Networks and Database Systems

Professor Tom Ellman
Lecture 9
(HTTP) Servlets

- Java program running on a web server.
- Receives an HTTP request.
- Dynamically constructs a web page.
- Sends constructed page in reply to request.
Adding a Servlet to a Web Application in NetBeans

- Make an empty web application.
- Invoke New-Other-Web-StandardDeploymentDescriptor.
- Click on project name and invoke New-Servlet.
- Check box: Add information to web.xml.
- Enter a name and package for the servlet class.
- Accept the Servlet Name and URL Pattern defaults.
- Edit the `processRequest` method of the servlet class. (Or edit the `doGet` or `doPost` methods.)
protected void doGet(HttpServletRequest request,
                      HttpServletResponse response)
  throws ServletException, IOException {
    processRequest(request, response);
  }

protected void doPost(HttpServletRequest request,
                       HttpServletResponse response)
  throws ServletException, IOException {
    processRequest(request, response);
  }
protected void processRequest(HttpServletRequest request,
                                      HttpServletResponse response)
          throws ServletException, IOException {
            response.setContentType("text/html;charset=UTF-8");
            PrintWriter out = response.getWriter();
            try {
                /* TODO output your page here
                 * out.println("<html>");
                 * ...
                 * out.println("</html>");
                 */
                finally {
                    out.close();
                }
            }
        }
EditConfirmProcessServletController Example

- Equivalent to EditConfirmProcessJSPController.

- But the Java code appears in the `doGet` method of a servlet.
public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException {
    String address;
    if (request.getParameter("confirmButton") != null) {
        address = "Confirm.jsp";
    } else if (request.getParameter("processButton") != null) {
        address = "Process.jsp";
    } else {
        address = "Edit.jsp";
    }
    request.getRequestDispatcher(address).forward(request, response);
}
Fragment of web.xml File

<servlet>
    <servlet-name>Controller</servlet-name>
    <servlet-class>controller.Controller</servlet-class>
</servlet>

The **servlet** tag associates a servlet name with a servlet class.
Fragment of web.xml File

```
<servlet-mapping>
    <servlet-name>Controller</servlet-name>
    <url-pattern>/Controller</url-pattern>
</servlet-mapping>
```

The `servlet-mapping` tag associates a servlet with one or more URL patterns. Each URL pattern indicates how the servlet may be referenced in a URL. For example, in this case, the servlet is referenced as:

`http://<HostURL>/EditConfirmProcessServletController/Controller`
Servlet Examples

• **ZerothServletApp:**
  – Invoked by http://<HostURL>/ZerothServletApp/*.  
  – Generates constant page dynamically.

• **FirstServletApp:**
  – Invoked by http://<HostURL>/FirstServletApp/FirstServlet/.  
  – Generates constant page dynamically.

• **PersonalServletApp:**
  – Welcome page request user’s first name and invokes servlet.  
  – Servlet reads client name from request parameters.  
  – Dynamically generates page greeting client.

• **AdderServletApp:**
  – Welcome page gets two numbers from user and invokes servlet.  
  – Servlet reads numbers from request parameters.  
  – Dynamically generates page showing sum of numbers.
Servlet Examples

• **ParameterSnoopApp:**
  – Displays all request parameters.
  – Illustrates parameters with multiple values.

• **HeaderSnoopApp:**
  – Displays all request headers.
  – Why are these useful?
Servlet Initialization

• Servlet’s `public void init()` method is called once, when the servlet is initialized.

• But only if web.xml file has the tag:
  `<load-on-startup>1</load-on-startup>`

• Initialization parameters may be placed in web.xml file:
  ```
  <init-param>
    <param-name>initialHitCount</param-name>
    <param-value>1</param-value>
  </init-param>
  ```

• Initialization parameters may be accessed using servlet’s method:
  ```java
  String getInitParameter(String name).
  ```

• Example: `HitCountWebApp` initializes `hitCount` variable from servlet initialization parameter `initialHitCount`. 
Servlet Destruction

• Servlet’s `public void destroy()` method is called once, when the servlet is taken out of service.

• E.g., Because the web application is stopped.

• Useful for closing sockets and database connections.
Servlet Member Variables

- Two different clients interacting with a single servlet may or may not be dealing with the same servlet class instance.
- Data that is shared among clients should be placed in (static) variables of the servlet class.
- Data that is not shared among clients should be placed in variables of a unique helper class instance that is constructed in response to each HTTP request.
- Example: **HitInstanceCountWebApp** tracks and displays both hit count and number of servlet instances created.
Context Creation/Destruction

• Declare `ServletContextListener` classes in the web.xml file.

• These classes implement methods:
  
  `contextInitialized`
  `contextDestroyed`

• Called when the entire web application is started and stopped.
Context Parameters

- Parameters associated with the entire web application.
- Available to all servlets in the web application.
Session Attributes

- A *session* is a data structure associated with a client across multiple requests to a server.
- Sessions are typically implemented using cookies.
- Data is stored in a session in the form of session attributes.
- Servlets may write attributes to and read attributes from the session.
- Alternative to hidden fields to pass data between pages.
- Example: `EditConfirmProcessServletControllerAlt`. 
Filters

• Click on project name and invoke New-Filter.
• Check box: Add information to web.xml.
• Filter allows for special processing before and after execution of a servlet.
• Filter mappings say what URLs are filtered.
• Useful for blocking direct access to pages that should only be accessed under special conditions.
• ServletFilterDemo: Attempts to directly access an error page are blocked and redirected to appropriate start page.
Exercise

- Create a web application: `MissingClientsWebApp`.
- The application has a `Controller` servlet with an initialization parameter called: “`initialLatestClient`” that is used to initialize a class variable called: “`latestClient`”.
- The initialization parameter should be set to “Kilroy”.
- The start page has a form asking the client to enter his/her first name.
- The form submit action invokes the `Controller` servlet, passing the client’s name as a parameter.
- The `Controller` servlet returns a page saying “You just missed `<latestClient>`. What a shame!”.
- The `Controller` servlet updates `latestClient` with the name of the new latest client.