CS 416
Web Programming

Intro to servlets

Dr. Williams
Central Connecticut State University

Simple servlet refresher

- Create a web page that has a field for the Bill amount and a percentage tip. When the user clicks calculate your servlet should calculate the tip and return it as a page that says “The tip amount is: X.XX” (Don’t worry about rounding the number)

- What is the difference between request forwarding and request redirection? (You don’t need it for this example)

Topics for Lecture 5

- Server side processing
- Servlet configuration
- App server in memory persitance
- Survey app

Separating logic/from display

- Insurance example
  - Receiving servlet looks up additional detail regarding entered data, possibly stores off data
  - Cost calculator uses enhanced information to do computation processing
  - Display servlet displays request that has been enhanced

```
request.setAttribute(“attrName”,valueObj);
(ValueObj)

request.getAttribute(“attrName”);
```

Insurance demo

Servlet configuration

- Java servlets can be configured in 2 ways web.xml file and annotations
- By default NetBeans creates code for annotations deployment but configuration can also be configured manually
- With Netbeans to create deployment XML you need to check to add deployment descriptor

web.xml

- Servlet web xml has two key elements
- <servlet>
  - <servlet-name> - name that will be used elsewhere in the web.xml to refer to the servlet
  - <servlet-class> - mapping to java servlet class

web.xml cont.

- <servlet-mapping>
  - <servlet> matches one of the defined servlet-name elements
  - <url-pattern> specifies the url patterns the servlet should match to
- url-pattern
  - Prefix path such as “/MyServlet”
  - Or wild card matching
    - “/serv/*”
    - “*.serv”
- Creating more than one mapping requires adding additional servlet-mapping elements

Initial parameters

- Within the servlet declaration <servlet> initial parameters can also be specified

```
<servlet>
   …
   <init-param>
      <description>my desc</description>
      <param-name>…</param-name>
      <param-value>…</param-value>
   </init-param>
```

Reading initial parameters

- To read the initial parameters you need to get an instance of the servlet config and then read each parameter

```
ServletConfig servletConfig = getServletConfig();
String param1 = servletConfig.getInitParameter(“param1”);
```

- Sometimes used to specify different debug behavior vs. production behavior
- Overriding behavior of parent class based on child classes initial values

Servlet annotation

- Alternate to using web.xml is to specify deployment information in the servlet itself
- @WebServlet annotation
  - name
  - url-patterns
  - init-params

@WebServlet

```
@WebServlet(name="MyServlet",
urlPatterns={"/MyServlet","*.serv"
 }, initParams={
     @WebInitParam(name="param1",
 value="value1"),
     @WebInitParam(name="param2",
 value="value2")
   }
)
```

Reading initial parameters

- The purpose of initial parameters is to allow servlets to be easily configured with minimal changes
- Initial parameters specified in the java file have simplicity of changes being made easily in the code
- Initial parameters in the web.xml file have the advantage of being able to be modified without needing to recompile code

web.xml cont.

- Web xml also allows you to specify the default page for a web site for the context

```
<welcome-file-list>
   <welcome-file>MyServlet</..>
</welcome-file-list>
```

- If not specified the server looks for index.html first, if that's not found it will check for index.jsp

Persisting request data

- Adding data to the request is very useful for processing particularly when request forwarding is used
- Setting attributes on the request allows complex server side tasks to be broken up into logical components
- Most common application is separating form processing from display
  - Store off and validate data, enhance request and forward
  - Display enhanced information

Persisting data across requests

- There are two options for persisting data across requests
  - Session data
  - Context data
- Session data is stored for a particular user session across all future requests while the session is alive, upon timeout all data is lost
- Context data is stored for life of application server and is accessible across all users

Common uses

- Session data is commonly used for:
  - Authentication – have the user authenticate once when they log on mark them as authenticated for the rest of the session
  - Data which is going to be unchanging during life of session. Personal info, preferences
  - Session tracking – personalization based on previous session activity

Session data

- Storing data in session is very similar to storing data in request
- Obtain a reference to the session then store or retrieve attributes from it:

```
HttpSession session = request.getSession();
session.setAttribute(“attrName”,attrObj);
attrObj = (AttrObj)session.getAttribute(“attrName”);
```

Common uses

- Context data
  - Information that is being tracked across users
  - Common data used across users – load from database once then reused

Context data

- Also similar to request and session data
- Get instance of the servlet’s context.

```
ServletContext context = getServletContext()
context.setAttribute(“attrName”,attrObj);
attrObj = (AttrObj)context.getAttribute(“attrName”);
```

PersistingServlet Demo

Processing form data part 1

- Create an HTML survey
  - Name, sex (Radio button not text) of user
- On clicking submit the form should submit to ResultsServlet and display the name and sex selected
- Display the percentage of males and females that have visited the site

Processing form data part 2

- Behavior – first time the user visits the page they must enter their name and sex, on returns to the page (refresh rather than back button) value shown non editable (ie. Straight HTML)
- Names of all visitors
- Feeling saucy? - Add some additional survey data – mode of transportation (car, bike, walk) checkbox and tabulate the results across session and users