《JavaEE 实验手册》

Java 教研室

版 本 1.3

文 档 提 供：Java 教 研 室　孙 丽 萍

修 改 记 录

修改时间	修改人	修改内容
2012.8.16	王伟	文档创建
2016.3.7	孙丽萍	修改

目录

安全(security)

一、内容概述

本章的教学内容介绍了应用与 Web 的程序的安全相关技术，基于 Servlet 规范，比较和对比以下安全机制：认证、授权、数据完整以及保密性。

在部署描述符中，声明安全性约束、Web 资源、传输保障、登陆配置和安全角色。

比较和对比认证类型：BASIC，DIGEST，FORM 以及 CLIENT-CERT。这些类型如何工作，给定一个情景，选择合适的类型。

二、实验一 BASIC 认证

2.1 实验目的

掌握基于 Tomcat6 服务器部署与实现 HTTP 基本认证机制的应用。

2.2 准备

保证 Eclipse 开发环境的正确配置。

2.3 实验步骤

步骤一：

在 Eclipse 中新建 Dynamic project “14_authenbase”，在该项目下新建 index.jsp 页面做为首页面，从该页面连接到有权限机制的页面，关键代码如下：

```
<body>
<a href="base.jsp">BASE</a>
<br />
</body>
```

步骤二：

从首页面可以连接到 base.jsp 页面，base.jsp 页面是受安全保护的页面，如访问需要输入用户名和密码，base.jsp 页面只包含一条输出信息：

```
<body>
这个页面使用HTTP基本验证机制进行访问。
</body>
```

步骤三：

有了测试页面，我们需要修改 web.xml 文件对认证机制进行定制，在 web.xml

文件中增加以下节点元素内容：

```
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>site14_authenbase</realm-name>
</login-config>
```

说明：使用 HTTP 基本认证，访问域为 site14_authenbase。

步骤四：

继续修改 web.xml 进行安全声明，在 web.xml 文件中增加以下节点元素：

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>basejsp</web-resource-name>
        <url-pattern>/base.jsp</url-pattern>
        <http-method>POST</http-method>
        <http-method>GET</http-method>
    </web-resource-collection>
    <auth-constraint>
        <role-name>tomcat</role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>NONE</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

说明：安全声明部分配置安全需求的细节，指明对哪些资源加以保护及限制对资源访问的相关信息。在这里指明 base.jsp 是受到保护的，对该页面的 POST 请求和 GET 请求都需要拥有“tomcat”角色权限的用户名和密码。用户数据不进行加密处理。

<auth-constraint>元素的子元素<role-name>指定了访问资源的角色，其值可以是*代表所有定义在 Web 应用程序中的角色，否则，其值必须是定义在 web.xml 中<security-role>元素中角色名称。

步骤五：

定义 Web 应用程序的角色，在 web.xml 文件中增加以下节点：

```
<security-role>
    <description></description>
    <role-name>tomcat</role-name>
</security-role>
```

说明：该应用中有一个“tomcat”角色。

步骤六：

添加 Tomcat 服务器的用户名和密码并指定角色，因为我们使用 Eclipse 插件来管理与配置 Tomcat6.0 服务器，在 eclipse 的包浏览器透视图中找到“Servers”项目并展开，在对应的 Tomcat 服务器配置文件夹中找到 tomcat-users.xm 文件并打开，如*图 2-1* 所示：

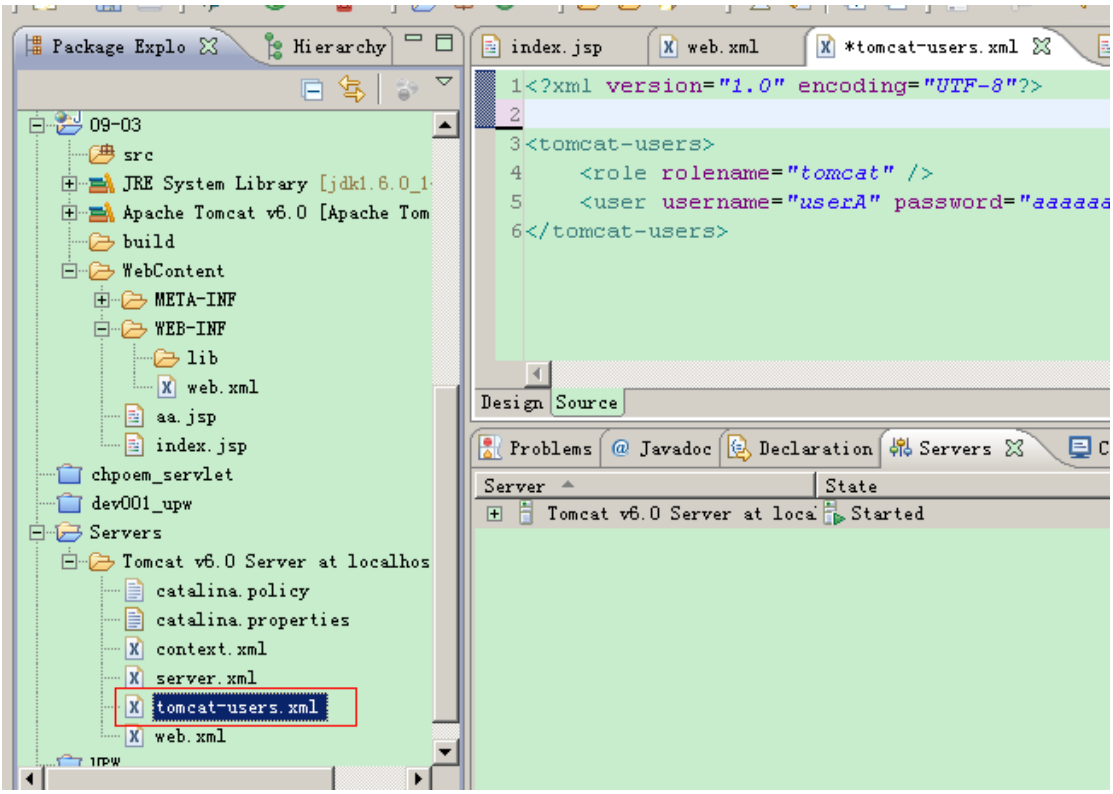

图 2-1

步骤七：

定义用户名和密码并分派角色，tomcat-users.xml 的内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>

<tomcat-users>
    <role rolename="tomcat" />
    <user username="userA" password="aaaaaa" roles="tomcat" />
</tomcat-users>
```

说明：指定一个角色，角色名为 tomcat，和 web.xml 中的定义相对应。定义一个用户“userA”(区分大小写)，密码为“aaaaaa”，并指定其角色为 tomcat。

步骤八：

运行项目 14_authenbase 进行测试，在首页面 index.jsp 中点击 base 连接请求 base.jsp 资源，因该资源被保护了，所以会弹出对话框要求输入用户名和密码，如*图 2-2* 所示：

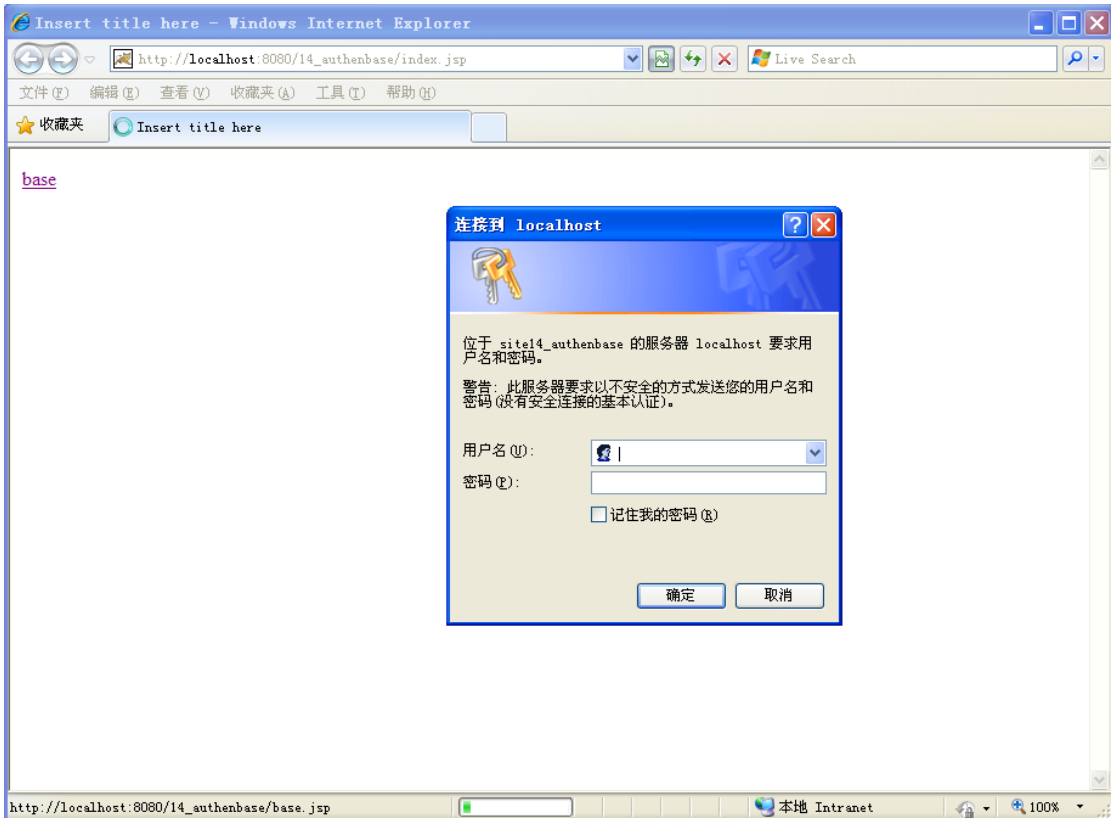

图 2-2

输入正确的用户名和密码可以将 base. jsp 资源响应给客户端，否则没有办法访问到 base. jsp。

2.4实验结论

通过以上实验要求学生掌握在 Tomcat 服务器环境下使用 HTTP 基本认证方式进行安全相关的设置。

三、实验二 FORM 认证

3.1 实验目的

掌握基于 Tomcat6 服务器部署与实现 FORM 认证机制的应用。

3.2 准备

完成实验一，在实验一的基础上进行。

3.3 实验步骤

步骤一：

修改 “14_authenbase”项目的 index.jsp 页面，从该页面连接到使用 FORM 权限机制的页面，关键代码如下：

```
<body>
<a href="form.jsp">Form机制</a>
<br />
</body>
```

步骤二：

从首页面可以连接到 form.jsp 页面，form.jsp 页面是受安全保护的页面，如访问需要输入用户名和密码，form.jsp 页面只包含一条输出信息：

```
<body>
这个页面使用FORM机制进行访问。
</body>
```

步骤三：

有了测试页面，我们需要修改 web.xml 文件对认证机制进行定制，修改<login-config>节点元素内容为以下内容：

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

说明：使用 FORM 认证机制，登录页面为 login.jsp，如果登录失败的错误页面为 error.jsp。

步骤四：

新创建 login.jsp 页面做为登录页面，该页面的关键代码如下：

```
<body>
<h2>HTTP表单认证的登录页面</h2>
<fieldset>
<legend>用户登录</legend>
<form action="j_security_check">
用户名：<input type="text" name="j_username" />
<br />
密  码：<input type="password" name="j_password" />
<br />
<input type="submit" name="登录" />
</form>
</fieldset>
</body>
```

说明：当用户访问受 FORM 认证机制保护的资源时(本例中该资源为 form.jsp 页面)要求输入正确的用户名和密码，与 HTTP 基本认证不同的是不使用浏览器弹出对话框，而是使用一个 HTML 的表单(本例中即：login.jsp 页面)来获取用户名和密码，并可以定制界面外观。但 Form 表单的 action 属性必须取值为 j_security_check,且提供两个文本输入域用于获取用户输入的用户名和密码。用户名文本域的 name 属性值为 j_username，密码文本域的 name 为 j_password。login.jsp 页面的创建除上面三点强制要求外，没有其他限制。

步骤五：

增加 error.jsp 页面，当登录失败时转到该页面，主要内容如下：

```
<body>
你没有得到授权，无法访问您所请求的资源。
</body>
```

说明：登录页面和错误页面我们都使用 JSP 实现，相比 HTML 静态页面我们可以使用 JSP 实现更多的动态信息，如错误原因等。

步骤六：

继续修改 web.xml 安全声明部分,将<security-constraint>节点元素更改为使用 FORM 认证，内容如下：

```
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>formjsp</web-resource-name>
            <url-pattern>/form.jsp</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>tomcat</role-name>
        </auth-constraint>
    </security-constraint>
```

说明：指明 form.jsp 资源是受到保护的，对该页面的任何请求都需要拥有“tomcat”角色权限的用户名和密码。

步骤七：

定义 Web 应用程序的角色及增加相应的用户名和密码，我们使用实验一中定义的角色和用户，对 web.xml 文件的<security-role>节点元素及 tomcat-users.xml 不做修改。

步骤八：

运行项目 14_authenbase 进行测试，在首页面 index.jsp 中点击“Form 机制”连接请求 form.jsp 资源，因该资源被保护了，所以转到 login.jsp 页面要求输入用户名和密码，如*图 3-1* 所示：

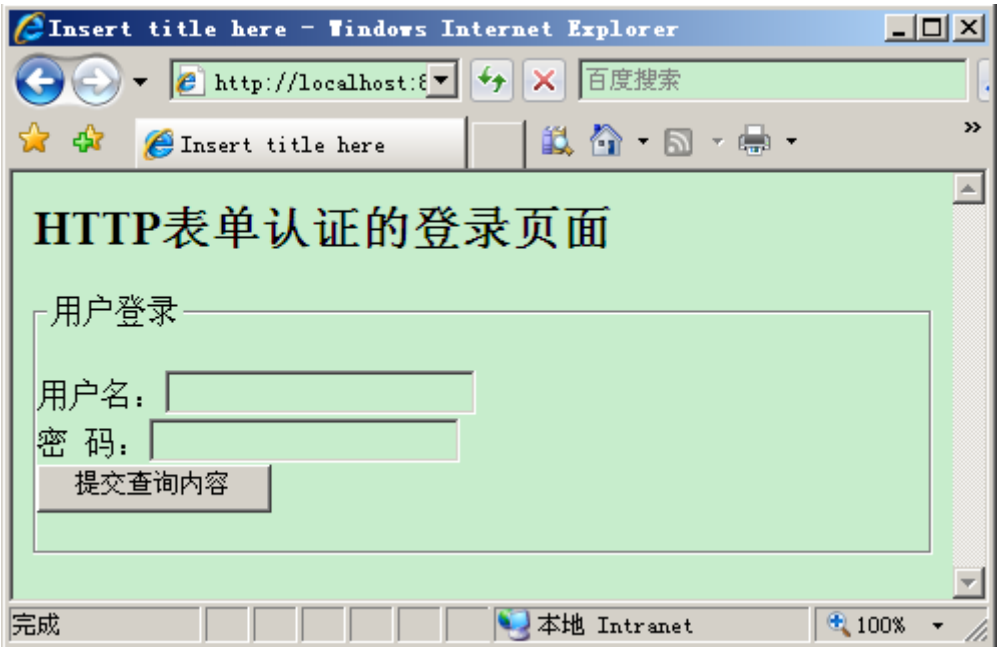

图 3-1

输入正确的用户名和密码可以成功访问 form.jsp 页面，否则如果输入的用户名和密码有误将转到 error.jsp 页面，无法访问 form.jsp。

3.4 实验结论

通过以上实验要求学生掌握在 Tomcat 服务器环境下使用 HTTP 表单认证方式进行安全相关的设置。

四、实验三 权限验证

4.1 实验目的

实现使用 Servlet 规范提供的 API 进行安全编程。

4.2 准备

完成实验二，在实验二的基础上进行实验。

4.3 实验步骤

步骤一：

修改 form.jsp 页面，根据登录用户的角色展示不同信息，代码如下：

```
<body>
这个页面使用HTTP基本验证机制进行访问。
<br />
<%
    String remoteUser = request.getRemoteUser();
    boolean isAdmin = request.isUserInRole("admin");
    request.setAttribute("remoteUser", remoteUser);
    request.setAttribute("isAdmin", isAdmin);
%>
欢迎你：${remoteUser}<br />
您拥有以下权限：<br />
${isAdmin?"管理，新增，修改，删除":"查询"}
</body>
```

说明：判断登录用户是否拥有“admin”权限，如果拥有，显示“管理，新增，修改，删除”。否则显示“查询”)。

运行项目并访问 form.jsp，输入用户名 userA 和密码 aaaaaa 能够访问 form.jsp，且其权限为“tomcat”，所以显示所拥有权限仅为“查询”。

步骤二：

增加新的用户 admin，密码也为 admin，且赋予 admin 角色。

修改 tomcat-users.xml 如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<tomcat-users>
    <role rolename="tomcat" />
    <user username="userA" password="aaaaaa" roles="tomcat" />
    <role rolename="admin" />
    <user username="admin" password="admin" roles="admin" />
</tomcat-users>
```

说明：增加一个角色 admin，并新增用户 admin，其密码也为 admin。

步骤三：

修改 web.xml 内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    id="WebApp_ID" version="2.5">
    <display-name>14_authenbase</display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
    <login-config>
        <auth-method>FORM</auth-method>
        <form-login-config>
            <form-login-page>/login.jsp</form-login-page>
            <form-error-page>/error.jsp</form-error-page>
        </form-login-config>
    </login-config>
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>formjsp</web-resource-name>
            <url-pattern>/form.jsp</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>tomcat</role-name>
            <role-name>admin</role-name>
        </auth-constraint>
    </security-constraint>
    <security-role>
        <role-name>tomcat</role-name>
    </security-role>
    <security-role>
        <role-name>admin</role-name>
    </security-role>
</web-app>
```

说明：注意新定义了角色名为 admin 的<security-role>元素，且<auth-constraint>元素中也新增了 admin 角色。

步骤四：

重新运行项目进行测试，访问 form.jsp，输入用户名 admin 和密码 admin，能够访问 form.jsp，且其权限为“admin”，所以显示所拥有权限为“管理，新增，修改，删除”。

4.4 实验结论

通过以上实验同学们应该能够掌握 HttpServletRequest 接口中识别用户和角色的方法的使用。