

Copyright © 2005-2007

1. What is This Guide about / Goals	1
2. The Project	2
3. JGuard Configuration	10

Chapter 1. What is This Guide about / Goals

This guide is the result of the JGuard Team's efforts in order to create a guide easy for beginners to follow and understand the way JGuard works, and, at the end, have a very basic Application secured by JGuard.

This guide is based on JGuard's Version 1.0.2

Goals of this Document:

- Point what is needed in order to secure an application using JGuard
- Explain how JGuard's flow is build based on the config files

To build the app, it was needed:

- JGuard 1.0.2
- Maven 2.0.5
- Sun Java JDK 1.6.5 (revision 11)
- Eclipse 3.2 M20070212-1330 and latest WTP available from update site
- Apache Tomcat version 5.5.20
- Apache Struts version 1.3.5
- Jakarta-Taglibs Standard-1.1 version 1.1.2

For any information regarding JGuard, please see <http://www.jguard.net/> and the foruns at <https://sourceforge.net/projects/jguard>

Remarks:

if you use maven to build your project, you will not have to add the libs in the project for yourself, because maven will do it for you and will also include the dependencies's depedencies, so, it's a much more comfortable and easy way to do it. In order to build the app using maven, use "**mvn -e -o clean package**". It will build your package in the offline way. If you need to check for updates while building, use "**mvn -e -U clean package**" instead. If you got a 'BUILD SUCCESSFULL' message, check for the .war at target dir.

Chapter 2. The Project

In this Chapter we will build, step by step, an Dynamic Web Project using WTP.

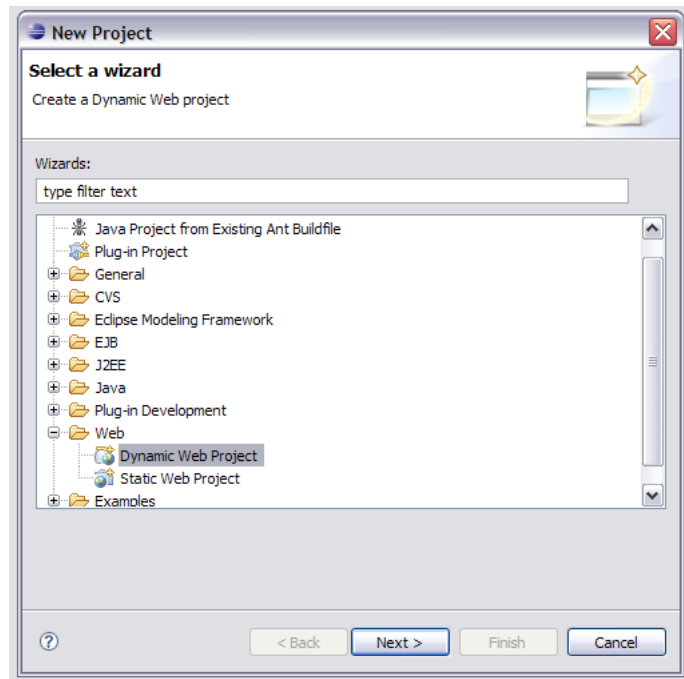


Figure 2.1. Step 1: Creating the Dynamic Web Project

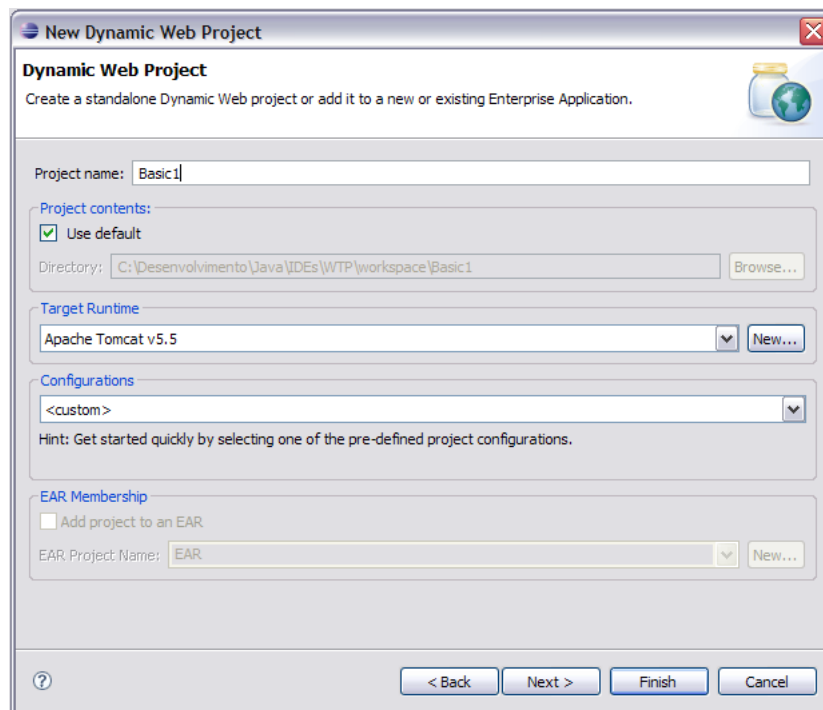


Figure 2.2. Step 2: Choosing the Project's Name and Target Runtime

Give the Project a name (suggestion: 'Basic1') and then click 'Next >'

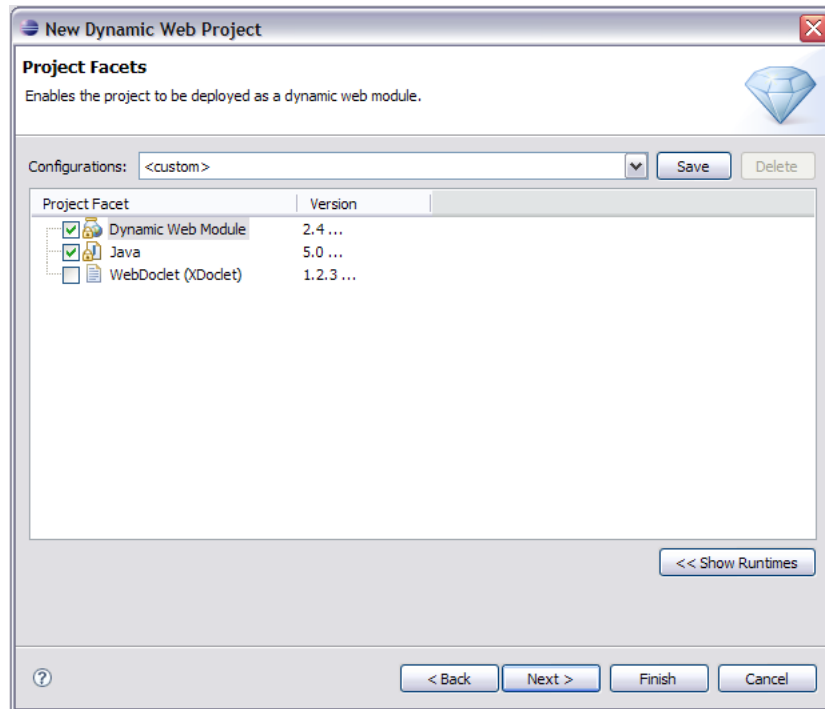


Figure 2.3. Step 3: Configurations

Just click 'Next >'

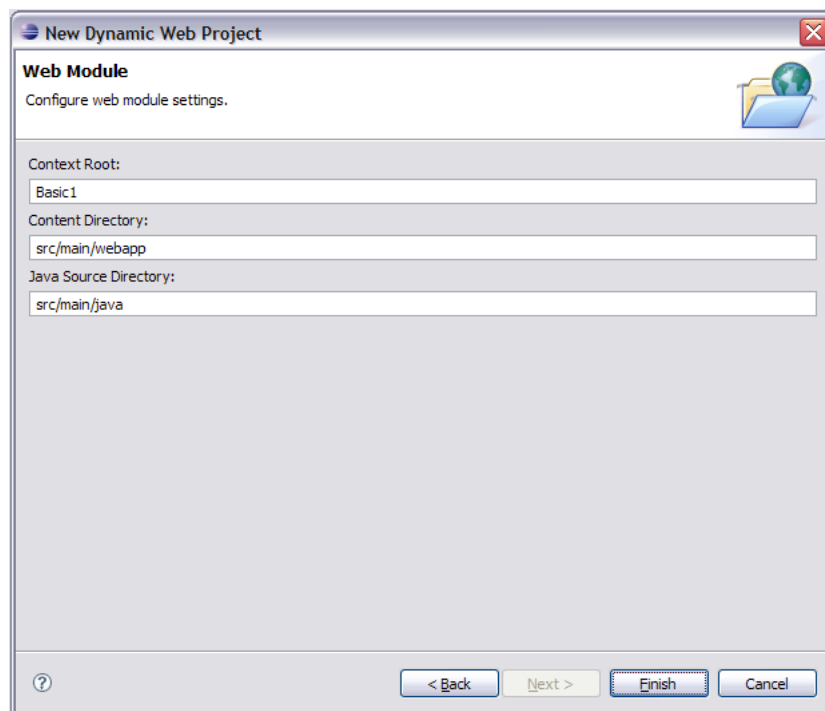


Figure 2.4. Step 4: Setting Java Source Directory

One note about this last step: As Maven 2 will be our official building tool, we changed Java Source Directory from src to src/main/java, in order to be compliant about Maven 2 source structure. Check out <http://jguard.xwiki.com/xwiki/bin/view/doc/devEclipse> if you have doubts.

Click 'Finish'

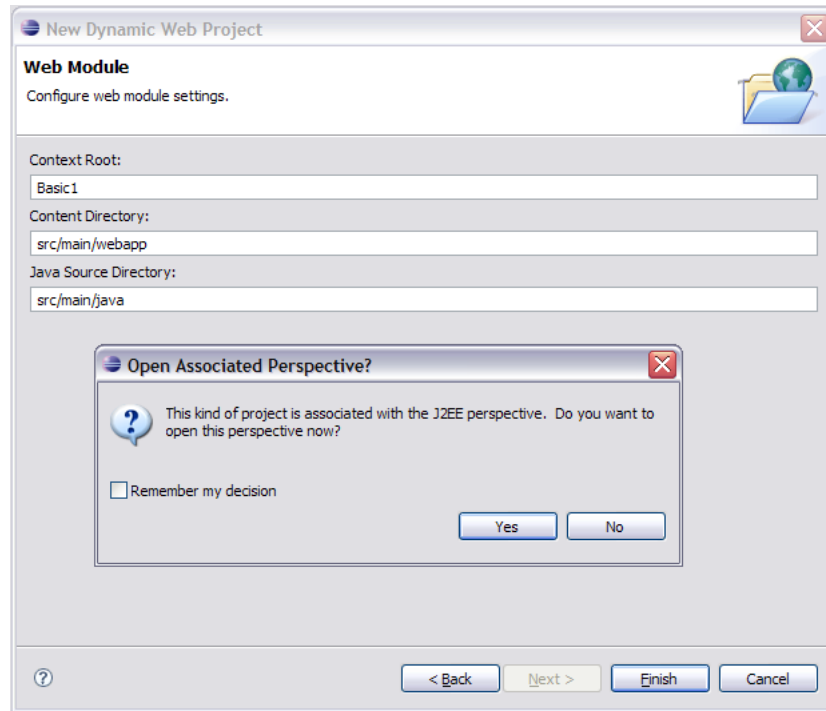


Figure 2.5. After Finished

If it is the first time you have created a project like this, you will be asked if you want to open the J2EE perspective, which is a personal choice.

After the Dynamic Web Project was created, we should create the structure which will hold the files and config we will use.

- inside WEB-INF create a folder named conf
- inside WEB-INF/conf create three folders: jGuard, struts and tld

The Folder Structure should look like this:

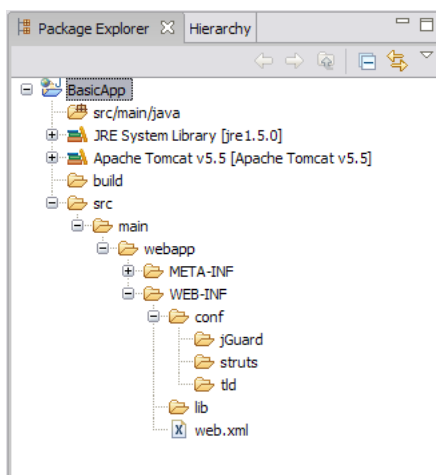


Figure 2.6. Folder Structure for Basic 1

First of all, we should create and run out build script, this way, the libs we will need will be downloaded by Maven 2 in case it was not downloaded yet. Save the file below as pom.xml in the project root dir (the dir where is .project and .classpath - usually workspace/projectname).

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-
v4_0_0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>net.sf.jguard</groupId>
<artifactId>jguard-example-basic1</artifactId>
<version>1.0.2</version>
<packaging>war</packaging>
<name>jGuard Basic-1 Example</name>
<url>http://jguard.net</url>
<dependencies>
  <!-- jee lib, use provided appserver lib at runtime -->
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.3</version>
    <scope>provided</scope>
  </dependency>
  <!-- struts core -->
  <dependency>
    <groupId>org.apache.struts</groupId>
    <artifactId>struts-core</artifactId>
    <version>1.3.5</version>
  </dependency>
  <!-- struts extras (ForwardAction) -->
  <dependency>
    <groupId>org.apache.struts</groupId>
    <artifactId>struts-extras</artifactId>
    <version>1.3.5</version>
  </dependency>
  <!-- struts taglib -->
  <dependency>
    <groupId>org.apache.struts</groupId>
    <artifactId>struts-taglib</artifactId>
    <version>1.3.5</version>
  </dependency>
  <!-- JSTL standard -->
  <dependency>
    <groupId>taglibs</groupId>
    <artifactId>standard</artifactId>
    <version>1.1.2</version>
  </dependency>
  <!-- JGuard core -->
  <dependency>
    <groupId>net.sf.jguard</groupId>
    <artifactId>jguard-core</artifactId>
    <version>1.0.2</version>
  </dependency>
  <!-- JGuard extensions -->
  <dependency>
    <groupId>net.sf.jguard</groupId>
    <artifactId>jguard-ext</artifactId>
    <version>1.0.2</version>
  </dependency>
  <!-- JGuard extensions for Java 1.5 -->
  <dependency>
    <groupId>net.sf.jguard</groupId>
    <artifactId>jguard-ext-java-5</artifactId>
    <version>1.0.2</version>
  </dependency>
  <!-- JGuard JEE extensions -->
  <dependency>
    <groupId>net.sf.jguard</groupId>
    <artifactId>jguard-jee</artifactId>
    <version>1.0.2</version>
  </dependency>
  <!-- JGuard JEE extras -->
  <dependency>
```

```

<groupId>net.sf.jguard</groupId>
<artifactId>jguard-jee-extras</artifactId>
<version>1.0.2</version>
</dependency>
</dependencies>
<build>
  <plugins />
</build>
</project>

```

after save the file, execute **mvn -e -U clean install**

if after Maven 2 ran you got the message BUILD SUCCESSFUL, everything went OK, else, look at the Maven 2 configuration page in order to see if you've missed something in your setup.

```

[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
[INFO] Total time: 23 seconds
[INFO] Finished at: Tue Mar 27 22:59:41 BRT 2007
[INFO] Final Memory: 6M/12M
[INFO] -----

```

After that, all libraries we will use will be available from M2_REPO, now we are going to add them to the app classpath.

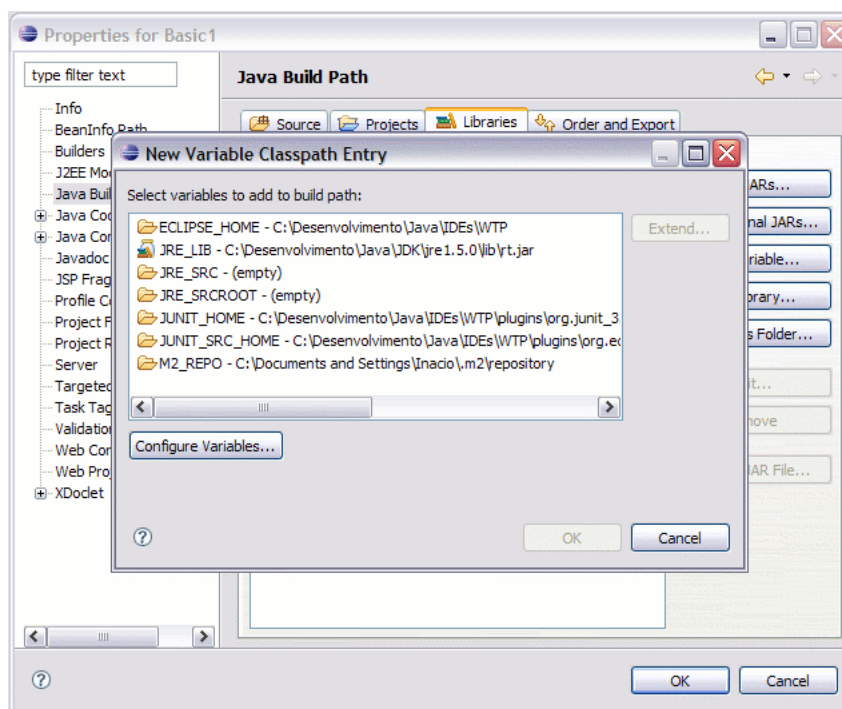


Figure 2.7. Adding libs to classpath

As M2_REPO is a directory, Eclipse will ask to extend M2_REPO so we can add what we really want to add. Extend and search for these libs:

- M2_REPO/net/sf/jguard/jguard-core/1.0.2/jguard-core-1.0.2.jar
- M2_REPO/net/sf/jguard/jguard-ext/1.0.2/jguard-ext-1.0.2.jar
- M2_REPO/net/sf/jguard/jguard-ext-java-5/1.0.2/jguard-ext-java-5-1.0.2.jar

- M2_REPO/net/sf/jguard/jguard-jee/1.0.2/jguard-jee-1.0.2.jar
- M2_REPO/net/sf/jguard/jguard-jee-extras/1.0.2/jguard-jee-extras-1.0.2.jar
- M2_REPO/org/apache/struts/struts-core/1.3.5/struts-core-1.3.5.jar
- M2_REPO/taglibs/standard/1.1.2/standard-1.1.2.jar

After added, it should look like this:

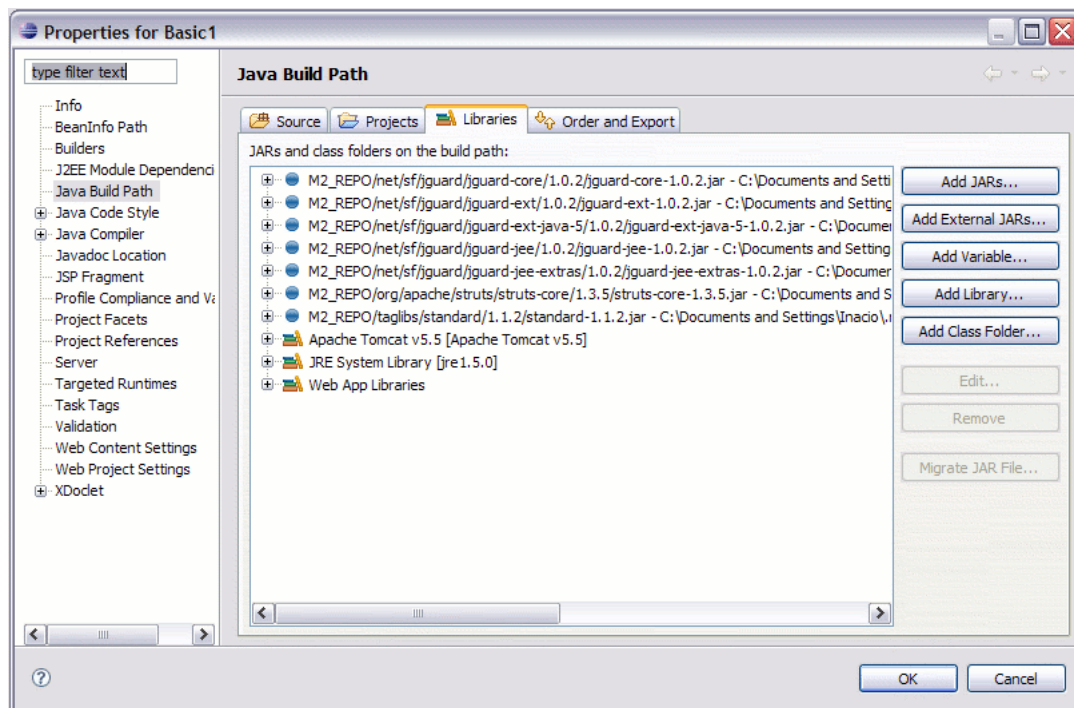


Figure 2.8. Adding libs to classpath

Before we can start writing our .jspxs, we will, at first, copy the .tlds files.

From standard-1.1.2.jar!/META-INF, copy c.tld to src/main/webapp/WEB-INF/config/tld

From struts-taglib-1.3.5.jar!/META-INF/tld, copy struts-html.tld to src/main/webapp/WEB-INF/config/tld

Now, we are going to create the .jsp files we will use.

Fist of all, we will write our **index.jsp** file, wich will only redirect the user to the login form (usually, every system requires that the user gets identified before anything else). Place this file at *src/main/webapp/*

```
<% /*
 * index.jsp
 * This page will redirect the user to the login page.
 */
%>

<%@taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
<c:redirect url="/LoginForm.do"/>
```

Now, let's write our login form.

In order to do that, create a folder called jsp in src/main/webapp/WEB-INF

```
<% /*
 * loginForm.jsp
 * This page will get the login and password info, and will send it
 * to LogonProcess.
 */
%>

<%@ taglib prefix="html" uri="http://struts.apache.org/tags-html" %>

<html>
<head>
  <title>Authentication</title>
</head>
<body>
  <html:form action="/LogonProcess" method="post">
    <label for="login">Login</label>
    <html:text property="login" maxlength="20" size="30" styleId="login" tabindex="1" />
    <p>
      <label for="login">Password</label>
      <html:password property="password" maxlength="20" size="30" styleId="login"
      tabindex="2" />
    <p>
      <html:submit tabindex="3" />
      <html:reset tabindex="4" />
    </html:form>
  </body>
</html>
```

Now, we will create a jsp in case authentication was successful, so, the user will be authenticated and will be able to unauthenticate.

```
<% /*
 * authenticated.jsp
 * This page will show the message and let the user get unauthenticated.
 */
%>

<%@ taglib prefix="html" uri="http://struts.apache.org/tags-html" %>

you are authenticated. You may <html:link forward="/LogOff" />.
```

After the files were created, we are going now to create the struts-config.xml file, which will be located in src/main/webapp/conf/struts

```
<!DOCTYPE struts-config
PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
"http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <form-beans>
    <form-bean name="loginForm" type="org.apache.struts.action.DynaActionForm">
      <form-property name="login" type="java.lang.String" />
      <form-property name="password" type="java.lang.String" />
    </form-bean>
  </form-beans>

  <action-mappings>
    <action path="/LoginForm" name="loginForm"
      type="org.apache.struts.actions.ForwardAction" parameter="/WEB-INF/jsp/
loginForm.jsp" />
```

```

<action path="/LogonProcess"
    type="org.apache.struts.actions.ForwardAction" parameter="/WEB-INF/jsp/
authenticated.jsp" />
<action path="/LogOff"
    type="org.apache.struts.actions.ForwardAction" parameter="/index.jsp" />
</action-mappings>
</struts-config>

```

After that, we will add struts-config.xml to our web.xml in order to have a struts-based web application.

So, now, our web.xml should look like this:

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4"
    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">

<display-name>Basic1</display-name>
<description>Provides a very basic webapp secured with jGuard</description>

<servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
        <param-name>config</param-name>
        <param-value>/WEB-INF/conf/struts/struts-config.xml</param-value>
    </init-param>
    <init-param>
        <param-name>locale</param-name>
        <param-value>>false</param-value>
    </init-param>
    <init-param>
        <param-name>debug</param-name>
        <param-value>2</param-value>
    </init-param>
    <init-param>
        <param-name>detail</param-name>
        <param-value>2</param-value>
    </init-param>
    <init-param>
        <param-name>convertNull</param-name>
        <param-value>>true</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>

<welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>

</web-app>

```

Now we have a basic **not secured yet** app. So, it's time to build and deploy it, and see if it works as a normal app. So, let's build it again.

If you got a build successfull message from mvn, you've got it, build was OK. You can deploy it and test it.

Chapter 3. JGuard Configuration

Now, that we have a basic app, it's time to make it secure.

The basic flow is: when you access the application, it will forward you to /LoginForm.do, which is the URI that presents the form that will collect the data which JGuard will use in order to solve how's trying to get authenticated and, if that was successful or failed.

<<PICTURE OF THE FLOW HERE?>>

In order to have JGuard working, we just need to configure five xml files and add just basic context information on our web.xml file.

This file is the **src/main/webapp/WEB-INF/conf/jGuard/jGuardFilter.xml**. Remember to copy the **jGuardFilter_1.00.dtd** file.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE configuration SYSTEM "jGuardFilter_1.00.dtd">
<configuration>
  <filter>
    <!--
      Index uri of your web application.
      This is the URI where JGuard will redirect the user case authentication succeed.
    -->
    <indexURI>/WEB-INF/jsp/authenticated.jsp</indexURI>
    <!-- Uri when the user authentication failed. -->
    <authenticationFailedURI>/AuthenticationFailed.do</authenticationFailedURI>
    <!--
      Uri to access to the authentication form.
      A Submit from this URI (logonURI) to logonProcessURI means to JGuard that
      it is an authentication cycle.
    -->
    <logonURI>/Logon.do</logonURI>
    <!-- uri to be authenticated. The action property of the authentication form MUST NOT
    be set to j_security_check. -->
    <logonProcessURI>/LogonProcess.do</logonProcessURI>
    <!-- uri related to register a user. not used by this example -->
    <registerURI></registerURI>
    <registerProcessURI></registerProcessURI>
    <!-- uri to to be unauthenticated -->
    <logoffURI>/LogOff.do</logoffURI>
    <!-- uri when access to a resource is denied, ie. user doesn't have access -->
    <accessDeniedURI>/AccessDenied.do</accessDeniedURI>
    <authScheme>FORM</authScheme>
    <!-- these names must be the same of the LogonURI's login and password fields. Remember
    the names in bold? -->
    <loginField>login</loginField>
    <passwordField>password</passwordField>
    <goToLastAccessDeniedUriOnSuccess>true</goToLastAccessDeniedUriOnSuccess>
  </filter>
</configuration>
```

This file is the **src/main/webapp/WEB-INF/conf/jGuard/jGuardAuthentication.xml**. Remember to copy the **jGuardAuthentication_1.00.dtd** file.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE configuration SYSTEM "jGuardAuthentication_1.00.dtd">
```

```

<configuration>
  <authentication>
    <!-- 'local' or 'jvm' -->
    <scope>local</scope>
    <!-- boolean option('true' or 'false'), to activate the authorization debug mode -->
    <debug>true</debug>
    <includeOldConfig>>false</includeOldConfig>
    <!-- java.security.auth.login.config -->
    <includeConfigFromJavaParam>>false</includeConfigFromJavaParam>
    <includePolicyFromJavaParam>>false</includePolicyFromJavaParam>
    <!-- <digestAlgorithm>MD5</digestAlgorithm> -->
    <!-- <salt>qsd846sdq6ds4</salt> -->
    <authenticationManager>
net.sf.jguard.ext.authentication.manager.XmlAuthenticationManager
</authenticationManager>
    <authenticationManagerOptions>
    <option>
      <name>authenticationXmlFileLocation</name>
      <value>WEB-INF/conf/jGuard/jGuardUsersPrincipals.xml</value>
    </option>
  </authenticationManagerOptions>
  <loginModules>
    <!-- specify which loginModules are used for authentication. -->
    <loginModule>
      <name>net.sf.jguard.ext.authentication.loginmodules.XmlLoginModule</name>
      <!-- flag : 'REQUIRED', 'OPTIONAL', 'REQUISITE' or 'SUFFICIENT' -->
      <flag>REQUIRED</flag>
      <loginModuleOptions>
        <option>
          <name>debug</name>
          <value>>false</value>
        </option>
      </loginModuleOptions>
    </loginModule>
  </loginModules>
</authentication>
</configuration>

```

This file is the `src/main/webapp/WEB-INF/conf/jGuard/jGuardAuthorization.xml`. Remember to copy the `jGuardAuthorization_1.00.dtd` file.

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE configuration SYSTEM "jGuardAuthorization_1.00.dtd">
<configuration>
  <authorization>
    <!-- 'local' or 'jvm' -->
    <scope>local</scope>
    <permissionResolutionCaching>true</permissionResolutionCaching>
    <authorizationManager>
net.sf.jguard.ext.authorization.manager.XmlAuthorizationManager
</authorizationManager>
    <authorizationManagerOptions>
    <option>
      <name>authorizationXmlFileLocation</name>
      <value>WEB-INF/conf/jGuard/jGuardPrincipalsPermissions.xml</value>
    </option>
    <option>
      <name>debug</name>
      <value>true</value>
    </option>
  </authorizationManagerOptions>
</authorization>
</configuration>

```

Now that we've defined the flow (controlled by JGuardFilter.xml) and the way authentication and authorization will happen, we have to define the roles (principals) and permissions related to these principals.

This file is the `src/main/webapp/WEB-INF/conf/jGuard/jGuardUsersPrincipals.xml`. Remember to copy the `jGuardUsersPrincipals_1.00.dtd` file.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE usersPrincipals SYSTEM "jGuardUsersPrincipals_1.00.dtd">
<usersPrincipals>
  <principals>
    <principal>
      <name>admin</name>
      <class>net.sf.jguard.core.principals.RolePrincipal</class>
      <applicationName>Basic1</applicationName> <!-- this value must be exactly the same as
web.xml's display-name -->
    </principal>
    <principal>
      <name>guest</name>
      <class>net.sf.jguard.core.principals.RolePrincipal</class>
      <applicationName>Basic1</applicationName>
    </principal>
  </principals>

  <users>
    <userTemplate>
      <name>default</name>
      <privateRequiredCredentials>
        <credTemplateId identity="true">login</credTemplateId>
        <credTemplateId digestNeeded="true">password</credTemplateId>
      </privateRequiredCredentials>
      <publicRequiredCredentials>
        <credTemplateId>firstname</credTemplateId>
        <credTemplateId>lastname</credTemplateId>
        <credTemplateId>location</credTemplateId>
      </publicRequiredCredentials>
      <privateOptionalCredentials>
        <credTemplateId>country</credTemplateId>
        <credTemplateId>religion</credTemplateId>
      </privateOptionalCredentials>
      <publicOptionalCredentials>
        <credTemplateId>hobbies</credTemplateId>
      </publicOptionalCredentials>
      <genericPrincipals>
        <principalRef name="admin" applicationName="Basic1"/>
      </genericPrincipals>
      <specificPrincipalFactories/>
    </userTemplate>
    <user>
      <privateCredentials>
        <credential>
          <id>login</id>
          <value>admin</value>
        </credential>
        <credential>
          <id>password</id>
          <value>admin</value>
        </credential>
      </privateCredentials>
      <publicCredentials>
        <credential>
          <id>firstname</id>
          <value>Rick</value>
        </credential>
      </publicCredentials>
    </user>
  </users>
</usersPrincipals>
```

```

<credential>
  <id>lastname</id>
  <value>Dangerous</value>
</credential>
<credential>
  <id>location</id>
  <value>Paris</value>
</credential>
</publicCredentials>
<principalsRef>
  <principalRef name="admin" applicationName="Basic1"
definition="{subject.publicCredentials.location.contains('Paris')}}" active="true"/>
</principalsRef>
</user>
<user>
  <privateCredentials>
    <credential>
      <id>login</id>
      <value>guest</value>
    </credential>
    <credential>
      <id>password</id>
      <value>guest</value>
    </credential>
  </privateCredentials>
  <publicCredentials/>
  <principalsRef>
    <principalRef name="guest" applicationName="jguard-struts-example" />
  </principalsRef>
</user>
</users>
</usersPrincipals>

```

This file is the **src/main/webapp/WEB-INF/conf/jGuardPrincipalsPermissions.xml**. Remember to copy the **jGuardPrincipalsPermissions_1.00.dtd** file.

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE configuration SYSTEM "jGuardPrincipalsPermissions_1.00.dtd">
<configuration>
  <permissions>
    <domain>
      <!--
        this domain will hold all permissions that everybody will be able to access.
      -->
      <name>public</name>
      <permission>
        <name>LoginForm</name>
        <class>net.sf.jguard.core.authorization.permissions.URLPermission</class>
        <actions>
          <action>/LoginForm.do</action>
          <action>ANY</action>
        </actions>
      </permission>

      <permission>
        <name>LogonProcess</name>
        <class>net.sf.jguard.core.authorization.permissions.URLPermission</class>
        <actions>
          <action>/LogonProcess.do</action>
          <action>ANY</action>
        </actions>
      </permission>
      <permission>
        <name>LogOff</name>
        <class>net.sf.jguard.core.authorization.permissions.URLPermission</class>

```

```

<actions>
  <action>/LogOff.do</action>
  <action>ANY</action>
</actions>
</permission>
</domain>

<domain>
<!--
  this domain will hold all permissions that only authenticated people
  having a related role will be able to access
-->
<name>secured</name>
<permission>
  <name>Authenticated</name>
  <class>net.sf.jguard.core.authorization.permissions.URLPermission</class>
  <actions>
    <action>/Authenticated.do</action>
    <action>ANY</action>
    <action>you have been authenticated</action>
  </actions>
</permission>
</domain>
</permissions>

<principals>
<principal>
  <name>guest</name>
  <class>net.sf.jguard.core.principals.RolePrincipal</class>
  <permissionsRef>
    <permissionRef name="LoginForm" />
    <permissionRef name="LogonProcess" />
    <permissionRef name="LogOff" />
  </permissionsRef>
</principal>
<principal>
  <name>admin</name>
  <class>net.sf.jguard.core.principals.RolePrincipal</class>
  <permissionsRef>
    <domainRef name="secured" />
  </permissionsRef>
</principal>
</principals>

</configuration>

```

Now we are going to reference them at web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4"
  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">

  <display-name>Basic1</display-name>
  <description>Provides a very basic webapp secured with jGuard</description>
  <!-- JGuard basic configuration -->

  <context-param>
    <param-name>authenticationConfigurationLocation</param-name>
    <param-value>/WEB-INF/conf/jGuard/jGuardAuthentication.xml</param-value>
  </context-param>
  <context-param>
    <param-name>authenticationScope</param-name>
    <param-value>local</param-value>

```

```
</context-param>
<context-param>
  <param-name>authorizationConfigurationLocation</param-name>
  <param-value>/WEB-INF/conf/jGuard/jGuardAuthorization.xml</param-value>
</context-param>
<context-param>
  <param-name>authorizationScope</param-name>
  <param-value>local</param-value>
</context-param>
<context-param>
  <param-name>enableJMX</param-name>
  <param-value>>false</param-value>
</context-param>

<filter>
  <filter-name>AccessFilter</filter-name>
  <filter-class>
    net.sf.jguard.jee.authentication.http.AccessFilter
  </filter-class>
  <init-param>
    <param-name>configurationLocation</param-name>
    <param-value>/WEB-INF/conf/jGuard/jGuardFilter.xml</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>AccessFilter</filter-name>
  <url-pattern>*.do</url-pattern>
</filter-mapping>
<listener>
  <listener-class>net.sf.jguard.jee.listeners.SessionListener</listener-class>
</listener>
<listener>
  <listener-class>net.sf.jguard.jee.listeners.ContextListener</listener-class>
</listener>
<!-- end of JGuard basic configuration -->

<session-config>
  <session-timeout>30</session-timeout>
</session-config>

<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/conf/struts/struts-config.xml</param-value>
  </init-param>
  <init-param>
    <param-name>locale</param-name>
    <param-value>>false</param-value>
  </init-param>
  <init-param>
    <param-name>debug</param-name>
    <param-value>2</param-value>
  </init-param>
  <init-param>
    <param-name>detail</param-name>
    <param-value>2</param-value>
  </init-param>
  <init-param>
    <param-name>convertNull</param-name>
    <param-value>>true</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
```

```

<servlet-name>action</servlet-name>
<url-pattern>*.do</url-pattern>
</servlet-mapping>

<welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
</web-app>

```

Now, we need to create the two remaining forwards, one which is related to authenticationFailedURI and accessDeniedURI.

```

<% /*
 * accessDenied.jsp
 * This page will show the Access Denied message.
 */
%>

<%@ taglib prefix="html" uri="http://struts.apache.org/tags-html" %>

Access Denied. You have not clearance to access the specified resource.

```

Now we need to add the forwards on the struts-config.xml file:

```

<!DOCTYPE struts-config
PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
"http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <form-beans>
    <form-bean name="loginForm" type="org.apache.struts.action.DynaActionForm">
      <form-property name="login" type="java.lang.String" />
      <form-property name="password" type="java.lang.String" />
    </form-bean>
  </form-beans>

  <action-mappings>
    <!-- errors -->
    <action path="/AccessDenied"
      type="org.apache.struts.actions.ForwardAction"
      parameter="/WEB-INF/jsp/accessDenied.jsp" />
    <action path="/AuthenticationFailed" name="loginForm"
      type="org.apache.struts.actions.ForwardAction"
      parameter="/LoginForm.do" />

    <action path="/LoginForm"
      type="org.apache.struts.actions.ForwardAction"
      parameter="/WEB-INF/jsp/loginForm.jsp" />
    <action path="/LogonProcess" name="loginForm"
      type="org.apache.struts.actions.ForwardAction"
      parameter="/Authenticated.do" />
    <action path="/Authenticated"
      type="org.apache.struts.actions.ForwardAction"
      parameter="/WEB-INF/jsp/authenticated.jsp" />
    <action path="/LogOff"
      type="org.apache.struts.actions.ForwardAction"
      parameter="/index.jsp" />
  </action-mappings>
</struts-config>

```

In order to deploy in tomcat, all you have to do is to copy the target/jguard-example-basic1-1.0.2.war file to tomcat's webapp dir and start tomcat (if it is not already started).