Java Web Application Components

- Compiled Java classes (.class files)
  - Servlets, beans, filters, ...
- Additional Java libraries (.jar files)
- JavaServer Pages (JSPs)
- Static resources
  - HTML, CSS, images, ...
- Metadata files
  - web.xml, ...

Directory Structure of a Java Web Application

- Application Root Directory
  - JSPs and static resources
    - WEB-INF
      - web.xml
      - classes
        - Compiled Java classes
      - lib
        - Additional Java libraries

Directory Structure on CS3

- Application Root Directory
  - JSPs and static resources
    - WEB-INF
      - web.xml
      - classes
        - Compiled Java classes
      - lib
        - Additional Java libraries

Directory Structure of an Eclipse Dynamic Web Project

- Application Root Directory
  - JSPs and static resources
    - WEB-INF
      - web.xml
      - classes
        - Compiled Java classes
      - lib
        - Additional Java libraries
      - build/classes

Servlet HelloWorld

```java
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

@WebServlet("/HelloWorld")
public class HelloWorld extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    out.println("Hello World");
  }
}
```
Some Simple Observations

- Inherits from HttpServlet
  - There’s no main() method
- doGet()
  - Input: HttpServletRequest
  - Output: HttpServletResponse
  - sent back to the client browser

Example: HelloWorld in HTML

- Modify the HelloWorld servlet to output in HTML

Generating HTML

- HttpServletResponse
- Set content type to “text/html”
  - setContentType()
- Generate an HTML page
  - getWriter().println()
    - <html>, <head>, <body> ...

Servlet Mapping

- @WebServlet(<URL Pattern(s)>)

Java Annotations

- Available since JDK 1.5 (Java 5)
- Data about a program that is not part of the program itself
- Can be used by compiler, VM, and other software tools for various purposes

Annotation Examples ...

- Error detection
  @Override
  protected void doGet()

- Suppress warning
  @SuppressWarnings("unchecked")
  public List<User> getAllUsers() {
    return (List<User>) new ArrayList();
  }
... Annotation Examples

- Servlet mapping in Servlet 3.x Specification
  ```java
  @WebServlet("/HelloServlet")
  public class HelloServlet extends HttpServlet
  ```
- Web service
  ```java
  @WebService
  public class HashService {
    @WebMethod
    public String md5(String text)
  }
  ```

About Annotations

- An annotation may have elements
- An element has a type (like a variable in Java)
- The default element is `value`
- `{}` can be omitted for array values if there’s only one value in the array

@WebServlet

- [http://download.oracle.com/javaee/6/api/java/servlet/annotation/WebServlet.html](http://download.oracle.com/javaee/6/api/java/servlet/annotation/WebServlet.html)

@WebServlet Elements for URL Patterns

- `value`
  - URL pattern(s) of the servlet
  - The default element
- `urlPatterns`
  - Same purpose as `value`
  - Usually used when more than one element is specified
  - Only one of `value` and `urlPatterns` can be specified

@WebServlet Examples

- `@WebServlet("/HelloServlet")`
- `@WebServlet("/HelloServlet","/member/*")`
- `@WebServlet(name="Hello", urlPatterns="(/HelloServlet","/*\.html")
```
- `@WebServlet(urlPatterns="/MyPattern",
               initParams=@WebInitParam(name="ccc", value="333")==>
```

Wildcard in Servlet Mapping

- A string beginning with a `/` and ending with a `/*`
  - E.g. `/*, /content/*`
- A string beginning with a `*`
  - E.g. `*.html, *.do`

See Servlet Specification 3.0, Section 12
Be Careful with URL Patterns

- Invalid patterns
  - E.g. /member/*.html, or member/index.html
- Conflicting patterns
  - E.g. two /HelloServlet
- Overlapping patterns
  - E.g. *.html and /member/*

Example: RequestCounter

- Display the number of times a servlet is requested

Servlet Life Cycle

- When the servlet is loaded – init()
  - Executed only once
  - Don't forget super.init(config)
- Per request – service()
  - dispatch to doXxx()
- When the servlet is unloaded – destroy()

Why Use init() Instead of Constructor

- Historical reasons – see http://csns.calstatela.edu/wiki/content/cysun/notes/servlet_data_init
- ServletException cannot be accessed in a constructor

Example: SharedRequestCounter

- Use one servlet to count the number of requests, and another servlet to display the count

Sharing Data among Servlets

- HttpServlet
  - getServletContext()
- HttpServletRequest
  - setAttribute(String name, Object value)
  - getAttribute(String name)
loadOnStartup

- By default, a servlet is not created until it is accessed for the first time
  - Could cause problem if one servlet must run before another servlet
- Use the loadOnStartup element of @WebServlet to have a servlet created during application startup

loadOnStartup Example

```java
@WebServlet(
    name="Hello",
    urlPatterns="{/HelloServlet", "/*.html"},
    loadOnStartup=1
)
```

The value for loadOnStartup is the order in which the application server will start the servlets.

About web.xml

- Web application deployment descriptor
  - <web-app>
    - version
  - <welcome-file-list>
- More about web.xml in Java Servlet Specification

Versions

<table>
<thead>
<tr>
<th>Servlet/JSP Spec</th>
<th>Tomcat</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0/2.2</td>
<td>7.0.x</td>
<td>1.6</td>
</tr>
<tr>
<td>2.5/2.1</td>
<td>6.0.x</td>
<td>1.5</td>
</tr>
<tr>
<td>1.4/2.0</td>
<td>5.5.x</td>
<td>1.4</td>
</tr>
</tbody>
</table>

The version attribute of <web-app> in web.xml

Debugging Servlets

- Using the Eclipse debugger
  - Set break points
  - Debug As ➔ Debug on Server
- View the source of the generated HTML
  - View Source in browser
  - Validation
    - http://validator.w3.org/
    - Use the Web Developer addon of Firefox