CS 416
Web Programming

Intro to servlets

Dr. Williams
Central Connecticut State University

What is a servlet

- Servlet is a Java class that extends capability of the server that hosts the website
- Servlets can respond to requests and generate responses
- Base class of all servlets is javax.servlet.GenericServlet which is protocol independent javax.servlet.http.HttpServlet is base for those using the HTTP protocol

Servlet methods

- To be useful a servlet must implement one or more of the 4 HTTP request types by overriding these methods all of which take 2 arguments

`(HttpServlet request, HttpServlet response)`

  - GET – doGet(…)
  - POST – doPost(…)
  - PUT – doPut(…)
  - DELETE – doDelete(…)

The names are rather intuitive for once ☺

doGet and doPost

- With a servlet the doGet() method gets called anytime a user
  - Enters the URL directly
  - Clicks on a link to a page
  - From a form submission of method="GET"
- The doPost() method gets called
  - From a form submission of method="POST"

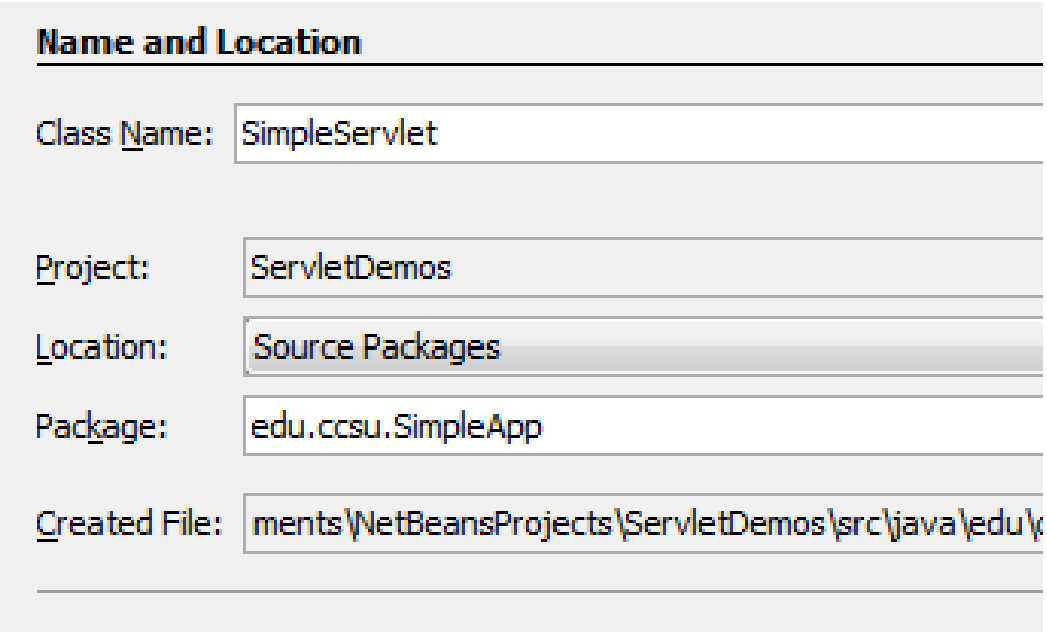
Developing a simple servlet

- Go to New Project, in the dialog select new web application

This creates an application that when run will start Apache (web server) and GlassFish (application server)

- Right click on the source packages folder and select New=>Servlet

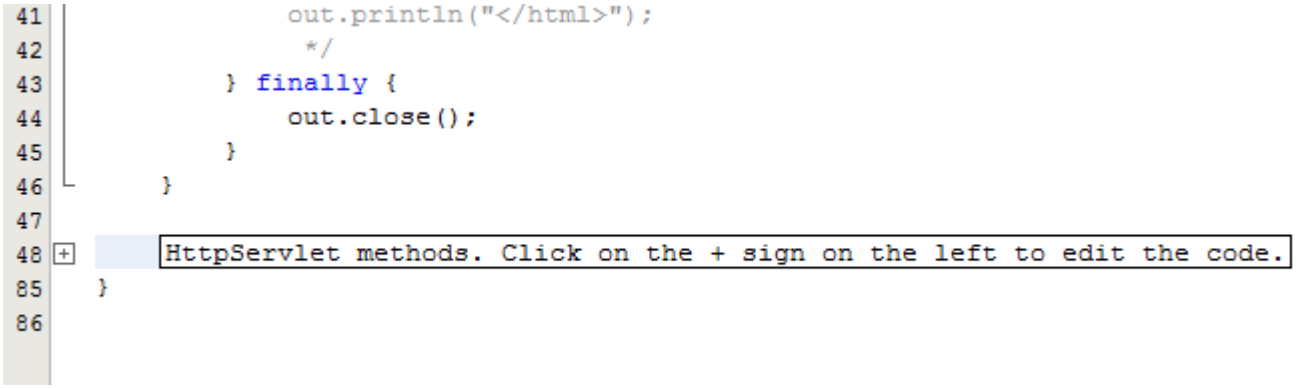
Create SimpleServlet



Press finish

Changing the code

- Netbeans creates a skeleton of the code for you, you just need to add your own code or overwrite the shells
- At the bottom of the generated code there is a button to expand HTTPServlet methods, expand it



Changing doGet

- By default doGet calls processRequest, we want to change that do our own code so delete it out
- Within our function we have two arguments the request and the response
  - Request allows us to read the information being sent to our form
  - Response is how we write the response

Changing the response

- First tell the response that we are going to be writing out HTML

```
response.setContentType("text/html");
```

- Next we use a PrintWriter to write to the response

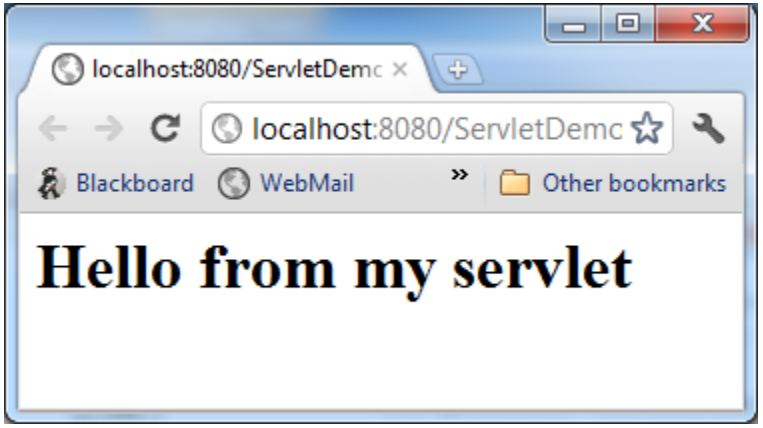
```
PrintWriter writer = response.getWriter();
```

- After that we can write any HTML we want to the output page

```
writer.println("<h1>Hello from my servlet </h1>");
```

Testing our servlet

- To test that our servlet works right click the file in the project and select Run file
- This will start up Apache and Glassfish (if they are not already started) and deploy your servlet and you should get



Using form GET

- If we create a form that GETs from our servlet
- Input name:

```
<form action="SimpleServlet"
method="GET">
  name<input type="textbox" name="name" />
  <input type="submit" value="Submit"
/>
</form>
```

When we hit submit it calls doGet on the servlet

Reading parameters

- Within a doGet or doPost the way you access input variables being sent to the server are through the **request**'s getParameter method which returns a string with that parameter's value
- If it is an input where more than one value can be selected you would use getParameterValues which returns an array of Strings

Reading parameters

```
<form name="myForm" action="SimpleServlet" method="GET">
    Input name:<input type="textbox" name="name" /><br/>

    <input name="activities" type="checkbox" value="Run">Run<br/>
    <input name="activities" type="checkbox" value="Fly">Fly<br/>
    <input name="activities" type="checkbox" value="Dance">Dance<br/>

    <input type="submit" value="Submit"/>

</form>
```

```
name = request.getParameter("name");
response.setContentType("text/html");
PrintWriter writer = response.getWriter();
writer.println("<h1>Hello "+name+" GET from my servlet </h1>");

String[] options = request.getParameterValues("activities");
for (int i=0; i<options.length;i++){
    writer.println("option:"+options[i]+"<br/>");
}
```

doPost

- To do the same thing for a POST method you would use the exact same code as the GET, this way if the form instead used method="POST" your servlet could handle it
- Frequently you will have the same handler for both (the processRequest method that was added by NetBeans in the beginning)

Request forwarding

- Request forwarding allows a servlet to perform some processing on the request before forwarding it to another servlet for additional processing
- Example: one servlet handles augmenting the request the second works on processing the complete data fields
- One handles charging the account another handles shipping

Request forwarding cont.

- In addition to doing processing on its own a servlet is able to add additional information to the request before sending it on

```
request.setAttribute("somename",myClass);
```

- It can then forward it on to the next Servlet to process it

Request forwarding

- After the initial servlet is done processing the request and adding information on the request it can forward the enhanced request using:

```
request.getRequestDispatcher( “SecondServlet”).forward(request,response);
```

- From the second servlet’s perspective it is just as if it was called directly just additional information may be available
- To the user they are unaware that multiple servlets may have been involved

Demo

- Form is posted to a servlet
  - Initial servlet inspects passed name if possible it enhances it with names of other family members (such as an interaction with a database)
  - Second servlet uses enhanced information to display content to the user

Initial servlet

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        String name;
        name = request.getParameter("name");
        List familyMembers = new ArrayList();
        familyMembers.add(name);
        if (name.equals("Chad")) {
            familyMembers.add("Grace");
            familyMembers.add("Kate");
            familyMembers.add("Patti");
        } else if (name.equals("Jon")) {
            familyMembers.add("Jane");
        }
        request.setAttribute("family", familyMembers);

        request.getRequestDispatcher("PrintFamilyServlet").forward(request, response);
    } finally {
        out.close();
    }
}
```

Second servlet

```
 protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        response.setContentType("text/html");
        PrintWriter writer = response.getWriter();
        writer.println("<h1>family</h1>");
        List family = (List)request.getAttribute("family");
        for (int i=0;i<family.size();i++){
            writer.println(family.get(i) +"<br />");
        }
    } finally {
        out.close();
    }
  }
```

Response redirection

- One disadvantage of request forwarding is request can only be forwarded to JSPs/Servlets in the **same** application server
- Response redirection allows a servlet to do some sort of processing of the request before redirecting the user to another page

Response redirection

- The initial servlet is able to do some processing before redirecting the user to some other site:

```
request.sendRedirect(url);
```

- From the redirected site's perspective the url being directed to is as if it is coming from the user directly
- From the user's perspective they will receive a response as if they typed in a different URL

Response redirect Servlet

```
 protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        String url = request.getParameter("website");
        if (url!=null&&url.length()>0){
            response.sendRedirect("http://"+url);
        }else{
            out.println("<h1>No website specified</h1>");
        }
    } finally {
        out.close();
    }
  }
```

Hands on

- Create your own servlets for
  - Create a form that has a series of options (maybe pizza toppings) in the form of check boxes plus submit button to POST the form
  - Your servlet should read the values selected and put them in to a TreeSet and add them to the request then forward to your second servlet
  - Your second servlet should output the pizza toppings selected in alphabetical order