How to Configure Security with Embedded Jetty

How to Configure Security with Embedded Jetty

This example shows you how to setup web application security programmatically. Firstly, we'll look at how to do it if you use a `web.xml` file to declare your <security-constraint>s on urls within your webapp. Then, we'll show you how to do it in code instead, so that you don't even need to have a `web.xml` file.

For both of these examples, we need to configure jetty with a Realm. A Realm represents the runtime security environment - the users, their credentials and their roles. Jetty has a number of different Realm implementations:

- org.mortbay.jetty.security.HashUserRealm obtains information from a properties file
- org.mortbay.jetty.security.JDBCUserRealm obtains information from a database
- org.mortbay.jetty.plus.jaas.JAASUserRealm uses JAAS for authentication and authorization

For these examples, we'll be using the org.mortbay.jetty.security.HashUserRealm. There is an example of a properties file for this Realm type in `$jetty.home/etc/realm.properties`.

We'll be using BASIC authentication for this example, but you can also set up FORM authentication in a similar way.

Using a web.xml file for security-constraints

If you're able to use a `WEB-INF/web.xml` file, you should configure it to use BASIC authentication, and to specify some urls that have security constraints. Here's an example:

```
<web-app>
  ...
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>A Protected
Page</web-resource-name>
      <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>admin</role-name>
      <role-name>user</role-name>
      <role-name>moderator</role-name>
    </auth-constraint>
  </security-constraint>

  ...
  <login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>MyRealm</realm-name>
  </login-config>
  ...
</web-app>
```

It is important to note the `<realm-name>MyRealm</realm-name>`. This is the linkage to the jetty Realm (a HashUserRealm in this instance). You'll specify this same name when you set up the Realm. Here's the code you need:

```
Server server = new Server();

Connector connector = new
SelectChannelConnector();
connector.setPort(8080);
server.setConnectors(new
Connector[]{connector});

WebAppContext webappcontext = new
WebAppContext();
webappcontext.setContextPath("/mywebapp");
webappcontext.setWar("./path/to/my/war/orExp
lodedwar");

HandlerCollection handlers= new
HandlerCollection();
handlers.setHandlers(new
Handler[]{webappcontext, new
DefaultHandler()});

server.setHandler(handlers);
HashUserRealm myrealm = new
HashUserRealm("MyRealm",System.getProperty("
jetty.home")+"/etc/realm.properties");
server.setUserRealms(new
UserRealm[]{myrealm});

server.start();
server.join();
```

Programmatic security constraints (no web.xml)

If you don't wish to use a web.xml file, you can specify your <security-constraint>s in code instead. Here's how we'd code the same security constraints for the `web.xml` file above:

```
import org.mortbay.jetty.security.*;

Server server = new Server();

Connector connector = new
SelectChannelConnector();
connector.setPort(8080);
server.setConnectors(new
Connector[]{connector});

Constraint constraint = new Constraint();
constraint.setName(Constraint.__BASIC_AUTH);
;
constraint.setRoles(new
String[]{"user","admin","moderator"});
constraint.setAuthenticate(true);

ConstraintMapping cm = new
ConstraintMapping();
cm.setConstraint(constraint);
cm.setPathSpec("/*");

SecurityHandler sh = new SecurityHandler();
sh.setUserRealm(new
HashUserRealm("MyRealm",System.getProperty("
jetty.home")+"/etc/realm.properties"));
sh.setConstraintMappings(new
```

```
ConstraintMapping[]{cm});

WebAppContext webappcontext = new
WebAppContext();
webappcontext.setContextPath("/mywebapp");
webappcontext.setWar("./path/to/my/war/orExp
lodedwar");
webappcontext.addHandler(sh);

HandlerCollection handlers= new
HandlerCollection();
handlers.setHandlers(new
Handler[]{webappcontext, new
DefaultHandler()});
```

```
server.setHandler(handlers);
server.start();
server.join();
```

Running

Assuming you deployed a webapp at the context path `/mywebapp` by running your application, then surfing to:

```
http://localhost:8080/mywebapp
```

will cause a dialog box to pop up prompting you for your username and password. This is how BASIC authentication obtains your authentication information.

The example properties file in `$jetty.home/etc/realm.properties` defines the following users and their roles:

- user `jetty` with role `user`
- user `admin` with roles `server-administrator,content-administrator,admin`

Our security constraint only allows users in these roles to access it:

- `admin`
- `user`
- `moderator`

Therefore, using the default `$jetty.home/etc/realm.properties`, only the users:

- jetty
- admin

would be able to access any of the pages protected by the <security-constraint> in the webapp.

Therefore, you would log in as either the user `jetty` or the user `admin` (with passwords `jetty` and `admin` respectively) to be able to access the webapp.