1. •认证是检验某人真正是ta自称的那个人的过程。一般是通过用户名/密码是否正确。

•授权是检查该级别的用户是否具有访问权限。通过在部署中声明安全约束描述符完成，称为声明式安全。

1. 声明式安全的缺点是，支持数据加密的身份验证发方法只能使用一个默认登陆对话框。

声明性安全的唯一方法是允许使用一个自定义HTML表单，但数据传输不加密。

1. 可以通过编辑conf目录中的tomcat-users.xml来创建用户和角色。

<?xml version='1.0' encoding='utf-8'?>

<tomcat-users>

<role rolename="manager" />

<role rolename="member" />

<user username="tom" password="secret" roles="manager,member"/>

<user username="xhsf" password="12345" roles="member"/>

</tomcat-users>

1. Security-constraint元素指定一个资源集合和角色或角色可以访问的资源。有两个子元素：web-resource-collection和auth-constraint。在部署描述文件中描述

示例:

<security-constraint>  
 <web-resource-collection>  
 <web-resource-name>JSP page</web-resource-name>  
 <url-pattern>\*.jsp</url-pattern> <!-- 指定资源路径 -->  
 <http-method>GET</http-method> <!-- 指定允许的HTTP方法 -->  
 </web-resource-collection>  
 <auth-constraint/> <!-- 使资源受保护 -->  
</security-constraint>

示例2：

<security-constraint>  
 <web-resource-collection>  
 <web-resource-name>index jsp</web-resource-name>  
 <url-pattern>index.jsp</url-pattern> <!-- 指定资源路径 -->  
 <http-method>GET</http-method> <!-- 指定允许的HTTP方法 -->  
 </web-resource-collection>  
 <auth-constraint>  
 <role-name>member</role-name>  
 <role-name>manager</role-name>  
 </auth-constraint> <!-- 使资源受保护 -->  
</security-constraint>  
<login-config>  
 <auth-method>BASIC</auth-method> <!-- 使可以使用Basic access authentication认证 -->  
 <realm-name>Members Only</realm-name> <!-- 在浏览器登录对话框中显示的realm-name -->  
</login-config>

1. 摘要加密，使用MD5算法，结合用户名、域名和密码的哈希值

在<login-config>标签里的<auth-method>的值设置为DIGEST

<login-config>  
 <auth-method>DIGEST</auth-method><!--摘要认证，使用MD5算法创建一个组合用户名、域名和密码的哈希值，发送到服务器-->  
 <realm-name>Members Only</realm-name>  
</login-config>

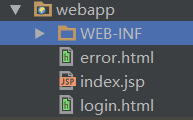
1. 基于表单的身份证认证，没有自带加密。

在<login-config>标签里的<auth-method>的值设置为FORM

<form-login-page>和<form-error-page>子标签设置登录页面和错误页面

示例：

<login-config>  
 <auth-method>FORM</auth-method>  
 <form-login-config>  
 <form-login-page>/login.html</form-login-page> <!-- 设置登录页面 -->  
 <form-error-page>/error.html</form-error-page> <!-- 设置错误页面 -->  
 </form-login-config>  
</login-config>



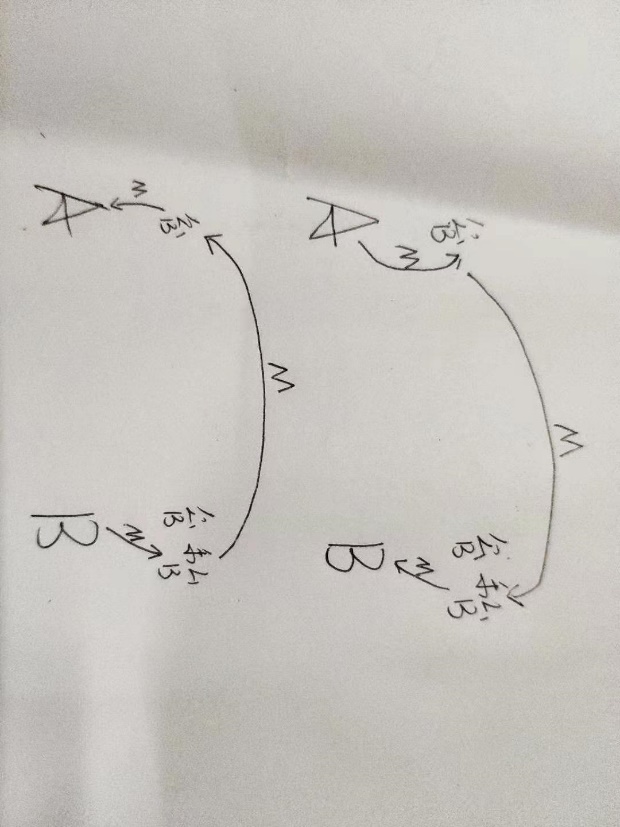
1. 非对称加密
   1. 有三个特点：

加密解密。信息对第三方进行加密隐藏。只有预期的接收者才能解密。

身份验证。验证确保实体就是声称者。

数据的完整性。许多计算机在互联网上发送的消息传递。它必须是确保发送的数据不变，完好无损。

* 1. 公钥加密，私钥解密。



* 1. SSL身份认证。通过引入证书。

•公钥

•关于主体的信息，即公开密钥的所有者。

•证书发行机构的名字。

•到证书到期时间的时间戳。

* 1. 进行数字签名（对一个文件）。

•签署一份文件设计创建一个文档的摘要，并使用签名者的私钥对摘要加密。

•验证的步骤是：

·使用签名者的公钥解密伴随文件的摘要。

·创建一个文档的摘要。

·比较前两个步骤的结果。如果两者匹配，那么该文件未被篡改。

•这种加密只有私钥的持有者才能加密，只能用相应的公钥解密。

* 1. SSL认证的流程

A -> B发起连接请求。

B -> A 发送证书。

B -> A 发送用B的私密加密的信息。

A 使用B证书里的公钥解密。A确实B的真实身份。

A -> B 发送用于后续信息交互的密钥。使用对称加密。

* 1. 数据的完整性

SSL引入消息认证码（MAC）。MAC是用一个密钥和传输数据计算出的数据块。

MD5是其中一个消息摘要算法。

* 1. SSL的工作原理

•使用https协议

•浏览器->服务器：你真的是Amazon.com吗？

服务器：是的，这是我的证书。

浏览器：单独的证书是不够的，请给一些别的东西。

服务器：我真的Amazon.com + [使用亚马逊网站的私钥加密同一信息的摘要]

浏览器使用Amazon的公钥解密摘要，并创建“我真的Amazon.com”的摘要。如果这两者匹配，验证成功。那么浏览器会产生一个随机密钥，使用Amazon公钥加密。这个随机密钥来加密和解密随后的信息（使用对称加密）。

1. 安全注释类型

示例1: 该Servlet要有member身份才可以访问

@ServletSecurity(value = @HttpConstraint(rolesAllowed = "member"))

示例2：该Servlet只有GET方法才需要进行member身份进行验证。

@ServletSecurity(httpMethodConstraints =   
@HttpMethodConstraint(value = "GET", rolesAllowed = "member"))

示例3：该Servlet可以给任何用户通过GET方法访问，其他方法只有身份为member的用户才可以访问

@ServletSecurity(value = @HttpConstraint(rolesAllowed = "member"),  
httpMethodConstraints = @HttpMethodConstraint("GET") )

示例4：该Servlet拒绝所有GET方法的请求，其他方法的请求需要有member身份才能访问。

@ServletSecurity(value = @HttpConstraint(rolesAllowed = "member"),  
httpMethodConstraints = @HttpMethodConstraint(value = "GET",   
 emptyRoleSemantic = ServletSecurity.EmptyRoleSemantic.*DENY*) ) //否认所有GET方法的请求。

1. Servlet的安全API

可以使用HttpServletRequest接口的以下方法实现安全性：

* 1. String getAuthType():返回用来保护servlet认证方法。如果没有安全约束则返回空。
  2. String getRemoteUser(): 返回发出此请求登录用户，如果用户尚未验证则返回空。
  3. boolean isUserInRole(String role): 返回一个指示用户是否属于指定角色的布尔值。

④java.security.Principal getUserPrincipal():返回包含当前通过验证的用户细节信息的⑤Principal，如果用户没有通过认证返回空。

⑤boolean authenticate(HttpServletResponse response) throws IOException: 通过指示浏览器显示登录表单来验证用户。

⑥void login(String username, String password) throws ServletException:试图使用所提供的用户名和密码进行登录。登录失败会抛出ServletExcepiton异常。

⑦void logout() throws ServletException 注销用户。

示例：

@WebServlet(urlPatterns = "/auth")  
public class AuthServlet extends HttpServlet {  
 @Override  
 protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 if (req.authenticate(resp)) {  
 resp.setContentType("text/html");  
 PrintWriter out = resp.getWriter();  
 out.println("Welcome");  
 } else {  
 System.*out*.println("User not authenticated");  
 }  
 }  
}