Advanced Java Application Development Using JavaEE

Day 3

University of Colombo School of Computing

Agenda

- Setting up environment
- Servlets continued from Day 2
- Sessions
- Events and Listeners
- Filters
- JSP

Sessions

Basics

- HTTP is a stateless protocol. So how does it know who we are?
- Four ways of managing sessions
  - Cookies
  - URL Rewriting
  - HttpSession
  - Hidden fields

HttpSession…(1)

- This is an interface that provides a way to identify a user across the website. (multiple visits, pages etc.)
- The web container generates an unique session ID and identifies the user.
- How does the request response model handle this?

HttpSession…(2)

- Creating a session

```
HttpSession session = request.getSession();
```

- Retrieving a session

```
HttpSession session = request.getSession(false);
```

- Destroying a session

```
session.invalidate();
```


Cookies…(1)

- Cookies are small files that are transferred from the server to the client.
- Initial cookie is transferred with the initial response to the client request.
- Cookies are stored in a predefined location in client machine.

Cookies…(2)

- Cookies have a predefined lifetime and it can be configured.
- Cookies are added to the response object using the `addCookie()` method.
- The `getCookies()` method retrieves the cookies from the request object.

Cookies…(3)

- Creating a cookie

```
Cookie cookie = new Cookie(<attribute name>, <value>);
```

- Adding to the response

```
response.addCookie(cookie);
```

- Retrieving from the request

```
Cookie[] cookies = request.getCookies();
```


URL Rewriting

- This method is used when the client browser has disable the cookies.
- Disabling cookies disables the browsers ability to identify the user.
- URL Rewriting will always work
- Standard syntax

```
response.encodeURL("redirect URL");
```


Events & Listeners

What is an Event?

- In general an event represents an occurrence of something.
- There are two types of servlet events
  - Servlet context-levels
  - Session level
- The servlet events can be of two other subcategories (4
  - Lifecycle changes (Initialization/Destroying of an application)
  - Attribute changes (modification of attributes)

Servlet Event Listeners

- Event listener listens to the events occurring in the web container.
- Listeners can be implemented by implementing the appropriate interfaces.
  - javax.servlet
  - javax.servlet.http
- Four categories of events
  - Servlet context lifecycle changes → javax.servlet.ServletContextListener
  - Servlet context attribute changes →javax.servlet.ServletAttributeListener
  - Session context lifecycle changes → javax.servlet.http.HttpSessionListener
  - Session context attribute changes → javax.servlet.http.HttpSessionAttributeListener

Setting up a Listener …(1)

- The event listeners are declared in the `web.xml` file directly under the `<web-app>` tag.
- Each listener declaration is within the `<listener></listener>` tags.
  - The `<listener-class>` specifies the actually class implementation of the listener.
- How is listeners setup at the start of a servlet?

Setting up a Listener …(2)

- Important methods

Listener	Methods
`ServletContextListener`	`contextDestroyed(ServletContextEvent e)` `contextInitialized(ServletContextEvent e)`

`ServletRequestAttributeListener`
`ServletContextAttibuteListener`
	`attributeAdded(…)`
	`attributeRemoved(….)`
	`attributeReplaced(…)`

`HttpSessionListener`	`sessionCreated(HttpSessionEvent e)` `sessionDestroyed(HttpSessionEvent e)`
`ServletRequestListener`	`requestDestroyed(servletRequestEvent e)` `requestInitialized(servletRequestEvent e)`

Filters

What are filters? …(1)

- As its name suggests filters are components that are capable of performing filtering tasks on responses and requests.
- Filters preprocess requests and post process responses.
- Generally used for preprocessing and post processing of requests and responses for a group of servlets.
  - For example encrypting and decrypting information sent and received before being processed by the servlet.
- Like other components filters too are configured using the `web.xml`

What are Filters? …(2)

- Like listeners, filters are not servlets.
- Filters implement `javax.servlet.Filter` interface.
- There are three main methods
  - `init()`
  - `destroy()`
  - `doFilter()`
- The order of execution of a filter is the order of specification in the `web.xml` in request and reverse in response.

Setting up a Filter

- Like defining servlets filters are defined using the `<filter></filter>` tags in the web.xml
  - Within these tags there are two mandatory tags
  - `<filter-name>` contains the name given to the filter
  - `<filer-class>` contains the class file of the filter implementation.
- After defining the filter, a URL mapping is needed to ensure the filter executes each time the URL pattern is found.
  - Within these tags two of the following are mandatory
  - `<filter-name>` contains the name given to the filter
  - `<url-pattern>` contains the URL pattern that the filter will execute
  - `<servlet-name>` contains the servlet name which the filter will fiter requests and responses.

Java Server Pages (JSP)

What is JSP? …(1)

- Nothing much but HTML and some java code.
- Presents dynamic content
- Handles the presentation logic in a MVC architecture.
- Primary difference between servlets and jsp are its purpose of use.
  - Servlets → for business logic processing (HTML in Java)
  - JSP → presentation logic processing (Java in HTML)

What is JSP? …(2)

- Has three lifecycle methods as the servlets
  - jspinit()
  - _jspService()
  - jspDestroy()
- In JSP methods that start with an underscore ("_") character cannot be overridden.
- What ever is written in the JSP file, it will eventually be translated to a servlet.

JSP Elements

- There are four types of JSP elements
  - Declarations
  - Scriptlets
  - Expressions
  - Directives

JSP Declarations

- Use `<%! … %>` to wrap declarations
- Instantiate objects/variables/methods for use in scriptlets
- E.g. `<%! int k = 5 %>`
- The code is placed outside of the `service()` method.
- Represents reusable code.

JSP Scriptlets

- Use `<%...%>` to wrap code blocks
- Contains pure java code.
- Business logic pertaining to view goes in scriptlets. (e.g. validations)
- May need to have imports using the page directives
- Code is directly placed in the `service()` method.

JSP Expressions

- Use `<%= … %>` to wrap an expression
- Evaluated and the output is displayed to the client.
- Whatever the code inserted in between the expression tags should be able to evaluate to a value.

JSP Directives … (1)

- These are preprocessing commands given to the JSP engine.
- These contains no java code or business logic
- Three types of directives (Include, Page, Tag Library)
- Use `<%@ directive …. %>` to wrap an expression
  - Include → <%@ include file="header.jsp" %>
  - Page → <%@ page import ="nav.html"%>
  - Tags → <%@ taglib uri="TAGS" prefix=".." %>

JSP Directives … (2)

- Page directive is applied to the current JSP page and contains a number of predefined attributes
  - Import → imported content like java.util.*;
  - isThreadSafe → true by default
  - isSession → true by default
  - isErrorPage → false by default
- The include directive is used whenever a file needs to be included in a JSP file.
  - For example the header and the footer of the webpage.
  - Included at the translation time (very important to observe this fact)

JSP Directives …(3)

- Tag libraries are used to clean up the JSP page with the addition of common custom JSP logic and HTML generation.
- Contains three basic components
  - Tag handler class
  - The mapping descriptor file
  - Taglib directive

JSP Implicit Objects

JAVA API	Implicit Object
HttpServletRequest	request
HttpServletReponse	response
HttpSession	session
ServletContext	application
ServletConfig	config
PrintWriter	out
Throwable [Exception]	Exception
PageContext	pageContext
this [Object]	page

Standard Actions

- Standard actions are available to JSP pages
- Standard actions take the form <jsp:…>…</jsp:…> and are pre-defined

Action	Description
forward	forward the request
useBean	work with objects [get/create]
setProperty	set the bean object properties
getProperty	get the bean object properties
include	Imports resources [html, jsp, etc]

Java Beans

- A class of any name that is serializable
- Contains a default, no-argument constructor
- All properties start with get/set (boolean get can be isProperty)
- Other properties named in camel-case
- Not the same thing as an Enterprise Java Bean