Access Control: Authentication & Authorization

Access Control Mechanisms

- **Declarative**
  - Separate from your jsp and servlet code
  - Based on adding authorization and security data in web app's deployment files
  - Policy is centralized in those files
  - Based on static Roles who are groups of users that have access to particular resources (typically pages)
- **Programmatic**
  - Your code is responsible
  - Choose when you need to create intricate access control strategies

Security Using the MemoryRealm or JDBC Realm

- **PLUS**: Really simple!
- **MINUS**: Clear text passwords, static policy
- Configure users and roles in tomcat-users.xml file
An example of declarative authorization w/ tomcat-users.xml

- Three classes of users
  - chair, committee, admin
- All http://host/app/admin/* pages will be accessed only by administrators
- All http://host/app/chair/* pages will be accessed only by the chairs
- All http://host/app/committee/* pages will be accessed only by chairs and committee
- "abk" is committee member
- "vianu" is chair
- "akovacev" is administrator

Security constraints (part of web.xml)

```xml
<!-- Security Constraints -->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Admin Area</web-resource-name>
    <!-- Define the context-relative URL(s) to be protected
    <url-pattern>/admin/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>admin</role-name>
  </auth-constraint>
  <user-data-constraint>
    <description>SSL Required</description>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

Security Constraints

```xml
<!-- Security Constraints -->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Committee Area</web-resource-name>
    <!-- Define the context-relative URL(s) to be protected
    <url-pattern>/committee/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>chair</role-name>
    <role-name>committee</role-name>
  </auth-constraint>
  <user-data-constraint>
    <description>SSL Required</description>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```
Dynamic Database-Driven Access Control

- **Tomcat-users.xml** is a kind of Security Realm, i.e., a provider of user credentials
- **JDBCRealm**: User credentials are stored in a database, accessed via JDBC
- **JNDIRealm**: User credentials are stored in directory server, accessed via JNDI

Realm in server.xml

```xml
<![CDATA[
<Realm className="org.apache.catalina.realm.JDBCRealm" debug="99"
driverName="org.gjt.mm.mysql.Driver"
connectionURL="jdbc:mysql://localhost/authority"
connectionName="test"
connectionPassword="test"
digest="MD5"
userTable="users"
userNameCol="username"
userCredCol="password"
userRoleTable="userroles"
rolenameCol="rolename" />
]]>
```

<table>
<thead>
<tr>
<th>users</th>
<th>userroles</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>password</td>
</tr>
<tr>
<td>vianu</td>
<td>foo</td>
</tr>
<tr>
<td>abk</td>
<td>foo</td>
</tr>
</tbody>
</table>
Hiding Passwords

Assume `pwd` has password.

`RealmBase.Digest(user, "MD5")` returns MD5 encoding
Which you insert in DB