Building a Better Servlet

by Jeff Volkheimer

Comment on this article

Servlet technology is rapidly becoming the optimal choice for server-side programmers. With its "Build Once Run Anywhere" ideal, Servlets allow developers viable and powerful alternatives to standard CGI programming. By inheriting all the power and flexibility of the Java API, they can do almost anything you want quickly and efficiently. For a more in-depth discussion of Servlet technology in general, click here.

A Brief Review

Let's briefly review our basic Servlet:

```
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet
{

   public void doGet (HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
   {
      res.setContentType("text/html");
      ServletOutputStream out = res.getOutputStream();
      out.println("<html>");
      out.println("<head><title>Hello World!</title></head>");
      out.println("<body>");
      out.println("<h1>Hello World!</h1>");
      out.println("</body></html>");
   }
}
```

Key features to note:

1. The HelloWorld class extends HttpServlet
2. We are overriding the doGet method
3. The output of this Servlet is plain text
4. doGet takes HttpServletRequest and HttpServletResponse objects as arguments which are created by the server when a client makes a request

Retrieving User Entered Data

The last example, while interesting, was not very useful. Not surprisingly, we would like to be able to accept incoming data from the user and do something with that data. Servlet technology allows us to do this quickly and easily...let's check it out:

```
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloUser extends HttpServlet
{

   public void doGet (HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
   {
      res.setContentType("text/html");
      ServletOutputStream out = res.getOutputStream();
      out.println("<html>");
      out.println("<head><title>Hello User!</title></head>");
      out.println("<body>");
      out.println("<h1>Hello " +
      "req.getParameter("User")" +
      "!</h1>");
      out.println("</body></html>");
   }
}
```

Not much has changed from our previous example, yet there is one key difference. We have now provided the user with some real dynamic content. We assume our data is sent via the GET method, which puts our variables into the querystring. Getting this data is usually quite a chore in CGI programming. When using Servlets, however, form parsing is done for us automatically. To get our data out of the querystring, we call the getParameter method of the HttpServletRequest object, like so:

```
req.getParameter("<VarName>")
```

where <VarName> is the name of the variable you are expecting. The returned value of getParameter is a string. If your parameter exists but has no value, getParameter will return an empty string. If your parameter does not exist at all, getParameter returns NULL. Finally, if your parameter may have multiple values, use getParameterValues which will return an array of strings.

Suppose you want to send your data through the POST method? Implementing this is about as easy as it gets. All you have to do is add the following to your class:

```
public void doPost (HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException
{
   doGet(req, res);
}
```

Your Servlet can now handle POST data just as gracefully as GET data.

Using Cookies

Cookies are one of the most indispensable tools used by web developers. They bring many elements to the "stateless Internet" which would otherwise make any task requiring the persistence of the clients state complicated. Without too much trouble, we can make our Servlet remember our user, so that each time the user visits our page, he will be greeted with a personalized message.

First of all, lets talk about some of the more useful cookie attribute get/set functions:

1. getName/setName - Without a doubt one of the most important attributes to set and you will almost always need to look up your cookie’s name.
2. getValue/setValue - Another crucial attribute which actually contains the values.
3. getMaxAge/setMaxAge - Allows you to specify/read how long the cookie has to live.
4. getComment/setComment - Reads/associates a comment with your cookie.
5. getSecure/setSecure - A Boolean value which specifies whether your cookie should be sent via SSL.

In order to store data in our cookie, we must:

1. Instantiate a cookie object
2. Set its attributes
3. Send the cookie

To retrieve data from our cookie, we must:

1. Retrieve all cookies from the users request.
2. Find the cookie you are interested in.
3. Get your cookie’s values.

Without further delay, lets add cookie creation to our Servlet:

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

public class HelloUser extends HttpServlet
{

   public void doGet (HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
   {
      String User = req.getParameter("User");
      int OneYear = 60*60*24*365;

      // Let’s instantiate our cookie
      Cookie UserNameCookie = new Cookie("Username", User);

      // We also want to set our cookie to live for a year
      UserNameCookie.setMaxAge(OneYear);

      // Finally, let’s send the cookie off
      response.addCookie(UserNameCookie);

      res.setContentType("text/html");

      ServletOutputStream out = res.getOutputStream();

      out.println("<html>");
      out.println("<head><title>Hello User!</title></head>");
      out.println("<body>");
      out.println("<h1>Hello " +         User + "!</h1>");
      out.println("</body></html>");
   }

   public void doPost (HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
   {
      doGet(req, res);
   }

}
```

As you can see, cookie creation is fairly straightforward. There are, however, some issues to be aware of.

First of all, cookie names must be a string not containing any of the following:

```
( ) | [ ] { } < > @ , . : ; \ " ? =
```

Moreover, cookie names may not start with $. Finally, if you want your cookie to comply with Netscape’s original cookie specification (and you probably do), do not include any whitespace.

Since all cookie information is sent to the client as a header, you must create your cookie before accessing the Writer (it is required that all headers must be written before accessing the Writer.

That is pretty much it...you have now baked your first cookie!

Now, let’s modify our original example (which took the user’s name from the querystring) and have it read the name from the cookie.

```
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloUser extends HttpServlet
{

   public void doGet (HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
   {
      // This returns an array of cookies
      Cookie[] cookies = req.getCookies();

      // We now need to traverse our array to find our cookie
      // since our cookie may be one of many.
      for(i=0; i < cookies.length; i++)
      {
         Cookie thisCookie = cookie[i];
         if (thisCookie.getName().equals("Username")
         {
            String User = thisCookie.getValue();
         }
      }

      res.setContentType("text/html");
      ServletOutputStream out = res.getOutputStream();

      out.println("<html>");
      out.println("<head><title>Hello User!</title></head>");
      out.println("<body>");
      out.println("<h1>Hello " + User + "!</h1>");
      out.println("</body></html>");
   }

}
```

Clearly, retrieving cookie data is not a difficult task either. We use getCookies() to give us an array of cookies. Once we have this array, we traverse it using a simple for loop. Upon finding the proper cookie, we store the information using getValue() and proceed with printing out the proper HTML. It’s that easy.

Conclusion

Java Servlets give programmers the power and ease of use of Java as an alternative to the traditional CGI programming techniques. From this article, you should have learned how to retrieve and process GET, POST, and cookie data. You should also have learned how to save the client’s state by using cookies. These are some invaluable techniques to learn in order to program effective and useful Servlets.

Developed Under:

Allaire's JRun

References

Java’s Servlet Tutorial Site:
http://java.sun.com/docs/books/tutorial/servlets/index.html

Java’s Servlet Technology Site:
http://java.sun.com/products/servlet/index.html

A nice forum for discussing Servlet Technology:
http://www.servletforum.com/