

Web Development on the SOEN 6011 Server

Stephen Barret

October 30, 2007

Introduction

Systems structured around Fowler's patterns of Enterprise Application Architecture (EAA) require a multi-tiered environment in which to operate. That requirement brings with it the need for a host of enabling technologies, among which are both client-sided and server-sided programming connected by some means, content servers, and databases. These days the client/server connectivity is often accomplished via the Internet.

A web deployment environment with all the requisite technologies has been set up on *runnerbean.cs.concordia.ca* for the use of students in SOEN 6011. Besides using *runnerbean* as a host for your applications, you may also choose to develop and test your code there.

The environment consists of a web browser as the front end – Mozilla or Opera on *runnerbean*, or whatever you happen to have on your own client – Tomcat acting as a web page server, and as a JSP/servlet container for dynamic web page creation, MySQL as a database back-end, and Connector/J to link the two.

Server and Tool Accounts

User accounts have been created for you on *runnerbean*. The accounts have been given the names **u01** to **u32**, all with a password of **pa22word**. For each user a Tomcat and MySQL account have also been set up. The user ID for both of these is **u##**, where **##** corresponds to your account number, and a password of **password** for MySQL and no password for Tomcat. Finally, an empty database named **soen6011u##** has been added to each MySQL account.

Under each user home directory, a web application structure has been built. Sample content has been added to it, and permissions have been set so as to allow Tomcat to serve that content to web clients. The structure is depicted below. Your HTML and JSP code go into the *webapps* directory, while your compiled servlet classes go under *classes*. Any package hierarchy you might create should be rooted here as well. Below you can see *soen6011*, *app*, *dom* and *ts*. These are all Java packages made in Eclipse that were used to partition

the design into layers, and to keep the code for this course from mingling with any other code that might be present. An important mapping file, *web.xml*, is kept in *WEB-INF*.

```
/home/u##/  
|  
|- bin                // Some useful scripts  
|  
|- public_html       // Apache served static HTML  
|  
|- webapps           // Tomcat static HTML & JSP code  
|  
|- WEB-INF           // web.xml (servlet to class map)  
|  
|   |- classes        // Java classes and packages  
|   |   |  
|   |   - soen6011    // Course package  
|   |   |  
|   |   - app         // Application layer package  
|   |   |  
|   |   - dom         // Domain layer package  
|   |   |  
|   |   - ts          // Tech. services or data  
|   |                 // sources layer package  
|- lib               // Java archives (.jar files)
```

Server Access

Okay, now that you're all excited to begin, you need to get access to *runnerbean*. This is a bit of a problem, because unless you're on its subnet, the ENCS firewall will prevent any contact. But fear not, that's what ssh port forwarding is for – to essentially construct a secure “tunnel” under the firewall through which data can be passed.

There are two script files under your *bin* directory which will do this for you. Unfortunately you can't get to them quite yet, so try this: on Windows execute the following commands from a DOS command line, but first change **user** to your ENCS ID, and **u##** to your *runnerbean* ID. **Note:** the whole thing must be run as *one* line! This will take a bit of cutting and pasting on your part.

```
ssh2 -t -l user -L 8080:localhost:3303 login.encs.concordia.ca  
ssh -l u## -L 3303:runnerbean.cs.concordia.ca:8080  
runnerbean.cs.concordia.ca
```

The last annoying thing is that for some reason DOS insists on echoing the password for the second ssh command back to the screen!

The Linux alternative doesn't have this problem, nor does it have to be run on one line – just execute the following in a terminal screen. Again, *after* making the appropriate substitutions.

```
ssh -t -l user -L 8080:localhost:3303 login.encs.concordia.ca \  
ssh -l u## -L 3303:runnerbean.cs.concordia.ca:8080 \  
runnerbean.cs.concordia.ca
```

If everything goes well you will now be viewing *runnerbean* through the just constructed tunnel. On Windows put aside the DOS box, but leave it running. Then open a regular *ssh* session against port 22 of *runnerbean.cs.concordia.ca* to your new account. On Linux you can either use the tunneling terminal directly, or open a new one.

Once logged in, the first thing you should do is change your password using *passwd*. Next transfer *bin/runner.bat* to your Windows space, and *runner.sh* to your Linux space. These take care of the double tunneling for you, so next time you can just run the scripts. If you're curious about this whole tunneling thing check out the script comments.

MySQL Setup

Once you are signed on to *runnerbean* you can set up your MySQL account. Execute the following commands to access your account and change its password:

```
[u##@runnerbean ~]$ mysql -p // Logs u## onto this host.  
Enter password:password // -p prompts for password.  
  
mysql> set password=password('u##_password'); // Change password.
```

Make sure to include the single quotes and the semicolon. Next execute the following MySQL commands to switch over to your database, explore it, add a new table, and insert a record into it:

```
mysql> use soen6011u## // Always switch to your database!  
  
mysql> show tables; // No tables - empty set.  
  
// Run SQL script from /home/u##/bin directory.  
mysql> source ~/bin/createpeople.sql;  
  
mysql> show tables; // Now shows people table.  
  
mysql> select * from people; // No rows - empty set.  
  
// Add person 'First Last' to table.  
mysql> insert into people values(0, 'Last', 'First');
```



Figure 1: Tomcat default web page.

```
mysql> select * from people;    // Now contains inserted record.
```

Notice that even though you specified an ID of 0, the database used 1. This is because the table definition in the *createpeople.sql* script specifies that the key column is an auto incremented integer. This will save you a bit of a headache when you have to insert a record programmatically.

Tomcat Access

Now that you've established a tunnel from your computer, to *login*, and then to *runnerbean* for data moving through port 8080 you can make a request of the web server on *runnerbean*. Enter **localhost:8080** in the address field of your web browser. Hopefully the response will be the default Tomcat web page of Fig. 1.

This indicates that an empty request made on the local machine's 8080 port has been forwarded to *runnerbean's* 8080 port, and that Tomcat which is listening on that port has responded. Otherwise it means that the tunnel has not been set up correctly (most likely), Tomcat is not running on the server, or the server is down.

Click the Manager link on the upper left of this page. It requires a user ID, but no password. Among other things, you will see a path to each student's

Tomcat applications. Locate your own and slide over to the right, there you will see start, stop, and reload commands. Because Tomcat keeps an application loaded until it is forced out of memory, you will need to do a **reload** after uploading a changed *.class*, *.jsp* or *web.xml* file. Your changes will not become visible until you do this reload.

Servlet Demonstration

Each student account has been set up with a default Tomcat page and two servlets. Account `u00` has three, so let's look at that one. On a machine connected to *runnerbean* enter **localhost:8080/u00** in a web browser. This should give you a list of URLs. Try the first. This one directly executes `u00's TimeServlet.class` which, according to its *web.xml* file, is referred to by the name **timeserv**. For each request, TimeServlet serves up a new web page constructed around the local time. You have the code for this servlet under your *public.html* directory.

The second URL displays a bare-bones data entry form. Pressing "Greet Person" invokes a *GreetServlet.class* through the form's **greet** action (again, see *web.xml*). Actions like this one that use the HTML Get method pass the entered data to the server by way of the request's URL. After pressing the button, you will be able to see the single name-value pair for this request in the browser's address line following the '?' at the end of the URL. The server packages the request's parameters into an `HttpServletRequest` object from which the servlet actually obtains the data. You also have the code for this servlet and its associated web page.

The third URL runs **addserv** – the *FrontController* servlet – which takes the two parameters gathered by its HTML entry form, and inserts them into the people table of the `soen6011` database as a new person. Because the table's key has been set to increment automatically, the servlet does not need to come up with a unique ID. Following the EAA patterns, *FrontController* doesn't directly store the data, but instead creates an appropriate Command object, which populates a Person domain object, which is then handed off to the proper `DataMapper` for database insertion. The results of this chain of events is then fed back to the client in the form of a dynamic web page.

Your own URL, **localhost:8080/u###** is a little less complete, but functional nonetheless. Give it a try as well. You can easily finish off your installation, should you chose to do so, by compiling the sample source code for project `Soen6011ServletAdd`, moving the resulting *.class* files along with *add.html* over to your *runnerbean* account, and modifying your *index.html* and *web.xml* files accordingly.

Points to Remember

When creating and deploying your own web application, keep these things in mind:

- To apply the EAA patterns the front end for your web-based projects will need to make use of dynamic HTML (as opposed to a Java applet).
- Since the HTML will be a little complicated it will probably be easier to implement as a JSP rather than a servlet.
- Establish a connection with *runnerbean* through two tunnels.
- Use **localhost:8080** as the base URL in order to contact *runnerbean* over the Web.
- Copy *.html* and *.jsp* code to your *webapps* directory.
- Copy packages and their *.class* files to the correct package under the *classes* directory.
- Update your *web.xml* file to map URL actions to the appropriate servlet class names.
- Should you change any of the above files, recopy them to the proper directories.
- After copying any files into the *public.html* directories reload your application with the Tomcat manager.
- When working in MySQL, make sure to switch over to your database first (i.e., **use soen6011u##**).

Good luck!