Com S 417
Software Testing

Fall 2017 – Week 9, Lecture 15

Announcements

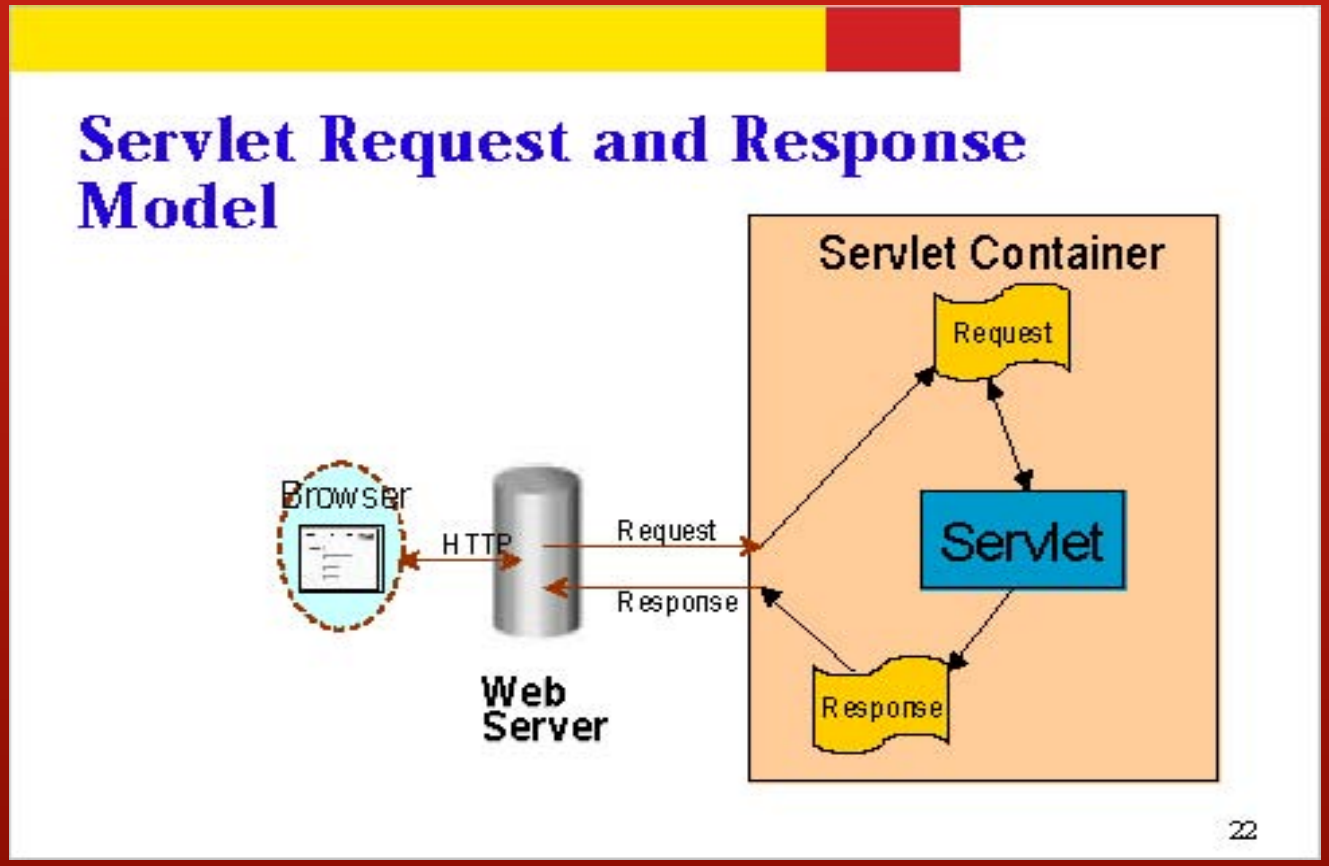
- Lab 4 will be available later today.
- We *will* have 5 labs.

Topics

- Servlet Life Cycle
- Java Web Application (.war) deployment model
- Hello World web demo
- Command Line test tools
- Issuing a request from a junit test.
- Template languages
- Hello World in JSP
- Suggested Readings.

Servlet Engine

Request/Response Model



From: materials created by S. Mitra

Request Processing

- On the client side, a URL is used to refer to the servlet. The browser connects to the server at port 80. A connection is established.
- On the server side, the web-server figures out from the URL which servlet to use. It is able to figure it out by looking at configuration files (web.xml)
- The web server packages the client request and delivers it to the servlet via the method:

```
service(ServletRequest req, ServletResponse res);
```

Response Population

- The servlet does its work and populates a response object.

- When the service() method exits, the web server sends the response object back to the client via the established HTTP connection.
  - The response is typically a html page which will be displayed at the client side by the client browser.
- The established connection is closed (in HTTP 1.0).
  - HTTP 1.1 supports a “keep-alive” mode that reduces overhead for multi-request sessions.

Servlet Life Cycle

Steps 1, 2, and 3 happen *exactly once* at container startup.

Step 4 happens once for every received request.

- Calling service() results in a call to doGet() or doPost(), depending upon the request method.

Step 5 happens once at container shutdown.

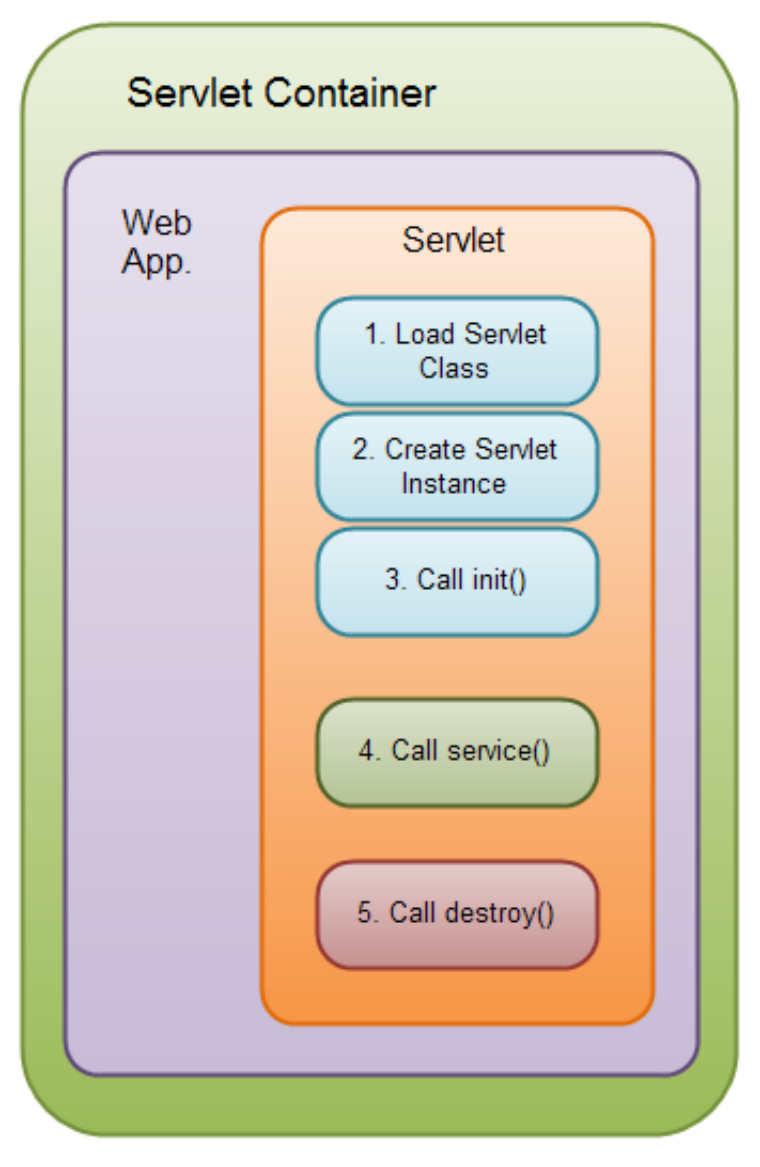


Image From: http://tutorials.jenkov.com/java-servlets/servlet-life-cycle.html

Servlet Life Cycle

- Each servlet instance is loaded once. Method init() is called once during initialization of the servlet.

- Method service() is invoked every time a request comes to the servlet. Don't override service()!
  - It spawns off threads
  - these threads perform doGet() or doPost()
  - You define the servlet's behavior by implementing (overriding) doGet and/or doPost.
    - service, doGet, and doPost are declared in class HttpServlet.

The ServletRequest object

- getServletContext(): returns the servlet context object – your connection to session and other info.
- getParameterMap(): returns the request parameters in a Map<String,String>
- getReader(): retrieves the body of the request

Plus 30-ish more methods.

Note: the ServletRequest object can be cast to HttpServletRequest to get access to more details of the HTTP request.

The ServletResponse object

- getWriter(): a handle where you can write your response.
- flushBuffer(): force output to be committed to the socket.

For control of response code, etc., you will need to cast to HttpServletResponse, where you find methods like:

- addHeader(), sendError(), sendRedirect(), setStatus() and more.

An Example (trivial) Servlet

```
public class SimpleHttpServlet extends HttpServlet {

  protected void doGet( HttpServletRequest request,
                        HttpServletResponse response)
        throws ServletException, IOException {

      response.getWriter().write("<html><body>GET response</body></html>");
  }
}
```

Note: this example makes no reference to the request object … so it's really not doing what Servlets are intended to do. It's just a minimal example of completing the loop.

Code From: http://tutorials.jenkov.com/java-servlets/httpservlet.html

Configuring Servlet Dispatch

A special file (web.xml) tells the servlet engine how to find the right servlet for a given request.

- web.xml is also important for many other container configuration details. See Homework 2.

- The mapping from request to servlet is specified in two steps:

  1. a URL pattern that maps the request to a convenient name <servlet-name> for the servlet.
  2. a mapping from the <servlet-name> to the implementing java class.

Servlet Mapping

- This web.xml would cause all requests for HTML files to be forwarded to “controlServlet”:

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

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>

  <servlet>
    <servlet-name>controlServlet</servlet-name>
    <servlet-class>com.jenkov.butterfly.ControlServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>controlServlet</servlet-name>
    <url-pattern>*.html</url-pattern>
  </servlet-mapping>
</web-app>
```

Code From: http://tutorials.jenkov.com/java-servlets/web-xml.html

Web-App deployment model

Java web-apps are packaged in special jar files with extension .war (web archive). These jars *must* have this structure.

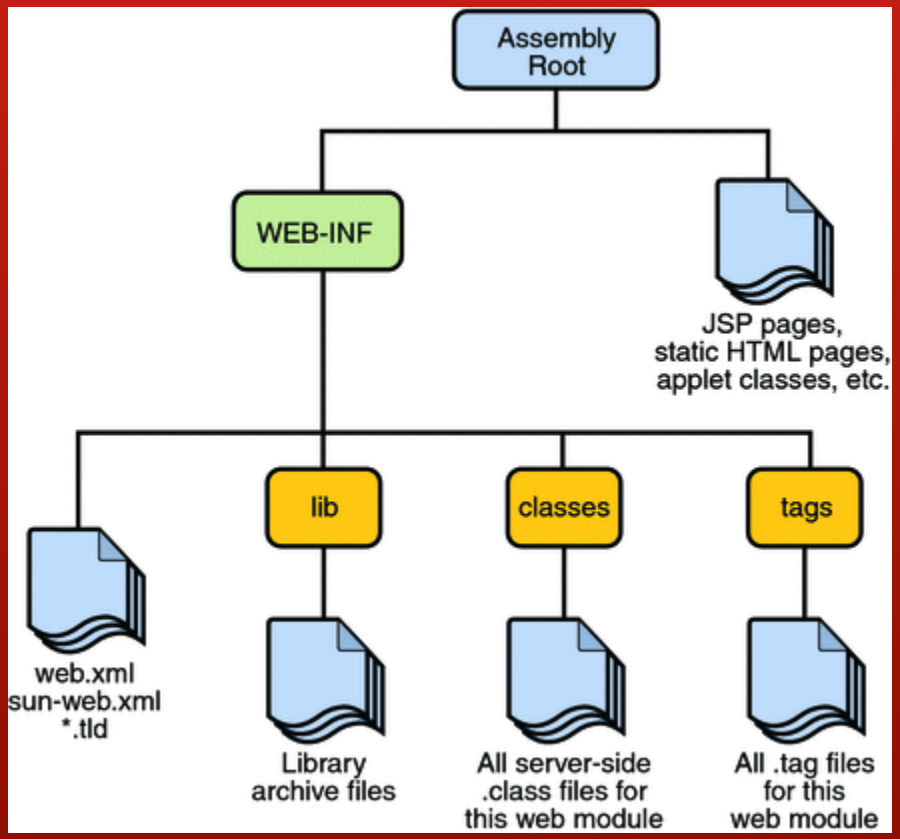


Figure from: http://docs.oracle.com/javaee/5/tutorial/doc/bnadx.html

Application Server Configuration

A Java application server is designed so it can handle more than one web application.

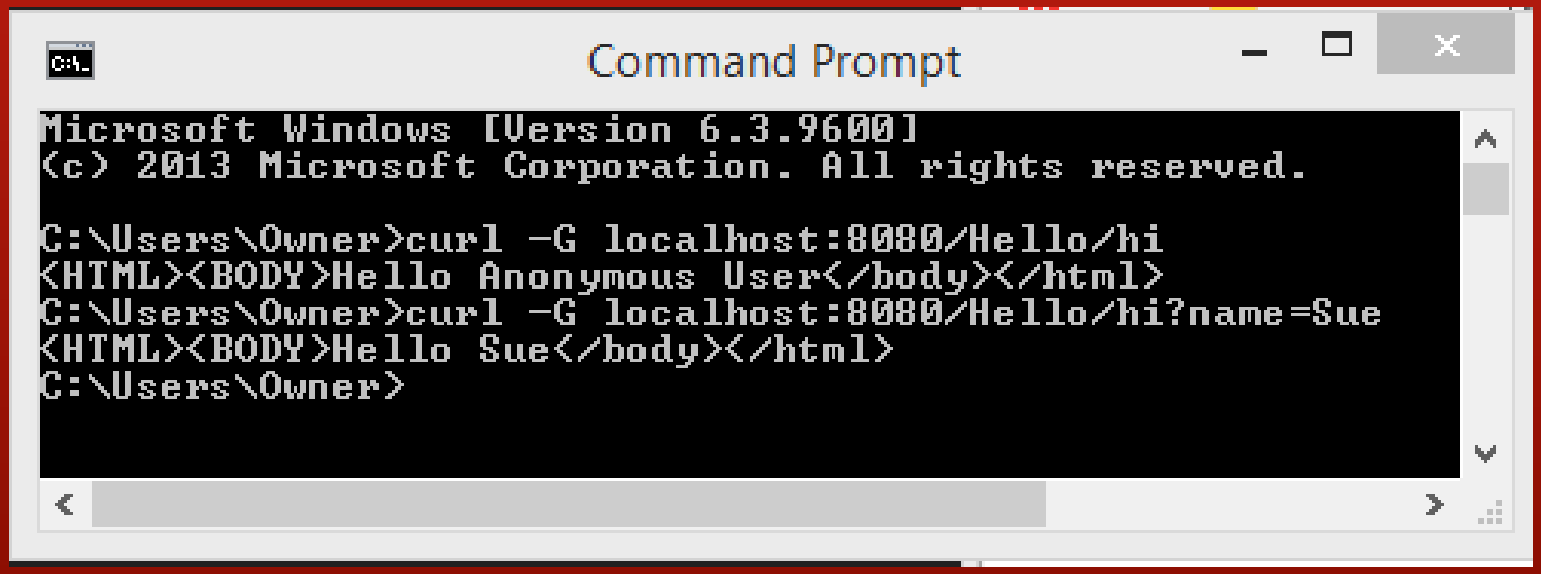
- Thus it needs to know how to find the right webapp for each request.
- Think of this as a dispatch level “above” the servlet dispatch level.
- Usually the first part of the URL (the word after the first “/”) is used to find the target webapp. This value is called the webapp context root.
- See the Java EE 5 and EE6 Tutorials for more information.

‘Hello World’ in a servlet container

- Download and install GlassFish.
- Start GlassFish and confirm that the admin page is available. (See the read-me in the zip file. You will need a console/command shell.)
- Create a new Dynamic Web project. This will let you easily create a war file and avoid J2EE .ear details.
- Create your servlet in “Java Resources/src”.
- Create a web-xml file in WebContent/WEB-INF
- Export a war file. (Pick the destination location carefully, so you can find it from the glassfish admin page.)
- deploy the war file, Launch the deployed file.
- check for the index and for your servlet.

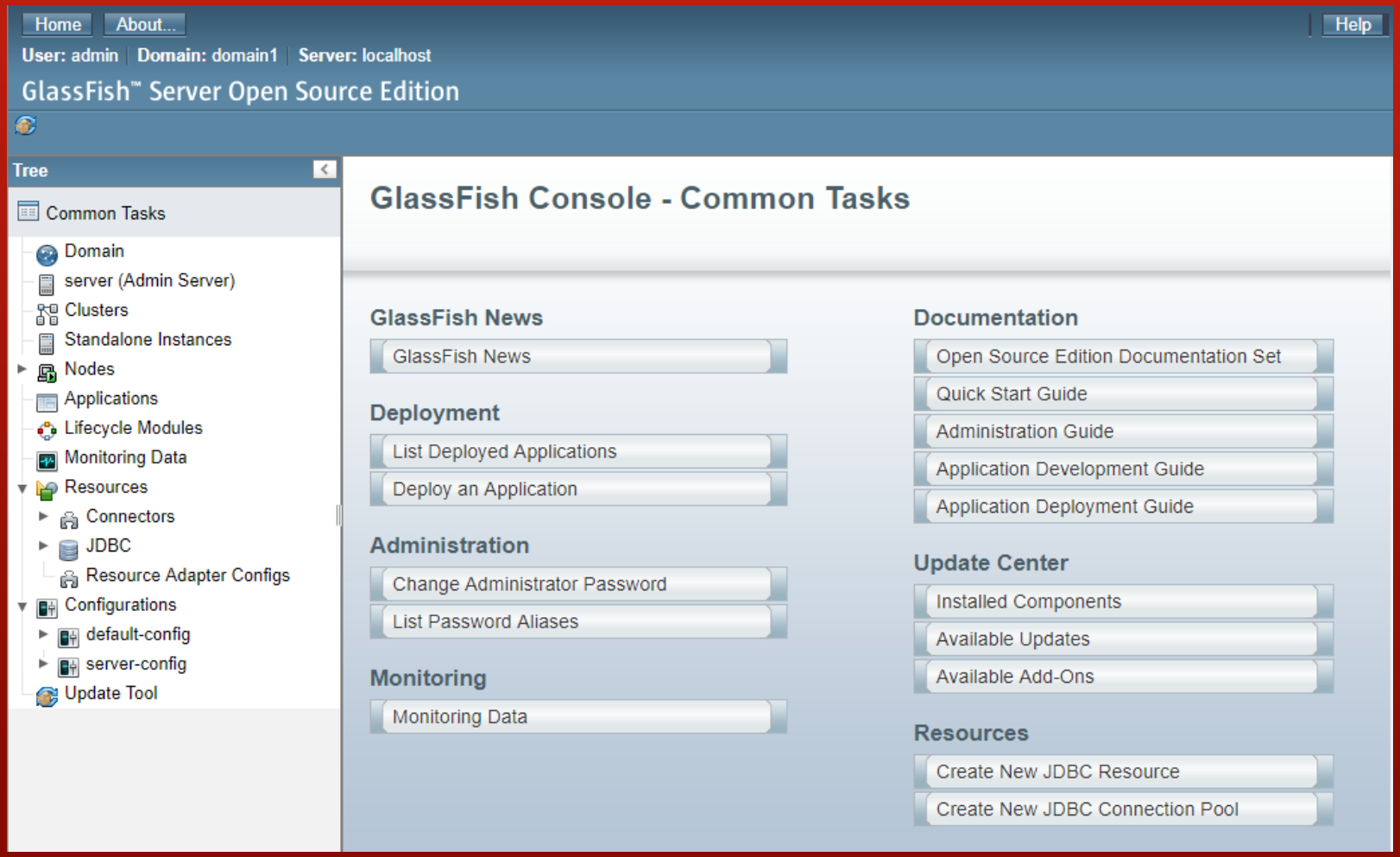
Cheat Sheet

- C:\Apps\glassfish-4.1.2-web\glassfish4\bin>asadmin start-domain
- >curl -G asus:8080/Hello/hi



GlassFish Admin Console

- Available at localhost:4848

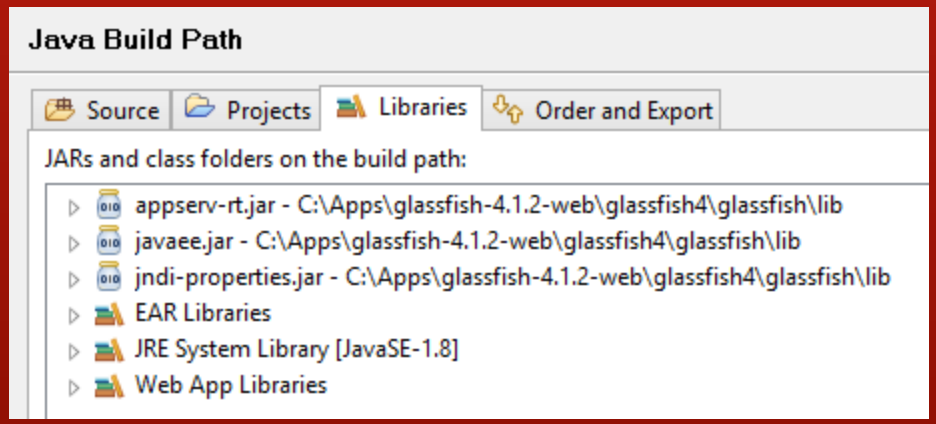


Testing without a Browser

- the unix/linux command “curl” knows how to send many different kinds of network commands.
- To send a get to the hello webapp:

Caveats

- When deploying, select the “upload” option. The “local directory” option only works if your war is within the servlet container’s directory space.
- Create your own index.html at the top level in WebContent.
- You will need to add the j2ee runtime from the glassfish distribution
- The war file name will be your context root unless you override that in the admin console.



Caveats

- You need to select the checkbox next to the running webapp in order to undeploy it.
- By default, glassfish assigns the webapp to “domain1”.
- You will probably need to give administrative permission for the servlet container to use network resources. By default, firewalls typically block the use of standard ports by unrecognized applications.
- See this page to turn on Access logging
- If you start getting memory errors or strange numbers in the context root, you probably have multiple instances running.

Enable Access Logging

- Nodes
- Applications
- Lifecycle Modules
- Monitoring Data
- Resources
  - Connectors
  - JDBC
  - Resource Adapter Configs
- Configurations
  - default-config
  - server-config
    - Admin Service
    - Connector Service
    - EJB Container
    - HTTP Service
      - Http Listeners
    - JVM Settings
    - Logger Settings
    - Monitoring
    - Network Config
    - Security
    - System Properties
    - Thread Pools

Configuration Name: server-config

SSO: ☐ Enabled

Access Logging

Access Logging: ☑ Enabled

Rotation: ☑ Enabled
Enable access log rotation

Rotation Policy: time
Rotate according to time

Rotation Interval: 1440 Minutes
Time interval between two successive rotations

Rotation Suffix: yyyy-MM-dd
Suffix to be added to the access log file name after rotation

Max File Count: -1
The maximum number of rotated access log files that are to be kept. Negative value indicates no limit.

Buffer Size: 0 Bytes
A value of 0 disables buffering

Write Interval: 10 Seconds
Interval between writing (updating) the access log. A value of 0 means the buffer is always written.

Format: common
Global format for the access log file

Tomcat vs GlassFish

- I added two slides to lecture 14 that address the key differences.

Reading Assignment

- See amended version of Lecture 14 for reading related to servlets.
- JSP Tutorial at
  https://www.tutorialspoint.com/jsp/jsp_architecture.htm
  Read sections “Architecture” through “http status codes.”
- JSTL Tutorials at
  https://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm
  Read sections :
  - Standard Tag Library, Java Beans, Custom Tags, Expression Language, Debugging