COMP201 Java Programming

Part III: Advanced Features

Topic 14: Servlets

Servlets and JavaServer Pages (JSP) 1.0:

A Tutorial

http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/Servlet-Tutorial-Intro.html

Objective & Outline

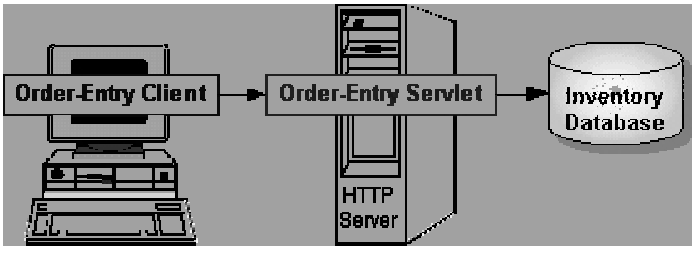
- Objective: Introduction to servlets
- Outline:
  - Introduction
  - A simple servlet
    - Environment for developing and testing servlets
  - General information about servlets
  - HTTP Servlets
  - Session tracking
  - Cookies

Resources

- Book: Marty Hall, *Core SERVLETS and JAVA SERVER PAGES*, A Sun Microsystems Press/Prentice Hall PTR Book. ISBN 0-13-089340-4
  - Available online: http://pdf.coreservlets.com/
- Online tutorials
  - Servlets and JavaServer Pages (JSP) 1.0: A Tutorial http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/Servlet-Tutorial-Intro.html
  - The J2EE Tutorial: http://java.sun.com/j2ee/tutorial/
- Apache Tomcat Software:
  - Standalone web server for servlet and JSP development
  - Download: http://jakarta.apache.org/builds/jakarta-tomcat-4.0/release/
  - Installation: http://www.moreservlets.com/Using-Tomcat-4.html
- Servlet API: http://java.sun.com/products/servlet/2.3/javadoc/index.html

Introduction

- Servlets are programs that run on server
  - Acting as a middle layer between client requests and databases or other applications.



- Example: Duke' Book store: http://monkey.cs.ust.hk:8080/bookstore.html

Introduction

- Traditionally, HTTP servers handle client requests by using CGI (common gateway interface) scripts.
  1. User fills out a form and submit.
  2. HTTP server gets URL requests from the net.
  3. HTTP server finds the CGI script specified in the HTML file, runs it with parameters from requesting URL
  4. HTTP server takes output from the CGI program (most often output is HTML text), fixes it up with a full complete HTTP header, and sends it back to the original requesting client

HTML:

```
<form action="cgi_bin/rpe" method=POST>
First Name: <input type=text size=20 name=firstname>
Last Name: <input type=text size=20 name=lastname>
<input type=submit>
```


Introduction

- Advantages of servlets over CGI scripts (Hall book)
  - More efficient
  - Easier to use
  - More powerful.
  - More portable
  - Safer
- Note:
  - CGI scripts written in scripting languages such as Perl, Python, Unix Shell, etc doe not need compilation.
  - CGI scripts written in programming languages such as C, C++ need compilation
  - Servlets need compilation

Servlet vs. Applet

- Servlets are to servers what applets are to browsers:
  - Applets run by browser, servlets run by server.
  - Applets are “client-side java”, servlets are “server-side java”.
  - Applets makes appearance of web pages alive, servlets makes contents of web pages dynamic.
  - Unlike applets, however, servlets have no graphical user interface. Implement only back-end processing.

Outline

- Introduction
- A simple servlet
  - Environment for developing and testing servlets
- General information about servlets
- HTTP Servlets
- Session tracking
- Cookies

A Simple Servlet


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

public class SimpleGenericServlet extends GenericServlet
{       public void service (ServletRequest request,
                             ServletResponse response)
              throws ServletException, IOException
        {
            response.setContentType("text/plain");
            PrintWriter out = response.getWriter();
            out.println("Hello World");
            out.close();
        }
}
```

Note: no **`main`** method. Servlet run by server, just as applet run by browser

A Simple Servlet

- **`service`**: The most important method in a servlet,
  Determines what the servlet does.
  Invoked automatically when a request comes in.
  Needs to be overridden.

- It takes two arguments:

```
service (ServletRequest request,
         ServletResponse response)
```

  - **`ServletRequest`** and **`ServletResponse`** are interfaces defined by the **`javax.servlet`**
  - Get information about a request from the **`ServletRequest`** object **`request`**.
  - Get information about a response via the **`ServletResponse`** object **`response`**.

Environment for developing and testing servlets

- Compile:
  - Need Servlet.jar. Available in Tomcat package
- Setup testing environment
  - Install and start Tomcat web server
  - Place compiled servlet at appropriate location
    - See http://www.moreservlets.com/Using-Tomcat-4.html
- Run example:
  - http://monkey.cs.ust.hk:8080/servlet/SimpleGenericServlet

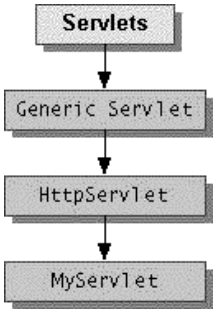
Outline

- Introduction
- A simple servlet
  - Environment for developing and testing servlets
- General information about servlets
- HTTP Servlets
- Session tracking
- Cookies

General Information about Servlets

Architecture of package **`javax.servlet`**

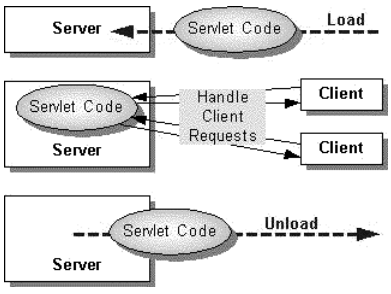
- **`Servlet`** interface: declares servlet methods (**`init`**, **`service`**, etc.)
- **`GenericServlet`** implements `Servlet`
- **`HttpServlet`** subclass adds features specific to HTTP
- Technically, a servlet is a program that extends either **`GenericServlet`** or **`HttpServlet`**.



General Information about Servlets

Servlet Life Cycle

- Servlets are controlled by servers
  1. A server loads and initializes the servlet
  2. The servlet handles zero or more client requests
  3. The server terminates the servlet



Servlet Life Cycle

- Methods
  - **`public void init():`**
    Called only once when servlet is being created.
    Good place for set up, open Database, etc.
  - **`public void service():`**
    Called once for each request.
    In **`HttpServlet`**, it delegates requests to **`doGet`**, **`doPost`**, etc.
  - **`public void destroy():`**
    Called when server decides to terminate the servlet.
    Release resources.

Outline

- Introduction
- A simple servlet
  - Environment for developing and testing servlets
- General information about servlets
- HTTP Servlets
- Session tracking
- Cookies

HTTP Servlets

- For HTTP requests.
  - HTTP requests include
    - GET, conditional GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS
    - The default is GET.
  - Type of request specified in HTML file:

```
<form action="/py/maps.py?Pyt=Tmap&YY=28457" method=GET>
  … </form>
<form method=POST action="/cgi-bin/ipc/idbsprd"> …
  </form>
```


HTTP Servlets

Methods of `HttpServlet` and HTTP requests

Methods	HTTP Requests	Comments
`doGet`	GET, HEAD	Usually overridden
`doPost`	POST	Usually overridden
`doPut`	PUT	Usually not overridden
`doOptions`	OPTIONS	Almost never overridden
`doTrace`	TRACE	Almost never overridden

- All methods take two arguments: an **`HttpServletRequest`** object and an **`HttpServletResponse`** object.
- Return a **`BAD_REQUEST`** `(400)` error by default.

HTTP Servlets

- Classes and interfaces in **`javax.servlet.http`** include
  - **`HttpServlet extends GenericServlet`**
  - **`HttpServletRequest extends ServletRequest`**
  - **`HttpServletResponse extends ServletResponse`**
  - …
- **`HttpServlet`** class has already overridden the **`service`** method to delegate requests to special purpose methods such as **`doGet`** and **`doPost`**.
  - Don't override the **`service`** method when sub classing **`HttpServlet`**. Instead, refine the special purpose methods, mostly **`doGet`** and **`doPost.`**

HTTP Servlets

`HttpServletRequest` Objects

- Provide access to HTTP header data and the arguments of the request.
- Values of individual parameters
  - **`getParameterNames`** method provides the names of the parameters
  - **`getParameter`** method returns the value of a named parameter.
  - **`getParameterValues`** method returns an array of all values of a parameter if it has more than one values.

HTTP Servlets

`HttpServletResponse` Objects

- Provide two ways of returning data to the user:
  - **`getWriter`** method returns a **`PrintWriter`** for sending text data to client
  - **`getOutputStream`** method returns a **`ServletOutputStream`** for sending binary data to client.
  - Need to close the **`Writer`** or **`ServletOutputStream`** after you send the response.
- HTTP Header Data
  - Must set HTTP header data before you access the **`Writer`** or **`OutputStream`**.
  - **`HttpServletResponse`** interface provides methods to manipulate the header data.
  - For example, the **`setContentType`** method sets the content type. (This header is often the only one manually set.)

Handling GET requests: Override the `doGet` method

```
public class BookDetailServlet extends HttpServlet
{   public void doGet (HttpServletRequest request,
                       HttpServletResponse response)
        throws ServletException, IOException
    {   ...
 // set content-type header before accessing the Writer
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println(...); // then write the response
        //Get the identifier of the book from request
        String bookId = request.getParameter("bookId");
        if (bookId != null)
       { out.println( information about the book );}
       out.close();
      ……
```

- Try `getRequest.html and getRequest2.html on` http://monkey.cs.ust.hk:8080/201html/index.html

`BookDetailServlet.java in bookstore example`

Handling POST requests: Override the **doGPost** method

```
public class ReceiptServlet extends HttpServlet
{   public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
        throws ServletException, IOException
    {    ...
// set content type header before accessing the Writer
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println(...); // then write the response

        out.println("<h3>Thank you for … " +
                    request.getParameter("cardname") +
                    ...);
        out.close();
    }
```

- Try `postRequest.html`

`ReceiptServlet.java` in bookstore example

HTTP Servlets

- Handling both GET and POST requests in the same way

```
public class BookDetailServlet extends HttpServlet
{   public void doGet (HttpServletRequest request,
                       HttpServletResponse response)
        throws ServletException, IOException
    {
         // codes for handling the request
    }
    public void doPost (HttpServletRequest request,
                       HttpServletResponse response)
        throws ServletException, IOException
    {
           doGet( request, response);
     }
}
```

Outline

- Introduction
- A simple servlet
  - Environment for developing and testing servlets
- General information about servlets
- HTTP Servlets
- Session tracking
- Cookies

Session Tracking

- Servlets in Duke's Bookstore
  - BookStoreServlet: Forward to main page
  - CatalogServlet: Show all books and allow selection
  - BookDetailServlet: Show details of a book and allow selection
  - ShowCartServlet: Show shopping cart
  - CashierServlet: Check out

Session Tracking

Motivation

- In the Duke's Bookstore example, suppose a client has selected several books. (Do this and check the page produced by `CatalogServlet`.)
- Problem 1:
  - The client requests
    - **`ShowCartServlet`** to show the books in his/her shopping cart.
  - Question:
    - How does **`ShowCartServlet`** know the selected books?
    - How communications between servlets are facilitated?
- Problem 2:
  - The client decides to leave the bookstore and visit some other pages.
  - Question: When the client comes back and makes further requests, how do the servlets know the books that have been selected previously?
    - How come servlets can remember things?

Session Tracking

- Session tracking is a mechanism that servlets use to maintain state about a series of requests
  - From the same user (that is, requests originating from the same browser)
  - Across some period of time

  Solution to Problem 2:

  Servlets use sessions as "notebook" and hence can remember
- Sessions are shared among the servlets accessed by the same client.

  Solution to Problem 1.

  Servlets communicate via sessions.

Session Tracking

- Session is associated with a request.
- To use session,
  - Get session from HttpServletRequest request:
    - `HttpSession getSession(boolean create)`
    - `HttpSession mySession = request.getSession(boolean create);`
    - **Case 1: create==true**
      - **Return the associated session if exists**
      - **Otherwise, create a new session, associate it with the request, and return the session**
    - **Case 2: create==false**
      - **Return the associated session if existis**
      - **Otherwise, return null.**
    - **Note: get session before accessing response streams**.

Session Tracking

- Store/retrieve information to/from `HttpSession` object.

```
public void setAttribute(String name, Object obj)
public Object getAttribute(String name).
```

- Invalidate the session (optional).
  - Manually: `Session.invalidate()`
  - Automatically when no request after certain time.

```
public void doGet(HttpServletRequest req,
                  HttpServletResponse resp)
            throws ServletException, IOException
{   resp.setContentType("text/html");
    HttpSession session = req.getSession(false);

    PrintWriter out = resp.getWriter();
    out.println("<HTML><BODY><h1>Count me!</h1><HR>");

    if (session == null)
   {    out.println("Welcome, I don't believe we've met!");
        session = req.getSession(true);
        session.setAttribute("Count", new Integer(1));
        out.println("I think of you as "+ session.getId());
    } else

    {   int n=((Integer)session.getAttribute("Count")).intValue();
        out.println("You again? " + session.getId());
        out.println("That makes " + n + " visits!");
        session.getAttribute("Count", new Integer(n + 1));
    }
    out.println("</BODY></HTML>");
    out.close();
} //HelloAgainServlet.java
```


Session Tracking

- Session tracking in Duke's Bookstore
  - Discussion the following in class
    - CatalogServlet.java
    - ShowCartServlet.java

Outline

- Introduction
- A simple servlet
  - Environment for developing and testing servlets
- General information about servlets
- HTTP Servlets
- Session tracking
- Cookies

Cookies

- Cookies
  - Made at server
  - Sent to client for storage
  - Retrieved by server when client connects again
- Cookies can be used for
  - Session tracking. **HttpSession** implemented using cookies.
  - Persistent state. E.g. name, address, email address. When user access some servlets again, no need to provide such information one more time.

Cookies

- Details:
  - Each cookie is a **name=value** pair.
  - Servlets send cookies to clients by adding fields to HTTP response headers.
  - Clients automatically return cookies by adding fields to HTTP request headers.
  - NOTE: Cookies shared among servlets on the server accessed by the same client.

Cookies

- Cookies are objects of class **`javax.servlet.http.Cookie`**
- To send a cookie,
  1. Create a **`Cookie`** object
     ```
     Cookie c = new Cookie(name, value);
     ```
  2. Set attributes if necessary
     ```
     c.setMaxAge(30); // expire after 30 seconds
     ```
  3. Send the cookie
     ```
     response.addCookie(c);
     ```
- To get information from a cookie,
  1. Retrieve all the cookies from the user's request
     ```
     Cookie[] cookies = request.getCookies();
     ```
  2. Find the cookie that you are interested in and get its value
     ```
     for (int i = 0; i < cookies.length; i++) {
         String name = cookies[i].getName();
         String value = cookies[i].getValue();
     }
     ```

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
        throws IOException, ServletException
{   response.setContentType("text/html");
    PrintWriter out = response.getWriter();
     out.println("Cookies received from client:<br>");
    Cookie[] cookies = request.getCookies();
    for (int i = 0; i < cookies.length; i++) {
        Cookie c = cookies[i];
        String name = c.getName();
        String value = c.getValue();
        out.println(name + " = " + value + "<br>");
    }

    out.println("<br>Cookies sent to client:><br>");
    String name = request.getParameter("cookieName");
    if (name != null && name.length() > 0) {
        String value =
             request.getParameter("cookieValue");
        Cookie c = new Cookie(name, value);
        c.setMaxAge(180);
         response.addCookie(c);
        out.println(name + " = " + value + "<br>");
    }} // CookieServlet.java
```