



Declarative Web Application Security

Core Servlets & JSP book: www.coreservlets.com
More Servlets & JSP book: www.moreservlets.com
Servlet and JSP Training Courses: courses.coreservlets.com

Slides © Marty Hall, <http://www.moreservlets.com>, book © Sun Microsystems Press

Agenda

- Major security concerns
- Declarative vs. programmatic security
- Using form-based authentication
 - Steps
 - Example
- Using BASIC authentication
 - Steps
 - Example

Major Issues

- **Preventing unauthorized users from accessing sensitive data.**
 - Access restriction
 - Identifying which resources need protection
 - Identifying who should have access to them
 - Authentication
 - Identifying users to determine if they are one of the authorized ones
- **Preventing attackers from stealing network data while it is in transit.**
 - Encryption (usually with SSL)

Declarative Security

- **None of the individual servlets or JSP pages need any security-aware code.**
 - Instead, both of the major security aspects are handled by the server.
- **To prevent unauthorized access**
 - Use the Web application deployment descriptor (*web.xml*) to declare that certain URLs need protection.
 - Designate authentication method that server uses to identify users.
 - At request time, the server automatically prompts users for usernames and passwords when they try to access restricted resources, automatically checks the results against a server-specific set of usernames and passwords, and automatically keeps track of which users have previously been authenticated. **This process is completely transparent to the servlets and JSP pages.**
- **To safeguard network data**
 - Use the deployment descriptor to stipulate that certain URLs should be accessible only with SSL. If users try to use a regular HTTP connection to access one of these URLs, the server automatically redirects them to the HTTPS (SSL) equivalent.

Programmatic Security

- **Protected servlets and JSP pages at least partially manage their own security.**
 - *Much* more work, but totally portable.
 - No server-specific piece. Also no web.xml entries needed and a bit more flexibility is possible.
- **To prevent unauthorized access**
 - Each servlet or JSP page must either authenticate the user or verify that the user has been authenticated previously.
- **To safeguard network data**
 - Each servlet or JSP page has to check the network protocol used to access it.
 - If users try to use a regular HTTP connection to access one of these URLs, the servlet or JSP page must manually redirect them to the HTTPS (SSL) equivalent.

6

Declarative Web Application Security

www.moreservlets.com

Form-Based Authentication (Declarative Security)

- **1) Set up usernames, passwords, and roles.**
 - Designate a list of users and associated passwords and abstract role(s) such as normal user or administrator.
 - **This is a completely server-specific process.**
 - Simplest Tomcat approach: use *install_dir/conf/tomcat-users.xml*:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tomcat-users>
  <user name="john" password="nhoj"
        roles="registered-user" />
  <user name="jane" password="enaj"
        roles="registered-user" />
  <user name="juan" password="nauj"
        roles="administrator" />
  <user name="juana" password="anauj"
        roles="administrator,registered-user" />
</tomcat-users>
```

7

Declarative Web Application Security

www.moreservlets.com

Form-Based Authentication (Continued)

- **2) Tell server that you are using form-based authentication. Designate locations of login and login-failure page.**
 - Use the *web.xml* login-config element with auth-method of FORM and form-login-config with locations of pages.

```
<web-app> ...  
  <login-config>  
    <auth-method>FORM</auth-method>  
    <form-login-config>  
      <form-login-page>/login.jsp</form-login-page>  
      <form-error-page>/login-error.html</form-error-page>  
    </form-login-config>  
  </login-config>  
...</web-app>
```

Form-Based Authentication (Continued)

- **3) Create a login page (HTML or JSP)**
 - HTML form with ACTION of j_security_check, METHOD of POST, textfield named j_username, and password field named j_password.

```
<FORM ACTION="j_security_check" METHOD="POST">  
  ...  
  <INPUT TYPE="TEXT" NAME="j_username">  
  ...  
  <INPUT TYPE="PASSWORD" NAME="j_password">  
  ...  
</FORM>
```

- For the username, you can use a list box, combo box, or set of radio buttons instead of a textfield.

Form-Based Authentication (Continued)

- **4) Create page for failed login attempts.**
 - No specific content is mandated.
 - Perhaps just “username and password not found” and give a link back to the login page.
 - This can be either an HTML or a JSP document.

Form-Based Authentication (Continued)

- **5) Specify URLs to be password protected.**
 - Use security-constraint element of *web.xml*. This element uses web-resource-collection and auth-constraint subelements. The first (web-resource-collection) designates URL patterns to which access should be restricted; the second (auth-constraint) specifies abstract roles that should have access to resources at the given URLs.

```
<web-app>...
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>Sensitive</web-resource-name>
      <url-pattern>/sensitive/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>administrator</role-name>
      <role-name>executive</role-name>
    </auth-constraint>
  </security-constraint>
  <login-config>...</login-config>...
</web-app>
```


Form-Based Authentication (Continued)

- **6) Specify which URLs require SSL.**

- If server supports SSL, you can stipulate that certain resources are available only through encrypted HTTPS (SSL) connections. Use the `user-data-constraint` subelement of `security-constraint`. Only full J2EE servers are *required* to support SSL.

```
<security-constraint>
...
<user-data-constraint>
  <transport-guarantee>
    CONFIDENTIAL
  </transport-guarantee>
</user-data-constraint>
</security-constraint>
```

Form-Based Authentication (Continued)

- **7) Turn off the invoker servlet.**

- You protect certain URLs that are associated with registered servlet or JSP names. The `http://host/prefix/servlet/Name` format of default servlet URLs will probably not match the pattern. Thus, the security restrictions are bypassed when the default URLs are used.
- Disabling it
 - In each Web application, redirect requests to other servlet by normal `web.xml` method

```
<url-pattern>/servlet/*</url-pattern>
```
 - Globally
 - Server-specific mechanism (e.g. `install_dir/conf/server.xml` for Tomcat).

Example: Form-Based Security



14

Declarative Web Application Security

Example: Step 1

- Set up usernames, passwords, and roles.

– *install_dir/conf/tomcat-users.xml*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tomcat-users>
  <user name="john" password="nhoj"
        roles="registered-user" />
  <user name="jane" password="enaj"
        roles="registered-user" />
  <user name="juan" password="nauj"
        roles="administrator" />
  <user name="juana" password="anauj"
        roles="administrator,registered-user" />
  <!-- ... -->
</tomcat-users>
```

15

Declarative Web Application Security

www.moreservlets.com

Example: Step 2

- Tell server that you are using form-based authentication. Designate locations of login and login-failure page.

```
<login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
    <form-login-page>
      /admin/login.jsp
    </form-login-page>
    <form-error-page>
      /admin/login-error.jsp
    </form-error-page>
  </form-login-config>
</login-config>
```

Example: Step 3

- Create a login page

```
...
<BODY>
<TABLE BORDER=5 ALIGN="CENTER">
  <TR><TH CLASS="TITLE">Log In</TH></TR>
<P>
<H3>Sorry, you must log in before
accessing this resource.</H3>
<FORM ACTION="j_security_check" METHOD="POST">
<TABLE>
<TR><TD>User name:
  <INPUT TYPE="TEXT" NAME="j_username">
<TR><TD>Password:
  <INPUT TYPE="PASSWORD" NAME="j_password">
<TR><TH><INPUT TYPE="SUBMIT" VALUE="Log In">
</TR></TABLE>
</FORM></BODY></HTML>
```

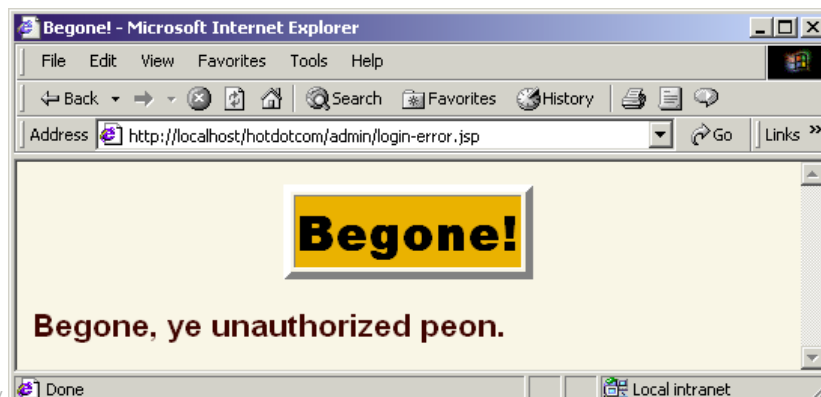

Example: Step 3 (Result)



Example: Step 4

- Create page for failed login attempts.

```
...  
<BODY>  
<TABLE BORDER=5 ALIGN="CENTER">  
  <TR><TH CLASS="TITLE">Begone!</TH></TR></TABLE>  
  
<H3>Begone, ye unauthorized peon.</H3>  
  
</BODY>  
</HTML>
```



Example: Access Rules

- **Home page**
 - Anyone
- **Investing page**
 - Registered users
 - Administrators
- **Stock purchase page**
 - Registered users
 - Via SSL only
- **Delete account page**
 - Administrators

Example: Step 5

- **Specify URLs to be password protected.**

```
<!-- Protect everything within
      the "investing" directory. -->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Investing
    </web-resource-name>
    <url-pattern>/investing/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>registered-user</role-name>
    <role-name>administrator</role-name>
  </auth-constraint>
</security-constraint>
```

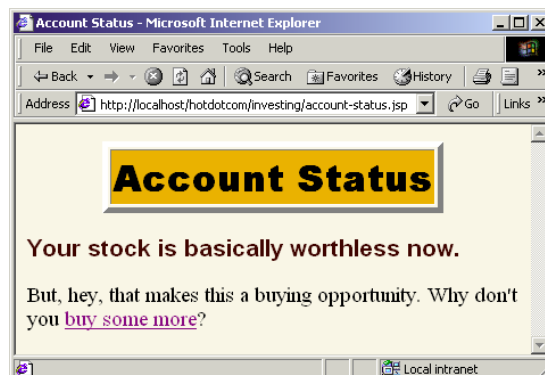
Example: Step 5 (Continued)

```
<!-- Only users in the administrator role
      can access the delete-account.jsp page
      within the admin directory. -->

<security-constraint>
  <web-resource-collection>
    <web-resource-name>Account Deletion
  </web-resource-name>
  <url-pattern>/admin/delete-account.jsp
</url-pattern>
</web-resource-collection>
  <auth-constraint>
    <role-name>administrator</role-name>
  </auth-constraint>
</security-constraint>
```

Example: Step 5 (Results)

- First attempt to access account status page
- Result of successful login and later attempts to access account status page



Example: Step 6

- Specify which URLs require SSL.

```
<!-- URLs of the form
      http://host/webAppPrefix/ssl/blah
      require SSL and are thus redirected to
      https://host/webAppPrefix/ssl/blah. -->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Purchase
    </web-resource-name>
    <url-pattern>/ssl/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>registered-user</role-name>
  </auth-constraint>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL
    </transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

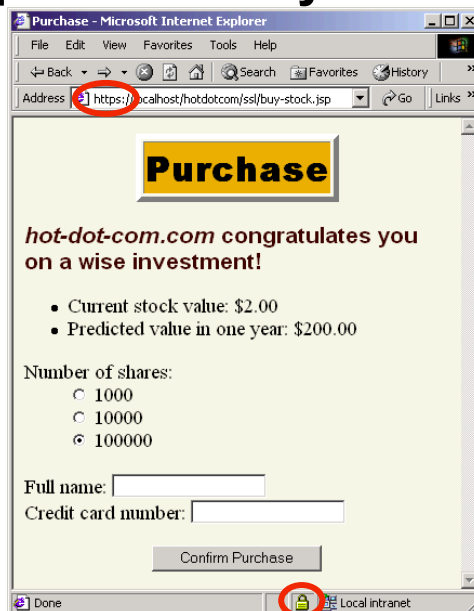
24

Declarative Web Application Security

www.moreservlets.com

Example: Step 6 (Results)

- <http://host/prefix/ssl/buy-stock.jsp> *or*
<https://host/prefix/ssl/buy-stock.jsp>



25

Declarative Web Application Security

www.moreservlets.com

Example: Step 7

- Turn off the invoker servlet

```
<!-- Turn off invoker. Send requests
      to index.jsp. -->

<servlet-mapping>
  <servlet-name>Redirector</servlet-name>
  <url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
...
<welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>index.html</welcome-file>
</welcome-file-list>
```

Example: Step 7 (Continued)

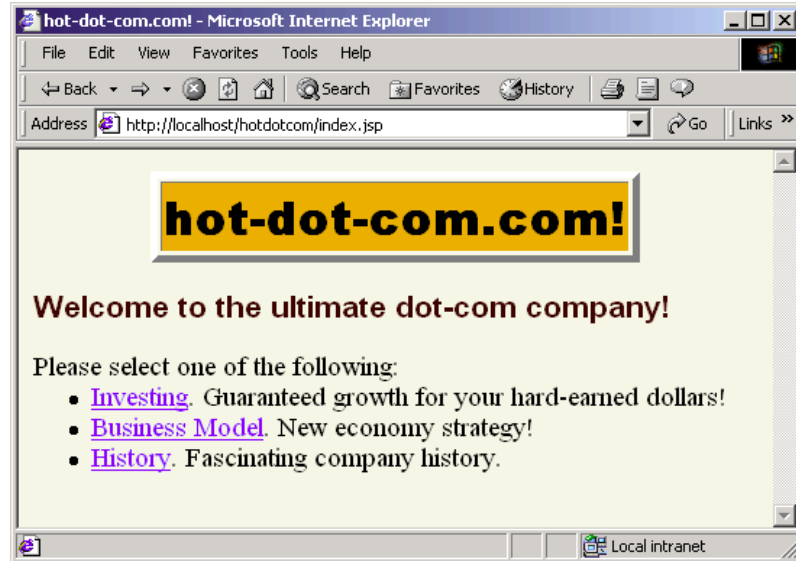
```
/** Servlet that simply redirects users to the
 *  Web application home page.
 */

public class RedirectorServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        response.sendRedirect(request.getContextPath());
    }

    public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
        throws ServletException, IOException {
        doGet(request, response);
    }
}
```


Example: Step 7 (Results)

- **Attempt to access**
`http://host/hotdotcom/servlet/Anything`



Form-Based vs. BASIC Authentication

- **Advantages of form-based**
 - Consistent look and feel
 - Fits model users expect from ecommerce sites
- **Disadvantage of form-based**
 - Can fail if server is using URL rewriting for session tracking. Can fail if browser has cookies disabled.
- **Advantages of BASIC**
 - Doesn't rely on session tracking
 - Easier when you are doing it yourself (programmatic security)
- **Disadvantage of BASIC**
 - Small popup dialog box seems less familiar to most users

BASIC Authentication

- 1. Set up usernames, passwords, and roles.**
 - Same as for form-based authentication. Server-specific.
- 2. Tell the server that you are using BASIC authentication. Designate the realm name.**
 - Use the *web.xml* `login-config` element with an `auth-method` subelement of BASIC and a `realm-name` subelement (generally used as part of the title of the dialog box that the browser opens).

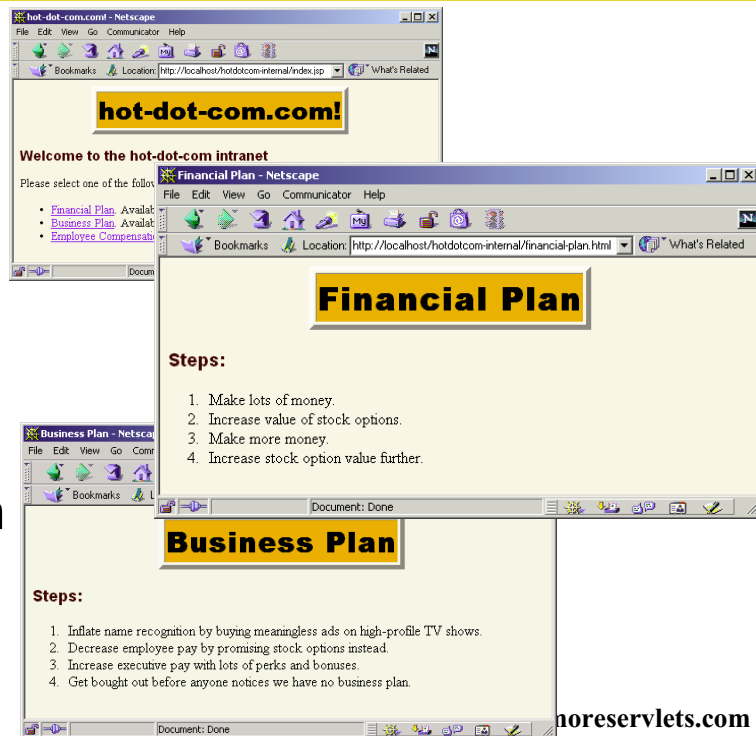
```
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Some Name</realm-name>
</login-config>
```

BASIC Authentication (Continued)

- 3. Specify which URLs should be password protected.**
 - Same as with form-based authentication.
- 4. Specify which URLs should be available only with SSL.**
 - Same as with form-based authentication.
- 5. Turn off the invoker servlet.**
 - Same as with form-based authentication.

Example: BASIC Authentication

- Home page
 - Anyone
- Financial plan
 - Employees or executives
- Business plan
 - Executives only



32

Declarative Web Application Security

Example: BASIC Authentication (Step 1)

- Set up usernames, passwords, and roles.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tomcat-users>

...
<user name="gates" password="llib"
      roles="employee" />
<user name="ellison" password="yrral"
      roles="employee" />
<user name="mcnealy" password="ttocs"
      roles="executive" />
</tomcat-users>
```
- Note: file that contains these passwords and those of declarative example is online at <http://archive.moreservlets.com/Security-Code/tomcat-users.xml>

33

Declarative Web Application Security

www.moreservlets.com

Example: BASIC Authentication (Step 2)

- Tell the server that you are using BASIC authentication. Designate the realm name.

```
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Intranet</realm-name>
</login-config>
```

Example: BASIC Authentication (Step 3)

- Specify which URLs should be password protected.

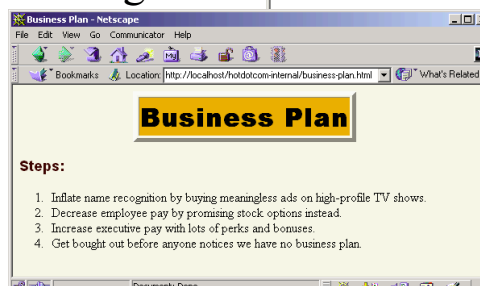
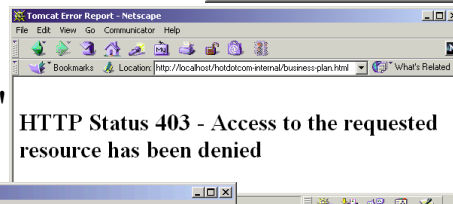
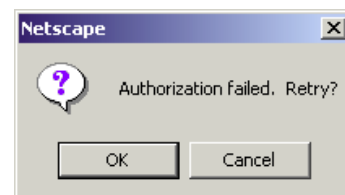
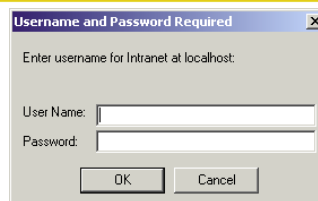
```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>
      Financial Plan
    </web-resource-name>
    <url-pattern>
      /financial-plan.html
    </url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>employee</role-name>
    <role-name>executive</role-name>
  </auth-constraint>
</security-constraint>
```

Example: BASIC Authentication (Step 3 Continued)

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>
      Business Plan
    </web-resource-name>
    <url-pattern>
      /business-plan.html
    </url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>executive</role-name>
  </auth-constraint>
</security-constraint>
```

Example: BASIC Authentication (Results)

- **First attempt**
 - For business plan
- **Failed login**
 - User not found or user not in executive role
- **Denied**
 - Result of selecting "Cancel" from previous dialog
- **Success**



Summary

- **Main security issues**
 - Preventing access by unauthorized users
 - Preventing attackers from stealing network data
- **Declarative security**
 - Much less work than programmatic security
 - Requires server-specific password setup
- **Form-based authentication**
 - Attempts to access restricted resources get redirected to login page. HTML form gathers username and password. Session tracking tracks authenticated users.
- **BASIC authentication**
 - Attempts to access restricted resources results in dialog box. Dialog gathers username and password. HTTP headers track authenticated users.



Questions?

Core Servlets & JSP book: www.coreservlets.com
More Servlets & JSP book: www.moreservlets.com
Servlet and JSP Training Courses: courses.coreservlets.com

More Information

- **Source code for examples**
 - <http://www.moreservlets.com>
- **More Servlets & JSP**
 - <http://www.moreservlets.com>
 - Site includes info on servlet and JSP training courses
- **Core Servlets & JSP**
 - Prequel to *More Servlets & JSP*
 - <http://www.coreservlets.com>
- **Servlet home page**
 - <http://java.sun.com/products/servlet/>
- **JavaServer Pages home page**
 - <http://java.sun.com/products/jsp/>

