

Security

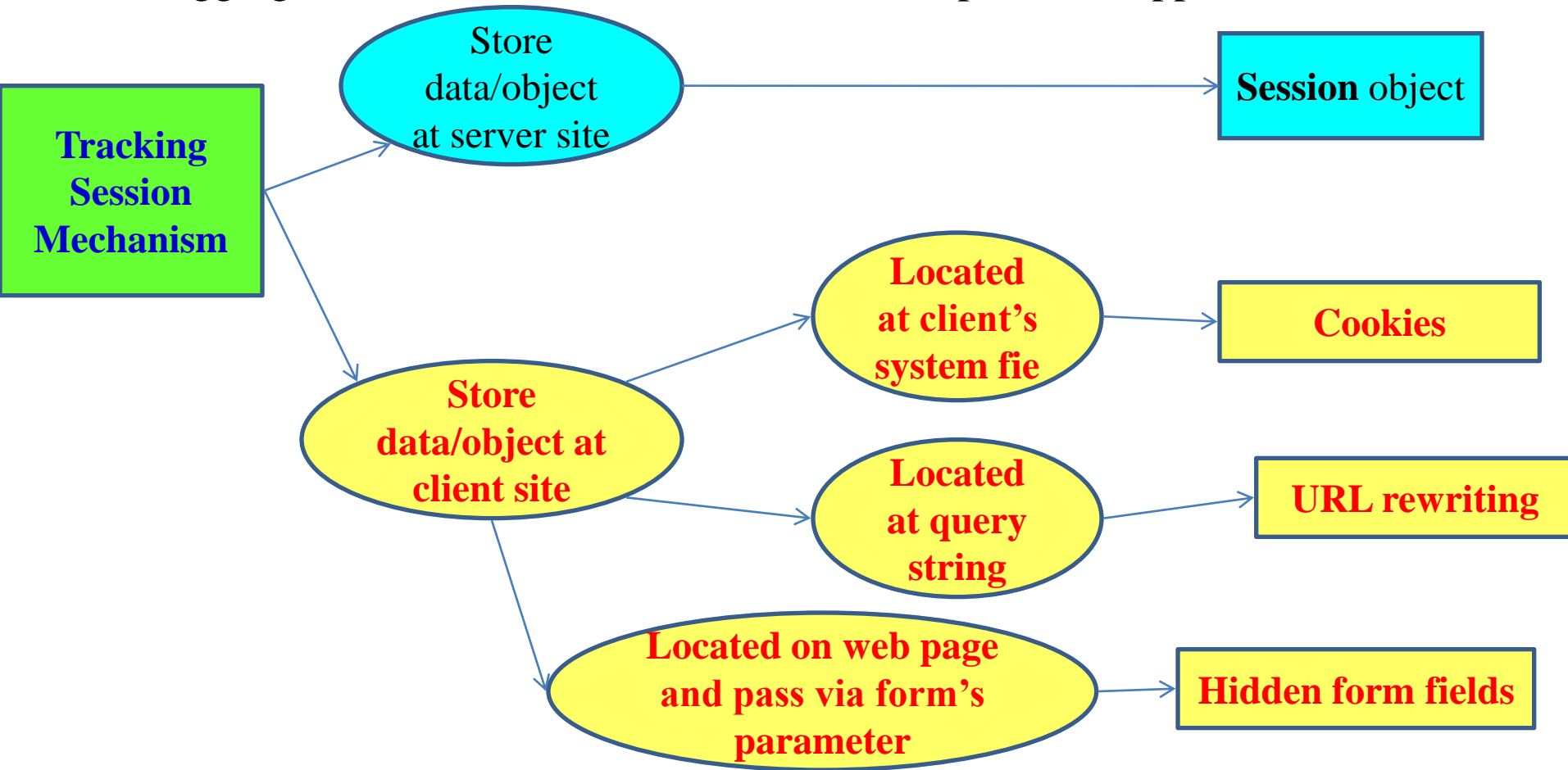
Security Mechanisms

Deployment Descriptor Security Declarations

Authentication Types

Review

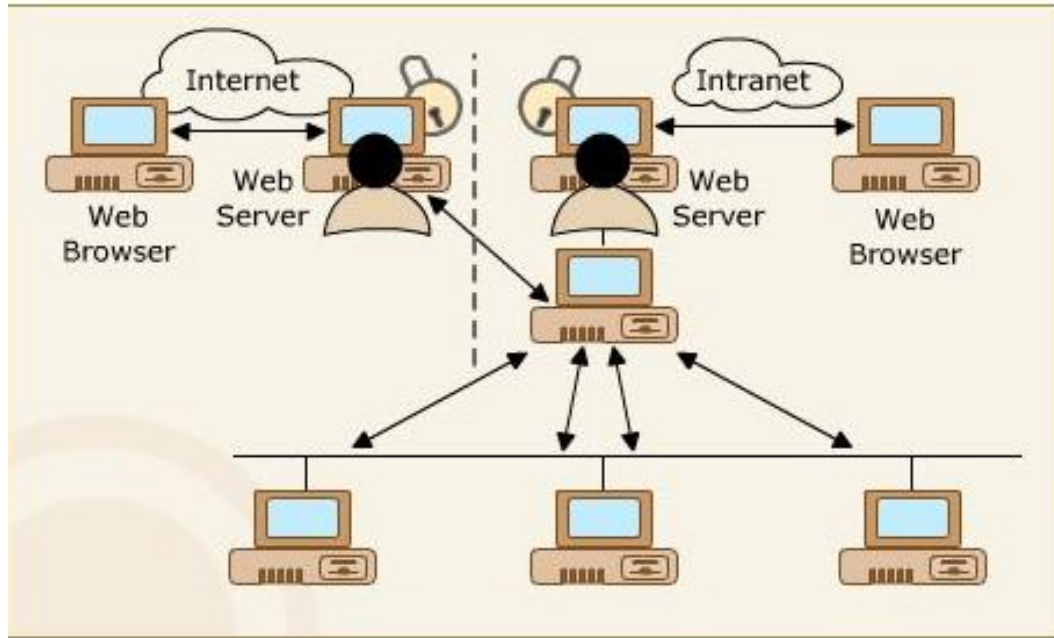
- **Session Tracking Mechanism**
 - Client must be stored the value that transfer to server in each its request
- **Error Handling**
 - Reporting Error: create the friendly UI to user when the system's errors occur.
 - Logging Error: store the errors to the file to improve the application



Objectives

- **Security Mechanisms**
 - Authentication
 - Authorization
 - Data Integrity
 - Confidentiality
- **Deployment Descriptor Security Declarations**
 - The <security-constraint> element
 - The <login-config> element
 - The <security-role> element
- **Authentication Types**
 - Basic Authentication
 - Digest Authentication
 - Form Authentication
 - CLIENT-CERT Authentication

Need of Securing Web Application



- Is accessed over a network such as Internet / Intranet
- Access to confidential information by unauthorized users
- Unauthorized use of resources
- Heavy traffic
- Malicious Code

Security Mechanisms

- There are 4 security mechanisms
 - **Authentication**
 - Ensures that the identity of the client is true
 - Can be achieved through basic means (user IDs and password) vs. complex means (digital certificates)
 - **Authorization** (access to controlled resources)
 - After the completion of authentication, permissions have to be granted to the user to perform all desired operations on resources. (Grant permissions to the correct user)
 - **Data integrity**
 - Is the process of ensuring that any messages passed through a network have not been tampered with in transit
 - **Confidentiality** (data privacy)
 - Goes one step further than data integrity by promising that the information in a message is available only to users authorized
 - The encryption process is used

Security Mechanisms



General User



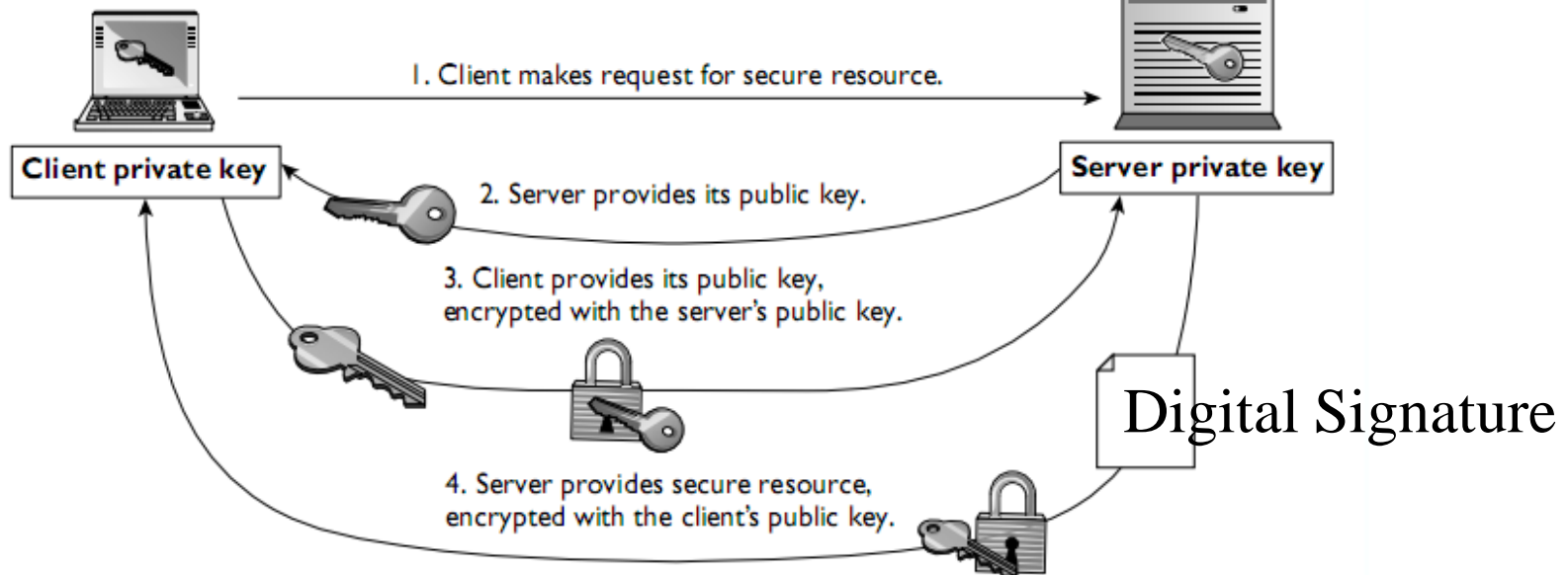
Logging In



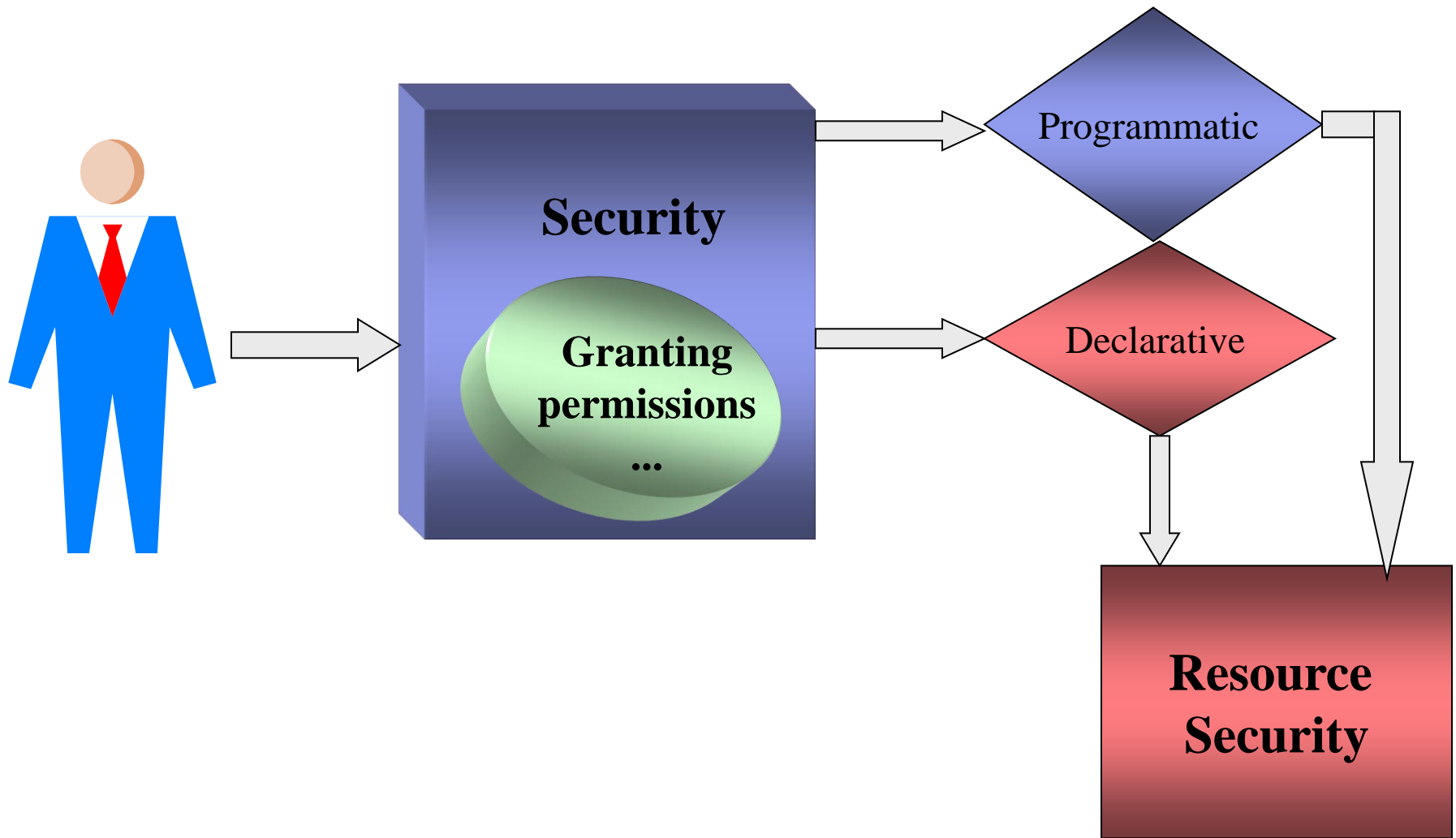
Authentication



Well-defined
security Role



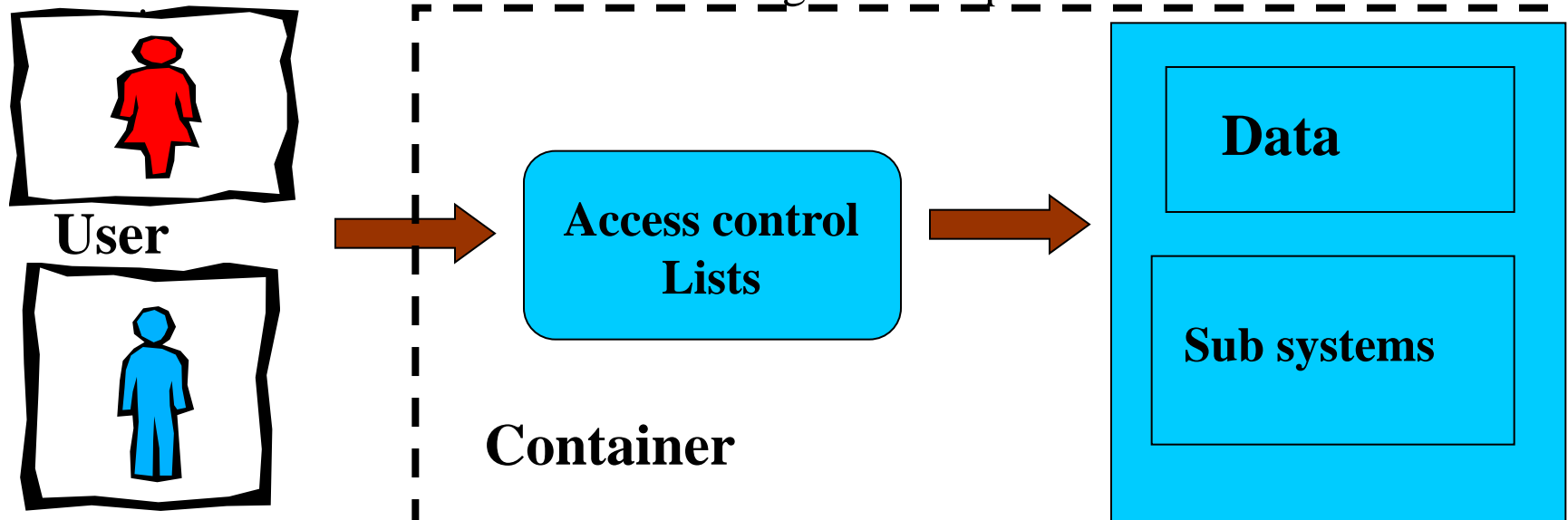
Security Mechanisms



Security Mechanisms

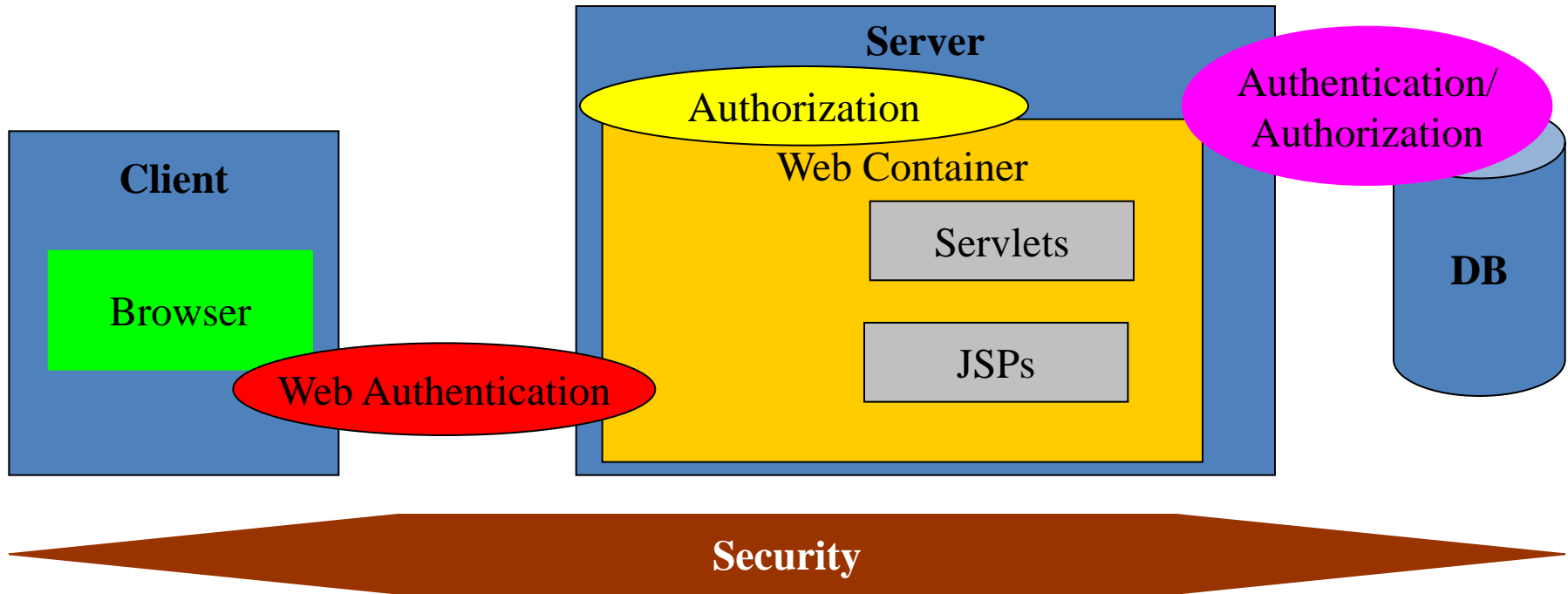
J2EE/JavaEE is applied JAAS

- Access Control List (ACL)
 - Permissions represent a right to access a particular resource or to perform some action on an application.
 - An administrator usually protects resources by creating lists of users and groups that have the permission to access a particular resource.
 - An ACL file is made up of entries, which contain a set of permissions for a particular resource and a set of users who can access those resources
 - Ex: In Web Tomcat Server, it is a tomcat-users.xml file
 - When user request any resources from the server, the container uses ACL to check the users' role for accessing the requested resource with his/her



Security Mechanisms

J2EE/JavaEE is applied JAAS



DD Security Declaration

The <security-constraint> element

<security-constraint>

<web-resource-collection>

<description>**description**</description>

<web-resource-name>**resourceName**</web-resource-name>

<url-pattern>**/resource**</url-pattern>

<http-method>**METHOD**</http-method>

</web-resource-collection>

<auth-constraint>

<description>**description**</description>

<role-name>**roleName**</role-name>

</auth-constraint>

<user-data-constraint>

<description>**description**</description>

<transport-guarantee>**NONE|INTEGRAL|CONFIDENTIAL**</transport-guarantee>

</user-data-constraint>

</security-constraint>

DD Security Declaration

The <security-constraint> element

- <security-constraint> is used to associate resources and HTTP methods with logical roles for
 - Authorization
 - Guaranteeing on resource security in transit
- <web-resource-collection> defines the resources to be secured
- <auth-constraint> lists the named roles authorized to the resources defined in the web resource collection. **If it is not exists, all clients can access the web resource without any permission**
- <user-data-constraint> serves to define guarantees on the network used to transmit resources to clients
 - NONE: no guarantee (is equivalent to omitting)
 - INTEGRAL: the web server must be able to detect any tampering with HTTP requests and responses for protected resources
 - CONFIDENTIAL: the web server must be ensure the content of HTTP requests and responses so that protected resources remain secret to all but authorized parties

The <security-constraint> element – Example

```
<security-constraint>
  <display-name>Constraint1</display-name>
  <web-resource-collection>
    <web-resource-name>AdminPage</web-resource-name>
    <description/>
    <url-pattern>/admin/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <description/>
    <role-name>manager</role-name>
  </auth-constraint>
</security-constraint>
```

The <security-constraint> element – Example

```
<security-constraint>
```

```
  <display-name>FormAuth</display-name>
```

```
  <web-resource-collection>
```

```
    <web-resource-name>FormResource</web-resource-name>
```

```
    <description/>
```

```
    <url-pattern>/*</url-pattern>
```

```
  </web-resource-collection>
```

```
  <auth-constraint>
```

```
    <description/>
```

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

```
  </auth-constraint>
```

```
  <user-data-constraint>
```

```
    <description/>
```

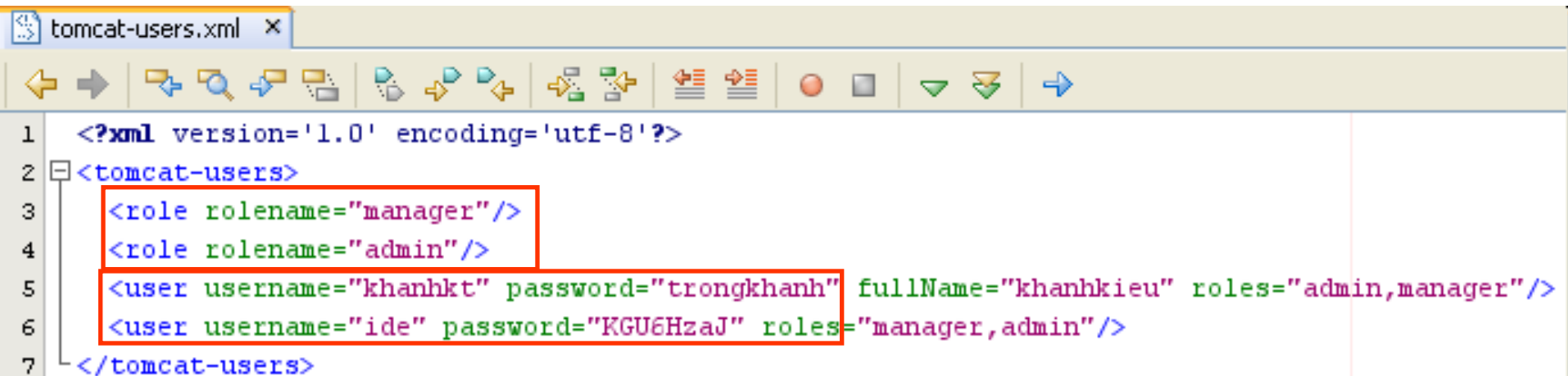
```
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
```

```
  </user-data-constraint>
```

```
</security-constraint>
```

DD Security Declaration

ACL on Tomcat Server – Example



```
1 <?xml version='1.0' encoding='utf-8'?>
2 <tomcat-users>
3   <role rolename="manager"/>
4   <role rolename="admin"/>
5   <user username="khanhkt" password="trongkhanh" fullName="khanhkieu" roles="admin,manager"/>
6   <user username="ide" password="KGU6HzaJ" roles="manager,admin"/>
7 </tomcat-users>
```

DD Security Declaration

The <login-config> element

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

```
<login-config>  
  <auth-method>FORM</auth-method>  
  <realm-name/>  
  <form-login-config>  
    <form-login-page>loginPage<form-login-page/>  
    <form-error-page>errorPage<form-error-page>  
  </form-login-config>  
</login-config>
```

```
<login-config>  
  <auth-method>DIGEST|CLIENT-CERT</auth-method>  
</login-config>
```

DD Security Declaration

The <login-config> element – Example

```
<security-constraint>
  <display-name>Constraint1</display-name>
  <web-resource-collection>
    <web-resource-name>BasicAuthentication</web-resource-name>
    <description/>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <description/>
    <role-name>manager</role-name>
  </auth-constraint>
</security-constraint>
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name/>
</login-config>
```


The <login-config> element – Example

```
<security-constraint>
  <display-name>FormAuth</display-name>
  <web-resource-collection>
    <web-resource-name>FormResource</web-resource-name>
    <description/>
    <url-pattern>/admin/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <description/>
    <role-name>manager</role-name>
  </auth-constraint>
</security-constraint>

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

DD Security Declaration

The <security-role> element

<security-role>

<description>**description**</description>

<role-name>**roleName**</role-name>

</security-role>

```
<security-role>
```

```
  <description/>
```

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

```
</security-role>
```

Additional

Configuring Tomcat

- Addition the username, password to the tomcat-users.xml file is located at

– C:\Documents and Settings\userNames\.netbeans\6.9\apache-tomcat-6.0.26_base\conf\tomcat-users.xml

– C:\Users\userNames\.netbeans\6.9\apache-tomcat-6.0.26_base\conf\tomcat-users.xml

- Ex:



```

19 <!--
20 NOTE: By default, no user is included in the "manager" role required
21 to operate the "/manager" web application. If you wish to use this app,
22 you must include such a user in the file. User and role names are arbitrary.
23 -->
24 <!--
25 NOTE: The sample user and role entries below are wrapped in a comment
26 and thus are ignored when reading this file. Do not forget to remove
27 <!-- ... --> that surrounds them.
28 -->
29 <role rolename="manager"/>
30 <role rolename="admin"/>
31 <role rolename="tomcat"/>
32 <role rolename="role1"/>
33 <user username="tomcat" password="tomcat" roles="tomcat"/>
34 <user username="both" password="tomcat" roles="tomcat,role1"/>
35 <user username="role1" password="tomcat" roles="role1"/>
36 <user username="ide" password="Wfnn2yCD" roles="manager,admin"/>
37 <user username="khanhkt" password="trongkhanh" roles="manager,admin"/>
38 </tomcat-users>
  
```

Authentication Types

BASIC

- The user sends request to access or request the web resource
- The **container uses ACL to authenticate the user permission when the secure web resource is accessed**
- The container **response that contains the authentication request attached in header response**
- Depending the response header, the **browser is triggered to show a standard dialog**. This dialog **allows entry of a username and password**, and displays a **realm name**
- When the **OK button** is pressed, the form “**username:password**” is processed on Base64 encoding and transmitted over the Internet/network to the container
- The container **checks the authenticated information on ACL**
 - If the user does **not exists** in ACL, the container **send error code 401**
 - If the **user exists** in ACL but he/she does **not permit to access resource**, the container **send error code 403 (access permission denied)**
 - **Otherwise**, the user can **access the requested web resource**

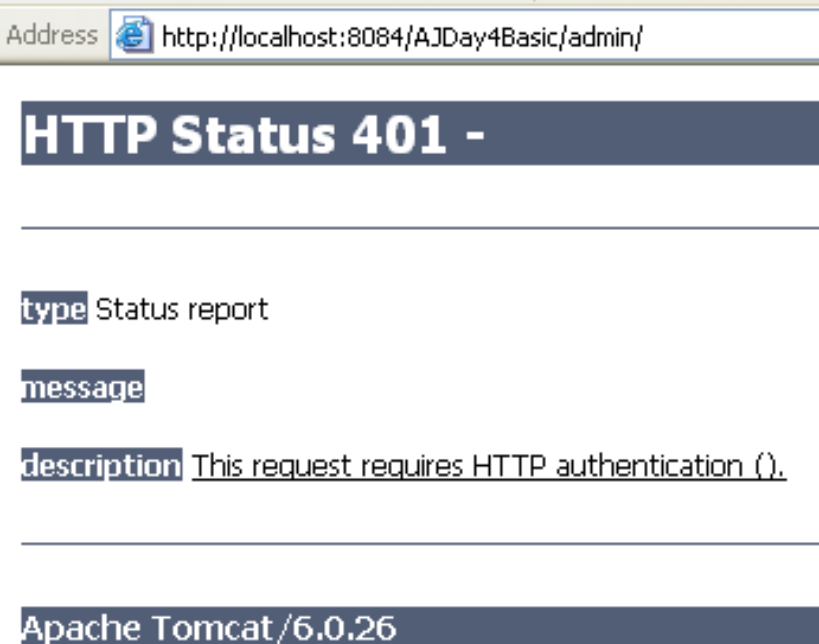
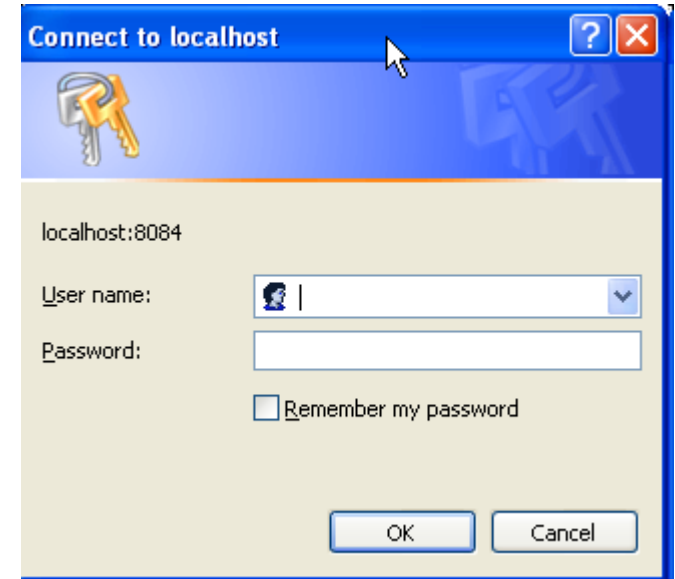
Authentication Types

BASIC

- **The realm**
 - Is the **registry** used to **store user account information**
 - The **authentication and authorization information** and the **enforcement policies**, as defined by an administrator.
 - Are typically **repositories** containing users, groups, permissions, and secured resources.
 - **Contains** the usernames and passwords and the authentication and authorization **policy details** that control these users.
 - Can be **stored** in an **XML file**, a **text file**, and sometimes **even** in a **database**
- **Advantages**
 - Browser **friendly**
- **Disadvantages**
 - It **does not provide any protection** for the information communicated between the client and the server.

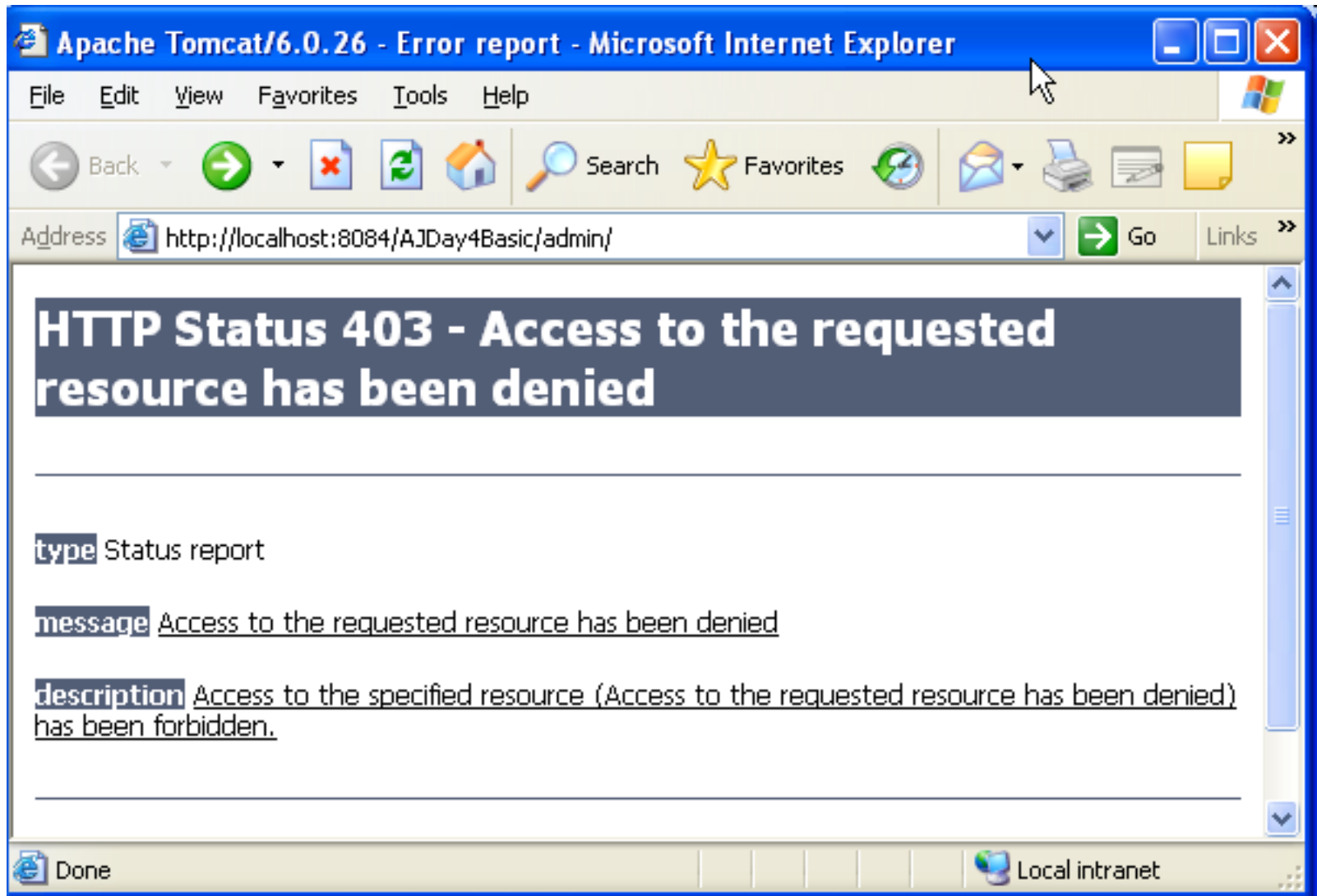
Authentication Types

BASIC



Authentication Types

BASIC



Authentication Types

BASIC

web.xml x

General Servlets Filters Pages References Security XML Login Configuration v

Login Configuration

☐ None
☐ Digest
☐ Client Certificate
☒ Basic
☐ Form

Form Login Page:

Form Error Page:

Realm Name:

Add Security Role [X]

Role Name:

Description:

OK Cancel

web.xml x

General Servlets Filters Pages References Security XML Security Roles v

Login Configuration

Security Roles

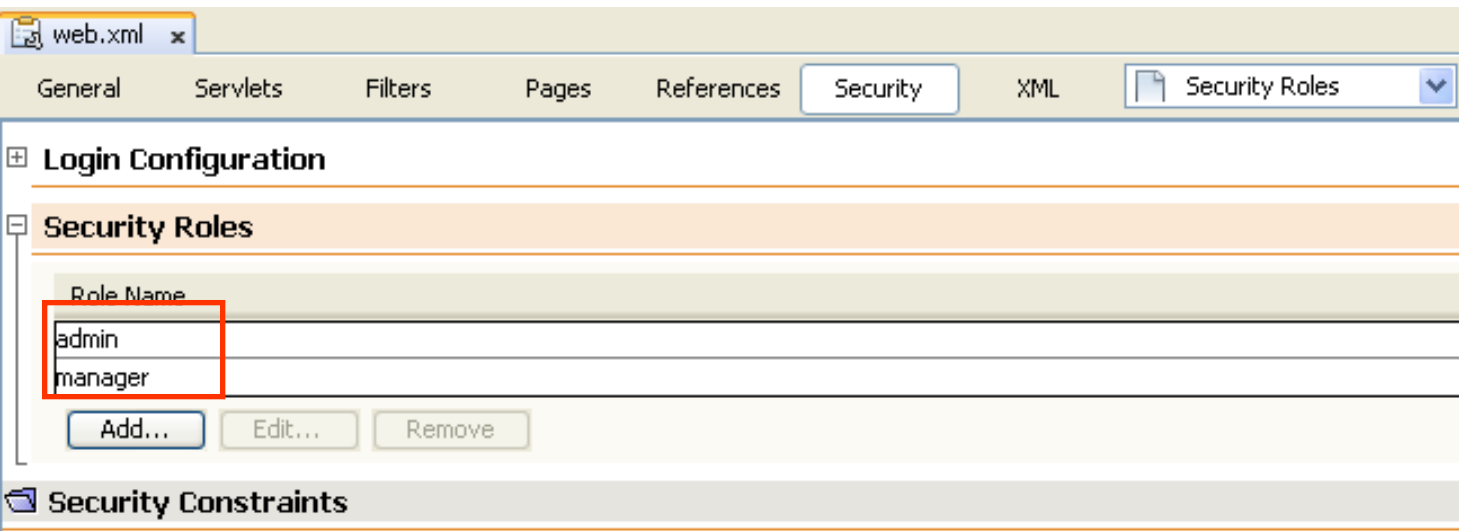
Role Name

Add... Edit... Remove

Security Constraints

Authentication Types

BASIC



web.xml x

General Servlets Filters Pages References Security XML Security Roles

+ Login Configuration

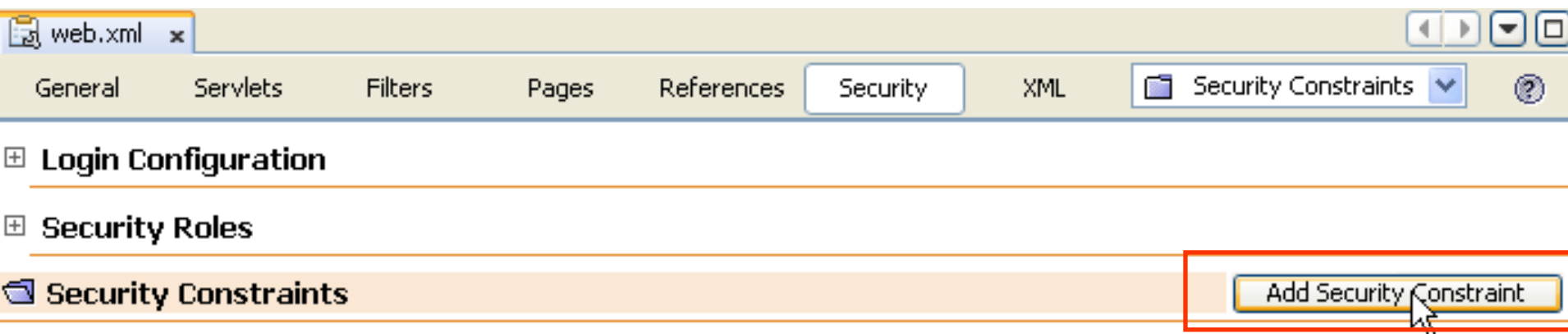
- Security Roles

Role Name

admin

manager

Add... Edit... Remove



web.xml x

General Servlets Filters Pages References Security XML Security Constraints

+ Login Configuration

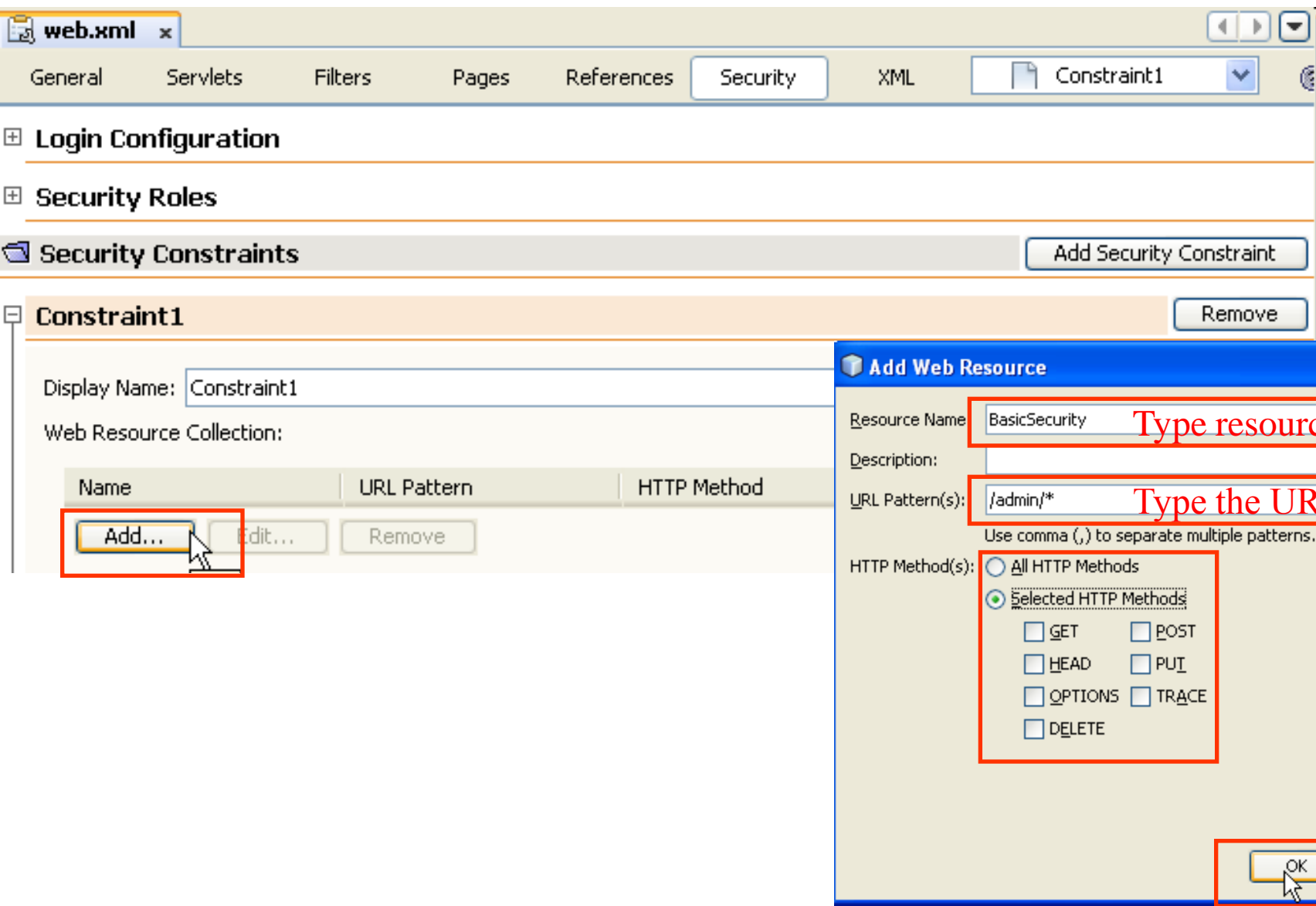
+ Security Roles

- Security Constraints

Add Security Constraint

Authentication Types

BASIC



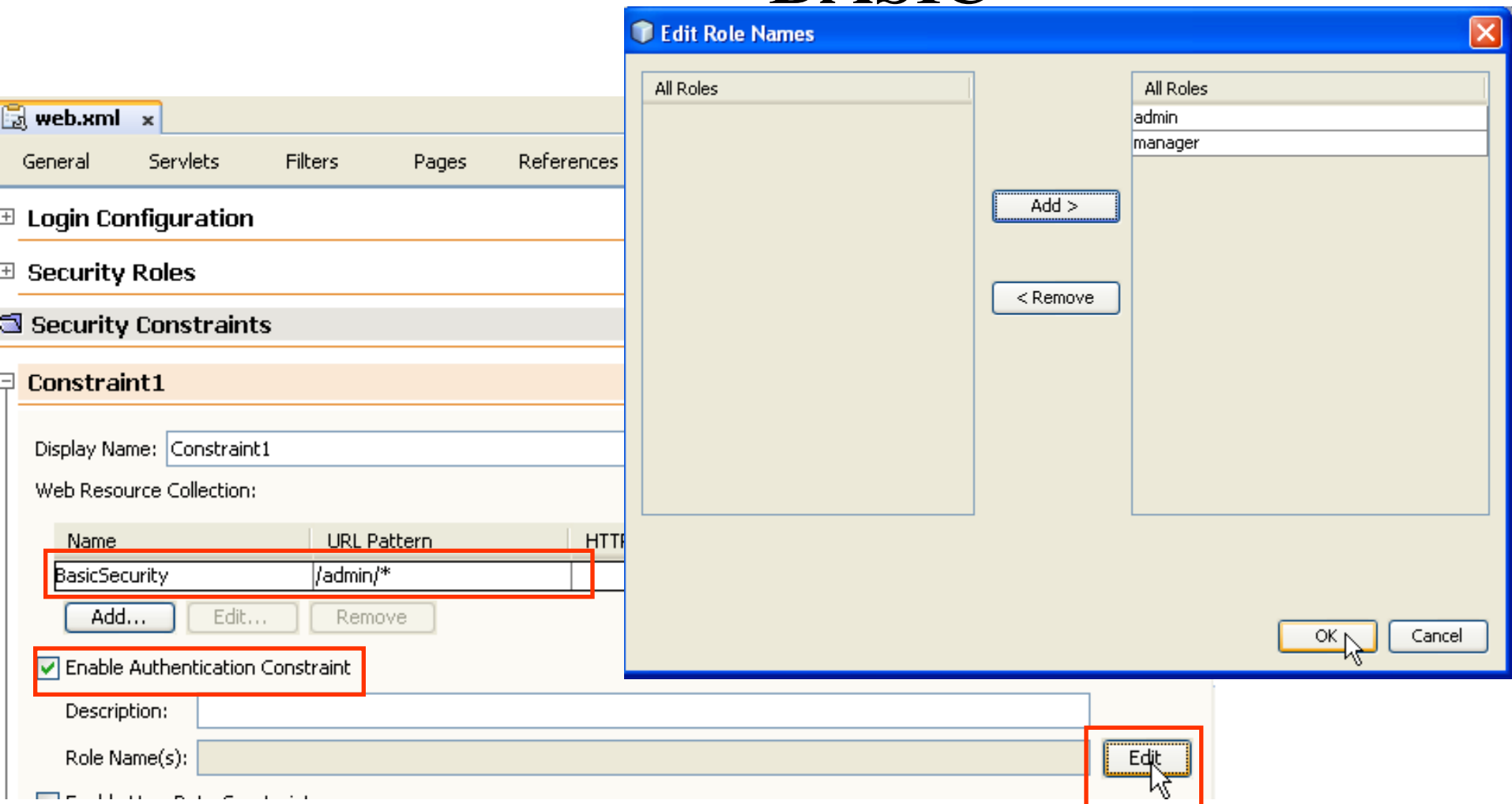
The screenshot shows the Eclipse IDE interface with the **web.xml** file open. The **Security** tab is selected, showing the **Constraint1** configuration. The **Display Name** is **Constraint1** and the **Web Resource Collection** is empty. The **Add...** button in the **Web Resource Collection** table is highlighted with a red box.

The **Add Web Resource** dialog is open, showing the following fields:

- Resource Name:** **BasicSecurity** (highlighted with a red box and the text "Type resource name")
- Description:** (empty)
- URL Pattern(s):** **/admin/*** (highlighted with a red box and the text "Type the URL /.../*")
- HTTP Method(s):** **Selected HTTP Methods** (selected, highlighted with a red box)
- HTTP Method(s) options:**
 - ☐ GET
 - ☐ POST
 - ☐ HEAD
 - ☐ PUT
 - ☐ OPTIONS
 - ☐ TRACE
 - ☐ DELETE
- OK** button (highlighted with a red box)
- Cancel** button

Authentication Types

BASIC



The screenshot displays a web application configuration interface with a tabbed menu (General, Servlets, Filters, Pages, References) and a 'web.xml' tab. The 'Login Configuration' section is expanded, showing 'Security Roles' and 'Security Constraints'. Under 'Security Constraints', 'Constraint1' is selected. Its configuration includes a 'Display Name' of 'Constraint1' and a 'Web Resource Collection' table with one entry: 'BasicSecurity' with a 'URL Pattern' of '/admin/*'. The 'Enable Authentication Constraint' checkbox is checked. An 'Edit' button is visible at the bottom right of the configuration area.

An 'Edit Role Names' dialog box is open, showing two 'All Roles' lists. The left list is empty, and the right list contains 'admin' and 'manager'. Between the lists are 'Add >' and '< Remove' buttons. 'OK' and 'Cancel' buttons are at the bottom right of the dialog.

Name	URL Pattern	HTTP
BasicSecurity	/admin/*	

Authentication Types

BASIC

web.xml

General
Servlets
Filters
Pages
References
Security
XML
Constraint1

+ Login Configuration

+ Security Roles

Security Constraints
Add Security Constraint

Constraint1
Remove

Display Name: Constraint1

Web Resource Collection:

Name	URL Pattern	HTTP Method	Description
BasicSecurity	/admin/*		

Add...
Edit...
Remove

☒ Enable Authentication Constraint

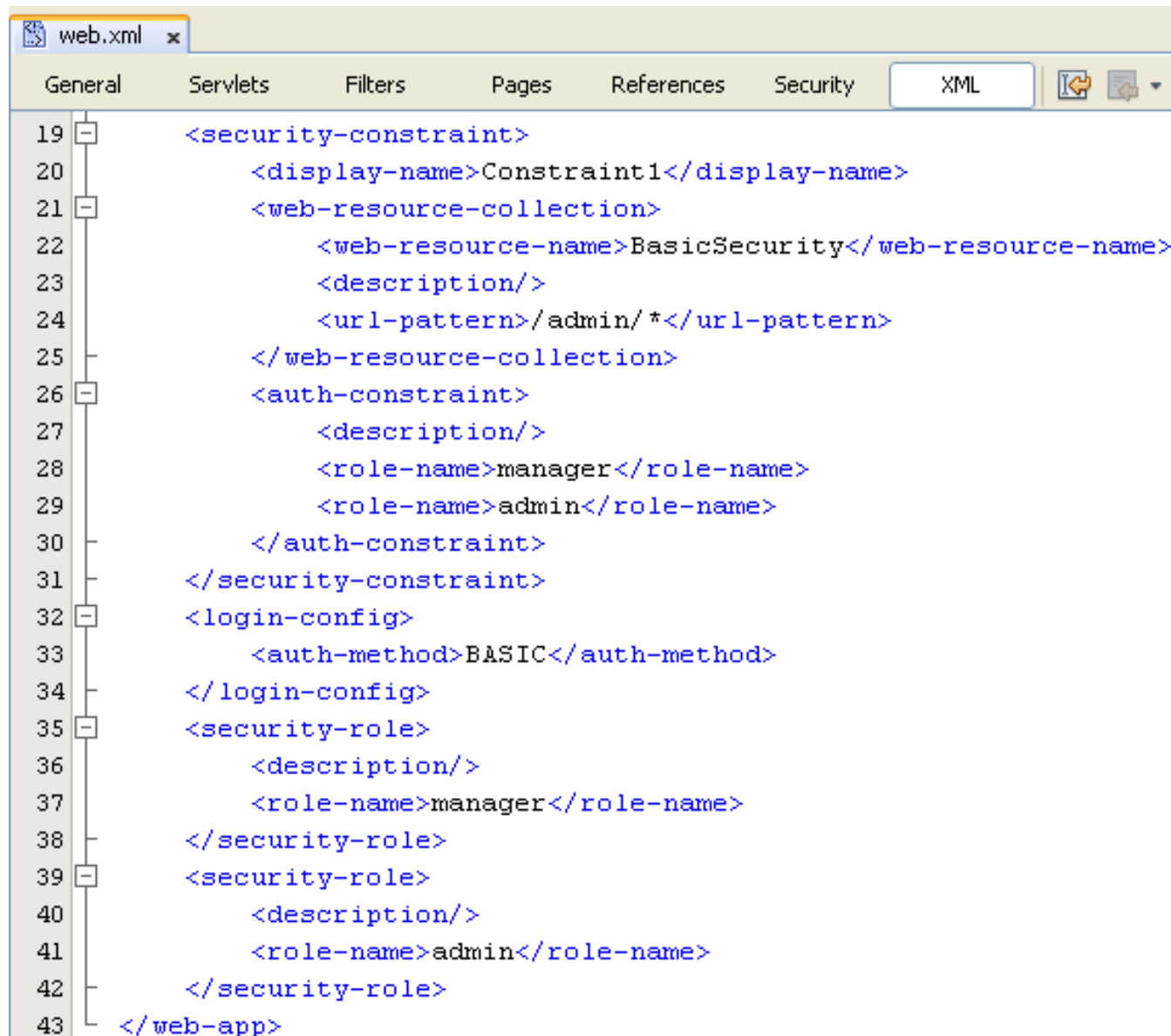
Description:

Role Name(s): manager, admin
Edit

☐ Enable User Data Constraint

Authentication Types

BASIC

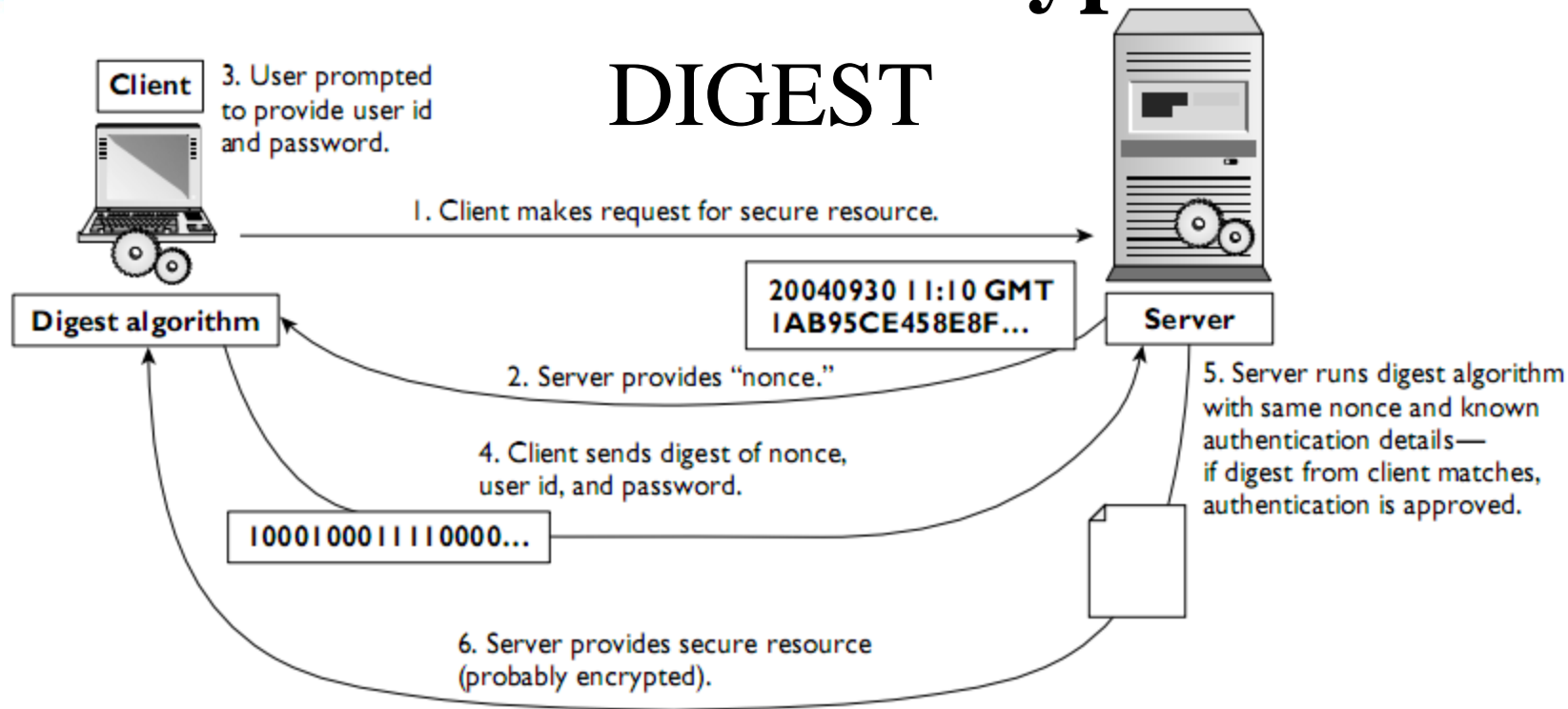


```

19  <security-constraint>
20      <display-name>Constraint1</display-name>
21  <web-resource-collection>
22      <web-resource-name>BasicSecurity</web-resource-name>
23      <description/>
24      <url-pattern>/admin/*</url-pattern>
25  </web-resource-collection>
26  <auth-constraint>
27      <description/>
28      <role-name>manager</role-name>
29      <role-name>admin</role-name>
30  </auth-constraint>
31 </security-constraint>
32 <login-config>
33     <auth-method>BASIC</auth-method>
34 </login-config>
35 <security-role>
36     <description/>
37     <role-name>manager</role-name>
38 </security-role>
39 <security-role>
40     <description/>
41     <role-name>admin</role-name>
42 </security-role>
43 </web-app>
  
```

Authentication Types

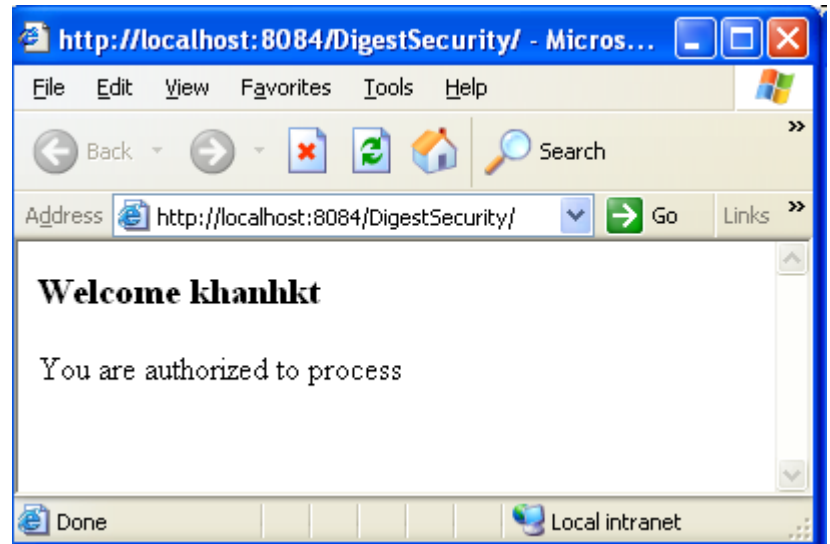
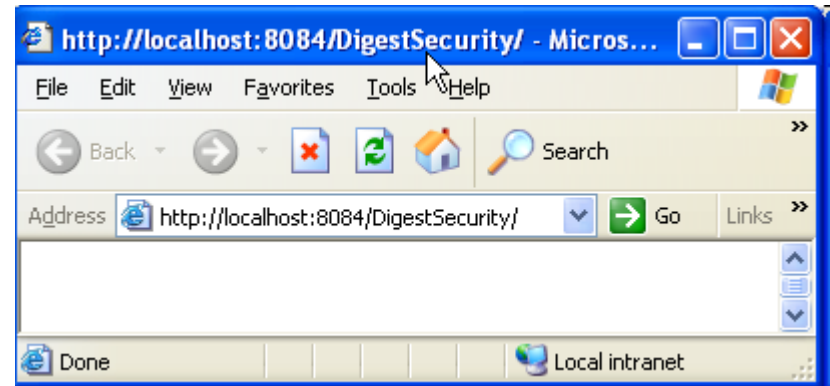
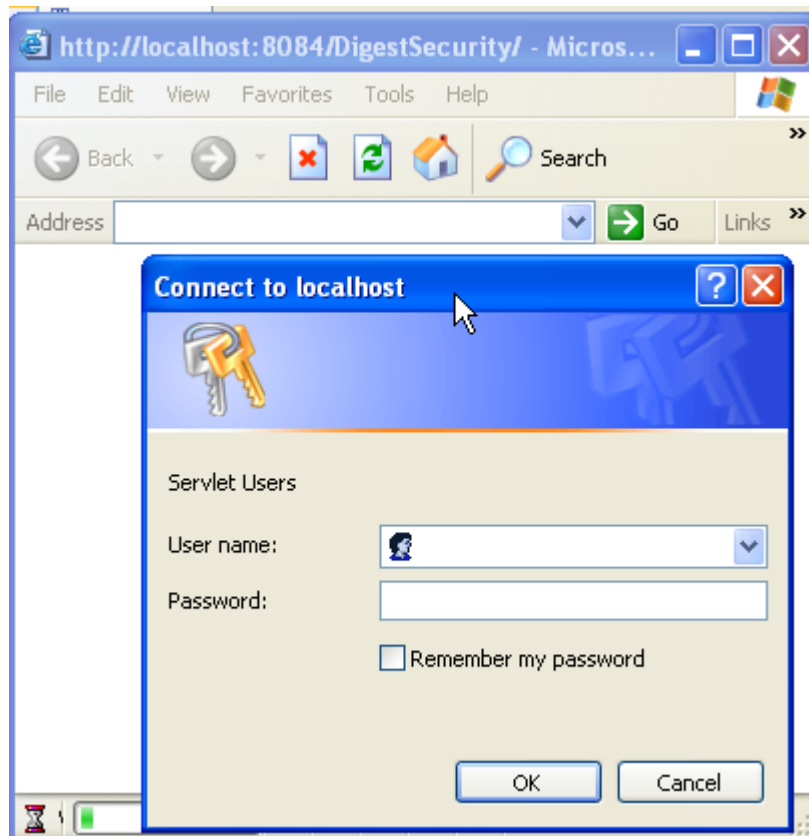
DIGEST



- Improves a little on BASIC by using a secure algorithm to encrypt the password and other security details
- Builds over the basic authentication but the password is first encrypted and then sent
- Use hash functions to secure web applications
- Hash function convert data into a small / complex no.
- A one-way system in process because it is difficult to know the original password according to the output
- Advantages: impossible hacking

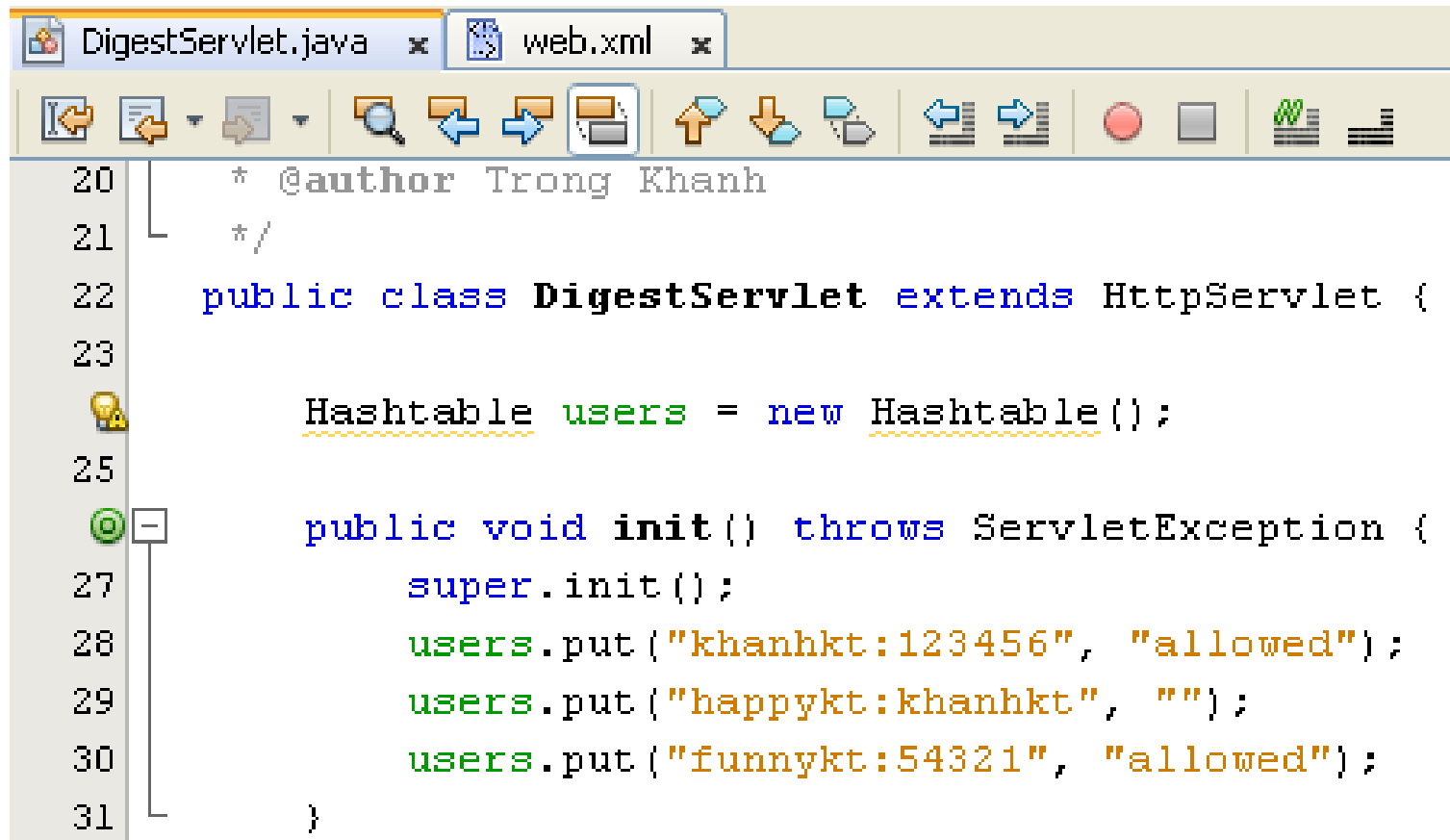
Authentication Types

DIGEST



Authentication Types

DIGEST



```
20  * @author Trong Khanh
21  */
22  public class DigestServlet extends HttpServlet {
23
24      Hashtable users = new Hashtable();
25
26      public void init() throws ServletException {
27          super.init();
28          users.put("khanhkt:123456", "allowed");
29          users.put("happykt:khanhkt", "");
30          users.put("funnykt:54321", "allowed");
31      }
```


Authentication Types

DIGEST

```
40 protected void processRequest(HttpServletRequest request, HttpServletResponse response)
41     throws ServletException, IOException {
42     response.setContentType("text/html;charset=UTF-8");
43     PrintWriter out = response.getWriter();
44     try {
45         String username = null;
46         boolean valid = false;
47         String authHeader = request.getHeader("Authorization");
48         if (authHeader != null) {
49             StringTokenizer st = new StringTokenizer(authHeader);
50             String basic = st.nextToken();
51             if (basic.equalsIgnoreCase("Basic")) {
52                 String credential = st.nextToken();
53                 BASE64Decoder decoder = new BASE64Decoder();
54                 String userPass = new String(decoder.decodeBuffer(credential));
55                 int p = userPass.indexOf(":");
56                 username = userPass.substring(0, p);
57                 valid = users.containsKey(userPass)
58                     && users.get(userPass).equals("allowed");
59             }
60         }
61         if (!valid) {
62             String s = "Basic realm=\"Servlet Users\"";
63             response.setHeader("WWW-Authenticate", s);
64             response.setStatus(401);
65         } else {
66             out.println("<h3> Welcome " + username + "</h3>");
67             out.println("You are authorized to process");
68         }
69     } finally {
```

Authentication Types

DIGEST

web.xml x

General Servlets Filters Pages References **Security**

Login Configuration


☐ None
☒ Digest
☐ Client Certificate
☐ Basic
☐ Form

Form Login Page:

Form Error Page:

Realm Name:

web.xml x

General Servlets Filters Pages References **Security** XML  Security Roles

Login Configuration

Security Roles

Security Constraints

Constraint1

Display Name:

Web Resource Collection:

Name	URL Pattern	H
DigestAuthen	/DigestServlet	

☐ Enable Authentication Constraint

Description:

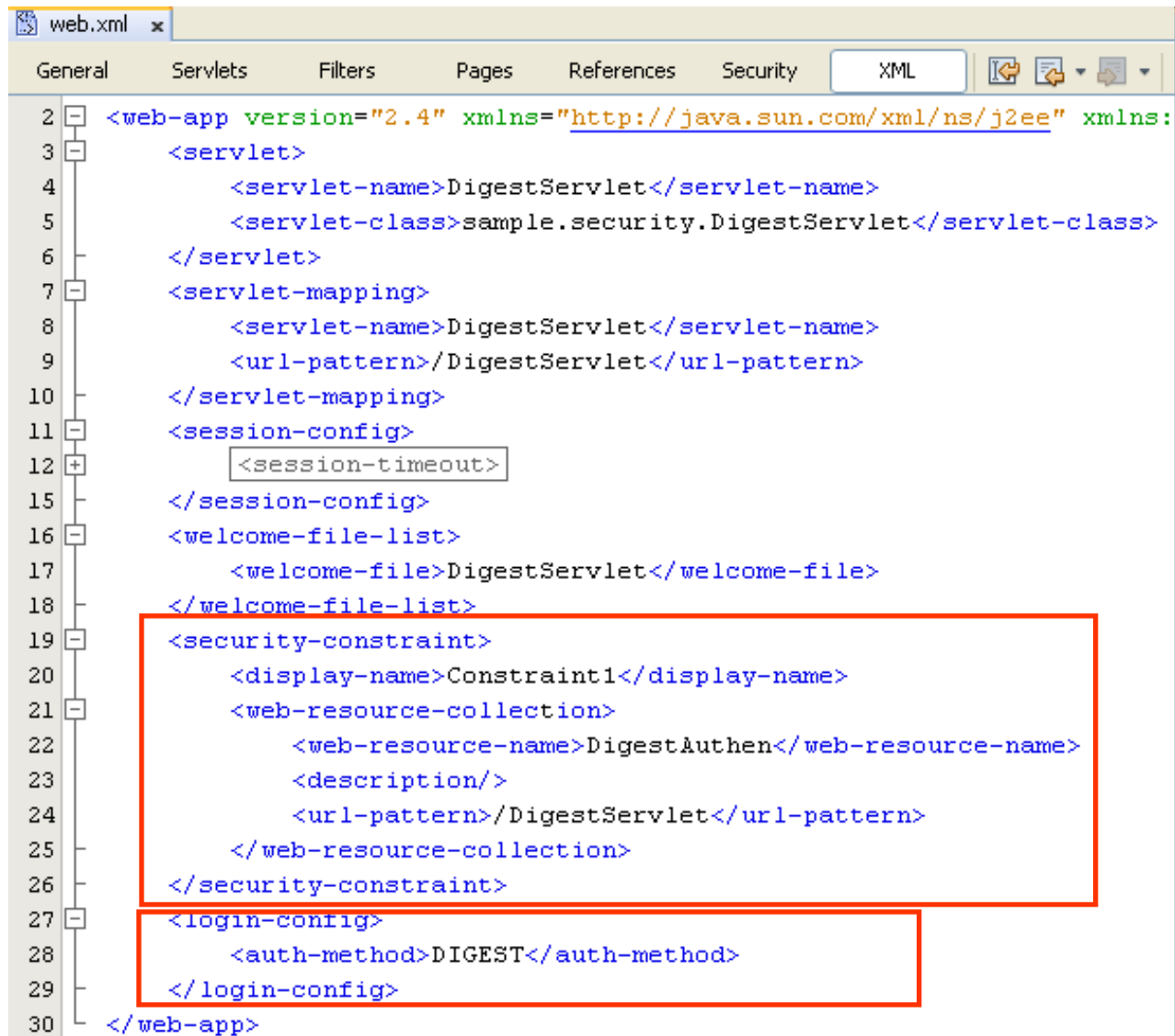
Role Name(s):

☐ Enable User Data Constraint

Description:

Authentication Types

DIGEST



```

web.xml x
General  Servlets  Filters  Pages  References  Security  XML
2  <web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:
3      <servlet>
4          <servlet-name>DigestServlet</servlet-name>
5          <servlet-class>sample.security.DigestServlet</servlet-class>
6      </servlet>
7  <servlet-mapping>
8      <servlet-name>DigestServlet</servlet-name>
9      <url-pattern>/DigestServlet</url-pattern>
10 </servlet-mapping>
11 <session-config>
12     <session-timeout>
15 </session-config>
16 <welcome-file-list>
17     <welcome-file>DigestServlet</welcome-file>
18 </welcome-file-list>
19 <security-constraint>
20     <display-name>Constraint1</display-name>
21     <web-resource-collection>
22         <web-resource-name>DigestAuthen</web-resource-name>
23         <description/>
24         <url-pattern>/DigestServlet</url-pattern>
25     </web-resource-collection>
26 </security-constraint>
27 <login-config>
28     <auth-method>DIGEST</auth-method>
29 </login-config>
30 </web-app>
  
```

Authentication Types

FORM

- Associates a custom web page with the login process, as an **alternative to a browser dialog**
- There are only a **few rules**
 - The HTML form **must use the POST method**
 - The form must have “**j_security_check**” as its **action**
 - The form must include an input-capable **field for user called “j_username”**
 - The form must also include an input capable **field for password called “j_password”**
 - The form-based authentication is **required an error page**
- The **mechanism**
 - When the web page is required, the server **caches the URL** that tries to reach and **redirects to the form login pages**
 - The username and password is provided; assuming that the server is happy with these credentials, the required URL is passed
 - If the login fails, the server redirects to the error page

Authentication Types

FORM – Example



Address  http://localhost:8084/AJDay4Form/

Welcome to Form Security

[Click here to go to Admin Page](#)

Address  http://localhost:8084/AJDay4Form/admin/index.html

Form Login Page

Username

Password

 http://localhost:8084/AJDay4Form/admin/index.html

```
<body>
  <h1>Form-based Authentication</h1>
  <form action="j_security_check" method="POST">
    Username <input type="text" name="j_username" value="" /><br/>
    Password <input type="password" name="j_password" value="" /><br/>
    <input type="submit" value="Login" />
  </form>
</body>
```

Authentication Types

FORM – Example



Welcome to Admin Page

FORM – Example

```
<security-constraint>
  <display-name>Constraint1</display-name>
  <web-resource-collection>
    <web-resource-name>Form Auth</web-resource-name>
    <description/>
    <url-pattern>/admin/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <description/>
    <role-name>manager</role-name>
  </auth-constraint>
</security-constraint>

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

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

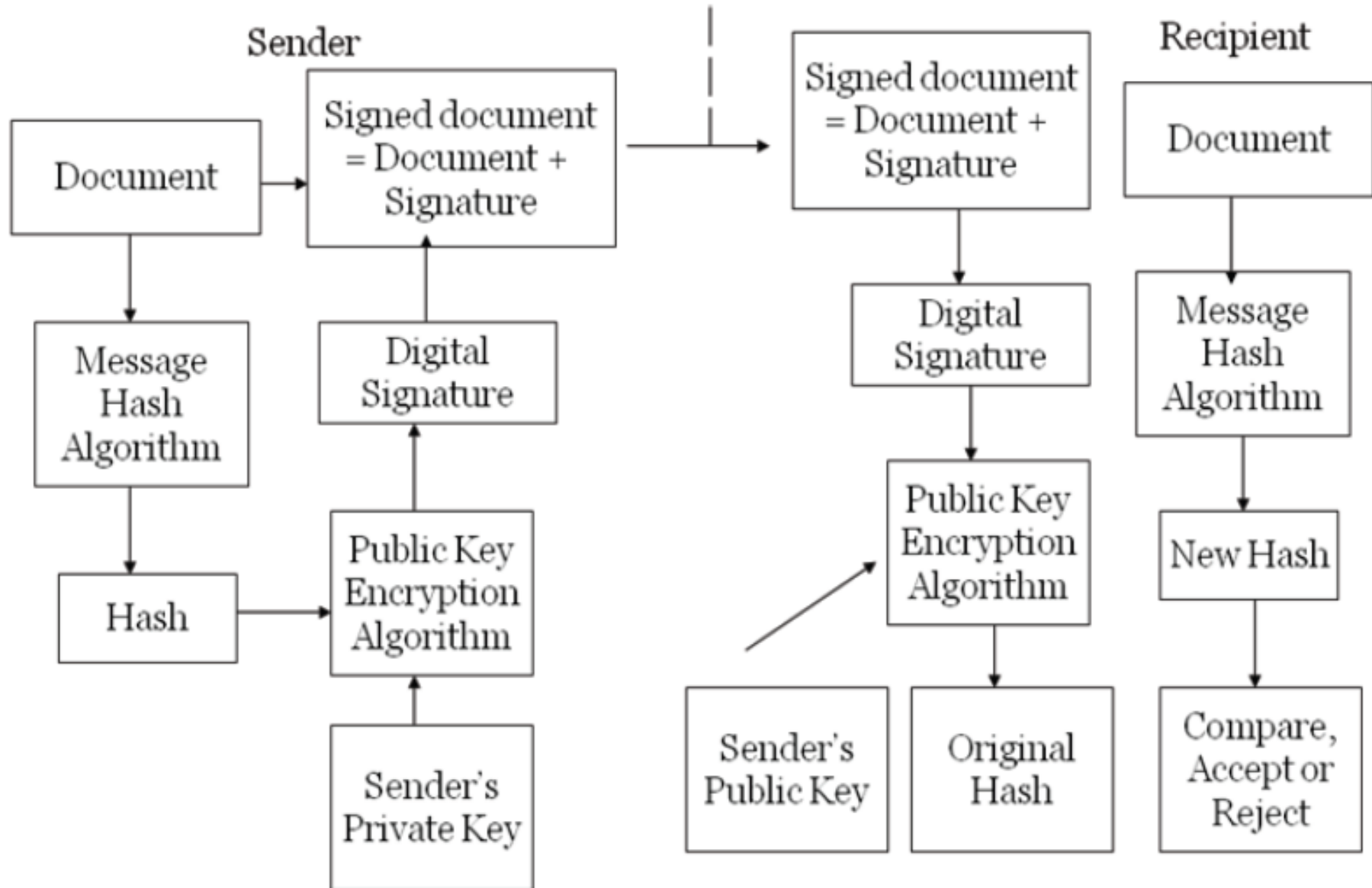
Authentication Types

CLIENT-CERT

- Uses **digital certificates** to achieve authentication
- Relies on **asymmetric keys** (public and private keys). Anything **encrypted with the public key can be decrypted with the private key and vice versa**
- The certificate is **generated by using specialized software** such as “**keytool**”
- **Mechanism**
 - The client, first **generates** a private and public key
 - The client **sends** the public key – and other information to a third-party certificate authority
 - The certificate authority **binds** this information and the client public key into a certificate
 - The certificate authority **adds** a digital encrypted with its private key, which makes a digest of information already in the certificate
 - The certificate is returned to the client, who installs it in his/her browser. When the server requested authentication from client browser, the browser supplies the certificate to server approving

Authentication Types

CLIENT-CERT – Mechanism



Authentication Types

HTTPS CLIENT

- Is a **secured client authentication** technique, which is based on **Public Key Certificates**.
- Authentication of users by **establishing a Secure Sockets Layer (SSL) connection** between sender and recipient
 - Sender – SSL Client
 - Recipient – SSL server
- **Extra authentication layer** in between HTTP and TCP
- This layer **confirms the client authentication**
- **Two kinds** of Certificated are used
 - **Server Certificates:** Contain information about server that allows a client to identify the server before sharing sensitive information
 - **Client Certificates:** Contains personal information about the user and introduces the SSL client to the server

Authentication Types

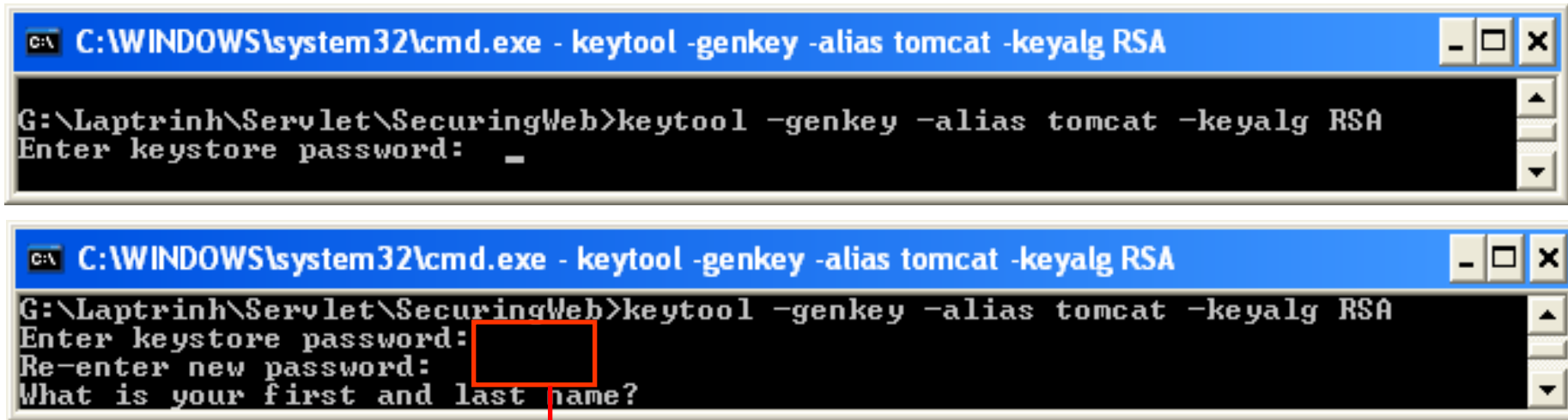
HTTPS CLIENT

- Steps in processing
 - **Step 1:** Generates Key (The key file (*.keystore) should be copied to **C:\Documents and Setting\username** or **C:\Users\username**)
 - **Step 2:** Configuring the server configuration files at Web Server
 - **Step 3:** Configuring the application deployment descriptor file (web.xml)
 - **Step 4:** Restart the Web server and execute the application

Authentication Types

HTTPS CLIENT

- **Step 1: Generates Key**
 - Using the key generator tool creates a keystore to store the keys generated as
`keytool -genkey -alias privatekeyName -keyalg EncodingAlgorithms`



```
C:\WINDOWS\system32\cmd.exe - keytool -genkey -alias tomcat -keyalg RSA

G:\Laptrinh\Servlet\SecuringWeb>keytool -genkey -alias tomcat -keyalg RSA
Enter keystore password: _

C:\WINDOWS\system32\cmd.exe - keytool -genkey -alias tomcat -keyalg RSA

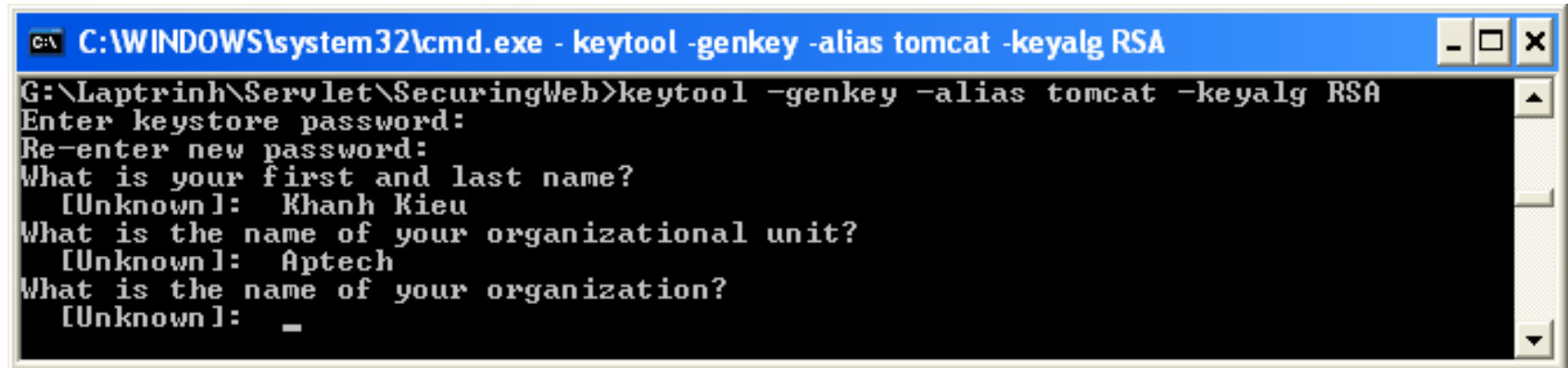
G:\Laptrinh\Servlet\SecuringWeb>keytool -genkey -alias tomcat -keyalg RSA
Enter keystore password: 
Re-enter new password: 
What is your first and last name?
```

Password and re-enter password can not be seen when typing (high security)

Authentication Types

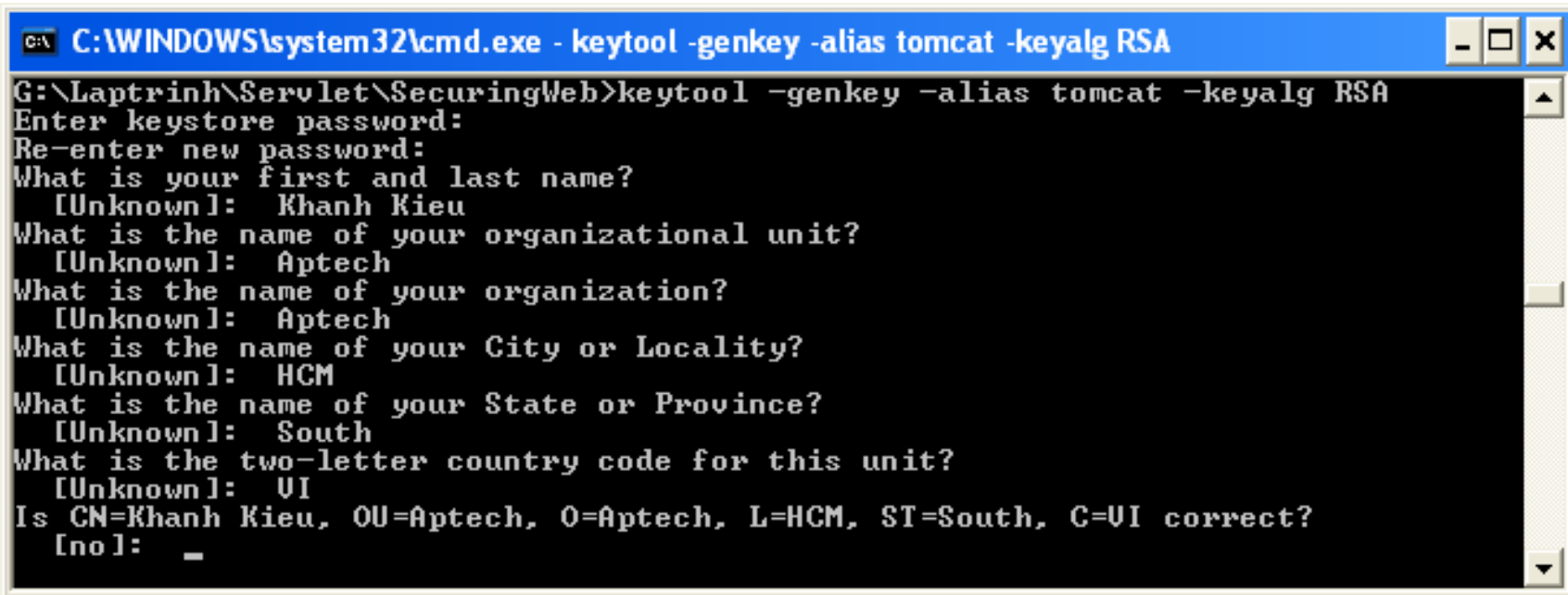
HTTPS CLIENT

- Step 1: Generates Key (cont)



```
C:\WINDOWS\system32\cmd.exe - keytool -genkey -alias tomcat -keyalg RSA

G:\Laptrinh\Servlet\SecuringWeb>keytool -genkey -alias tomcat -keyalg RSA
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  Khanh Kieu
What is the name of your organizational unit?
  [Unknown]:  Aptech
What is the name of your organization?
  [Unknown]:  _
```



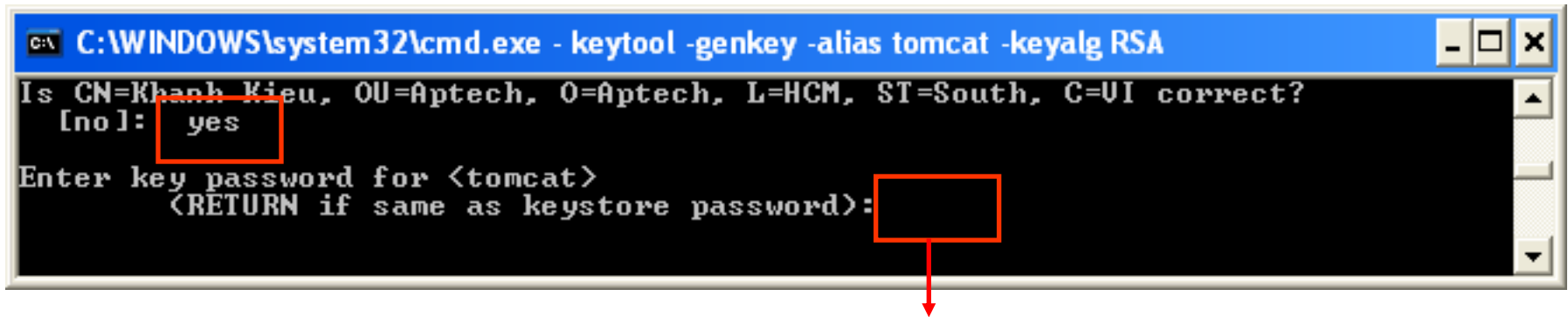
```
C:\WINDOWS\system32\cmd.exe - keytool -genkey -alias tomcat -keyalg RSA

G:\Laptrinh\Servlet\SecuringWeb>keytool -genkey -alias tomcat -keyalg RSA
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  Khanh Kieu
What is the name of your organizational unit?
  [Unknown]:  Aptech
What is the name of your organization?
  [Unknown]:  Aptech
What is the name of your City or Locality?
  [Unknown]:  HCM
What is the name of your State or Province?
  [Unknown]:  South
What is the two-letter country code for this unit?
  [Unknown]:  VI
Is CN=Khanh Kieu, OU=Aptech, O=Aptech, L=HCM, ST=South, C=VI correct?
[no]:  _
```

Authentication Types

HTTPS CLIENT

- **Step 1: Generates Key (cont)**



```
C:\WINDOWS\system32\cmd.exe - keytool -genkey -alias tomcat -keyalg RSA
Is CN=Khánh Kieu, OU=Aptech, O=Aptech, L=HCM, ST=South, C=VI correct?
[no]: yes
Enter key password for <tomcat>
(RETURN if same as keystore password):
```

Should be pressed Enter

- The .keystore file is created at the directory
 - C:\Documents and Setting\username\.keystore
 - C:\Users\username\.keystore

Authentication Types

HTTPS CLIENT – Step 2

- Go to the
 - C:\Documents and Settings\currentUser\.netbeans\6.9\apache-tomcat-6.0.26_base\conf\server.xml
 - C:\Users\currentUser\.netbeans\6.9\apache-tomcat-6.0.26_base\conf\server.xml
 - Or right click on the Server in tab Runtime, click edit server.xml
- Uncomment the `<!-- ... -->` as

```
<Connector protocol="org.apache.coyote.http11.Http11Protocol" port="8443"
maxHttpHeaderSize="8192" SSLEnabled="true"
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" disableUploadTimeout="true"
acceptCount="100" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS" keystorePass="trongkhanh" />
```

Search key SSL on the server.xml file

Keystore
password

Authentication Types

HTTPS CLIENT – Step 3

☒ Enable User Data Constraint

Description:

Transport Guarantee:

CONFIDENTIAL ▼

```
<security-constraint>
```

```
  <display-name>Constraint1</display-name>
```

```
  <web-resource-collection>
```

```
    <auth-constraint>
```

```
      <user-data-constraint>
```

```
        <description/>
```

```
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
```

```
      </user-data-constraint>
```

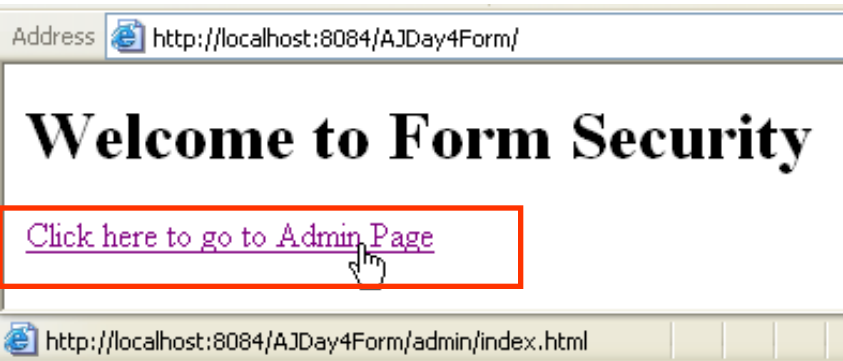
```
</security-constraint>
```

```
<login-config>
```

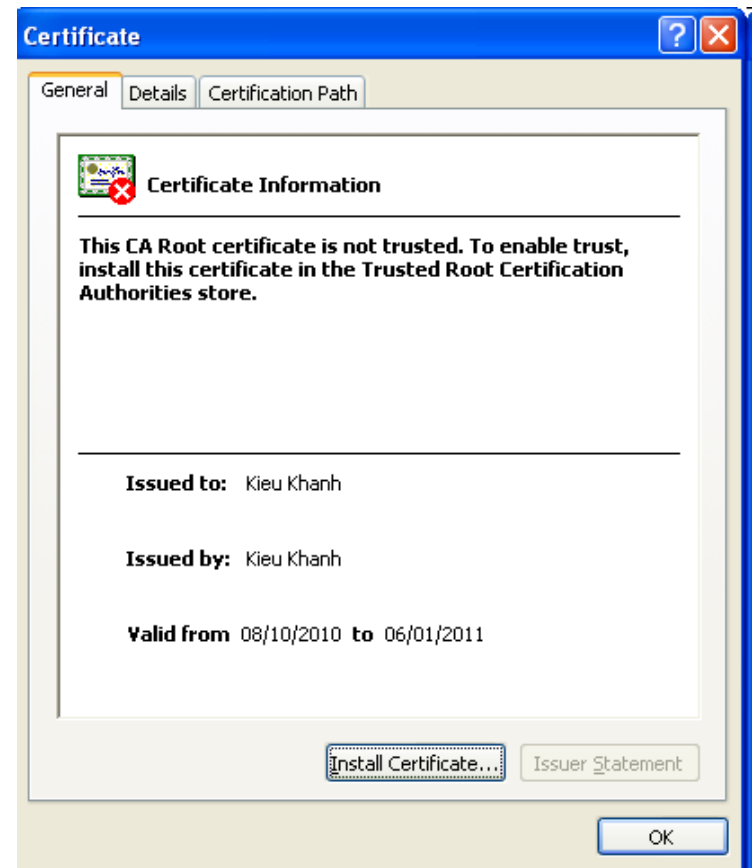

Authentication Types

HTTPS CLIENT – Step 4

- Restart the Web server and execute the application



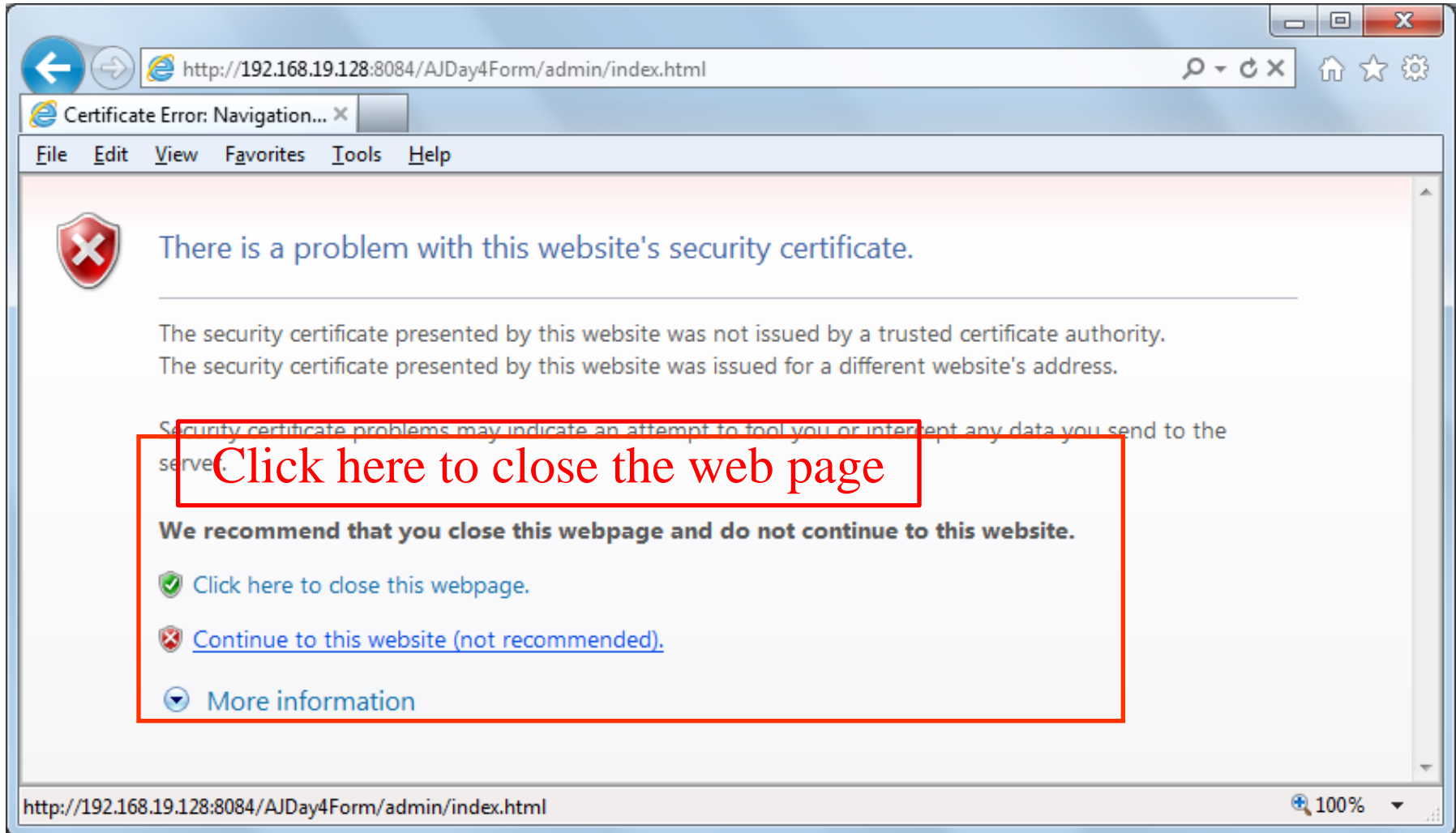
Click yes to go the login form



Authentication Types

HTTPS CLIENT – Step 4

- Restart the Web server and execute the application – Windows Vista or Wins 7



Authentication Types

HTTPS CLIENT – Step 4

- Restart the Web server and execute the application

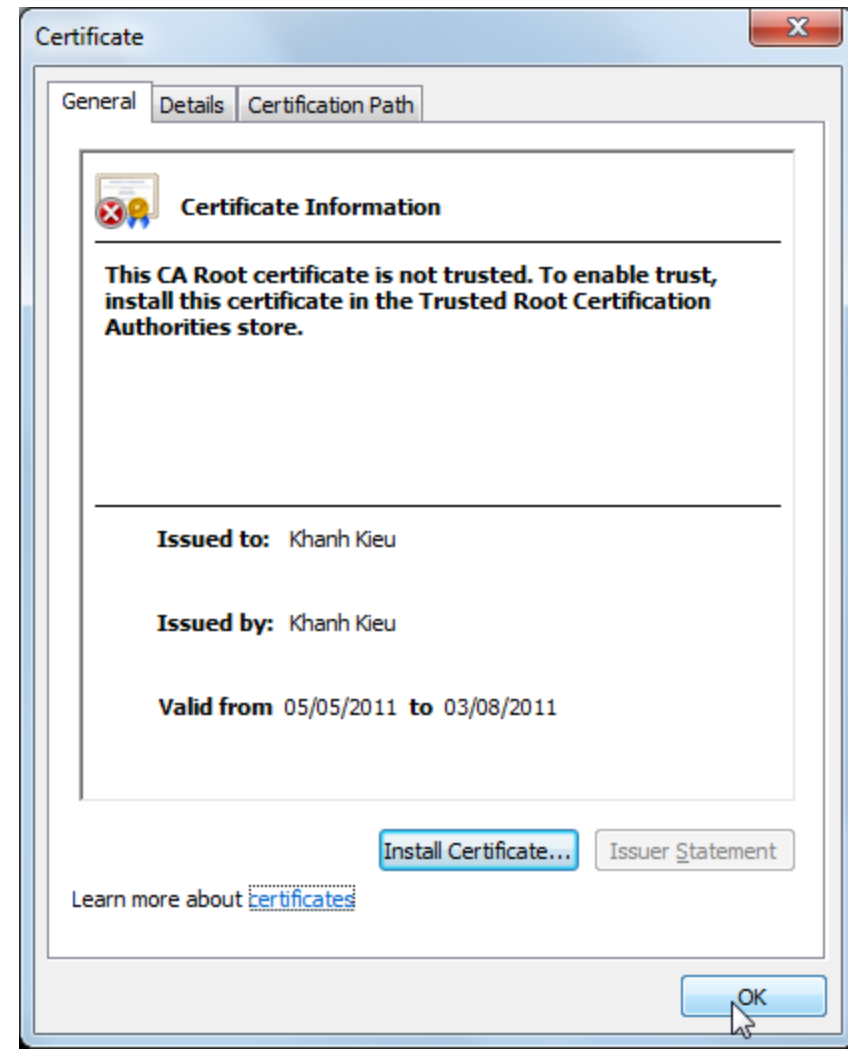
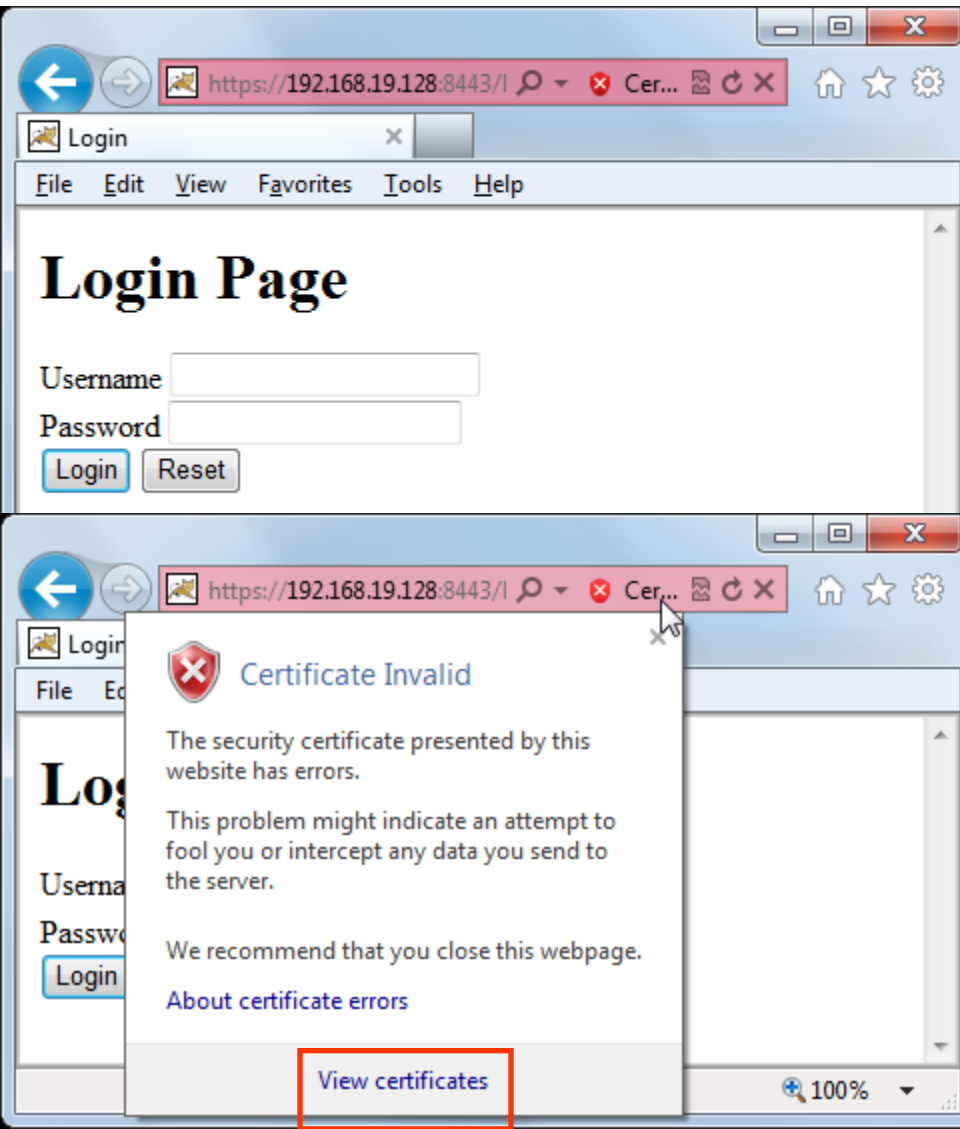


Click No Button on Security Alert Dialog to close web page

Authentication Types

HTTPS CLIENT – Step 4


- Win Vista – Win7



Authentication Types

HTTPS CLIENT – Step 4

- Modify one of following contents to check application
 - *Comment above configuration contents on server.xml file*
 - *Restart server*
 - *Running the application again*
 - *Delete the .keystore file*
 - *Restart Server*


Address  https://localhost:8443/AJDay4Form/admin/index.html



The page cannot be displayed

The page you are looking for is currently unavailable. The Web site might be experiencing technical difficulties, or you may need to adjust your browser settings.

Please try the following:

- Click the  Refresh button, or try again later.
- If you typed the page address in the Address bar, make sure that it is spelled correctly.
- To check your connection settings, click the **Tools** menu, and then click **Internet Options**. On the **Connections** tab, click **Settings**. The settings should match those provided by your local area network (LAN) administrator or Internet service provider (ISP).
- See if your Internet connection settings are being detected. You can set Microsoft Windows to examine your network and automatically discover network connection settings (if your network administrator has enabled this setting).

Output

Apache Tomcat 6.0.26 Log x Apache Tomcat 6.0.26 x Day4Form1005F2 (run) x

```
SEVERE: Failed to load keystore type JKS with path C:\Documents and Settings\Trong Khanh\.keystore due to C:\Documents and
java.io.FileNotFoundException: C:\Documents and Settings\Trong Khanh\.keystore (The system cannot find the file specified)
  at java.io.FileInputStream.open(Native Method)
  at java.io.FileInputStream.<init>(FileInputStream.java:106)
  at org.apache.tomcat.util.net.jsse.JSSESocketFactory.getStore(JSSESocketFactory.java:347)
  at org.apache.tomcat.util.net.jsse.JSSESocketFactory.getKeyStore(JSSESocketFactory.java:269)
  at org.apache.tomcat.util.net.jsse.JSSESocketFactory.getKeyManagers(JSSESocketFactory.java:482)
  at org.apache.tomcat.util.net.jsse.JSSESocketFactory.init(JSSESocketFactory.java:419)
  at org.apache.tomcat.util.net.jsse.JSSESocketFactory.createSocket(JSSESocketFactory.java:130)
```

Authentication Types

JDBC Realms

- Is an **implementation of a tomcat Realm** that use a set of **configurable tables inside a RDMS** to store user's data, this tables are accessed by means of standard JDBC drivers
- **Step 1:** Creating DB on RDBMS
- **Step 2:** Config JDBCRealm on Tomcat
- **Step 3:** Start Server and run application again

Authentication Types

Creating the table on RDBMS

Table - dbo.users			
Summary			
	Column Name	Data Type	Allow Nulls
▶	user_name	varchar(15)	<input type="checkbox"/>
	user_pass	varchar(15)	<input type="checkbox"/>

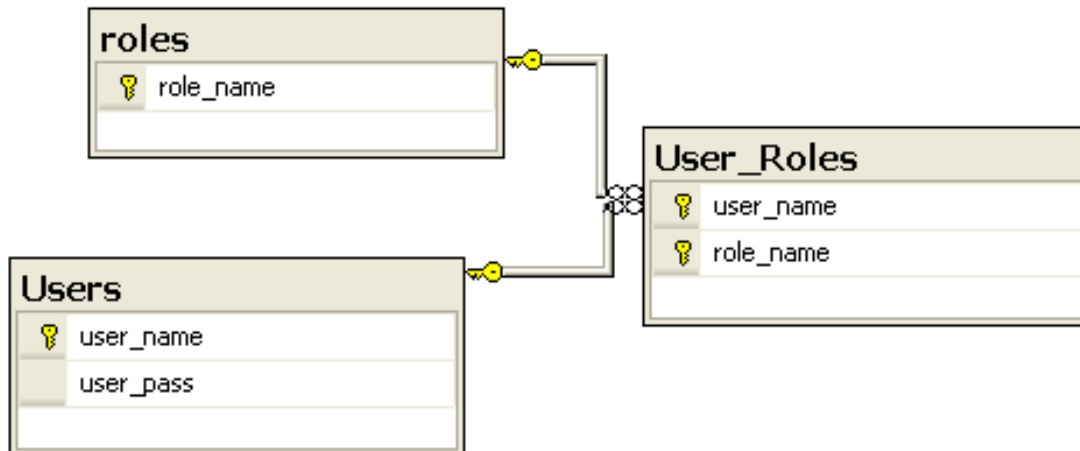
Table - dbo.roles			
Summary			
	Column Name	Data Type	Allow Nulls
▶	role_name	varchar(15)	<input type="checkbox"/>
			<input type="checkbox"/>

Table - dbo.user_roles			
Summary			
	Column Name	Data Type	Allow Nulls
▶	user_name	varchar(15)	<input type="checkbox"/>
▶	role_name	varchar(15)	<input type="checkbox"/>

Table - dbo.users		
Summary		
	user_name	user_pass
▶	guest	guest
	ide	BUuvwKWD
	khanh	khanhkieu
	tomcat	tomcat

role_name	
▶	admin
	manager
	user

Table - dbo.user_roles		
Summary		
	user_name	role_name
▶	guest	user
	ide	manager
	khanh	manager
	tomcat	admin



Authentication Types

Configuring JDBCRealm on Tomcat

- Go the server.xml to update/ modify the following content

```
<Realm className="org.apache.catalina.realm.UserDatabaseRealm" resourceName="UserDatabase"/>
<Realm className="org.apache.catalina.realm.JDBCRealm"
  driverName="com.microsoft.sqlserver.jdbc.SQLServerDriver"
  connectionURL="jdbc:sqlserver://localhost:1433;instanceName=SQL2005;databaseName=Sinhvien"
  connectionName="sa" connectionPassword=""
  userTable="users" userNameCol="user_name" userCredCol="user_pass"
  userRoleTable="user_roles" roleNameCol="role_name" />
```

Search key word Realm on the server.xml file

- **Step 3: Start Server and run application again**
 - **Before the server is started, the DB driver (sqljdbc.jar) must be located at c:\Program Files\Apache Software Foundation\Apache Tomcat 6.0.26\lib**

Security Mechanisms

Declarative Security

- Provides security to resource with the help of the **server configuration**
- **Works as a different layer from the web component** which it works.
- **Advantages**
 - Gives scope to the programmer to **ignore** the **constraints** of the programming environment
 - **Updating the mechanism** does not require total change in Security model
 - It is easily maintainable
- **Limitation**
 - Access is **provided to all or denied**
 - Access is **provided by the Server only if the password matches**
 - All the **pages use same authentication mechanism**
 - It can **not use both form-based and basic authentication for different page**

Security Mechanisms

Declarative Security Implementation

- Setting up User Names, Passwords, Roles
- Setting Authentication mechanism to Authentication Type
 - Creating Login Page and Error Page with Form-based authentication
 - Defining all the deployment descriptor security declaration
- Specify URLs that should be password protected
- Specify URLs that Should be available only with SSL (if necessary)

Security Mechanisms

Programmatic Security

- Authenticates users and grant access to the users
- Servlet either authenticates the user or verify that the user has authenticates earlier
- There are **2 types**:
 - Hard vs. Soft (combining between code and declaration)
- **Advantages**
 - Ensure total portability
 - Allowed password matching strategies
- **Limitation**
 - Much harder to code and maintain
 - Every resource must use the code
- **Implementation**
 - Check whether there is an authorization request header (**checking header**)
 - Get the String, which contains the encoded user name / password
 - Reverse the base64 encoding of the user name / password String (**digest auth**)
 - Check the user name and password (**login form with web server or DB**)
 - If authentication fails, send the proper response to the client (**error handling**)

Security Mechanisms

Programmatic Security

- Some **method** supporting from `HttpServletRequest`

Methods	Descriptions
getAuthType	- public string getAuthType() - returns the authentication scheme name
getHeader	- Reference details in previous topic
getRemoteUser	- public String getRemoteUser() - If the user is authenticated it returns the login name of the user else it returns null.
isUserInRole	- public boolean isUserInRole(String role) - Returns a Boolean value, which indicates whether the authenticated user is included in the logical “role”.
getUserPrincipal	- public Principal getUserPrincipal() - Returns a <code>java.security.Principal</code> object

Security Mechanisms

Programmatic Security

- **Securing on Servlet**

`<servlet>`

`<servlet-name>servletName</servlet-name>`

`<servlet-class>servletClass</servlet-class>`

`<security-role-ref>`

`<description/>`

`<role-name>aliasName_map_rolelink</role-name>`

`<role-link>realRoleName</role-link>`

`</security-role-ref>`

`</servlet>`

Security Mechanisms

Programmatic Security – Example

```

AuthServlet.java x
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Programmatically Security</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Programmatically Security Demo</h1>");

        String user = request.getUserPrincipal().getName();
        String userName = request.getRemoteUser();

        out.println("<h1>The accessed user is: " + user + "</h1>");
        out.println("<h2>The username is: " + userName + "</h2>");
        out.println("The Auth Type: " + request.getAuthType() + "</h3>");

        if(request.isUserInRole("MGR")){
            out.println("<h2>Are you a manager???</h2>");
        } else if(request.isUserInRole("user")){
            out.println("<h2>Are you a user???</h2>");
        } else if (request.isUserInRole("guest")){
            out.println("<h2>Are you a guest???</h2>");
        }
        out.println("</body>");
        out.println("</html>");
    } finally {

```

Security Mechanisms

Programmatic Security – Example

- Applied the Form Security to this Application, then run directly the AuthServlet and the web.xml does not secure the AuthServlet
 - The errors occur at the server consoles.

```
<servlet>
  <servlet-name>AuthServlet</servlet-name>
  <servlet-class>sample.auth.AuthServlet</servlet-class>
  <security-role-ref>
    <description/>
    <role-name>MGR</role-name>
    <role-link>*</role-link>
  </security-role-ref>
</servlet>
<servlet-mapping>
<session-config>
<welcome-file-list>
<security-constraint>
  <display-name>Constraint1</display-name>
  <web-resource-collection>
    <web-resource-name>Servlet Auth</web-resource-name>
    <description/>
    <url-pattern>/index.jsp</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <description/>
    <role-name>*</role-name>
```

Output

Apache Tomcat 6.0.26 x Apache Tomcat 6.0.26 Log x ProgrammaticallySec (run) x

```
23-06-2011 12:00:11 org.apache.catalina.core.StandardWrapperValve invoke
SEVERE: Servlet.service() for servlet AuthServlet threw exception
java.lang.NullPointerException
    at sample.auth.AuthServlet.processRequest (AuthServlet.java:40)
    at sample.auth.AuthServlet.doGet (AuthServlet.java:71)
    at javax.servlet.http.HttpServlet.service (HttpServlet.java:617)
```

Security Mechanisms

Programmatic Security – Example

```
<servlet>
  <servlet-name>AuthServlet</servlet-name>
  <servlet-class>sample.auth.AuthServlet</servlet-class>
```

```
  <security-role-ref>
    <description/>
    <role-name>MGR</role-name>
    <role-link>*</role-link>
  </security-role-ref>
```

```
</servlet>
```

```
<servlet-mapping>
```

```
<session-config>
```

```
<welcome-file-list>
```

```
<security-constraint>
```

```
  <display-name>Constraint1</display-name>
```

```
  <web-resource-collection>
```

```
    <web-resource-name>Servlet Auth</web-resource-name>
```

```
    <description/>
```

```
    <url-pattern>/AuthServlet</url-pattern>
```

```
  </web-resource-collection>
```

```
  <auth-constraint>
```

```
    <description/>
```

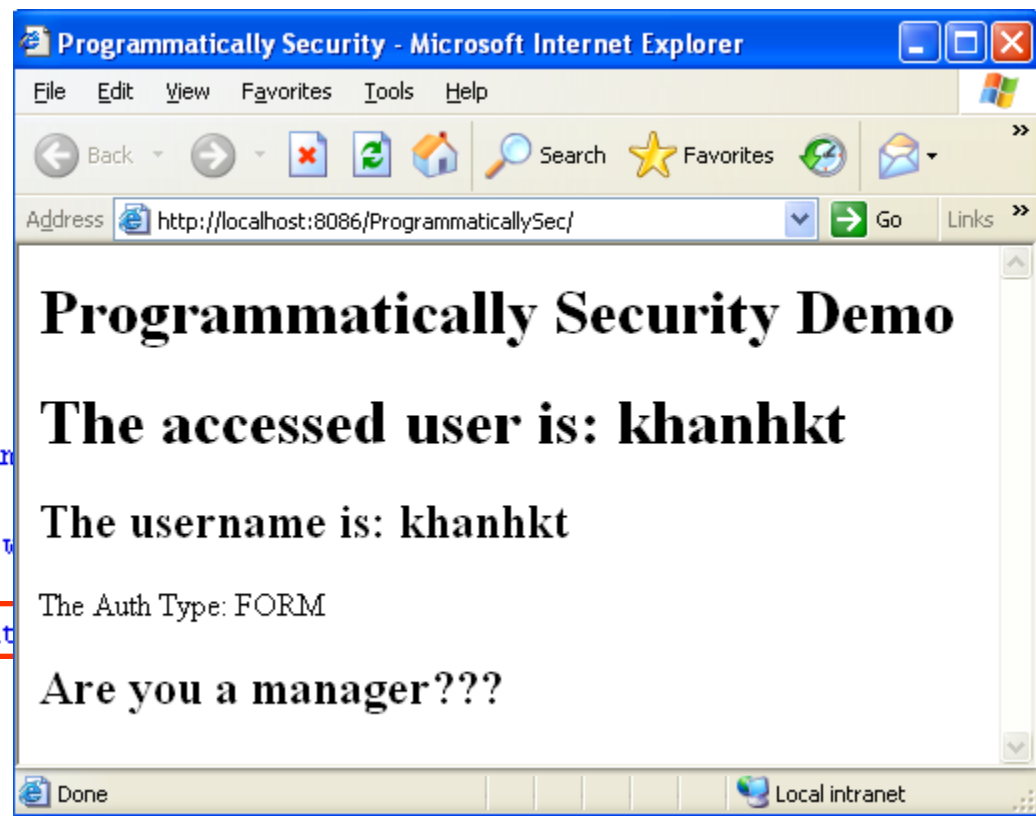
```
    <role-name>*</role-name>
```

```
  </auth-constraint>
```

```
</security-constraint>
```

```
<login-config>
```

```
<security-role>
```



Output

Apache Tomcat 6.0.26 x Apache Tomcat 6.0.26 Log x ProgrammaticallySec (run) x

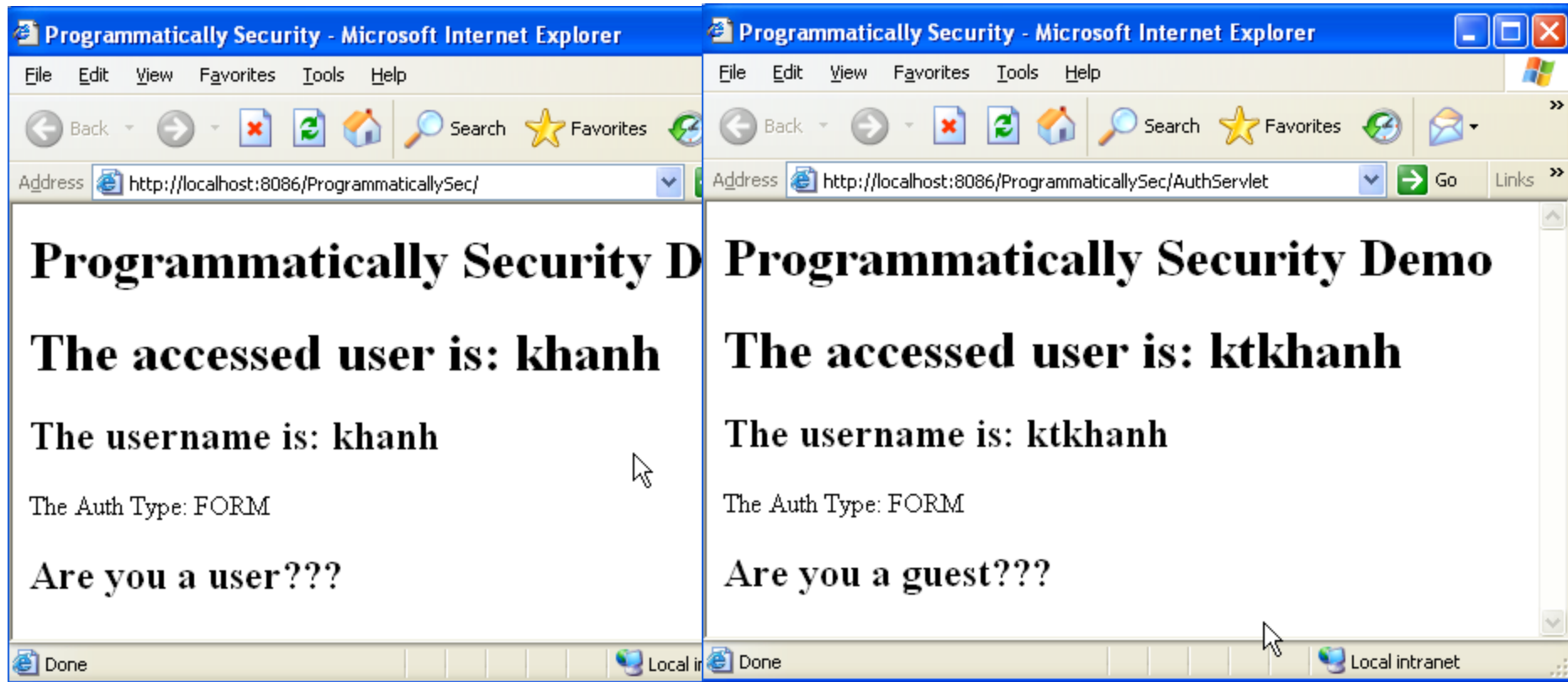
23-06-2011 12:04:01 org.apache.catalina.startup.ContextConfig validateSecurityRoles

INFO: WARNING: Security role name * used in a <role-link> without being defined in a <security-role>

Security Mechanisms

Programmatic Security – Example

```
if (request.isUserInRole("manager")) {
```



Security Mechanisms

Programmatic Security – Example

```
<security-constraint>
  <display-name>Constraint1</display-name>
  <web-resource-collection>
    <web-resource-name>Servlet Auth</web-resource-name>
    <description/>
    <url-pattern>/AuthServlet</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <description/>
    <role-name>manager</role-name>
  </auth-constraint>
</security-constraint>
```

Only user with manager role can access the protected web resources. Others users' role will get 403 error

Address  http://localhost:8084/ProgrammaticallySec/AuthServlet   Go

HTTP Status 403 - Access to the requested resource has been denied

type Status report

message Access to the requested resource has been denied

description Access to the specified resource (Access to the requested resource has been denied) has been forbidden.

Apache Tomcat/6.0.26

Security Mechanisms

Programmatic Security – Example

```
<servlet>
  <servlet-name>AuthServlet</servlet-name>
  <servlet-class>sample.auth.AuthServlet</servlet-class>
  <security-role-ref>
    <description/>
    <role-name>MGR</role-name>
    <role-link>manager</role-link>
  </security-role-ref>
</servlet>
<servlet-mapping>
  <session-config>
  <welcome-file-list>
</servlet-mapping>
<security-constraint>
  <display-name>Constraint1</display-name>
  <web-resource-collection>
    <web-resource-name>Servlet Auth</web-resource-name>
    <description/>
    <url-pattern>/AuthServlet</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <description/>
    <role-name>*</role-name>
  </auth-constraint>
</security-constraint>
```

- Same result with previous slides because only manage role can access
- Same result with checking MGR in servlet code

Summary

- **Security Mechanisms**
- **Deployment Descriptor Security Declarations**
- **Authentication Types**

Q&A

Next Lecture

- **JSP**
 - Introduction
 - Syntax:
 - Declaration
 - Scriptlet
 - Directives
 - Expression
 - Comments
 - Implicit Objects
- **MVC Pattern Design**