**安全的Web应用**

# 授课内容

授课科目：JavaEE

授课内容：安全的Web应用

授课类型：讲授+实践

授课时间：3+1学时

主讲教师：孟双英

# 教学目标要求

## 能力目标：

1. 培养学生认识网络安全的重要性；
2. 培养学生认识Web应用的安全的设置；
3. 培养学生掌握如何保障Web应用的安全性。

## 知识目标：

1. 了解网络安全相关的概念；
2. 掌握JavaEE中的安全认证机制；
3. 掌握JavaEE中的安全的声明；
4. 了解Servlet的安全编程。

# 课件分析

概 述： 本章围绕着网络安全相关的概念阐述Web应用安全的重要性以及在JavaEE应用中如何通过一些机制保障资源的安全性，主要介绍Servlet规范中的4种认证机制、安全声明以及安全编程。

教学重点：JavaEE应用中4种认证机制的设置、安全声明、安全编程。

教学难点：JavaEE应用中4种认证机制的设置、安全声明。

# 教学方法

问题教学法、类比法、讲授法、演示法等。

# 教学过程

## 课前补充

（0分钟）

## 上节回顾

（0分钟）

## 作业点评

（0分钟）

## 导入新课

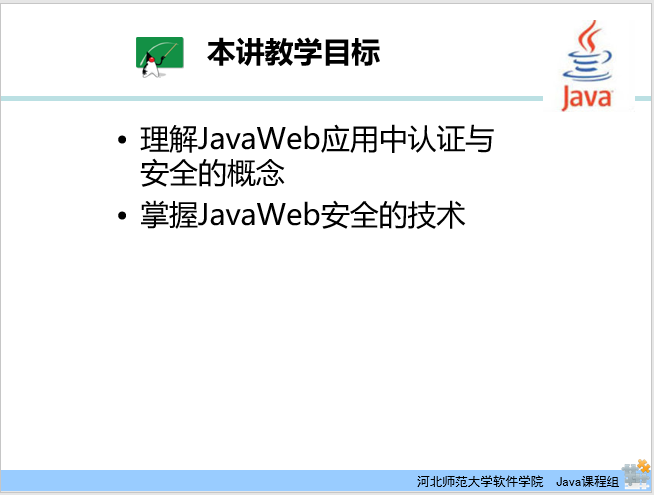
（5分钟）

互联网作为商业交易的工具快速发展，越来越多的公司提供了网络交易服务，所有的商业活动都将在网上进行，成千上万的网民在网上购物，在网上传递个人信息，每天在网上发生各种各样的商业活动。那么，这些网上交易的安全性就显得尤为重要，目前有许多公司专门从事互联网安全事业，比如启明星辰、绿盟、亿阳、华为等等。

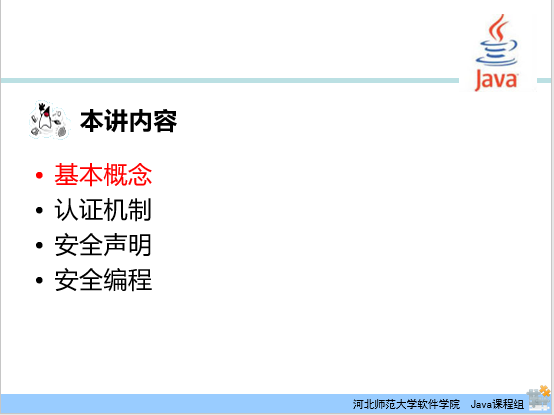
本章内容主要介绍网络安全相关的一些概念，以及在JavaEE应用中从哪些方面来保障应用的安全，本章所介绍的安全保障机制都是基于应用本身的资源安全的设置，更多有关于安全的内容和知识有待大家后期的实践中接触和学习。

## 讲授新课

（120分钟）

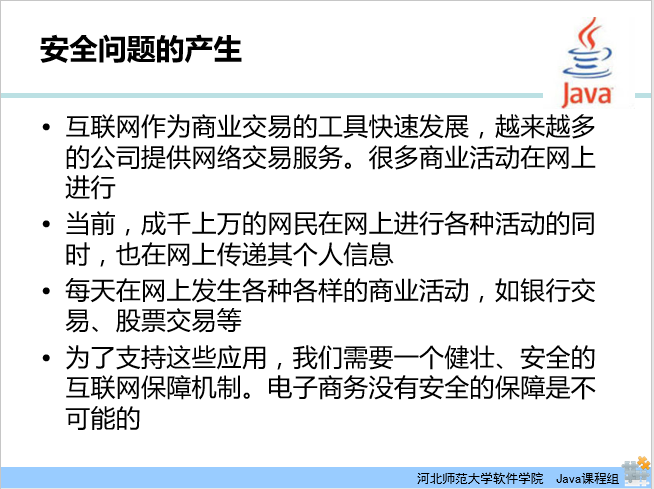


本章教学目标：了解Web应用中安全相关的概念，理解JavaWeb应用中的认证与安全的机制，掌握JavaWeb中如何通过一系列的方法保证应用中资源的安全。

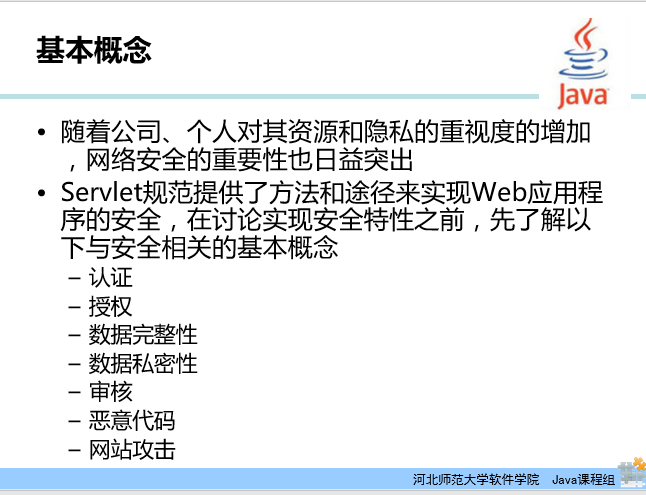


本章教学内容共分四个部分，首先介绍安全相关的基本概念，然后介绍JavaWeb应用中几种认证机制，安全声明，以及Servlet规范中安全编程相关的内容。

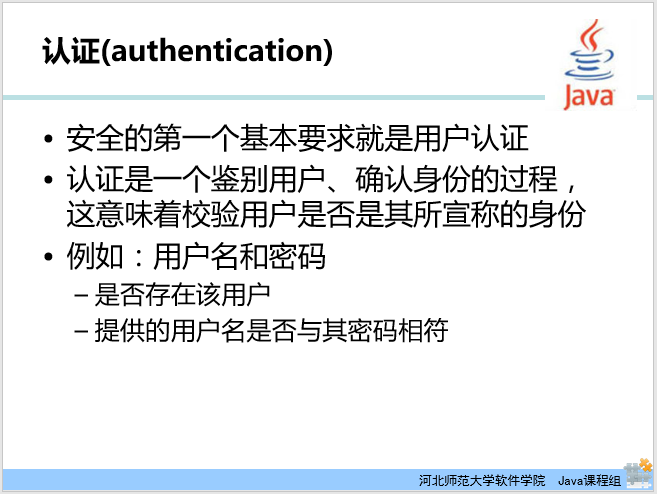
首先来认识一下网络安全的基本概念。



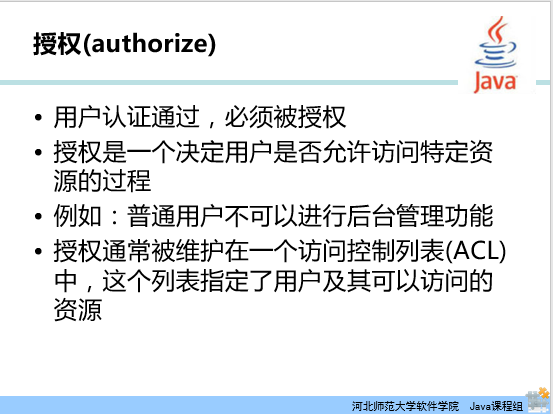
互联网已经深入到人们工作和生活的方方面面，比如：购物、人际沟通、生活缴费、在线学习等等。网民的个人信息在互联网的各种活动中已经暴露无遗，如果各网络应用不够健壮、安全，则网民的人身和财产安全都将受到侵害。任何一款互联网应用产品，不管其功能的优劣，前提必须保障使用者的人身和财产安全，即必须要有足够的安全保障机制和措施。



由于人们的安全意识越来越强烈，对个人的隐私信息重视程度也日益增加，所以网络安全的重要性日益突出，任何一个Web应用的实现技术都会有其自身的安全保障机制，作为企业应用主流技术实现的Java，在JavaEE的规范中提供了一系列的方法和途径来保障Web应用资源的安全，在讨论这些安全机制之前，首先来认识几个与安全相关的基本概念，分别是：认证、授权、数据完整性、数据私密性、审核、恶意代码、网站攻击。

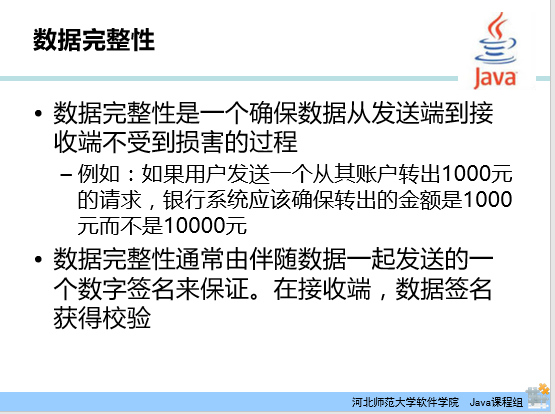


认证英文名称authentication，是安全的一个基本要求，所谓的认证是一个用户身份鉴别和确认的过程，是指校验用户是否是其所宣称的身份。一个现实生活中的认证例子就是当飞机乘客登机前必需出示护照，这个护照是乘客的ID，这个ID的认证过程就是检验乘客的合法身份。在互联网世界中，最基本、最典型的认证就是用户提供用户名和密码登录，首先检查是否存在该用户名，其次验证用户提供的密码是否与其用户名相符。

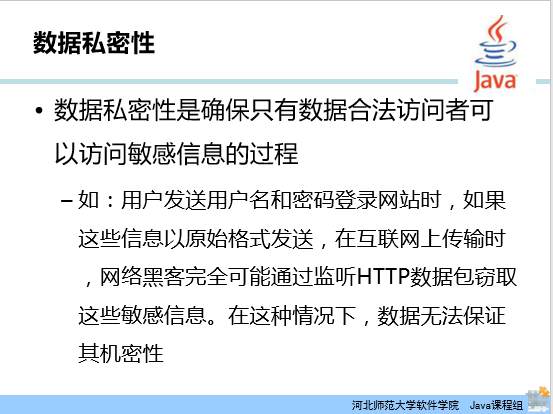


授权英文名称是authorize，用户认证通过后必须被授权，授权是一个决定用户是否被允许访问某一特定资源的过程。还拿刚才生活中的登机例子来说，一名乘客认证通过，被允许登机，但登机后乘客是不被允许进入驾驶室的，甚至飞机起飞前乘客必须要把手机关机，系好安全带，安静地坐在自己的座位上，这就是普通乘客的权限，普通乘客只被授权了这些权限，其他的操作不在被允许的范围。互联网中最典型的授权的例子是普通用户登录系统后不能进行后台管理功能操作。

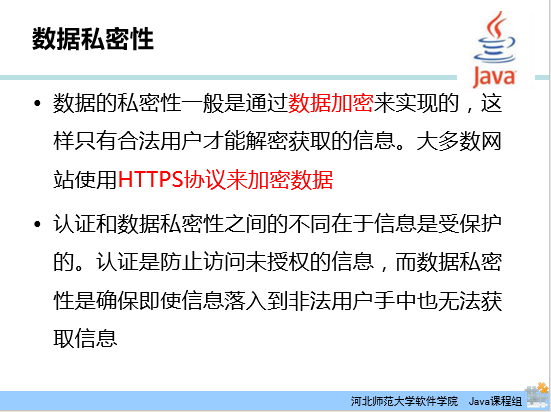
通常授权的实现是维护在一个访问控制列表，在这个列表中记录了用户以及其可以访问的资源。



数据完整性是一个确保数据从发送端到接收端不受损害的过程，比如：用户发送一个从其账户转出1000元的请求，银行系统应该确保转出的金额是1000而不是10000或者其他的。数据完整性通常是由数字签名来保障的。所谓数字签名，就是只有信息的发送者才能产生的，别人无法伪造的一段数字串，它是对接受者接收到的信息的真实性的一个证明。

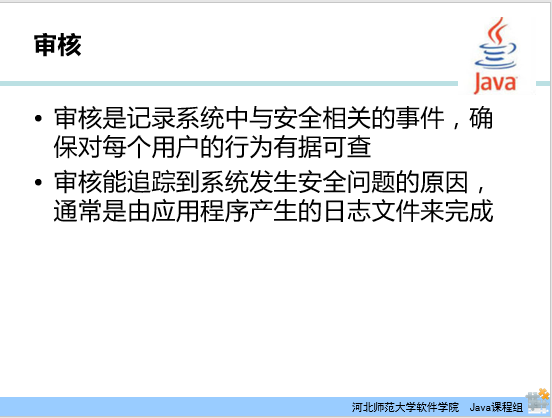


数据私密性是确保只有数据合法的访问者才可以访问的敏感信息过程。比如：用户登录时向服务器端发送用户名和密码时，如果用户名和密码不是私密信息则可以直接发送，数据在互联网传输过程中如果被网络黑客截获，则信息被窃取；如果用户名和密码是私密信息可以使用密文传输，即使传输过程中被黑客截获，黑客也无法解密获取到正确的信息。

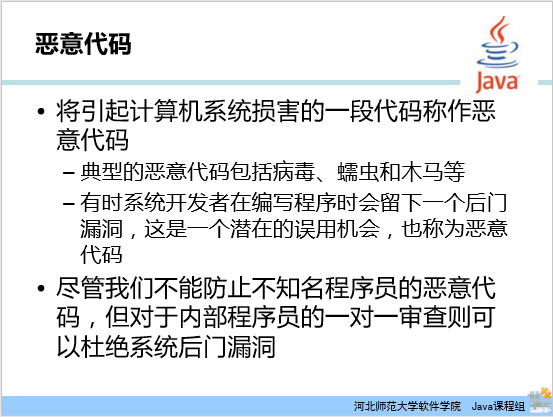


通常情况下，数据的私密性是通过数据加密来实现的，只有合法的用户才能解密获取正确的信息。多数网站使用HTTPS协议来加密数据传送。

认证和数据私密性是有区别的，同学们一定不能混淆两个概念，认证是指防止非法用户获取信息，而数据私密性是确保即使信息落入非法用户手中也无法获取正确信息。

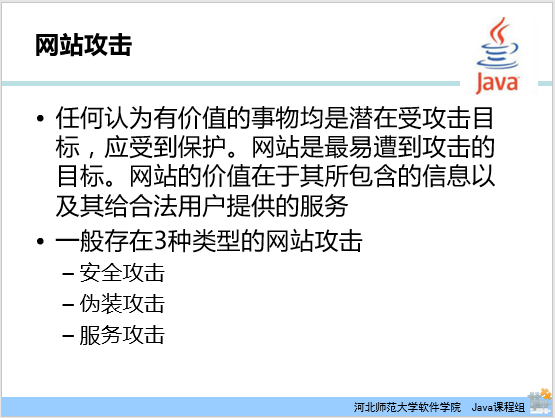


审核是记录系统中与安全相关的事件，确保每个用户的行为有据可查。审核能够追踪到系统发生安全问题的原因，通常是由应用程序产生的日志来完成审核。

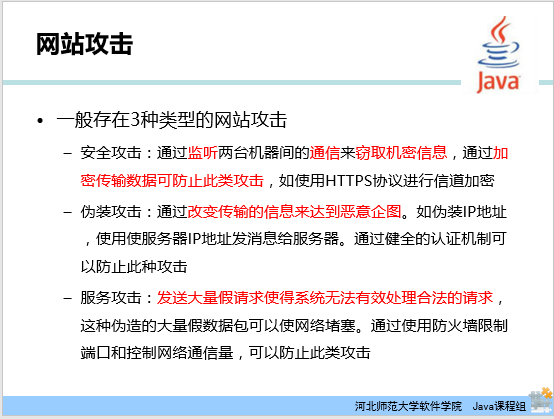


恶意代码是指引起计算机系统损害的一段代码。典型的恶意代码包括病毒、蠕虫和木马等。

病毒、蠕虫、木马是同学们很容易想到的恶意代码的典型代表，但除了这些外界威胁，有时候系统开发者在程序编写的时候故意留下的漏洞也称为恶意代码，因为这也是一潜在的被别人利用的机会。对于后者虽然无法防止，但可以通过一对一的程序审查尽可能地做好后期检查，杜绝系统的后门漏洞。



在互联网的诸多应用中，网站是最易收到攻击的目标，互联网应用中任何有价值的事物均是受攻击的目标，都应该收到保护。网站的价值就在于其所包含的信息以及其给合法用户提供的服务。一个网站可能被不同的人处于不同的目的攻击，比如：黑客的自我娱乐，被解雇员工的报复行为，专业盗取信用卡号的目的性很强的窃贼等。

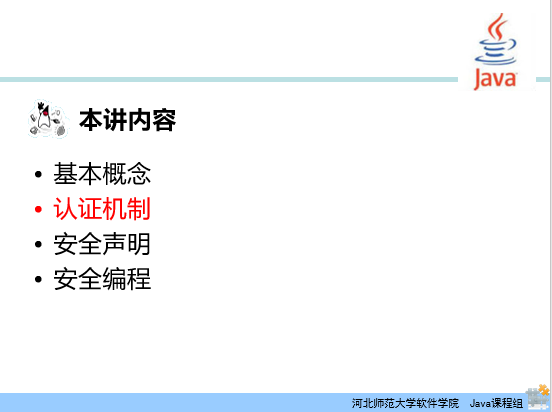


总体来说网站攻击大体存在以下三类：

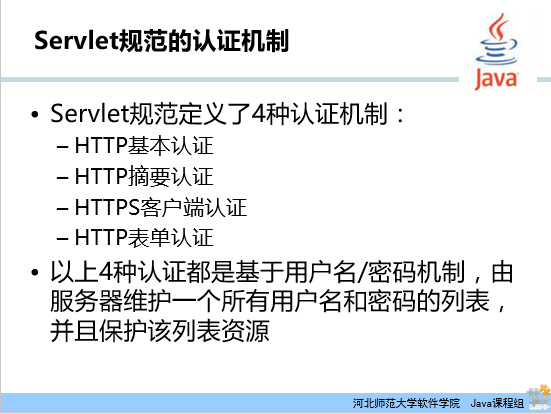
1.安全攻击，通过监听两台机器之间的通信来窃取机密信息。通过加密传输数据可以防止此类攻击，比如：金融机构一般都采用HTTPS协议完成在线交易活动。

2.伪装攻击，通过改变传输的信息来达到恶意企图，伪装IP地址是此类攻击的主要手段，在这种技术中，攻击者使用一个可使服务器信任的IP地址来发送消息给服务器，服务器因此受到欺骗放行攻击者。通过健全的认证机制可以防止此类攻击。

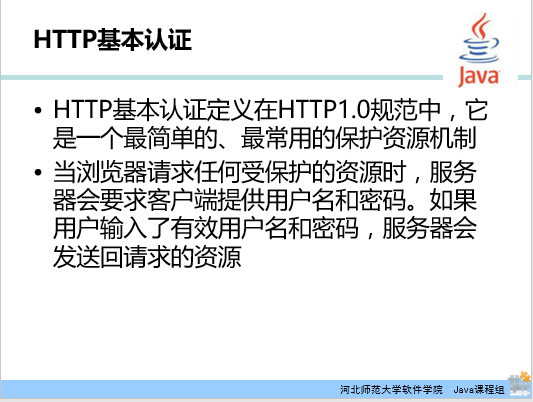
3.服务攻击，发送大量假请求使得系统无法有效处理合法请求，这种伪造的大量的假数据包可以使网络发生堵塞。通过使用防火墙限制端口和控制网络通信量，可以防止此类攻击。



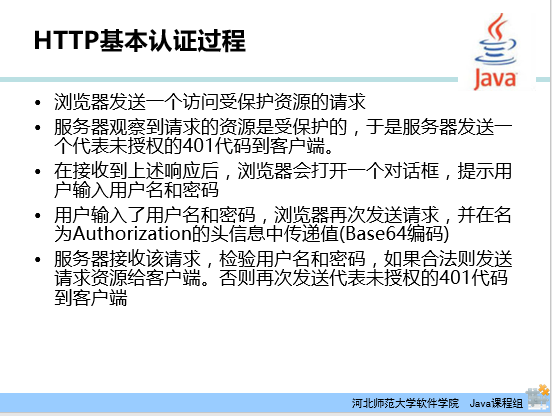
了解了基本的网络安全相关的概念后，接下来学习JavaEE应用中的保障应用安全的措施和方法。首先来看Servlet规范中的认证机制。



Servlet规范中定义了四种认证机制，分别是HTTP基本认证、HTTP摘要认证、HTTPS客户端认证、HTTP表单认证。四种认证都是基于用户名/密码机制的，在服务器端维护一个所有的用户名和密码的列表，并保护该列表资源。



HTTP基本认证是定义在HTTP1.1规范中，是一个简单的、常用的保护资源的机制。当浏览器请求任何受保护的资源时，服务器都要求客户端提供用户名和密码。



HTTP基本认证过程是：

1.浏览器发送一个访问资源的请求

2.服务器观察到请求的资源是受保护的，服务器会发送一个代表未授权的401代码到客户端，这个响应信息中还包含了一个通知浏览器访问受保护资源需要基本认证的消息头。

3.客户端接收到响应后，浏览器会打开一个对话框，提示用户输入用户名和密码。

4.一旦用户输入了用户名和密码，浏览器会在此发送请求，并在名为authorization的头信息中传递值。

5.当服务器接收到该请求后，会校验用户名和密码，如果用户名和密码合法，则发送请求的资源给客户端，如果不合法再次发送一个代表未授权的401代码到客户端。

6.客户端接收响应，展示响应回的资源，或者再次显示输入用户名密码的对话框。



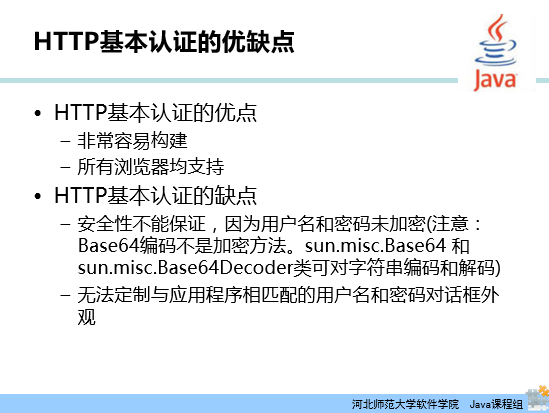
Tomcat中的Manager站点就是HTTP基本认证方式，若在JavaEE应用中使用HTTP基本认证，需要首先配置Tomcat服务器中的用户信息，在Tomcat的安装目录找到tomcat-users.xml文件，打开文件，在其中<tomcat-users>标签内添加如下配置：

|  |
| --- |
| <role rolename="role1"/>  <user name="mengsy" password="12345678" roles="role1"/> |

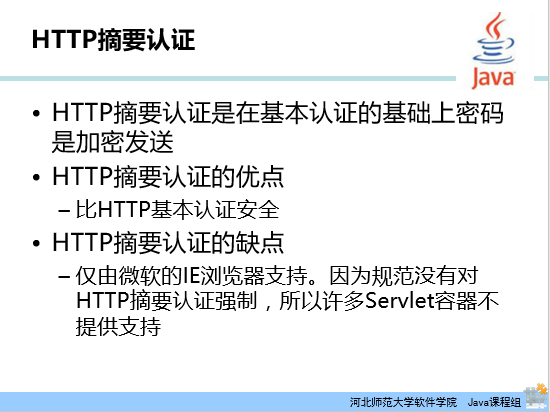
然后在Web应用中的web.xml文件中添加认证的配置，配置文件如下：

|  |
| --- |
| <security-constraint>       <web-resource-collection>                 <web-resource-name>                  protected Resource                 </web-resource-name>                 <url-pattern>/BasicVerify/\*</url-pattern><!-- 受保护资源的配置 -->      </web-resource-collection>     <auth-constraint>          <role-name>role1</role-name><!—设置允许访问资源的角色名 -->     </auth-constraint>    </security-constraint>     <login-config>      <auth-method>BASIC</auth-method>      <realm-name>Default</realm-name>    </login-config> |

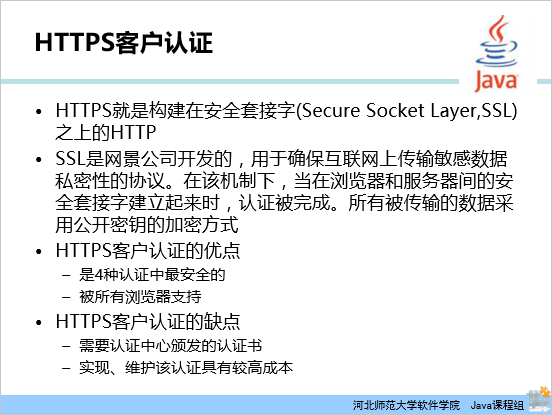
login-config元素用来指定服务器使用何种认证机制。此元素在web.xml中应放在security-constraint 的紧后面。其子元素auth-method用于设置认证机制的类型。



通过示例配置可以看到HTTP基本认证配置简单方便，且所有的浏览器都支持。由于HTTP基本认证的用户名和密码都是明文传输和保存的，其安全性不能保证，另外一个缺点是用户的登录框是浏览器自动设置的，用户无法定制登录框的外观样式。

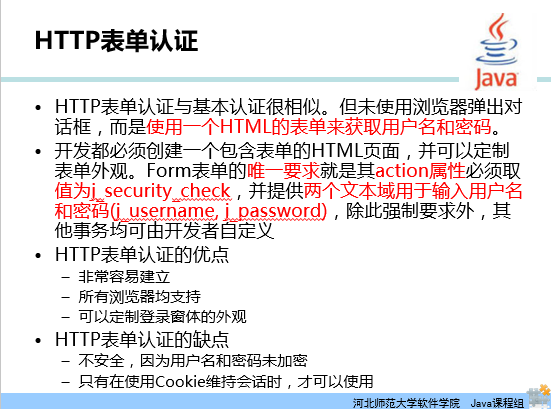


HTTP摘要认证是HTTP基本认证的升级，是在基本认证的基础上，密码加密发送，较基本认证安全，但并不通用，只有少数浏览器和Servlet容器支持。



HTTPS客户认证是构建在安全套接字上的HTTP认证，是用于确保互联网上传输敏感数据私密性的协议。浏览器和服务器之间传输的数据使用公开秘钥的加密方式来进行传输。

HTTPS客户认证是四种认证机制中最安全的一种，也被所有的浏览器支持，但其需要专门的认证中心颁发认证证书，实现、维护认证具有较高的成本。

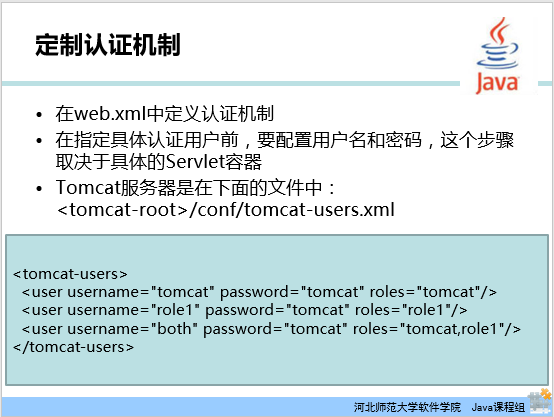


HTTP表单认证与HTTP基本认证类似，不使用浏览器自动弹出的对话框，而是使用一个自定制的HTML的表单来呈现登录框，要求其form的action属性取值为j\_security\_check，用户名和密码框表单域的name属性必须为j\_username，j\_password。HTTP表单认证最大的优点是可定制登录框的外观。缺点是不安全，这一点与HTTP基本认证一致，用户名和密码没有加密存储和传输，HTTP表单认证是有在浏览器使用Cookie来维持会话时才可使用，即禁用Cookie后不可用。

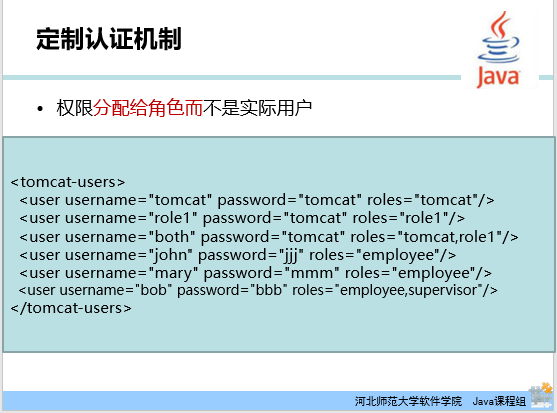
演示：在Web应用的web.xml文件中修改配置，改为如下代码：

|  |
| --- |
| <security-constraint>  <web-resource-collection>  <web-resource-name>protected Resource</web-resource-name>  <url-pattern>/BasicVerify/\*</url-pattern>  <http-method>POST</http-method>  <http-method>GET</http-method>  </web-resource-collection>  <auth-constraint>  <role-name>role1</role-name>  </auth-constraint>  </security-constraint>  <login-config>  <!-- HTTP表单证配置-->  <auth-method>FORM</auth-method>  <form-login-config>  <form-login-page>/formlogin.html</form-login-page>  <form-error-page>/formerror.html</form-error-page>  </form-login-config>  </login-config> |

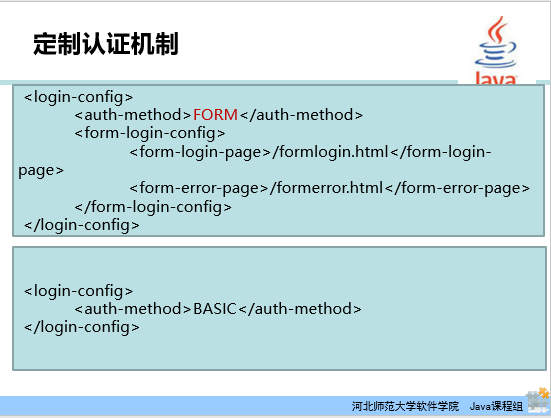
form-login-config元素用于设定表单认证的登录页面和错误页面。



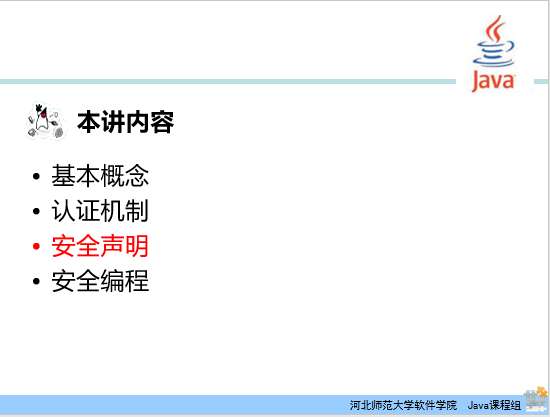
在Web应用中使用认证机制，必须先在Tomcat服务器中设置角色、用户信息，然后，在Web应用的web.xml中定义使用的认证机制和受保护的资源。



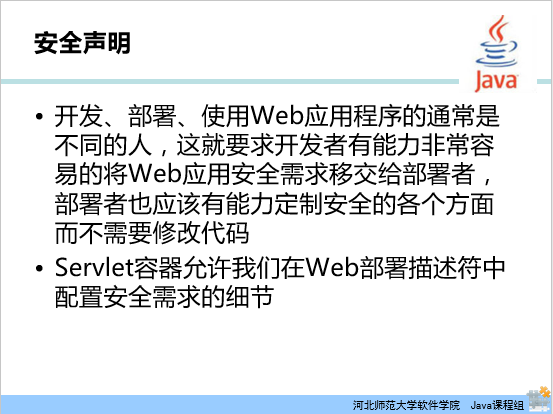
前面HTTP基本认证的例子中已经配置过一次Tomcat的角色和用户，这里再重点强调一点，在Tomcat的配置中权限是分配给角色，而不是分配给用户的，每个用户可以属于一个或多个角色。在上述代码中，我们又添加了3个用户名：john mary bob 。john mary的角色是员工，而bob的角色即是员工又是经理，在指定了用户名、密码和角色后，就可以在Web应用程序的部署描述符中使用<login-config>元素来定义认证机制了。



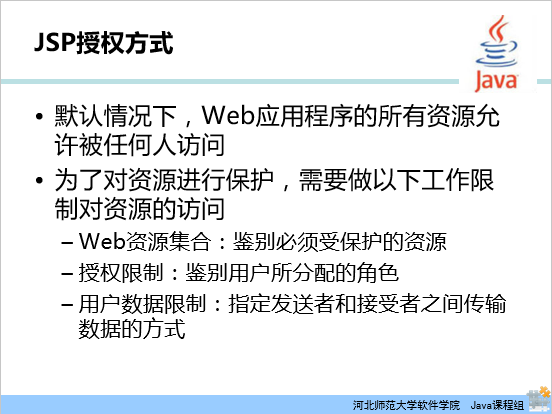
<login-config>标签是用来定义认证机制的，<auth-method>标签是定义使用哪种认证机制，<form-login-page>标签是定义认证的登录页面和错误页面的。



接下来我们继续学习JavaEE应用中的安全声明机制。



在实际的项目开发中，往往开发Web应用程序的人和部署系统的人是不同的，比如许多软件公司专门致力于开发针对行业需求的Web应用产品，这意味着开发者有能力非常容易地将Web应用程序的安全需求移交给Web部署者，Web应用部署者也有能力定制Web应用程序安全的各个方面，而不需要修改代码。Servlet容器允许我们在Web部署描述符中配置安全需求细节的能力称为安全声明。即，所谓的安全声明是指不改动源代码，直接通过配置文件来声明受保护资源相关信息的机制。

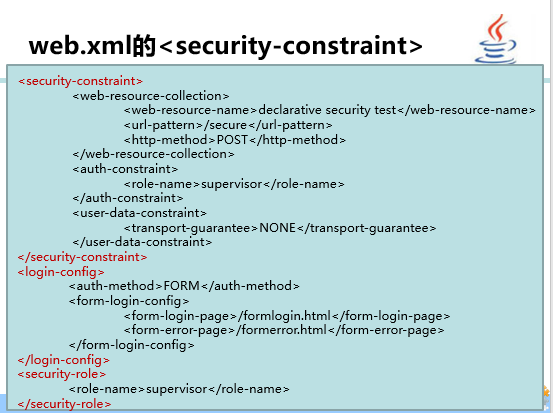


默认情况，Web应用程序的所有资源是允许所有人访问的，为了对资源进行保护，我们可以做以下工作来限制对资源的访问，并结合Servlet四种认证机制对Web应用的资源安全起到一定的保护作用：

1.Web资源集合，鉴别必须受保护的Web应用程序资源，用户必须拥有合法的授权才能访问受保护资源。

2.授权限制，鉴别用户所分配的角色。许可权应分配给角色而不是独立的用户，有助于减少实际用户和许可权之间的紧密程度。

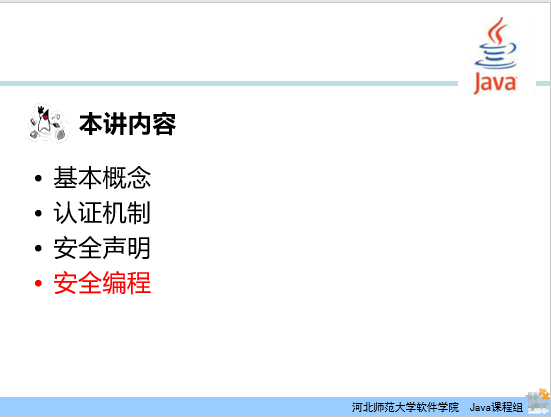
3.用户数据限制，指定在发送者和接受者之间的数据传输方式，即限制指定了Web应用程序对传输层的要求，指定了维护数据完整性和机密性的策略。



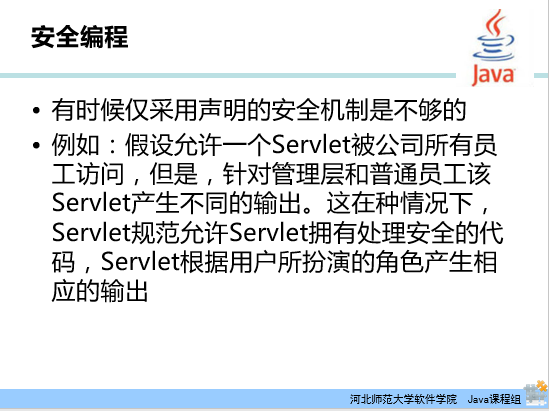
PPT中的是一个JavaEE应用程序中的安全声明的示例程序。与HTTP基本认证示例中使用过的配置一致。

security-constriaint元素是用来指示服务器使用何种验证方法。web-resource-collection子元素用于设置受保护的资源，auth-constraint子元素定义哪些用户具有访问受保护资源的权限，user-data-constraint是可选子元素定义在访问相关资源时使用任何传输层保护。

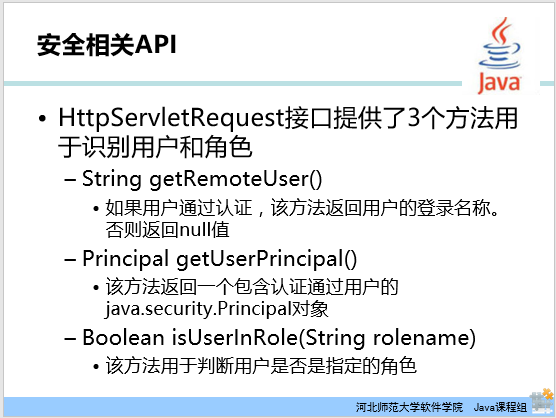
web-resource-collection中的web-resource-name子元素定义受保护资源的名称，url-pattern子元素定义受保护资源的访问URL映射，http-method子元素定义受保护的请求方式，即只有同时满足URL映射和请求方式才会被拦截验证是否登录。



仅仅依靠安全声明和认证机制是不足以细致地管理JavaEE应用的资源的安全性，接下来介绍另一种安全保障机制——安全编程。



安全编程是在程序开发的过程中来保障应用运行的安全性和可靠性，比如：允许一个资源可以被所有的用户访问，但不同的用户访问该资源会得到不同的响应结果，则这样的安全控制通过前面介绍的安全声明和认证机制则无法做到，只能借助于程序实现。



Servlet规范中提供了三个接口方法来鉴别不同的用户身份。封装在HttpServletRequest接口中：

1.getRemoteUser()方法，如果用户通过认证，则方法返回登录的用户名，如果未通过认证，返回null。

2.getUserPrincipal()方法，返回包含认证通过用户的Principal对象。

3.isUserInRole(roleName)方法，用于判断用户是否被包含在指定的角色列表中。

## 课时小结

（10分钟）

1. 了解与安全相关的一些概念：认证、授权、数据完整性、数据私密性、审核等等。
2. 掌握Servlet规范中的四种认证机制：HTTP基本认证、HTTP摘要认证、HTTPS客户 认证、HTTP表单认证。
3. 了解安全声明及其主要的设置标签。
4. 了解安全编程的三个方法。

# 布置作业

（5分钟）

完成实验十五。