

## Servlets

- Generic Java2EE API for invoking and connecting to “mini-servers” (lightweight, short-lived, server-side interactions).
- Mostly HTTPServlets
  - Generate HTTP responses (as for CGI)
  - Embedded via <servlet> tag into HTML pages for server-side includes
  - Servlet code snippets can be embedded directly into HTML pages: JSP = Java Server Pages
    - Competitor: Microsoft ASP = Active Server Pages

1

## Servlet Architecture

- **Clients:** web browsers i.e., IE, Netscape
- **Web server:** Apache, Netscape Enterprise, IIS
- **CGI Protocol:** Specifying what a request/response looks like
- **Servlet container:** Running JVM, hooked into the web server, loads and maintains servlets, session info, object store
- **Servlet:** A Java class used to handle a specific request.

2

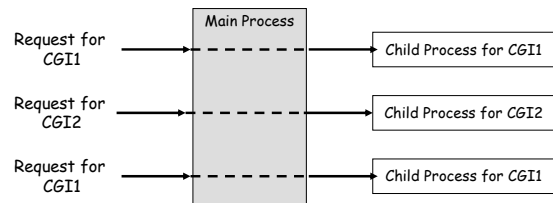
## Servlet Interaction

- Client
  - Makes a request by specifying a URL+additional info
  - Basically a method call, the method and arguments.
- Web Server
  - Receives the request.
  - Identifies the request as a servlet request
  - Passes the request to the servlet container.
- Servlet Container
  - Locates the servlet (Java code, loaded and running in the container JVM)
  - Feeds the request parameters to the servlet
- Servlet
  - Executes in a separate thread
  - The servlet can store/retrieve objects (possibly session scoped) from the servlet container.
  - Output is sent back to the requesting web browser.
  - Servlet continues to be available in the servlet container.

3

## CGI vs. Servlets

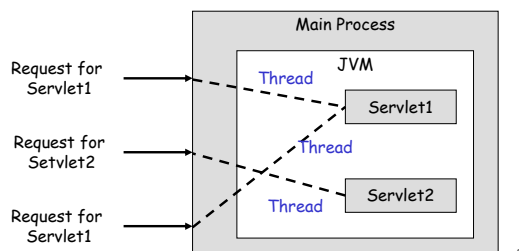
### CGI-Based Web Server



4

## CGI vs. Servlets

### Java Servlet-Based Web Server



5

## Servlets vs CGI

- Servlets...
  - Faster than CGI scripts because a different process model is used.
    - Also Java just-in-time compilation increases performance
  - Process model allows resource sharing (e.g., database connection sharing).
    - Container is always available.
  - Has all the advantages of the Java programming language, including ease of development and platform independence.
    - Same language for both client-side and server-side development
    - Can access the large set of APIs available for the Java platform.
  - Has access to CGI support through the Servlet API
  - Has session awareness, ability to store and retrieve session scoped objects from servlet container.

6

## Servlet API (javax.Servlet)

- Servlet Container
- Servlet Interface
- HttpServletResponse
- HttpServletRequest
- ServletContext

7

## Servlet Container

- Contains and manages servlets through their lifecycle
- Provides the network services over which requests and responses are sent
- Decodes requests
- Formats MIME based responses

8

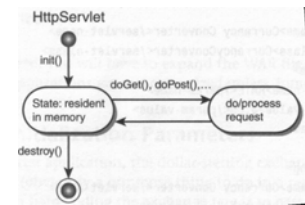
## Servlet Interface

- Central abstraction of the servlet API
- All servlets either implement this interface, or extend a class that implements this interface
- `GenericServlet` and `HttpServlet` two classes in the servlet API that implement the Servlet interface

9

## HttpServlet

- Web developers extend `HttpServlet` to implement their servlets.
- Inherited from `GenericServlet`
  - `init()`
  - `destroy()`
- Defined by `HTTPServlet`
  - `doGet()`
  - `doPost()`



10

## doGet() and doPost()

```
protected void doGet(HttpServletRequest req,  
HttpServletResponse resp)
```

```
protected void doPost(HttpServletRequest req,  
HttpServletResponse resp)
```

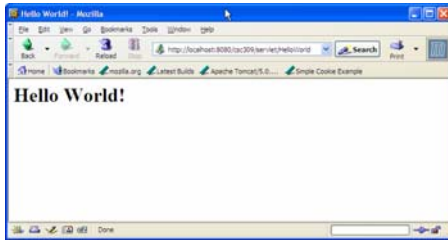
11

## HttpServletResponse

- Encapsulates all information to be returned from the server to the client
- In the HTTP protocol, this information is transmitted from the server to the client either by HTTP headers or the message body of the request.
- **Headers:** Can set/manipulate http headers
  - primitive manipulation: `setStatus()`, `setHeader()`, `addHeader()`
  - convenience methods: `setContentType()`, `sendRedirect()`, `sendError()`
- **Body:** Obtain a `PrintWriter`, used to return character data to the client
  - `getWriter()`

12

## Example: Hello World



13

## Example: Redirection

<http://localhost:8080/csc309/servlet/OldPage>

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

public class OldPage extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    {
        response.sendRedirect("NewPage");
    }
}
```

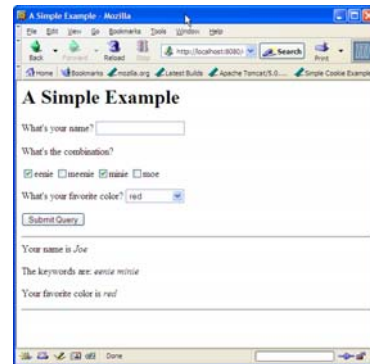
14

## HttpServletRequest

- Encapsulates all information from the client request
- In the HTTP protocol, this information is transmitted from the client to the server in the HTTP headers and the message body of the request.
- Manipulate 'form' variables:  
`getParameter()`,  
`getParameterNames()`,  
`getParameterValues()`

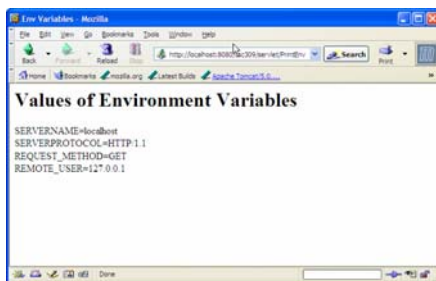
15

## Example: Form



16

## Example: Environment Variables



17

## Session Maintenance

- HTTP is stateless by default
  - No memory of browsers state
- Preserving state between browser/web server interactions
  - Probably determined by previous forms filled in and pages viewed
  - Kept at the browser or/and web server.
- **Example:** Want to allow access to a given script only once a user is logged in. Don't want to require a user to login at every page.

18

## Session Maintenance

- Adding state information
- Where
  - At client
    - Cookies
    - Hidden variables
    - URL rewriting
  - At server
    - Database, file system, etc.

19

## Cookies

- Store at the browser
- Are **name=value** pairs (like parameters in a HTTP query)
- Servlets can create cookies and send them to the browser in the HTTP header.
- The browser maintains a list of cookies that belong to a particular Web server, and returns them to the web server in subsequent interactions.
- Supported by Netscape 1.1 (and later), Internet Explorer
  - **Note:**
    - Browsers limit the size of cookies.
    - Users can refuse to accept them

20

## Creating Cookies

HTTPServletResponse      addCookie(Cookie cookie)

Cookie      Cookie(String name, String value)  
              setMaxAge(int expiry)  
              setDomain(String pattern)  
              setPath(String uri)  
              setSecure(boolean flag)

**MaxAge** - Stored until expiration time. If negative, cookie persists until browser exits.

**Domain** - Return cookie to any server in this domain. Otherwise only servers on host sending cookie.

**Path** - Cookies sent only to scripts within the path. If not specified, return cookies to any script.

**Secure flag** - Return cookies only on a secure (SSL) channel

21

## Retrieving Cookie Values

HttpServletRequest      public Cookie[] getCookies()

Cookie      getName()  
              getValue()  
              getDomain()  
              getPath()  
              getSecure()

22

## CookieExample.java

- User logs in
  - Check password
- Count how many times user has visited page
- User logs out

23

## CookieExample.java



24

## CookieExample.java

- Two cookies: loggedIn & numTimes
- Two parameters: login & password
- If loggedIn cookie exists
  - numTimes++
- else if login!=hello or password!=world
  - Error, ask user to try again
  - Return
- Set cookie loggedIn
- Set cookie numTimes
- Display numTimes
- Add link to CookieExample servlet
- Add link to CookieLogout servlet

25

## CookieLogout.java

- Set lifetime of loggedIn and numTimes to zero
- Add link to CookieExample servlet

26

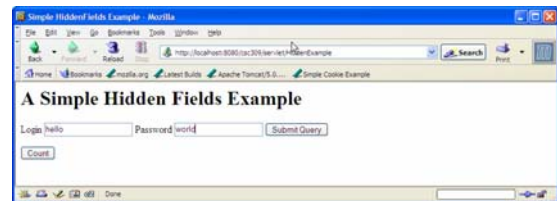
## Hidden Variables

- Store state in web pages: Add hidden variables.
- Supported by all browsers
- Requires all hidden variables to appear in all forms.  

```
<input type="hidden" name="secret" value="Don't tell anyone!!">
```
- State is sent inside each web page.
- For form based applications only. Following hyperlinks causes a loss of state.
- Current submitted page represents current state independent of what was done previously.

27

## HiddenExample.java



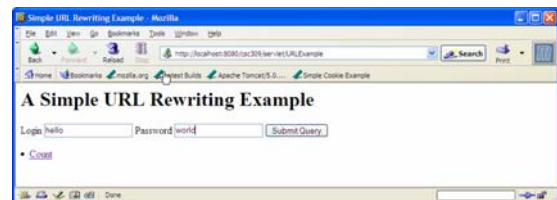
28

## URL Rewriting

- Store state in the URL: Rewrite URLs so that they include state variables
- Each URL is now a CGI-get request
- Supported by all browsers
- Requires all URLs contain all state information (long URLs)
- Current submitted page represents current state independent of what was done previously.

29

## URLExample.java



30

## Store State At Server

- **Current state is stored at the server** (i.e., in a file, database, or in JVM's memory)
- Each request includes a **token** identifying the browser's session (tokens can be passed via cookies, hidden variables, URL rewriting).
- At each request, the executing servlet uses the token to fetch session state
- **Careful!** Browser back button problem. The page the user is viewing may not reflect state stored at the server.

31

## HttpSessions

- Allows a servlet container to use any of several approaches (cookie, hidden variables, URL rewriting) to track a user's session without involving the Application Developer in the nuances of any one approach.
  - Usually associated with a single user browsing within a single Web applications

32

## Session Tracking Mechanisms

- A **session token** is passed back and forth between the client and servlet container. The token allows the container to associate a session with a particular user.
- The **servlet container adapts** to whether the client (browser) is accepting cookies.
- **Cookies:** If the client accepts cookies, sessions will be identified by a token stored on the browser as a cookie.
- **URL Rewriting:** If the client is not accepting cookies, the session token is communicated via the URL. Use the `HttpServletResponse.encodeURL()` method.

33

## HttpSession

- One instance for each active session
- Accessing/creating a session

```
HttpSession session = request.getSession();
```
- Session attributes

```
Object o = session.getAttribute("name");  
session.setAttribute("name", o);
```

34

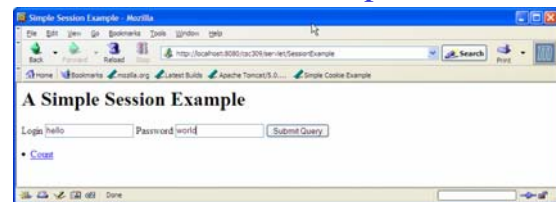
## Session Duration

- Default: expire a session if idle (no http requests) for 30 minutes
  - Delete all session related attributes from server
  - Does not remove cookie from client
- Changing session timeout
  - Configuration file (web.xml)

```
<session-config>  
<session-timeout>30</session-timeout>  
</session-config>
```
  - Run-time
    - `session.getMaxInactiveInterval()`
    - `session.setMaxInactiveInterval()`
- Terminating a session
  - `session.invalidate()`
  - Deletes session related attributes from server
  - Removes cookie from client

35

## Session Example



Run with cookies off and on, notice the URLs.  
Shut down and restart your browser between accepting and rejecting cookies.

36

## Other Issues

- Multithreading
  - HttpSession is not thread safe
  - Use proper synchronization to prevent concurrent property modifications
- Persistent sessions
  - Tomcat has support for persistent sessions
    - Survive server restart
  - Saves session data to a file
  - Serializes properties
    - Property values need to be serializable
- Session event listeners
  - Possible to create servlets that execute when a session times out
  - Can be used to garbage collect resources outside of the JVM

37

## ServletContext

- Interface defines a servlet's view of the web application within which the servlet is running.
- Container Provider is responsible for providing an implementation of the ServletContext interface in the servlet container.
- Using the ServletContext object, a servlet can
  - log events
  - obtain URL references to resources
  - set attributes that other servlets in the context can access.

38

## ServletContext Example

All servlets associated with this lecture are part of one application, the `csc309` servlet context.

References to servlets within that application are via the URL

`http://127.0.0.1:8080/csc309/`

39

## ServletContext Continued

- One instance of ServletContext per web application per container

```
ServletContext context = getServletContext();
```

- Provides access to Initialization Parameters
  - Defined in the deployment descriptor
    - A file called `web.xml` in the `WEB-INF` subdirectory of the app directory

- **Methods:**

```
getInitParameter(),  
getInitParameterNames()
```

40

## ServletContext Attributes

- Context level shared objects. Can be bound into the context and accessed by **any** servlet that is part of the application.
- **Methods:**

```
setAttribute()  
getAttribute()  
getAttributeNames()  
removeAttributes()
```
- More than 1 thread may be accessing a shared object.
- Application developer responsible for synchronizing access to shared objects

41

## ServletContext Resources

- Gives servlet access to local documents associated with the application. (i.e. static web pages, images)
- Access to server logs

- **Methods:**

```
getResource()  
getResourceAsStream()  
getResourcePaths()
```

```
log("Error! File does not exist");
```

42

## File Uploading

- Client side
  - `<input type="file" name="myFile" />`
    - Allows the user to enter a filename that can be used as input for the form.
  - `<form action="upload" method=post enctype="multipart/form-data" />`
    - RFC2388 (<http://www.faqs.org/rfcs/rfc2388.html>)
    - Internet media type for communicating a set of values as the result of a user filling out a form.
    - Follow MIME multipart format
- Server side
  - Process multipart/form-data from stdin
  - commons-fileupload-1.0.jar

43

## FileUpload.java

```
<FORM action="/servlet/UploadFile" enctype="multipart/form-data" method="post">
  What is your name? <INPUT type="text" name="submit-name"><BR>
  What files are you sending? <INPUT type="file" name="files"><BR>
  <INPUT type="submit" value="Send">
  <INPUT type="reset">
</FORM>
```

```
Content-Type: multipart/form-data; boundary=AaB03x
```

```
--AaB03x
Content-Disposition: form-data; name="submit-name"
```

```
Larry
--AaB03x
Content-Disposition: form-data; name="files"; filename="file1.txt"
Content-Type: text/plain
```

```
... contents of file1.txt ...
--AaB03x--
```

44

## JSP

- Servlets disadvantage
  - A little java in a whole lot of HTML
- Java Server Pages
  - Java embedded in HTML
  - <http://localhost:8080/csc309/hello.jsp?name=Eyal>
- Deployed in textual form (as opposed to byte code)
- Translated into a servlet on the fly

45

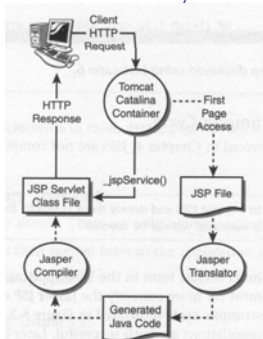
## JSP Example

```
<html>
<head>
<title>JSP Example</title>
</head>

<body>
<h1>JSP Example</h1>
<hr>
<% out.println("Hello " + request.getParameter("name")); %>
<p>
<%= "Hello again " + request.getParameter("name") + "!"%>
</body>
</html>
```

46

## JSP Life Cycle

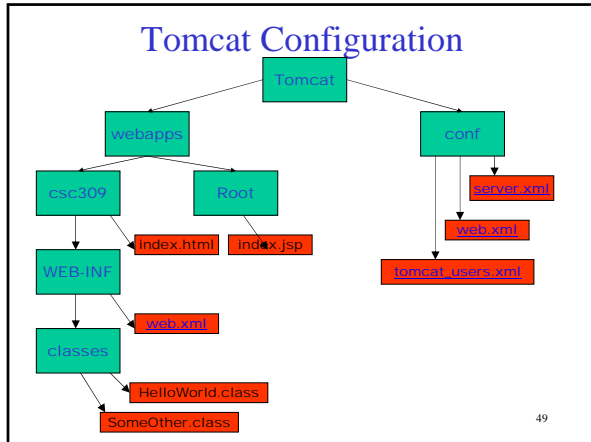


47

## ASP

- Microsoft-specific technology
- Active Server Pages
- Very similar to JSP
  - Uses VisualBasic instead of Java
  - Can access a vast array of COM and DCOM objects

48



- ### Environment Variables
- CATALINA\_HOME
    - Location of the Tomcat root directory
  - JAVA\_HOME
    - Location of the JVM
- 50

- ### \$CATALINA\_HOME/conf/server.xml
- Main configuration file
  - Defines a number of containers (connectors)
  - Connectors
    - Handle request for a specific protocol
    - Coyote (HTTP and Apache Jserv Protocol (AJP))
      - AJP integrates Tomcat with another web server
    - Webapp (Web Appl. Remote (AccessControl)+ Protocol (WARP))
      - Integrates Tomcat with Apache
    - Port number
    - Maximum number of threads
    - Request queue size
    - Default context
- 51

- ### Deployment Descriptor (web.xml)
- \$CATALINA\_HOME/conf/web.xml
    - Default for all web applications
  - \$CATALINA\_HOME/webapp/csc309/WEB\_INF/web.xml
    - Specific to csc309 folder
- ```

<servlet>
  <servlet-name>SampleServerName</servlet-name>
  <servlet-class>HelloWorld</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>SampleServerName</servlet-name>
  <url-pattern>/myDirectory/Alias</url-pattern>
</servlet-mapping>
  
```
- <http://127.0.0.1:8080/csc309/myDirectory/Alias>  
 Maps to: \$CATALINA\_HOME/webapp/csc309/WEB\_INF/classes/HelloWorld.class
- 52

### Tomcat Web Application Manager

- Deploy applications
- Start/stop/reload

53

### Tomcat Administrator Tool

- Server properties
  - Host name
  - Ports
- App. Properties
  - Reloadable
  - # active sessions
- Users and roles

54

## `$CATALINA_HOME/conf/tomcat_users.xml`

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <role rolename="manager"/>
  <role rolename="admin"/>
  <user username=" " password=" " fullName="name" roles="admin,manager"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="role1" password="tomcat" roles="role1"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
</tomcat-users>
```

55

## Servlet Debugging

- Use Java JPDA interface
  - Allows an external program to “attach” to process
- Start Tomcat in debugging mode
  - `$CATALINA_HOME/bin/catalina.sh jpda start`
- Start IDE debugger
- Attach debugger to running Tomcat instance

56