</th>
<th>Interpretation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <code>&lt;jsp:forward&gt;</code> Action</td>
<td><code>&lt;jsp:forward page=&quot;relative URL&quot;/&gt;</code></td>
<td>Forwards request to another page.</td>
<td></td>
</tr>
</tbody>
</table>
| 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. |       |
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.
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.
1. Java Server Pages – JSP Technology

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.
1. Java Server Pages – JSP Technology

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.
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.
JSP Tag libs, JSP MVC Concept, Web Server Clusters
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 URLs for the libraries are as follows:* 

**Core:** [http://java.sun.com/jsp/jstl/core](http://java.sun.com/jsp/jstl/core)

**XML:** [http://java.sun.com/jsp/jstl/xml](http://java.sun.com/jsp/jstl/xml)

**Internationalization:** [http://java.sun.com/jsp/jstl/fmt](http://java.sun.com/jsp/jstl/fmt)

**SQL:** [http://java.sun.com/jsp/jstl/sql](http://java.sun.com/jsp/jstl/sql)

**Functions:** [http://java.sun.com/jsp/jstl/functions](http://java.sun.com/jsp/jstl/functions)
## 2. Advanced Java Server Page – JSP Tech Topics

### JSP DEMO Tag-lib

<table>
<thead>
<tr>
<th>Area</th>
<th>Subfunction</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>Variable support</td>
<td>c</td>
</tr>
<tr>
<td></td>
<td>Flow control</td>
<td></td>
</tr>
<tr>
<td></td>
<td>URL management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Miscellaneous</td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td>Core</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Flow control</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transformation</td>
<td></td>
</tr>
<tr>
<td>I18N</td>
<td>Locale</td>
<td>fmt</td>
</tr>
<tr>
<td></td>
<td>Message formatting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number and date formatting</td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td>SQL</td>
<td>sql</td>
</tr>
<tr>
<td>Functions</td>
<td>Collection length</td>
<td>fn</td>
</tr>
<tr>
<td></td>
<td>String manipulation</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Area</th>
<th>Function</th>
<th>Tags</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>Variable support</td>
<td>remove set</td>
<td>c</td>
</tr>
<tr>
<td></td>
<td>Flow control</td>
<td>choose</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>when</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>otherwise</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>forEach</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>forEach</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>forTokens</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>if</td>
<td></td>
</tr>
<tr>
<td></td>
<td>URL management</td>
<td>import</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>param</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>redirect</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>url</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>param</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Miscellaneous</td>
<td>catch</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>out</td>
<td></td>
</tr>
</tbody>
</table>
## JSP DEMO Tag-lib

<table>
<thead>
<tr>
<th>Area</th>
<th>Function</th>
<th>Tags</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>I18N</td>
<td>Setting Locale</td>
<td>setLocale, requestEncoding</td>
<td>fmt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Messaging</td>
<td>bundle, message, param, setBundle</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number and Date Formatting</td>
<td>formatNumber, formatDate, parseDate, parseNumber, setTimeZone, timeZone</td>
<td></td>
</tr>
</tbody>
</table>

### Area: XML

<table>
<thead>
<tr>
<th>Function</th>
<th>Tags</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>out, parse, set</td>
<td>x</td>
</tr>
</tbody>
</table>

### Area: Flow control

<table>
<thead>
<tr>
<th>Function</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>choose, when, otherwise, forEach, if</td>
</tr>
</tbody>
</table>

### Area: Transformation

<table>
<thead>
<tr>
<th>Function</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>transform, param</td>
</tr>
</tbody>
</table>
2. Advanced Java Server Page – JSP Tech Topics

JSP DEMO Tag-lib

<table>
<thead>
<tr>
<th>Area</th>
<th>Function</th>
<th>Tags</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Setting the data source</td>
<td>setDataSource</td>
<td>sql</td>
</tr>
<tr>
<td>SQL</td>
<td></td>
<td>query</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>dateParam</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>param</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>transaction</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>update</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>dateParam</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>param</td>
<td></td>
</tr>
</tbody>
</table>
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
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

Sequence diagram

- User
- Servlet
- Action
- JSP
- Service
- Data

User requests access via HTTP
Servlet executes the request and calls the Action
Action performs the business logic and calls the Service
Service retrieves data and returns it to the Action
Action returns the result to the JSP
JSP generates the HTML page
Page is displayed to the user
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

Browser ➔ Apache Server ➔ TomcatA ➔ TomcatB ➔ TomcatC

Browser ➔ Apache Server ➔ TomcatB ➔ TomcatC

http://host:8000
http://host:8000/ws

http://host:8080/ws

Machine 1 ➔ TomcatA ➔ Machine 2 ➔ TomcatB ➔ TomcatC ➔ Machine 3

JK12 adapter

/examples
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

```conf
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

```conf
worker.tomcatC.lbfactor=100
```

increase and decrease request to this tomcatC instance
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 through 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 than “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!
Before Virtualization:
- Single OS image per machine
- Software and hardware tightly coupled
- Running multiple applications on same machine often creates conflict
- Underutilized resources
- Inflexible and costly infrastructure

After Virtualization:
- Hardware-independence of operating system and applications
- Virtual machines can be provisioned to any system
- Can manage OS and application as a single unit by encapsulating them into virtual machines
Hosted Architecture
- Installs and runs as an application
- Relies on host OS for device support and physical resource management

Bare-Metal (Hypervisor) Architecture
- Lean virtualization-centric kernel
- Service Console for agents and helper applications
Cloud Concepts – Intro – Virtualization Overview

**TYPE 1**

*native*
*(bare metal)*

**TYPE 2**

*hosted*
Cloud Concepts – Intro – IaaS, PaaS, SaaS

Separation of Responsibilities

On-Premises
- Applications
- Data
- Runtime
- Middleware
- O/S
- Virtualization
- Servers
- Storage
- Networking

You manage

Infrastructure (as a Service)
- Applications
- Data
- Runtime
- Middleware
- O/S
- Virtualization
- Servers
- Storage
- Networking

Other manages

Platform (as a Service)
- Applications
- Data
- Runtime
- Middleware
- O/S
- Virtualization
- Servers
- Storage
- Networking

Other manages

Software (as a Service)
- Applications
- Data
- Runtime
- Middleware
- O/S
- Virtualization
- Servers
- Storage
- Networking

Other manages

Copyright to
Cloud Concepts – Intro – IaaS, PaaS, SaaS

SaaS

SalesForce.com, Google Apps

Google App Engine for:
Java, Ruby, Python & GO

VMForce.com, MS Azure

PaaS

Tomcat App Server
Java VM
MySQL DB
Ubuntu Linux

IaaS

vCloud Express/Datacenter,
Amazon EC2
Cloud Concepts – Intro – IaaS, PaaS, SaaS

Copyright to
Cloud Concepts – Intro

Figure 6: Virtual Infrastructure Management

Copyright to www.vmware.com
Cloud Concepts – Public vs. Private vs. Hybrid Cloud Intro
Cloud Concepts – IaaS Intro – Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers.

Amazon EC2 Functionality

Amazon EC2 presents a true virtual computing environment, allowing you to use web service interfaces to launch instances with a variety of operating systems, load them with your custom application environment, manage your network’s access permissions, and run your image using as many or few systems as you desire.

To use Amazon EC2, you simply:

- Select a pre-configured, template image to get up and running immediately. Or create an Amazon Machine Image (AMI) containing your applications, libraries, data, and associated configuration settings.
- Configure security and network access on your Amazon EC2 instance.
- Choose which instance type(s) and operating system you want, then start, terminate, and monitor as many instances of your AMI as needed, using the web service APIs or the variety of management tools provided.
- Determine whether you want to run in multiple locations, utilize static IP endpoints, or attach persistent block storage to your instances.
- Pay only for the resources that you actually consume, like instance-hours or data transfer.
Cloud Concepts – IaaS Intro – Amazon EC2

Instance Types - Standard Instances

First Generation

First generation (M1) Standard instances provide customers with a balanced set of resources and a low cost platform that is well suited for a wide variety of applications.

- M1 Small Instance (Default) 1.7 GiB of memory, 1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit), 160 GB of local instance storage, 32-bit or 64-bit platform
- M1 Medium Instance 3.75 GiB of memory, 2 EC2 Compute Units (1 virtual core with 2 EC2 Compute Units each), 410 GB of local instance storage, 32-bit or 64-bit platform
- M1 Large Instance 7.5 GiB of memory, 4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each), 850 GB of local instance storage, 64-bit platform
- M1 Extra Large Instance 15 GiB of memory, 8 EC2 Compute Units (4 virtual cores with 2 EC2 Compute Units each), 1690 GB of local instance storage, 64-bit platform

Second Generation

Second generation (M3) Standard instances provide customers with a balanced set of resources and a higher level of processing performance compared to First Generation Standard instances. Instances in this family are ideal for applications that require higher absolute CPU and memory performance. Examples of applications that will benefit from the performance of Second Generation Standard instances include encoding, high traffic content management systems, and mem-cached.

- M3 Extra Large Instance 15 GiB of memory, 13 EC2 Compute Units (4 virtual cores with 3.25 EC2 Compute Units each), EBS storage only, 64-bit platform
- M3 Double Extra Large Instance 30 GiB of memory, 26 EC2 Compute Units (8 virtual cores with 3.25 EC2 Compute Units each), EBS storage only, 64-bit platform

Copyright to http://aws.amazon.com/ec2/
Cloud Concepts – IaaS Intro – Amazon EC2

Operating Systems

Amazon Machine Images (AMIs) are preconfigured with an ever-growing list of operating systems. We work with our partners and community to provide you with the most choice possible. You are also empowered to use our bundling tools to upload your own operating systems. The operating systems currently available to use with your Amazon EC2 instances include:

<table>
<thead>
<tr>
<th>Operating Systems</th>
<th>Windows Server 2003/2008</th>
<th>Oracle Enterprise Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Hat Enterprise Linux</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenSolaris</td>
<td>Amazon Linux AMI</td>
<td>Ubuntu Linux</td>
</tr>
<tr>
<td>Fedora</td>
<td>Gentoo Linux</td>
<td>Debian</td>
</tr>
<tr>
<td></td>
<td>SUSE Linux Enterprise</td>
<td></td>
</tr>
</tbody>
</table>

Software

Amazon EC2 enables our partners and customers to build and customize Amazon Machine Images (AMIs) with software based on your needs. We have hundreds of free and paid AMIs available for you to use. A small sampling of the software available for use today within Amazon EC2 includes:

<table>
<thead>
<tr>
<th>Databases</th>
<th>Batch Processing</th>
<th>Web Hosting</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM DB2</td>
<td>Hadoop</td>
<td>Apache HTTP</td>
</tr>
<tr>
<td>IBM Informix Dynamic Server</td>
<td>Condor</td>
<td>IIS/Asp.Net</td>
</tr>
<tr>
<td>Microsoft SQL Server Standard 2005/2008</td>
<td>Open MPI</td>
<td>IBM Lotus Web Content Management</td>
</tr>
<tr>
<td>MySQL Enterprise</td>
<td></td>
<td>IBM WebSphere Portal Server</td>
</tr>
<tr>
<td>Oracle Database 11g</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Application Development Environments</th>
<th>Application Servers</th>
<th>Video Encoding &amp; Streaming</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM sMash</td>
<td>IBM WebSphere Application Server</td>
<td>Wowza Media Server Pro</td>
</tr>
<tr>
<td>JBoss Enterprise Application Platform</td>
<td>Java Application Server</td>
<td>Windows Media Server</td>
</tr>
<tr>
<td>Ruby on Rails</td>
<td>Oracle WebLogic Server</td>
<td></td>
</tr>
</tbody>
</table>
Cloud Concepts – IaaS Intro – Amazon EC2

The pricing below includes the cost to run private and public AMIs on the specified operating system ("Windows Usage" prices apply to Windows Server® 2003 R2, 2008, 2008 R2 and 2012). Amazon also provides you with additional instances for Amazon EC2 running Microsoft Windows with SQL Server, Amazon EC2 running SUSE Linux Enterprise Server, Amazon EC2 running Red Hat Enterprise Linux and Amazon EC2 running IBM that are priced differently.

<table>
<thead>
<tr>
<th>Instance Type</th>
<th>Linux/UNIX Usage</th>
<th>Windows Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard On-Demand Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small (Default)</td>
<td>$0.065 per Hour</td>
<td>$0.115 per Hour</td>
</tr>
<tr>
<td>Medium</td>
<td>$0.130 per Hour</td>
<td>$0.230 per Hour</td>
</tr>
<tr>
<td>Large</td>
<td>$0.260 per Hour</td>
<td>$0.460 per Hour</td>
</tr>
<tr>
<td>Extra Large</td>
<td>$0.520 per Hour</td>
<td>$0.920 per Hour</td>
</tr>
<tr>
<td>Second Generation Standard On-Demand Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extra Large</td>
<td>$0.550 per Hour</td>
<td>$0.980 per Hour</td>
</tr>
<tr>
<td>Double Extra Large</td>
<td>$1.100 per Hour</td>
<td>$1.960 per Hour</td>
</tr>
<tr>
<td>Micro On-Demand Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Micro</td>
<td>$0.020 per Hour</td>
<td>$0.035 per Hour</td>
</tr>
<tr>
<td>High-Memory On-Demand Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extra Large</td>
<td>$0.460 per Hour</td>
<td>$0.570 per Hour</td>
</tr>
<tr>
<td>Double Extra Large</td>
<td>$0.920 per Hour</td>
<td>$1.140 per Hour</td>
</tr>
<tr>
<td>Quadruple Extra Large</td>
<td>$1.840 per Hour</td>
<td>$2.280 per Hour</td>
</tr>
<tr>
<td>High-CPU On-Demand Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>$0.165 per Hour</td>
<td>$0.285 per Hour</td>
</tr>
<tr>
<td>Extra Large</td>
<td>$0.660 per Hour</td>
<td>$1.140 per Hour</td>
</tr>
<tr>
<td>Cluster Compute Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eight Extra Large</td>
<td>$2.700 per Hour</td>
<td>$2.970 per Hour</td>
</tr>
<tr>
<td>Cluster GPU Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quadruple Extra Large</td>
<td>$2.36 per Hour</td>
<td>$2.60 per Hour</td>
</tr>
</tbody>
</table>
In JEE – Java Enterprise Edition:

Fundamentals: Annotations + Reflection + (Filters + Tag-libs + Generics + Servlet/JSP)

⇒ Advanced: MVC + EJB (incl. JPA) + JTA + JMS + Rules Engine/BPM-BPEL = LOVE

⇒ in Cloud PaaS Solutions for Java / Ruby / Python or GO ... Google App Engine
Thanks!

DAD – Distributed Application Development
End of Lecture 7