Bruce Phillips Blog on Java, ColdFusion, Flex and Spry

An Introduction to Shiro (formerly JSecurity) – A Beginner's Tutorial Part 2

Posted At : April 5, 2009 12:48 PM | Posted By : Bruce Phillips
Related Categories: **Java**

Introduction

NOTE: Updated in January 2011.

In **part 1 of this tutorial**, I explained what Shiro is and created a simple web application that had no security. In part 2, I'll demonstrate how to use Shiro to apply basic security to the example web application. Please understand that I'm just a beginner in using Shiro. Any mistakes in my explanations or code are my responsibility. If you do notice that something is wrong, please post a comment.

Shiro provides powerful security capabilities including protecting areas of your application based on whether a user has been authenticated, whether a user has a specific role, or even whether a user's role has a specific permission. See the Shiro (formerly called JSecurity and Ki) references below for more about what Shiro can provide.

Database Realm

Shiro can use different realms for authenticating a user. A realm is basically a resource Shiro should use to authenticate a user. In this tutorial I'm using a database which stores the usernames and passwords of people authorized to log in and visit the secure area of the web application. See **part 1** for more information about the database.

As I mentioned in part 1, if you can use the defaults Shiro expects then configuring Shiro to work with your web application is pretty simple. However, even if you cannot use the defaults Shiro expects, you can provide Shiro with the information it needs to adapt to your configuration. To keep this tutorial simple, I'm using the defaults Shiro expects.

When using a database as the place Shiro should look for user authentication, Shiro's default is that you'll provide a DataSource that will enable Shiro to get a connection to a database that has a table named users. In the users table there must be username and password columns. In the Derby database (see part 1) provided with this tutorial there is a users table with those columns. Note that in my example users table, the passwords are stored in plain text. This is not a good security practice and Shiro supports storing passwords encrypted with different types of hashing (see: **class HashedCredentialsMatcher**).

The users table has two users:

username: bruce@hotmail.com   password: bruce

username: sue@hotmail.com   password: sue

Part 2 Example Application

In part 2's example application, I've added basic authentication using Shiro. Now only users who have successfully logged in will be able to view JSPs stored in the secure folder.

You can download the part 2 example application at **http://www.brucephillips.name/jsecurity_examples/somesecurity_mvn.zip**. The download is an archived Eclipse dynamic web project (that uses Maven). You can import this archived project (named somesecurity) into Eclipse and then run it on a Tomcat server from within Eclipse. Be sure to **review part 1** of this tutorial for how to setup the Derby database.

You can also use the Maven jetty plugin (see reference below for how to install Maven if you don't already have Maven) to run the web application if you're not using Eclipse and Tomcat. Just open a command window and navigate to where you unzipped the somesecurity_mvn.zip download. Make sure you're in the somesecurity directory. Then do the following (in this example I unzipped somesecurity_mvn.zip to c:\jsecurity_examples):

```
c:\jsecurity_examples\somesecurity\mvn clean
```

```
c:\jsecurity_examples\somesecurity\mvn jetty:run
```

Once you see [INFO] Started Jetty Server in the command window, open your web browser and go to this URL: **http://localhost:8080/somesecurity/**. You should see the contents of the index.jsp. To stop the Jetty server type control-c in the command window.

Since this web application has some security you should NOT be able to open the web pages that are in the secure folder (secure/index.jsp and secure/users.jsp) without first logging in using one of the users listed above.

Incorporating Shiro Into The Web Application

Here are the general steps I followed to add security using Shiro to the **project I described in part 1**. I'll explain these steps more fully below.

1. Added dependencies for shiro-core and shiro-web artifacts to pom.xml so that Maven will get the Shiro jar files.
2. Added ehcache.xml to src/main/resources. I just copied this file from the sample applications that come with

ARCHIVES BY SUBJECT

ColdFusion (26) [**RSS**]
Flex (75) [**RSS**]
Java (76) [**RSS**]
Running (10) [**RSS**]
Spry (13) [**RSS**]

CALENDAR

<< August 2012 >>

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

LATEST FROM CFBLOGGERS.ORG

Why I Still Love ColdFusion
Java Shots 9 - Java EE7 and HTML 5, SAML Explained, and Git Internals
New Product Marketing Manager for ColdFusion
Getting HTML5 Ready - ContentEditable
NCDevCon Session Spotlight: Designing the mobile experience

LINKS

Bruce Phillips' Main Web

Blog Home

Flex.org - The Directory for Flex

NAVIGATION

Home

RECENT ENTRIES

No recent entries.

RECENT COMMENTS

Android App Development - Using The ActionBar Widget
mike said: That was fantastic, been looking for that for a while now. thanks for putting in the effort. regards **[More]**

Using Blackboard Learn 9 Web Services - Part 4 Getting, Creating, Updating, Deleting Users
Bruce said: Baci - I"ve not used that method before and don't know much about course organizations. **[More]**

Using Blackboard Learn 9 Web Services - Part 4 Getting, Creating, Updating, Deleting Users
Baci said: Bruce, Thanks for being really the only resource as we're trying to implement some Bb webservic stuf... **[More]**

An Introduction to Shiro (formerly

the Shiro download. I believe it's used to configure caching of authentication information on the server.
3. Added the IniShiroFilter and its configuration to web.xml. This is a Servlet filter that configures and enables all Shiro functions within a web application (see: **class IniShiroFilter**)
4. Added class RoleSecurityJdbcRealm to the dao package
5. Added additional Servlet classes to handle logging in and logging out users

Here is the code from web.xml where I've setup the IniShiroFilter. (Note if you're not familiar with filters see: **http://java.sun.com/javaee/5/docs/tutorial/doc/bnagb.html**)

```
<filter>
<filter-name>ShiroFilter</filter-name>
<filter-class>org.apache.shiro.web.servlet.IniShiroFilter</filter-class>
<init-param>
<param-name>config</param-name>
<param-value>
#See Shiro API http://shiro.apache.org/static/current/apidocs/org/apache/shiro
/web/servlet/IniShiroFilter.html

#create an object of the RoleSecurityJdbcRealm
#IniShiroFilter will inject that object into the SecurityManager
[main]
```

In the config param-value, you specify the information that the IniShiroFilter needs. Under the [main] heading I tell the filter to create an object of class RoleSecurityJdbcRealm to use as a realm to validate users. If you examine the code for RoleSecurityJdbcRealm you'll see that the class just extends class JdbcRealm, which is provided by Shiro (see: **class JdbcRealm**). In RoleSecurityJdbcRealm I just need a no-argument constructor. In the constructor I setup the DataSource the class should use to get connections to the Derby database. Remember, I'm storing my usernames and passwords in the users table in that database.

By setting up this subclass of JdbcRealm to use a DataSource that gets connections to the Derby database, Shiro will be able to query the users table to authenticate the users who try to log in. This works automatically because I've configured the users table following Shiro's defaults (see above). Consult the Shiro references below for information about how to setup Shiro if you cannot use Shiro's defaults (for example your project already has a table with usernames and passwords but the table and/or columns have different names).

The only other configuration I needed to specify in the config param-values was under the [urls] section. Here I specified that any JSPs in the secure folder can only be viewed by authenticated users. If a user tries to view a JSP in the secure folder and that user has not logged in, the IniShiroFilter will redirect the user to /login.jsp. Again this is part of the default and you can override this if you want the IniShiroFilter to forward to a different login JSP.

When a user fills in the form on /login.jsp, the application calls the LoginUser servlet. If you view the source code for this Servlet's doPost method you'll see how I used Shiro to login in the user. There are extensive comments in the source code, but basically there are just a few steps you need to implement:

> Create a UsernamePasswordToken object using the username and password provided by the user (see: **class UsernamePasswordToken**)
>
> Have the SecurityUtils class get the Subject object, which is the user associated with this request. If the user has not yet logged in, this will be an anonymous user who is not authenticated. (see: **class SecurityUtils**)
>
> Have the Subject object execute its login method, passing the method the UserNamePasswordToken object created earlier. (see: **interface SubjectI**)

Consult the Shiro API for more information about these classes. The JavaDoc provided with the Shiro API is pretty good with providing explanations about Shiro's classes and their methods.

After calling the login method, Shiro will get a connection to the database and query the users table for the password associated with the provided username. If the username provided cannot be found in the users table, an UnknownAccountException is thrown. If the username is found, but the password associated with that username doesn't match the password the user provided, an IncorrectCredentialsException is found.

If the login is successful, a new Subject object is created and stored in the session scope associated with this request.

Shiro will automatically check that the current user has been authenticated before allowing the user to view any JSPs stored in the secure folder because of the configuration in web.xml. If you have a Servlet that needs to verify if a user is authenticated (logged in successfully), you can programmatically check a user's status. For example, see the doPost method in the Servlet class GetAllUsers. The basic steps are:

> Have the SecurityUtils class get the Subject object.
>
> Use the Subject object's isAuthenticated method, which returns true if "this Subject/user has proven their identity *during their current session* by providing valid credentials matching those known to the system, false otherwise" (**see method isAuthenticated**).

See the Servlet class LogoutUser for how to log out a user.

Summary and What's Next?

With minimal additions to the web application I introduced in part 1, I was able to add security using Shiro. Again, any mistakes in my explanations or code are my fault. Please comment if you find something that needs to be corrected.

Please note that I've just touched on the very basics of using Shiro to provide security to a web application. Shiro has extensive capabilities and you should read through the references to understand it better.

JSecurity) – A Beginner's Tutorial Part 4
mark said: In secure/index.jsp had to change jsec:principal to shiro:principal. **[More]**

NetBeans Java IDE 7.2 - Comments From An Eclipse User
Ralf Eichinger said: I used Eclipse for years now and tried to switch to Netbeans 7.1.2. Basically Netbeans does the job,... **[More]**

RSS

RSS FEED

SEARCH

Search

SUBSCRIBE

Enter your email address to subscribe to this blog.

Subscribe

TAGS

coldfusion flex java running spry

In the **next tutorial**, I'll examine how to implement security based on a user's role.

References:

1. An Introduction to Shiro (formerly JSecurity) – A Beginner's Tutorial Part 1, **http://www.brucephillips.name /blog/index.cfm/2009/4/5/An-Introduction-to-Ki-formerly-JSecurity--A-Beginners--Tutorial-Part-1**
2. Some Security Example Application, **http://www.brucephillips.name/jsecurity_examples /somesecurity_mvn.zip**
3. Apache Shiro **http://shiro.apache.org/**
4. Apache Shiro API, **http://shiro.apache.org/static/current/apidocs/**
5. Apache Shiro Mailing Lists, **http://shiro.apache.org/mailing-lists.html**
6. Presentation on JSecurity to the Charlotte Java Users Group, **http://www.jsecurity.org/files/JSecurity.pdf**
7. Apache Derby, **http://db.apache.org/derby/**
8. Apache Tomcat, **http://tomcat.apache.org/**
9. Jetty, **http://jetty.mortbay.org/jetty5/index.html**
10. Maven: The Definitive Guide, **http://www.sonatype.com/books/maven-book/reference/public-book.html**
11. Developing with Eclipse and Maven, **http://www.sonatype.com/books/m2eclipse-book/reference /index.html**

Comments (4) | **Print** | **Send** | **del.icio.us** | **Digg It!** | **Linking Blogs** | 17454 Views

Related Blog Entries

- **An Introduction to Shiro (formerly JSecurity) – A Beginner's Tutorial Part 3** (April 5, 2009)
- **An Introduction to Shiro (formerly JSecurity) – A Beginner's Tutorial Part 1** (April 5, 2009)

Comments (Comment Moderation is enabled. Your comment will not appear until approved.)

[**Add Comment**] [**Subscribe to Comments**]

Hi,

I'm testing Ki (formerly JSecurity), using your tutorial and maybe there is a bug in it. In the somesecurity project, if i click on the link list-of-users, there is no control. So i added the following line in web.xml :

/GetAllUsers/** = authc

I'm not sure the bug is yours because i had to change a lot the project to run it :)

Best regards

JC

# Posted By JC Vidal | 4/27/09 5:16 AM

JC

I downloaded the somesecurity project, unzipped it, and ran the maven clean and maven jetty:run commands. I then opened a web browser and went to the URL **http://localhost:8080/somesecurity**. Everything ran correctly and I'm forced to login after clicking on the link for the secret user's list or the link for pages in the secure area.

So I don't understand why it didn't work for you. You should not have to modify anything in the project if you've followed the steps in part 1 for setting up the Derby database. You should be able to just download the project, unzip it, and use maven as this article describes to run the project in your web browser.

Can you explain further why you weren't able to do that?

Bruce

# Posted By Bruce | 4/27/09 7:06 AM

Dear Bruce

My PC configuration is a mess ! So i was obliged to modify some trunks.

But it daes'nt matter, now. For a better understanding of JSecurity, i use now a more touchy Realm (an XML file) to try a deep comprehension of JSecurity arcanes :)

Thanks for everythink !

# Posted By JC Vidak | 4/27/09 8:16 AM

Steps are pretty clear. So far everything is working as expected for part1 and part2.

-Santosh

# Posted By Santosh | 7/22/09 6:04 AM

[**Add Comment**]