Web application security sample

{scrollbar}

top

This article focuses on the web application security related features of the Apache Geronimo server. The sample application covered in this article is a basic time reporting system that uses Servlets, JSPs and J2EE declarative security. In addition to above features it uses Geronimo's embedded Derby database to store user information of the system. Even though this application uses a database to hold user information, it is merely for configuration purposes. For detailed information on the usage of JDBC in Geronimo, refer the Simple database access sample application article.

After reading this article you should be able to configure Geronimo application server for web applications with declarative security features.

This article is organized in to following sections.

Web Applications in Geronimoweb

Apache Geronimo includes a Web application container supporting J2EE Web applications. The Web container itself supports basic configuration such as network ports and SSL options, and each Web application may include Geronimo-specific configuration information as well. Web applications participate in the Geronimo security infrastructure, so authenticating to a Web application allows access to secure EJBs and Connectors as well.

Apache Geronimo currently supports two Web containers: Jetty and Tomcat.

Jetty

Jetty is a 100% Java HTTP Server and Servlet Container. This means that you do not need to configure and run a separate Web server in order to use servlets and JSPs to generate dynamic content. Jetty is a fully featured Web server for static and dynamic content.

Unlike separate server/container solutions, Jetty's Web server and Web application run in the same process without interconnection overheads and complications. Furthermore, as a pure java component, Jetty can be easily included in your application for demonstration, distribution or deployment. Jetty is available on all Java supported platforms.
http://jetty.mortbay.org/jetty/index.html

Tomcat

Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies.
http://tomcat.apache.org/

Application Overview overview

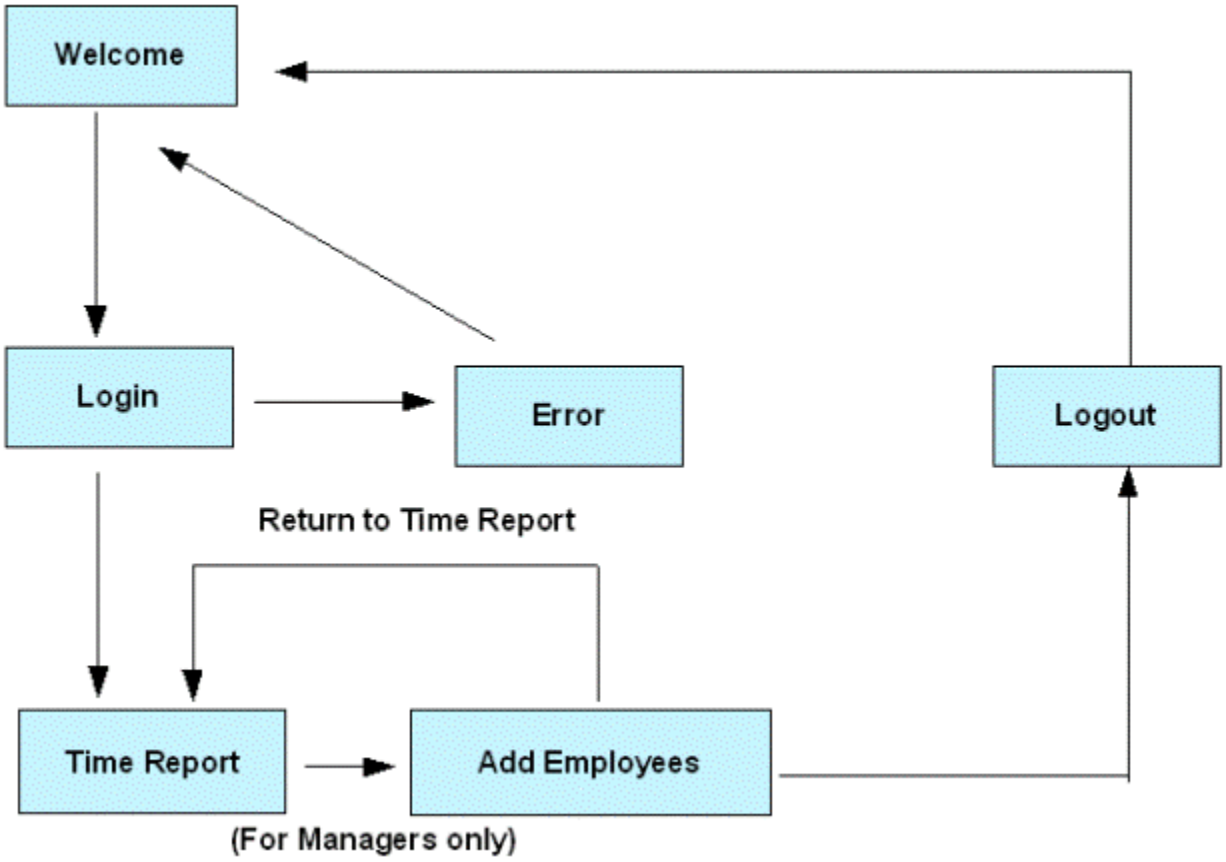
The Time Report application helps to report working times of different projects. Even though this is not a full blown time reporting application, it covers most of the displaying and security related features web applications in Apache Geronimo.

This sample application allows two types of user groups to report their time tasks to the system, namely managers and employees. Both type of users have to provide their credentials before reporting time tasks. Managers are more of super users of the system, so they can add employees to the system additionally.

The Time Report application has the following list of pages.

- Welcome
- Login
- Time Report
- Add Employees
- Logout

The following figure illustrates overview of application flow:



By default the given sample application is directed to the Welcome page with a link to the Time Report functionality. The users can access the Time Report page by providing a valid user name and password to the Login page. If those provided user credentials are from a manager role, Time Report page will display an additional link to the Add Employees functionality too.

Application contents

Following is the main folder hierarchy of the Time Reporting application. It display both JSPs and configuration files used in the application.

java |- employee |- index.jsp |- login |- login.jsp |- login_error.jsp |- logout.jsp |- manager |- index.jsp |- WEB_INF |- geronimo-web.xml |- web.xml |- index.jsp

In addition to the above JSPs and configurations, two other servlets are also required to fulfill the business logic of the application.

- AddTimeRecordServlet - Read the input data from the Time Report page
- AddEmployeeServlet - Capture input information from Add Employee page

Security configuration of the Time Report application is handled by **geronimo-web.xml** and **web.xml** files. **geronimo-web.xml** is used to define user roles of the application with **TimeReportRealm**.

The first part of **geronimo-web.xml** is straight forward. However, the security configuration is tricky. The <security-realm-name> is described in the <security> element through a sequence of declarations in the <realms> element.

While the web.xml specifies the security roles, the geronimo-web.xml maps to which specific users or groups in the Geronimo security realms they belong to. If there is a user that is not logged in, it defaults to what is defined in the <default-principal> element.

There are two roles that are issued in this project: manager and employee. Since a manager is also an employee of the company, it will be listed under employee too. However, it also has its spot under the 'manager' role.

xmlsolidgeronimo-web.xml <?xml version="1.0" encoding="UTF-8"?> <web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.1"> <environment> <moduleId> <groupId>${pom.groupId}</groupId> <artifactId>${pom.artifactId}</artifactId> <version>${version}</version> <type>war</type> </moduleId> < /environment> <context-root>/timereport</context-root> <security-realm-name>TimeReportRealm</security-realm-name> <security> <default-principal realm-name="TimeReportRealm"> <principal name="anonymous" class="org.apache.geronimo.security.realm.providers.GeronimoUserPrincipal" /> < /default-principal> <role-mappings> <role role-name="employee"> <realm realm-name="TimeReportRealm"> <principal name="EmployeeGroup" class=" org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal" /> </realm> <realm realm-name="TimeReportRealm"> <principal name=" ManagerGroup" class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal" /> </realm> </role> <role role-name="manager"> <realm realm-name="TimeReportRealm"> <principal name="ManagerGroup" class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal" /> < /realm> </role> </role-mappings> </security> </web-app>

web.xml will map the defined user roles to resources in the web application. It also defines the login configurations of the application.

xmlsolidweb.xml <?xml version="1.0" encoding="UTF-8"?> <web-app xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001 /XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4"> <welcome-file-list> <welcome-file>index.jsp</welcome-file> </welcome-file-list> <security-constraint> <web-resource-collection> <web-resource-name>employee< /web-resource-name> <url-pattern>/employee/*</url-pattern> </web-resource-collection> <auth-constraint> <role-name>employee</role-name> </auth-constraint> </security-constraint> <security-constraint> <web-resource-collection> <web-resource-name>manager</web-resource-name> <url-pattern> /manager/*</url-pattern> </web-resource-collection> <auth-constraint> <role-name>manager</role-name> </auth-constraint> </security-constraint> <login-config> <auth-method>FORM</auth-method> <realm-name>TimeReportRealm</realm-name> <form-login-config> <form-login-page>/login/login.jsp< /form-login-page> <form-error-page>/login/login_error.jsp</form-error-page> </form-login-config> </login-config> <security-role> <role-name>employee< /role-name> </security-role> <security-role> <role-name>manager</role-name> </security-role> <servlet> <display-name>AddTimeRecordServlet< /display-name> <servlet-name>AddTimeRecordServlet</servlet-name> <servlet-class>org.apache.geronimo.samples.timereport.web. AddTimeRecordServlet</servlet-class> </servlet> <servlet> <display-name>AddEmployeeServlet</display-name> <servlet-name>AddEmployeeServlet< /servlet-name> <servlet-class>org.apache.geronimo.samples.timereport.web.AddEmployeeServlet</servlet-class> </servlet> <servlet-mapping> <servlet-name>AddTimeRecordServlet</servlet-name> <url-pattern>/employee/add_timerecord</url-pattern> </servlet-mapping> <servlet-mapping> <servlet-name>AddEmployeeServlet</servlet-name> <url-pattern>/manager/add_employee</url-pattern> </servlet-mapping> </web-app>

To restrict access to the Add Employee functionality from Time Report page, programmatic authentication has beeen used as in indicated below.

javasolidemployee/index.jsp ...
 <%if(request.isUserInRole("manager")){%> <A href="../manager/">Add Employees</A>
 ...

geronimo-application.xml tells the application that there is a database pool that needs to be deployed as well. The **security realm** configurations are included along with this db pool. The db pool is defined in TimeReportPool.xml and the driver that is needs in order to be deployed is the tranql-connector-ra-1.3.rar file--these two files will reside on the top level layer of the resultant EAR file.

xmlsolidgeronimo-application.xml <?xml version="1.0" encoding="UTF-8"?> <application xmlns="http://geronimo.apache.org/xml/ns/j2ee/application-1.2"> <environment xmlns="http://geronimo.apache.org/xml/ns/deployment-1.2"> <moduleId> <groupId>${pom.groupId}</groupId> <artifactId>${pom.artifactId}< /artifactId> <version>${version}</version> <type>ear</type> </moduleId> </environment> <module> <connector>tranql-connector-ra-1.3.rar</connector> <alt-dd>TimeReportPool.xml</alt-dd> </module> </application>

TimeReportPool.xml defines two things: the database pool itself and a security realm. As shown, the first part is similar to any other db pool plan. The second part, are the essentials for a security realm plan. By combining the two into a separate file, we can ship a db pool and a security realm with the application so it will require less things to install.

xmlsolidTimeReportPool.xml <?xml version="1.0" encoding="UTF-8"?> <connector xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector-1.2"> <dep: environment xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2"> <dep:moduleId> <dep:groupId>console.dbpool</dep:groupId> <dep: artifactId>TimeReportPool</dep:artifactId> <dep:version>1.0</dep:version> <dep:type>rar</dep:type> </dep:moduleId> <dep:dependencies> <dep: dependency> <dep:groupId>org.apache.geronimo.configs</dep:groupId> <dep:artifactId>j2ee-security</dep:artifactId> <dep:type>car</dep:type> </dep: dependency> <dep:dependency> <dep:groupId>org.apache.geronimo.configs</dep:groupId> <dep:artifactId>system-database</dep:artifactId> <dep: type>car</dep:type> </dep:dependency> </dep:dependencies> </dep:environment> <!--db pool fragment--> <resourceadapter> <outbound-resourceadapter> <connection-definition> <connectionfactory-interface>javax.sql.DataSource</connectionfactory-interface> <connectiondefinition-instance> <name>TimeReportPool</name> <config-property-setting name="Driver">org.apache.derby.jdbc.EmbeddedDriver</config-property-setting> <config-property-setting name="UserName">app</config-property-setting> <config-property-setting name="ConnectionURL">jdbc:derby:TimeReportDB< /config-property-setting> <connectionmanager> <local-transaction/> <single-pool> <max-size>10</max-size> <min-size>0</min-size> <match-one/> < /single-pool> </connectionmanager> </connectiondefinition-instance> </connection-definition> </outbound-resourceadapter> </resourceadapter> <!-- security realm fragment--> <gbean name="TimeReportRealm" class="org.apache.geronimo.security.realm.GenericSecurityRealm"> <attribute name=" realmName">TimeReportRealm</attribute> <reference name="ServerInfo"> <name>ServerInfo</name> </reference> <xml-reference name=" LoginModuleConfiguration"> <log:login-config xmlns:log="http://geronimo.apache.org/xml/ns/loginconfig-1.1"> <log:login-module control-flag="REQUIRED" wrap-principals="false"> <log:login-domain-name>TimeReportRealm</log:login-domain-name> <log:login-module-class>org.apache.geronimo.security. realm.providers.SQLLoginModule</log:login-module-class> <log:option name="jdbcDriver">org.apache.derby.jdbc.EmbeddedDriver</log:option> <log: option name="jdbcUser">app</log:option> <log:option name="userSelect">select userid, password from users where userid=?</log:option> <log:option name="groupSelect">select userid, groupname from usergroups where userid=?</log:option> <log:option name="jdbcURL">jdbc:derby:TimeReportDB< /log:option> </log:login-module> </log:login-config> </xml-reference> </gbean> </connector>

Tools used

The tools used for developing and building the Time Reporting sample application are:

Apache Maven 2

Maven is a popular open source build tool for enterprise Java projects, designed to take much of the hard work out of the build process. Maven uses a declarative approach, where the project structure and contents are described, rather than the task-based approach used in Ant or in traditional make files, for example. This helps enforce company-wide development standards and reduces the time needed to write and maintain build scripts. The declarative, lifecycle-based approach used by Maven 1 is, for many, a radical departure from more traditional build techniques, and Maven 2 goes even further in this regard. Maven 2 can be download from the following URL:
http://maven.apache.org

Configuring, Building and Deploying the Sample Application configure

Download the Time Reporting application from the following link:
Time Report

After extracting the zip file, the <time_report> directory is created.

Source Code

You can checkout the source code of this sample from SVN:

svn checkout http://svn.apache.org/repos/asf/geronimo/samples/trunk/samples/timereport

Configuring

Since Time Reporting application is going to use J2EE declarative security, user needs to create a database to hold the information and deploy the security realm.

Create Database to hold User Information

After starting Apache Geronimo server, log into the console and follow the given steps to create the **TimeReportDB** to hold user information for the application.

solidTimeReportDB.sql CREATE TABLE users(userid VARCHAR(15) PRIMARY KEY, password VARCHAR(15), name VARCHAR(40)); CREATE TABLE usergroups(userid VARCHAR(15), groupname VARCHAR(20), PRIMARY KEY (userid, groupname)); INSERT INTO users VALUES('emp1', 'pass1', 'Employee 1'); INSERT INTO users VALUES('emp2', 'pass2', 'Employee 2'); INSERT INTO users VALUES('mgm1', 'pass3', 'Manager 1'); INSERT INTO users VALUES('mgm2', 'pass4', 'Manager 2'); INSERT INTO usergroups VALUES('emp1', 'EmployeeGroup'); INSERT INTO usergroups VALUES ('emp2', 'EmployeeGroup'); INSERT INTO usergroups VALUES('mgm1', 'ManagerGroup'); INSERT INTO usergroups VALUES('mgm2', 'ManagerGroup');

1. Select **DB Manager** link from the **Console Navigation** in the left.
2. Give the database name as **TimeReportDB** in the **Create DB** field and click **Create** button.
3. Select TimeReportDB to the **Use DB** field.
4. Open **TimeReportDB.sql** in the **time_report/config** directory.
5. Paste the content **TimeReportDB.sql** to the **SQL Commands** text area and press **Run SQL** button.

Building

Time Report application comes with an pom.xml to help users to build from source code. Open a command prompt window and navigate to the **timereport** directory and just give **mvn install site** command to build. This will create a **timereport-ear-2.0-SNAPSHOT.ear** under the **timereport** folder. Now, you are ready to deploy the Time Report application in the Geronimo Application server.

Deploying

Deploying the sample application is pretty straight forward, since we are using the Geronimo Console.

1. Scroll down to **Deploy New** from the **Console Navigation** panel.
2. Load **timereport-ear-2.0-SNAPSHOT.ear** from **time_report** folder in to the **Archive** input box.
3. Press **Install** button to deploy application in the server.

Back to Top

Testing of the Sample Application testing

To test the sample application open a browser and type http://localhost:8080/timereport. It will forward to the Welcome page of the application.

User can access Time Report page providing username as **emp1** and password with **pass1**. To login to the application as a Manager provide **mgm1** and **pass3** credentials.

Summary summary

This article has shown you how to deploy web application in to the Geronimo Application server with J2EE declarative security features. You followed step-by-step instructions to build, deploy and test the sample application.

Some highlights of the article are:-

- Apache Geronimo provides two different web containers namely Jetty and Tomcat.
- Create a database to hold security data with built-in Derby.
- Define security roles in Geronimo Web applications.
- Deploy deployment plans and web archives using the Geronimo Console.