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
        - WebContent/WEB-INF/lib

Servlet HelloWorld

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

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

About web.xml

- Web application deployment descriptor
  - `<welcome-file-list>`
  - `<servlet>` and `<servlet-mapping>`
- More about web.xml in Java Servlet Specification

Example: HelloWorld in HTML

- Modify the HelloWorld servlet to output in HTML

Generating HTML

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

Example: RequestCounter

- Display the number of times a servlet is requested

Servlet Life Cycle

- When the servlet is loaded – `init()`
  - Executed only once
- Per request – `service()`
  - dispatch to `doXxx()`
- When the servlet is unloaded – `destroy()`
Example:
SharedRequestCounter

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

Sharing Data among Servlets

HttpServlet

- getServletContext()

HttpServletContext

- setAttribute(String name, Object value)
- getAttribute(String name)

<load-on-startup> in web.xml

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 <load-on-startup> to have a servlet created during application startup

<load-on-startup> Example

<servlet>
  <servlet-name>ServletA</servlet-name>
  <servlet-class>cs320.cysun.ServletA</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet>
  <servlet-name>ServletA</servlet-name>
  <servlet-class>cs320.cysun.ServletB</servlet-class>
  <load-on-startup>2</load-on-startup>
</servlet>

<error-page> Example

<error-page>
  <error-code>404</error-code>
  <location>/404.html</location>
</error-page>

<error-page>
  <error-code>403</error-code>
  <location>/403.html</location>
</error-page>

Debugging Servlets

Using the Eclipse debugger

- Set break points
- Debug As → Debug on Server

View the source of the generated HTML

- Validation - http://validator.w3.org/