第9章

Web安全性入门

本章内容

- 9.1 Web安全性措施
- 9.2 安全域模型
- 9.3 定义安全约束
- 9.4 编程式的安全
- 9.5 小 结

9.1 Web安全性措施

- Web应用程序通常包含许多资源，这些资源可被多个用户访问，有些资源要求用户必须具有一定权限才能访问。可以通过多种措施来保护这些资源。

9.1.1　理解验证机制

- Web应用的安全性措施主要包括下面4个方面：
  1. 身份验证
  2. 授　权
  3. 数据完整性
  4. 数据保密性

9.1.2 验证的类型

- 在Servlet规范中定义了如下4种用户验证的机制：
  ① HTTP Basic 验证；
  ② HTTP Digest 验证；
  ③ HTTPS Client 验证；
  ④ HTTP FORM-based 验证。

1. HTTP Basic 验证

- 这种验证称为HTTP基本验证。当浏览器请求任何受保护资源时，服务器都要求一个用户名和口令。如果用户输入了合法的用户名/口令，服务器才发送资源。
- 优点：实现较容易，所有的浏览器都支持。
- 缺点：不能自定义对话框的外观。

2. HTTP Digest 验证

- 这种验证称为HTTP摘要验证，它除了口令是以加密的方式发送的，其他与基本验证都一样，但比基本验证安全。
- 缺点：它只能被IE 5以上版本支持；许多Servlet容器不支持，因为规范并没有强制要求。

3. FORM-based 验证

- 这种验证称为基于表单的验证，它类似于基本验证，但它使用用户自定义的表单来获得用户名和口令而不是使用浏览器的弹出对话框。
- 优点：所有的浏览器都支持，且很容易实现，客户可以定制登录页面的外观（Look And Feel）。
- 缺点：不安全。

4. HTTPS Client 验证

- 这种验证称为客户证书验证，它采HTTPS传输信息。
- 优点：它是4种验证类型中最安全的；所有常用的浏览器都支持这种验证。
- 缺点：它需要一个证书授权机构（如VeriSign）的证书；它的实现和维护的成本较高。

9.1.3 基本验证的过程

（1）浏览器向某个受保护资源（Servlet或JSP）发送请求，浏览器并不知道资源是受保护的，所以它发送的请求是一般的HTTP请求，例：GET /login.do HTTP/1.1

（2）当服务器接收到对资源的请求后，首先在访问控制列表（ACL）中查看该资源是否是受保护资源，如果不是，服务器将该资源发送给用户。如果发现该资源是受保护的，它并不直接发送该资源，而是向客户发送一个401 Unauthorized（非授权）消息。

9.1.3　基本验证的过程

（3）当浏览器收到上面响应，打开一个对话框提示输入用户名和密码。

（4）用户一旦输入了用户名和密码并单击【确定】按钮，浏览器再次发送请求并在名为Authorization的请求头中传递用户名和密码的值。

（5）当服务器接收到该请求，它将在访问控制列表中检验用户名和密码，如果是合法用户它将发送资源并在浏览器中显示出来，否则浏览器再一次显示对话框。

9.1.4 声明式安全与编程式安全

- 在Servlet规范中提到，实施Web应用程序的安全性可以有两种方法：声明式安全和编程式安全。
- 所谓声明式安全（declarative security）是一个应用程序的安全结构。包括角色、访问控制及验证需求都在应用程序外部表示。
- 编程式安全（programmatic security）主的有关方法实现。在9.4节将介绍编程式安全。

9.2 安全域模型

- 9.2.1 安全域概述
- 9.2.2 定义角色与用户

9.2.1　安全域概述

- 安全域是Web服务器保护Web资源的一种机制。所谓安全域（realm）是标识一个Web应用程序的合法的用户名和口令的“数据库”，其中包括与用户相关的角色。
- 角色的概念来自于现实世界，例如，一个公司可能只允许销售经理访问销售数据，而销售经理是谁没有关系。实际上，销售经理可能更换。任何时候，销售经理实际是一个充当销售经理角色的用户。

9.2.1　安全域概述

- 每个用户可以拥有一个或多个角色，每个角色限定了可访问的Web资源。在Web应用中，对资源的访问权限一般是分配给角色而不是实际的用户。把权限分配给角色而不是用户使对权限的改变更灵活。一个用户可以访问其拥有的所有角色对应的Web资源。

Tomcat的安全域类型

安全域类型	类名	说明
内存域	MemoryRealm	在Tomcat服务器初始化阶段，从一个XML文档中读取验证信息，并把它们以一组对象的形式存放在内存中
JDBC域	JDBCRealm	通过JDBC驱动程序访问存放在数据库中的验证信息
数据源域	DataSourceRealm	通过JNDI数据源访问存放在数据库中的验证信息
JNDI域	JNDIRealm	通过JNDI provider访问存放在基于LDAP的目录服务器中的安全验证信息
JAAS域	JAASRealm	通过JAAS（Java验证授权服务）框架访问验证信息

9.2.1　安全域概述

- 不管使用哪一种安全域模型，都要包含下列步骤。

  （1）定义角色、用户以及用户与角色的映射。

  （2）为Web资源设置安全约束。

9.2.2 定义角色与用户

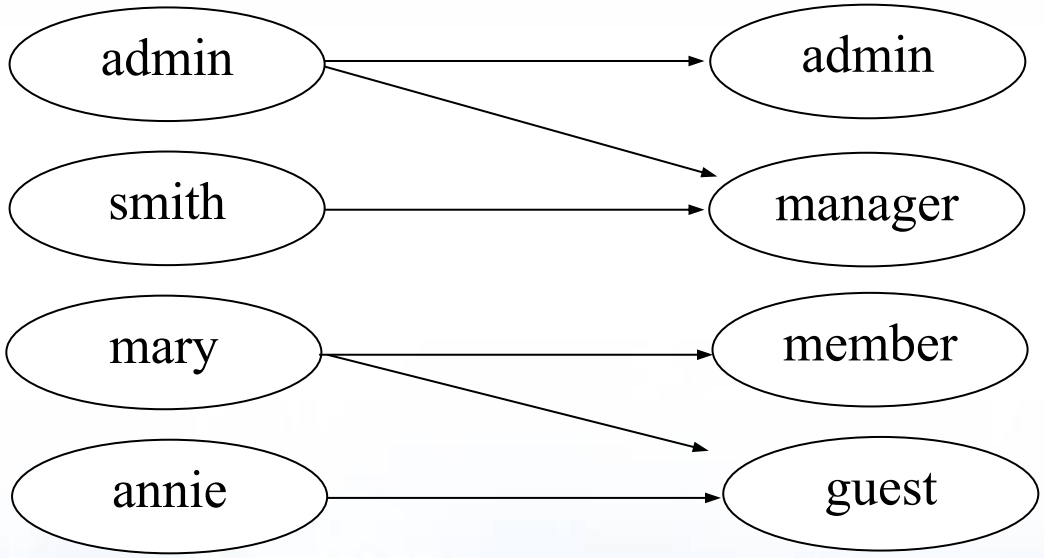
- 使用的安全域模型不同，用户和角色的定义也不同。
- 内存域安全模型通过org.apache.catalina.realm.MemoryRealm类实现的。它将用户和角色信息存储在一个XML文件中，当应用程序启动时将其读入内存中。这默认情况下，该XML文件为

<tomcat-install>\conf\tomcat-users.xml。

程序9.1 tomcat-users.xml

9.2.2 定义角色与用户

- 这些用户与角色的映射关系如图9-2所示。



9.3 定义安全约束

- 为Web资源定义安全约束是通过web.xml文件实现的，这里主要配置哪些角色可以访问哪些资源。
- 安全约束的配置主要是通过三个元素实现:

  <security-constraint>

  <login-config>

  <security-role>

9.3.1　安全约束定义

- 1. <security-constraint>元素
- 2. <login-config>元素
- 3. <security-role>元素

1. <security-constraint>元素

- 该元素用来定义受保护的Web资源集合、访问资源的角色以及用户数据的约束。该元素的DTD格式如下。

<!ELEMENT security-constraint (display-name?,
web-resource-collection+,
auth-constraint?, user-data-constraint?)>

<web-resource-collection>元素

- 该元素定义一个或多个Web资源集合，它的DTD定义:

```
<!ELEMENT web-resource-collection
(description?,web-resource-name,
        url-pattern*, http-method*)>
```

- web-resource-name指定资源的名称。url-pattern指定受保护的资源。http-method元素指定该约束适用的HTTP方法。

- <web-resource-collection>
- <web-resource-name>admin resource</web-resource-name>
- <url-pattern>/examination.do</url-pattern>
- <url-pattern>/admin/*</url-pattern>
- <http-method>GET</http-method>
- </web-resource-collection>

<auth-constraint>元素

- 该元素指定可以访问受限资源的角色。它的DTD定义：

  <!ELEMENT auth-constraint (description?, role-name*)>

- role-name元素指定可以访问受限资源的角色。它可以是*（表示Web应用程序中定义的所有角色），或者是<security-role>元素中定义的名称。

- <auth-constraint>
- <description>accessible to all manager</description>
- <role-name>manager</role-name>
- <auth-constraint>

<user-data-constraint>元素

- 该元素指定数据应该如何在客户与服务器之间传输，它的DTD定义：

  <!ELEMENT user-data-constraint (description?, transport-guarantee)>

- transport-guarantee元素指定数据传输的方式，它的取值为下面三者之一：NONE、INTEGRAL或CONFIDENTIAL。

- <user – data – constraint>
- <transport – guarantee> INTEGRAL</ transport – guarantee>
- </user – data – constraint>

2. <login-config>元素

- <login-config>元素定义使用的验证机制。验证机制不同，当Web客户访问受保护的Web资源时，系统弹出的对话框不同。该元素的DTD定义：

  <!ELEMENT login-config (auth-method?, realm-name?, form-login-config?)>

2. <login-config>元素

- 这里，<auth-method>元素指定使用的验证方法，其值可以为BASIC、DIGEST、FORM和CLIENT-CERT。
- <realm-name>元素仅用在HTTP基本验证（BASIC）中，指定安全域（realm）名称。
- <form-login-config>元素仅用在基于表单的验证（FORM）中，指定登录页面的URL和错误页面的URL。

2. <login-config>元素

- 下面是web.xml的代码片段，它是BASIC验证机制的配置。

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

3. <security-role>元素

- 该元素用来定义安全约束中引用的所有角色名，该元素的定义格式为：

  <!ELEMENT security-role (description?, role-name*)>

- 其中，<role-name>元素定义角色名。

3. <security-role>元素

- 例如，假设在helloweb应用程序中要引用manager角色和member角色，则该元素可定义为：

```
<security-role>
        <role-name>manager</role-name>
        <role-name>member</role-name>
</security-role>
```

9.3.2　安全验证示例

- 下面通过一个实际的例子说明安全性的使用。首先建立一个名为AccountServlet的Servlet，然后分别采用基本验证和基于表单的方式测试安全性。
- 程序9.2 AccountServlet.java

1. 基本身份验证方法

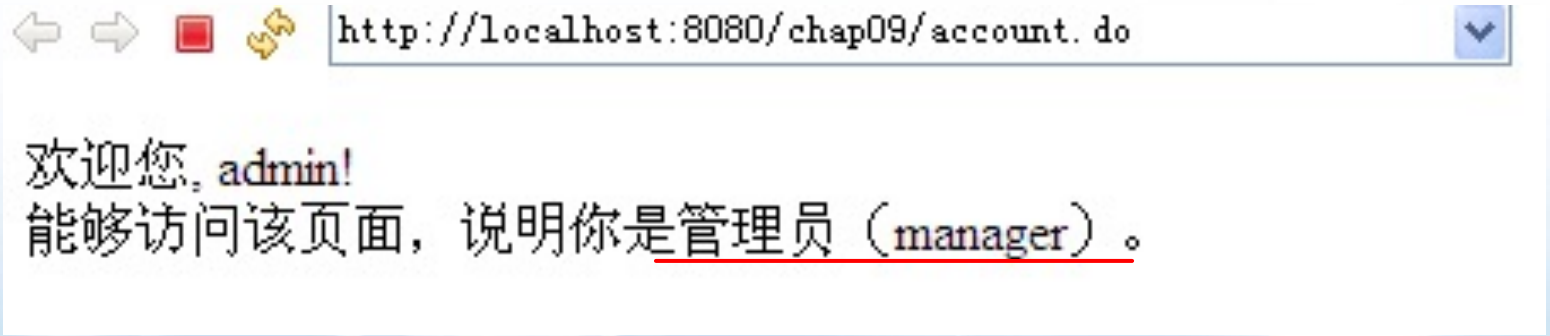
- Web应用程序的所有安全的需要都可以在DD中指定，如下面代码所示。

```
<security-constraint>
        <web-resource-collection>
          <web-resource-name>Account Servlet</web-resource-name>
                <url-pattern>/account.do</url-pattern>
                <http-method>GET</http-method>
        </web-resource-collection>
        <auth-constraint>
                <role-name>manager</role-name>
        </auth-constraint>
    <user-data-constraint>
    <transport-guarantee>NONE</transport-guarantee>
```

```
        </user-data-constraint>
    </security-constraint>
    <login-config>
        <auth-method>BASIC</auth-method>
        <realm-name>Account Servlet</realm-name>
    </login-config>
    <security-role>
        <role-name>manager</role-name>
    </security-role>
    <security-role>
        <role-name>member</role-name>
    </security-role>
```

1. 基本身份验证方法

- 当在浏览器中使用http://localhost:8080/helloweb/account.do访问AccountServlet时，这将向它发送一个GET请求。浏览器将弹出一个如图9.1所示的对话框，输入正确的用户名和密码，可以看到该Servlet发回的响应页面。



2. 基于表单的验证方法

- 基于表单的验证方法需要使用自定义的登录页面代替标准的登录对话框同时需要在web.xml的<login-config>元素的子元素<form-login-config>元素中指定登录页面和出错页面。
- 程序9.3 loginPage.jsp
- 程序9.4 errorPage.jsp
- 下面修改web.xml文件，需将<login-config>元素的内容修改为：

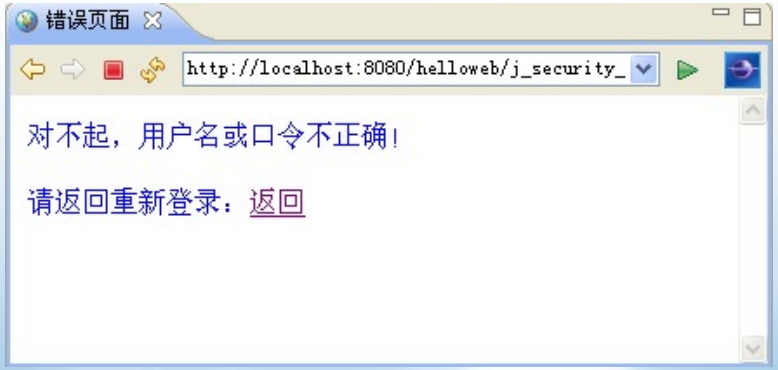
2. 基于表单的验证方法

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

<realm
className="org.apache.catalina.realm.JDBCrealm"
driverName="com.may "

2. 基于表单的验证方法

- 重启服务器，使用http://localhost:8080/helloweb/account.do访问AccountServlet时，这将发送一个GET请求。浏览器将显示loginPage.jsp页面的表单，如图9-3所示。输入正确用户名和口令即可访问AccountServlet。如果输入用户名和口令不正确，将显示如图9-4所示的错误页面（errorPage.jsp）。



9.4 编程式的安全

- Web应用程序的安全属于声明式的安全。这种安全机制是部署者在部署Web应用时通过web.xml配置的，而在资源或Servlet中并不涉及安全信息。
- 然而在某些情况下，声明式安全对应用程序来说还不够精细。
- 例如，假设希望一个Servlet能够被所有职员访问。但我们希望服务器为主管（director）产生的输出与为其他职员（employee）产生的输出不同。

- 在HttpServletRequest接口中定义了如下三个用于识别用户和角色的方法：

public String getRemoteUser()：如果用户已被验证，该方法返回验证用户名。

public boolean isUserInRole (String rolename)：返回一个布尔值，表示验证的用户是否属于指定的角色。

public Principal getUserPrincipal()：返回java.security.Principal对象，它包含当前验证的用户名。

- 下面程序通过编程为主管和雇员产生不同的输出页面。
- 程序9.5 AuthorizationServlet.java
- 程序9.6 web.xml

- 基于Servlet和JSP技术实现一个学生成绩录入功能。用户需求：设计一个表单页面（如：score.jsp），用户录入学生学号和学生成绩，提交到后台Servlet （如：ScoreServlet）, 并保存在数据库（如: stu_score）中；成绩录入功能需要验证用户权限（录入功能需要用户有ROLE_ADMIN的角色权限）；根据需求给出以下代码或配置示例：（1）给出保存学生成绩的数据表设计；（2）给出成绩提交表单页面的form主要代码片段；（3）给出servlet主要代码片段；（4）给出数据保存到数据库中的主要代码片段；（5）给出权限验证相关配置；（6）添加来访IP地址过滤器，只允许211。20段的IP访问。

9.5 小 结

- 随着企业在Internet上处理的业务越来越多，安全性问题也变的越来越重要。Servlet规范定义了4种验证用户的机制：BASIC、CLIENT-CERT、FORM和DIGEST。验证机制是在应用程序的部署描述文件（web.xml）中定义的。
- 本章讨论了如何建立安全的Web应用程序。重点讨论了如何在部署描述文件中配置安全性来实现声明式的安全。