Advanced Programming Concepts in Java

Java Servlets

11–06–2013
Outline

- Java Servlets & JSP
  - J2EE
  - Apache tomcat
  - JavaBeans

Read:
- Java EE Tutorial on Servlets & JSP

Exam#2 is scheduled for Tues., Nov. 19, 7:00 pm, Snell 213
review session:  Monday, 11/11, 5:30 to 7:00 pm, ITL
J2EE Web Components

Web Browser
Web Pages, Applets, and Optional JavaBeans Components
Application Client and Optional JavaBeans Components

JSP Pages
Servlets

JavaBeans Components (Optional)

Business Tier

J2EE Server
Web applications

- Web browsers – Mozilla, Safari, Chrome,…
- Web server apps – Apache
Basic Architecture

- Web client lets a user request a resource and shows the user the result of the request
- Web server gets the request, finds the resource, and returns something to the user
  - resources: HTML page, image, video, sound, ...
- HTTP protocol – simple request and response conversations (runs on top of TCP/IP)
Java2 Enterprise Edition (J2EE)
- J2EE extends J2SE (standard edition) adding functionality for server programming, including networking, distributed computing, and web services

Servlet
- runs on a HTTP web server, and extends the functionality of the server

Enterprise JavaBean (EJB)
- encapsulates server-side computations, particularly for business logic

Apache Tomcat
- open source web server and servlet container for Java servlets and Java Server Pages (JSP)

Key idea: J2EE combines the use of servlets and EJB’s
Advantages of Servlets

- fast – unlike CGI scripts which run a separate process, servlets run in a thread within the web service

- platform independence – CGI scripts are platform dependent and cannot access all the server’s capabilities (like writing to a server’s log file)

*CGI (Common Gateway Interface)– Perl script, C, Python, PHP, …*
Other applications of Servlets

- allows collaboration among people – servlet can handle multiple requests concurrently, so can support systems such as on-line conferencing

- servlets can forward requests to other servers and servlets, so can be used to balance the load among a group of servers that mirror the same content
Getting Started with Java Enterprise Edition

- Need Java EE 7 SDK
  - comes with the GlassFish Open Source Server
- Contains packages:
  - javax.servlet
  - javax.servlet.http
- Scroll way down the page to “Additional Resources”
  - Download Java EE 7 SDK documentation
- Head First Servlets & JSP, 2nd Ed. is a good reference
Servlets don’t have a main() method. They are under the control of another application called a container. When the web server gets a request for a servlet, it gives it to the container in which the servlet is “deployed”.

Examples of containers:
- Apache tomcat (used in this course)
  - [http://tomcat.apache.org](http://tomcat.apache.org)
- GlassFish (packaged with NetBeans)
- standalone servletrunner
- IDEs such as WebSphere provide tools to create and compile servlets
Containers

- Communications support
  - provides an easy way for your servlets to talk to the web server (no need for ServerSockets, etc.)

- Lifecycle management
  - instantiates and initializes the servlet, invokes servlet methods, marks servlet for garbage collection when the time comes

- Multithreading support
  - container automatically creates a new Java thread for each servlet request and manages threads for multiple requests
Containers, continues

- Declarative Security
  - uses XML deployment descriptor to configure security
- JSP support
  - translates JSP code into Java

cf. Head First Servlets & JSP
Apache Tomcat

Apache Software Foundation: supports open-source software projects; one is Tomcat

- webpage: tomcat.apache.org
- download Tomcat 7.0.47
  - binary distributions and documentation
  - e.g. c:\apache-tomcat-7.0.47-windows-x64
- install Tomcat and set Environment Variables
  - read the text file “running” in apache-tomcat-7.0.47
    - Environment Variables: $CATALINA_HOME and $CATALINA_BASE (also $JAVA_HOME & $JRE_HOME)
Running Tomcat

- $CATALINA_HOME\bin$ contains the following:
  - startup.bat & shutdown.bat (for Windows)
  - startup.sh & shutdown.sh (for Unix)

- Run startup
  - if successful, the Tomcat server will start running on your computer
  - open a web browser & type the following URL: localhost:8080
  - you will see the startup Tomcat page

- Check out the examples of servlets & JSP

- Run shutdown to terminate Tomcat
Servlet Examples with Code

This is a collection of examples which demonstrate some of the more frequently used parts of the Servlet API. Familiarity with the Java(tm) Programming Language is assumed.

These examples will only work when viewed via an http URL. They will not work if you are viewing these pages via a "file://..." URL. Please refer to the README file provide with this Tomcat release regarding how to configure and start the provided web server.

Wherever you see a form, enter some data and see how the servlet reacts. When playing with the Cookie and Session Examples, jump back to the Headers Example to see exactly what your browser is sending the server.

To navigate your way through the examples, the following icons will help:

- Execute the example
- Look at the source code for the example
- Return to this screen

Tip: To see the cookie interactions with your browser, try turning on the "notify when setting a cookie" option in your browser preferences. This will let you see when a session is created and give some feedback when looking at the cookie demo.

Hello World
Request Info
Request Headers
Request Parameters
Cookies
Sessions

Note: The source code for these examples does not contain all of the source code that is actually in the example, only the important
JSP Samples

This is a collection of samples demonstrating the usage of different parts of the Java Server Pages (JSP) specification. Both JSP 2.0 and JSP 1.2 examples are presented below.

These examples will only work when these pages are being served by a servlet engine; of course, we recommend Tomcat. They will not work if you are viewing these pages via a "file://..." URL.

To navigate your way through the examples, the following icons will help:

- Execute the example
- Look at the source code for the example
- Return to this screen

Tip: For session scoped beans to work, the cookies must be enabled. This can be done using browser options.

JSP 2.0 Examples

Expression Language

Basic Arithmetic  
Basic Comparisons  
Implicit Objects  
Functions  
Composite Expressions
More JSP Examples

**Other JSP 2.0 Features**
- `<jsp:attribute>` and `<jsp:body>`
- Shuffle Example
- Attributes With Dynamic Names
- JSP Configuration

**JSP 1.2 Examples**
- Numberguess
- Date
- Snoop
- ErrorPage
- Carts
- Checkbox
- Color
- Calendar
- Include
**HTTP requests**

- **HTTP GET**
  - example – user clicks a link to a new page
  - browser sends an HTTP GET to the server asking it to GET the page

- **HTTP POST**
  - example – user types in a form and hits the submit button
  - browser sends an HTTP POST to the server, giving it the data the user typed in

HTTP response contains a header (protocol, status, MIME type, date, etc.) followed by a body containing HTML or whatever content is to be displayed.
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Hello World!</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello World!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloDate extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response) throws IOException {
        PrintWriter out = response.getWriter();
        java.util.Date today = new java.util.Date();
        out.println("<html>
            <body>
                <h1 align=center>Hello Date!</h1>
                <br>
                today
            </body>
        </html>" );
    }
}
Java Server Pages (JSP)

- specifies the static HTML code and the dynamic information that the user of the web application sees in a browser
- provides a connection to JavaBeans that carry out computations
- each JSP page is automatically translated into a servlet

idea: separate programming & presentation
What is JSP?

- Mostly HTML page, with extension .jsp
- Include JSP tags to enable dynamic content creation
- Translation: JSP → Servlet class
- Compiled at Request time (first request takes a bit longer)
- Execution: Request → JSP Servlet's service method
JSP objectives

- To implement dynamic web pages with JavaServer Pages technology
- To learn the syntactical elements of JavaServer Pages
- To structure a web application as a sequence of JavaServer Pages
- To understand the relationship between JavaServer Pages and servlets
A JavaServer Page (JSP) page contains HTML tags and Java instructions.

The Java instructions are executed each time the page is served to the browser.

An instruction to insert the current date and time into a web page:

<%= new java.util.Date() %>
JSP/ASP/PHP vs CGI/Servlets

- CGI & Servlets -- **Mostly Code** with some HTML via print & out.println

- JSP/ASP/PHP -- **Mostly HTML**, with code snippets thrown in
  - No explicit recompile
  - Great for small problems
  - Easier to program
  - Not for large computations
<html>
<head>
<title>HelloDate JSP</title>
</head>
<body>
<h1>HelloDate JSP</h1>
<p>The current time is: <%= new java.util.Date() %></p>
</body>
</html>

Where you put this file is very important!
Placement of Web Application Files

$CATALINA_BASE

webapps

WEB-INF

classes

HTML, JSP files

web.xml

bean, servlet class files

cs242
To Deploy the Date Page

1. Type the JSP file into a text editor
2. If you use Tomcat, you may want to create a subdirectory for the JSP file in $CATALINA_HOME\webapps
c:\apache-tomcat-7.0.23\webapps\cs242
4. Place the date.jsp file into that directory
5. Startup Tomcat
6. Point your browser to localhost:8080/cs242/date.jsp
Executing the Date Page

HelloDate JSP

The current time is: Wed Nov 14 13:14:11 EST 2012