Web Component Development Using Java

Session: 15

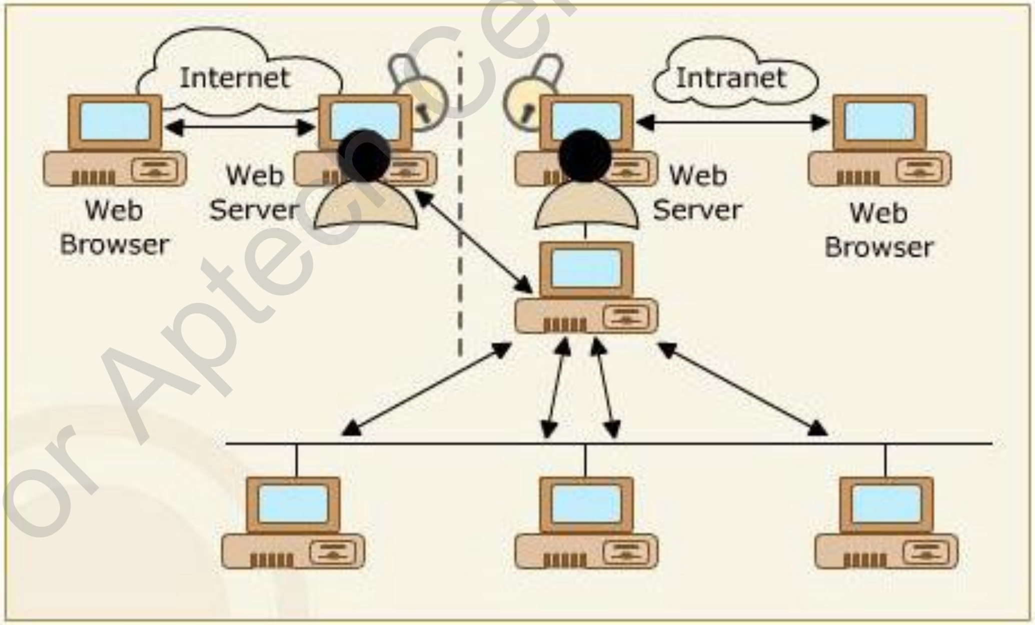
Securing Web Applications

Objectives

- Explain the need and features of securing Web applications
- Describe the HTTP basic, digest, client, and form-based authentication method of ensuring security
- Explain how to configure users in Tomcat
- Explain how to specify authentication mechanisms using web.xml
- Describe the seven steps to implement declarative security
- Explain the concept and five steps to implement programmatic security
- Describe the HttpServletRequest methods for identifying users

Introduction 1-2

- A Web application can be accessed using a Web browser over a network, such as Intranet or the Internet.
- Easier accessibility to Web applications opens door to attackers to access or modify the confidential information.
- It is necessary to secure the Web application to keep the information safe using some of the security mechanisms.
- Figure depicts unauthorized access on the Internet.



Introduction 2-2

- Three important issues to be considered in case of security:

Authentication

- Enables client verification through correct username and password.

Confidentiality

- Keeps information hidden from people other than the involved client and server.

Integrity

- Ensures that the content of the communication is not changed during transmission.

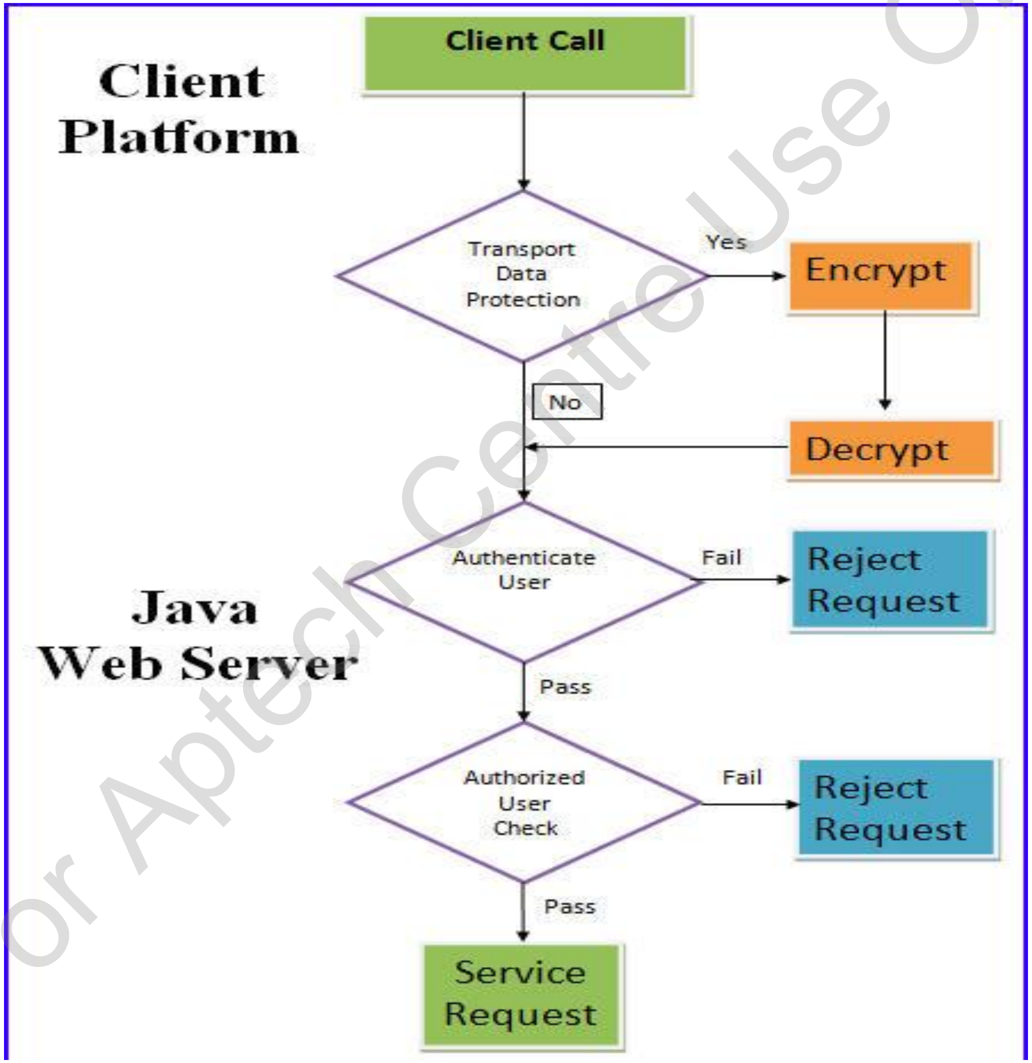
Understanding Authentication and Authorization 1-2

- When the user accesses the Web application resources, he/she needs to be identified.

- Identifying the user allows the Web application to understand:
  - Identity of the user
  - Operations that can be performed by the user
  - Maintaining integrity and confidentiality of the accessed resources

Understanding Authentication and Authorization 2-2

- Figure shows the various runtime security mechanism applied in a Web application.



Securing Web Applications

- Web applications are secured by:
  - **Application Developer** - Set up the security for the application by using annotations or deployment descriptors.
  - **Application Deployer** - Assigns method permissions for security roles, set up authentication and transport mechanisms.
- Terms related to applying security in Java Web application are as follows:

User

Is an identity that has been defined in the Web server.

Group

Is a set of authorized users classified with common features such as job title or department.

Role

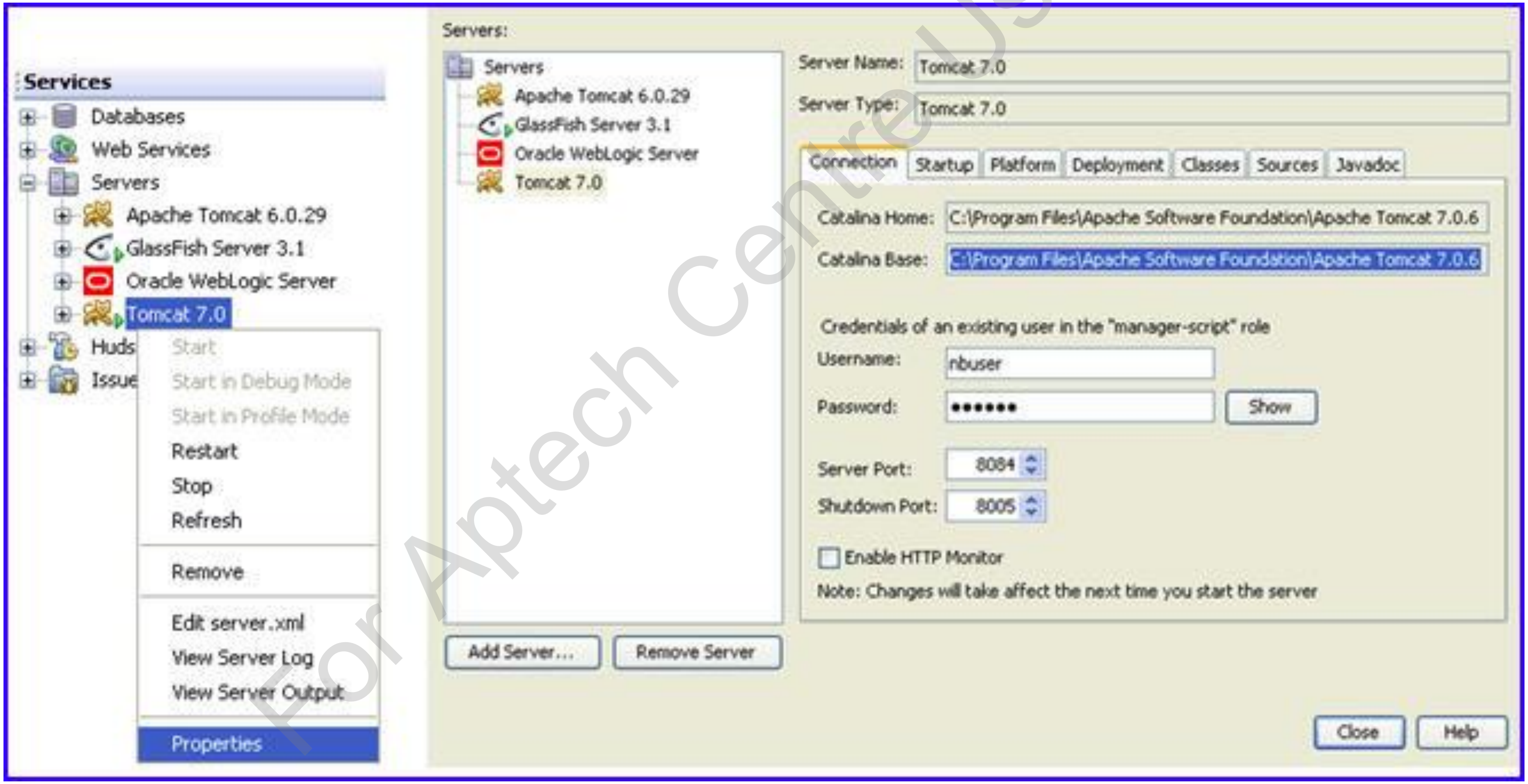
A role is a set of permissions that is applied to a resources in an application.

Realm

A realm is a database of users and groups identifying valid users for the Web applications.

Configuring Users in Tomcat 1-2

- For Tomcat 7, create a user with the manager-script role and a password for that users.
- Figure shows how to register the Tomcat server with NetBeans IDE.



Server Name: Tomcat 7.0

Server Type: Tomcat 7.0

Catalina Home: C:\Program Files\Apache Software Foundation\Apache Tomcat 7.0.6

Catalina Base: C:\Program Files\Apache Software Foundation\Apache Tomcat 7.0.6

Credentials of an existing user in the "manager-script" role

Username: nbuser

Password: ••••••

Server Port: 8084

Shutdown Port: 8005

□ Enable HTTP Monitor

Note: Changes will take affect the next time you start the server

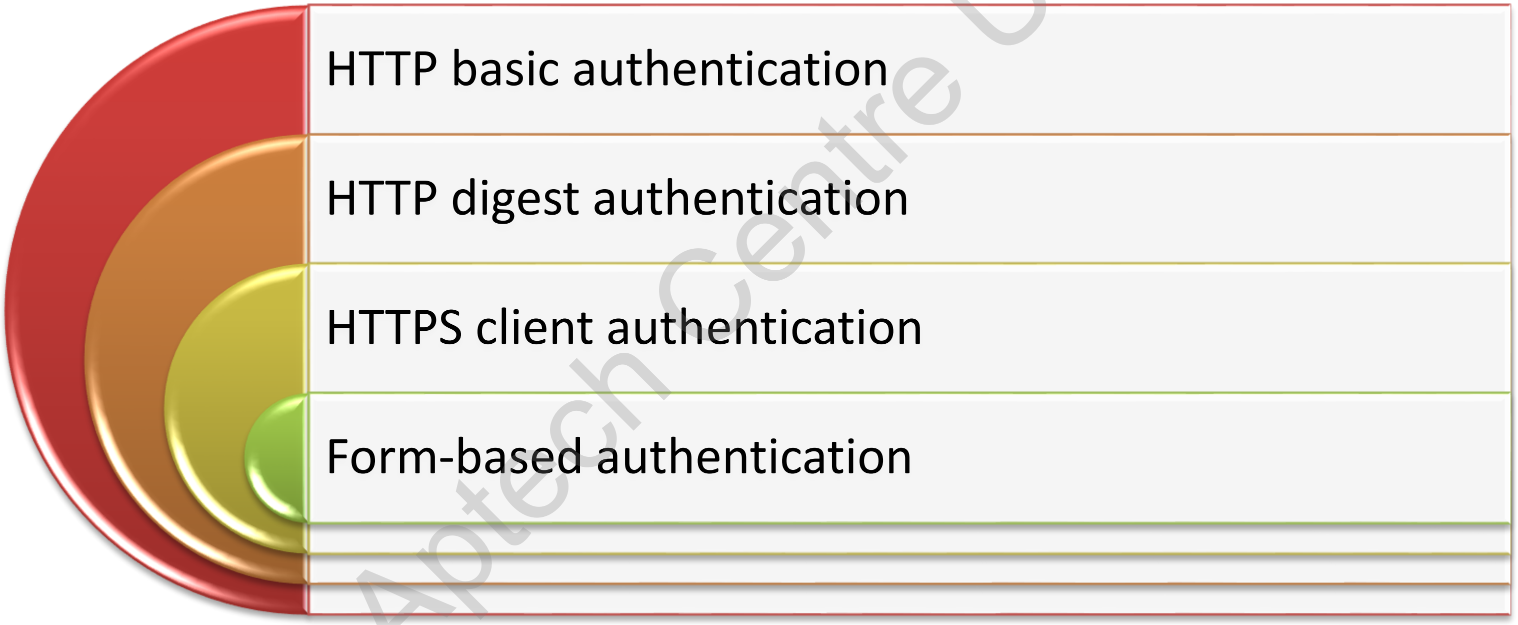
Configuring Users in Tomcat 2-2

- The basic users and roles for the Tomcat server are in `tomcat-users.xml` which is available in:
  - `<CATALINA_BASE>\conf` directory under the installed directory of the machine.
- The code snippet demonstrates how to add users and roles in the `tomcat-users.xml` file.

```
// To include a new user
<?xml version=’1.0’ encoding=’utf-8’?>
<tomcat-users>
   <role rolename=”manager”/>
   <role rolename=”admin”/>
   <user username=”Tom” password=”tempo”
                                roles=”admin,manager”/>
   <user username=”admin” password=”aptech”
                                roles=”admin,manager”/>
</tomcat-users>
```

Authentication Mechanism 1-3

- The security mechanisms are based on authentication and authorization techniques.
- Some of the commonly used authentication mechanisms are as follows:



HTTP basic authentication

HTTP digest authentication

HTTPS client authentication

Form-based authentication

Authentication Mechanism 2-3

HTTP Basic Authentication Method

- In HTTP basic authentication, the Web browser pops-up a login page, which prompts the user for credentials in the form of username and password.
- A user will be allowed to access information only if the credentials match a stored pair of username and password.

HTTP Digest Authentication Method

- In HTTP digest authentication, the server has the password based on a hash function.
- By using the hash function, the database stored the hash value rather than the password in plain text, so that it is difficult to decipher the data.

Authentication Mechanism 3-3

Form-based Authentication Method

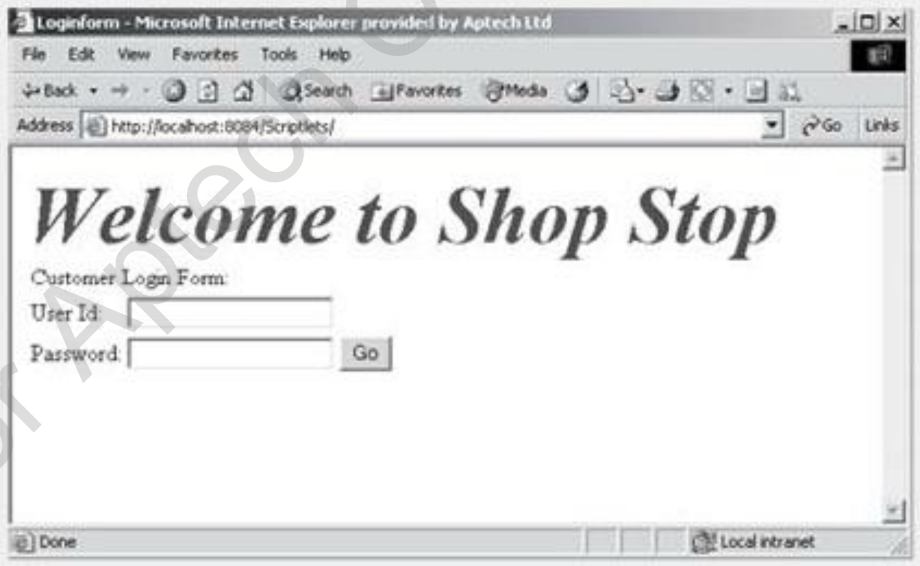
- In form-based authentication, the browser provides a login form, which appears in response to a request.
- The user can enter a username and password in the form.
- If the entered credentials match a stored pair of username and password, the user will be allowed to access information.
- The user will be directed to an error page if the login fails

HTTPS Client Authentication Method

- In HTTPS client authentication, the browser uses HTTPS protocol, which is identical with the http but the URL indicates to use a default TCP port and an extra authentication layer in between HTTP and TCP.
- This extra security layer conform the client's authentication.

HTTP Basic Authentication

- HTTP basic authentication:
  - Web browser pops-up a login page in response to a request.
  - Login page prompts the user for credentials, such as username and password.
  - Works on the assumption that the client-server communication is reliable.
  - Does not provide any protection for the information communicated between the client and the server.
- The advantage of using basic authentication is that it is browser friendly.
- Figure depicts basic authentication.



Loginform - Microsoft Internet Explorer provided by Aptech Ltd
File
Edit
View
Favorites
Tools
Help
Back
Search
Favorites
Media
Address
http://localhost:8084/Scriptlets/
Go
Links
Done
Local intranet

Welcome to Shop Stop

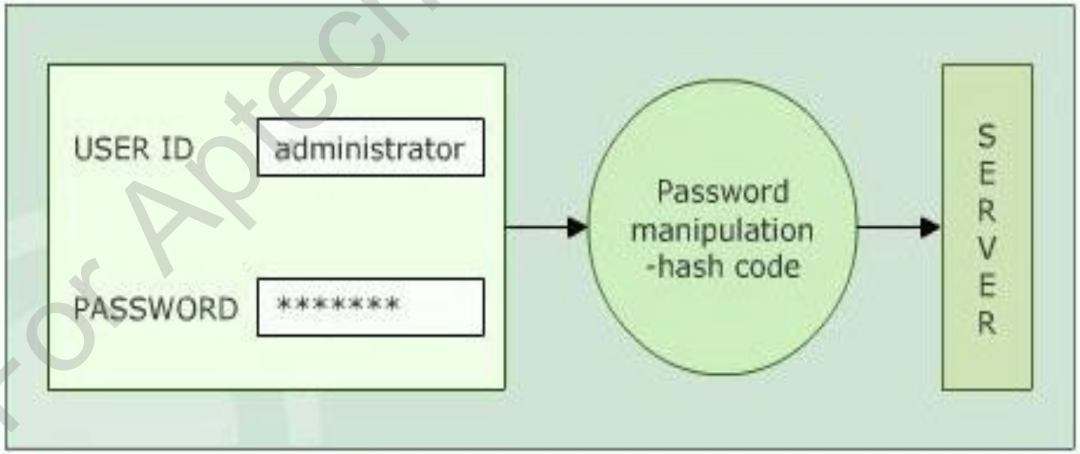
Customer Login Form:

User Id:

Password: Go

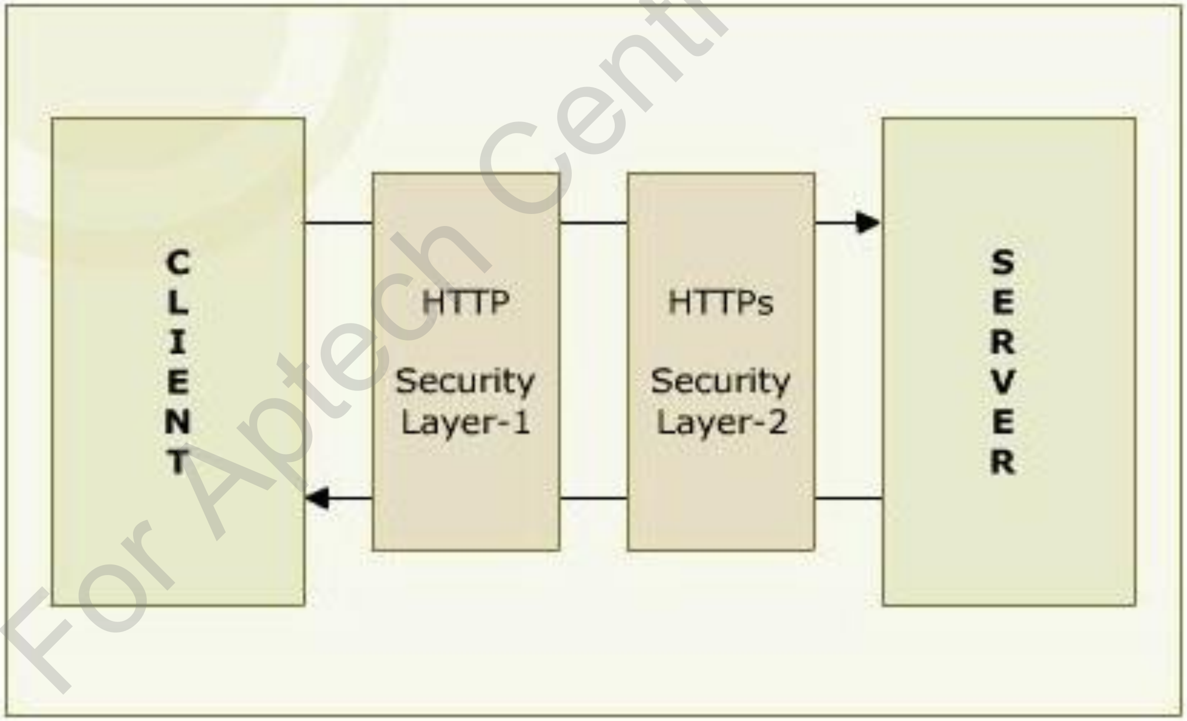
HTTP Digest Authentication

- HTTP digest authentication:
  - Authentication is based on a hash function. The database stores a hash value rather than the password.
  - Authenticates the user identity without sending the password as plain text to the server.
  - Password is first encrypted and then sent.
  - Process is a one-way system because it is difficult to know the original password according to the output.
  - Impossible for the attacker to breach this authentication mechanism.
- Figure depicts digest authentication.



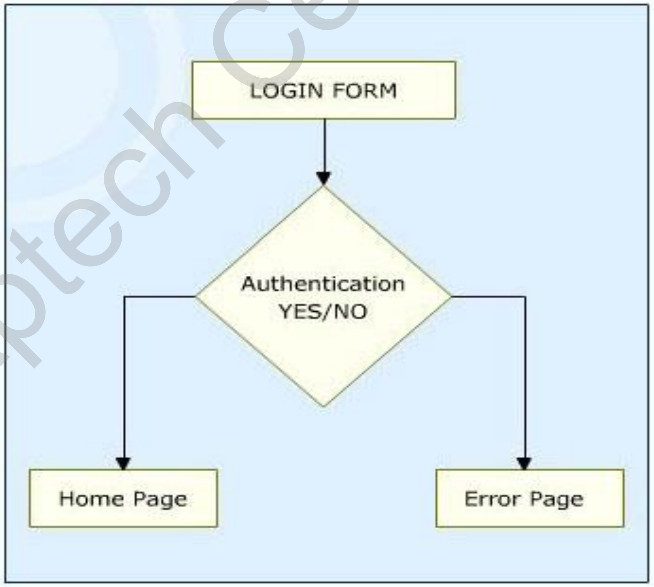
HTTPS Client Authentication

- HTTP client authentication:
  - Is a secured client authentication technique based on Public Key Certificates.
  - Is similar to http but uses the https, which is HTTP over Secure Socket Layer (SSL).
  - Based on URL that instructs the browser to use a default TCP port with an extra authentication layer in between HTTP and TCP.
- Figure depicts HTTPS client authentication.



Form-based Authentication

- Form-based authentication:
  - Provides a login page, which appears in response to a request from the user.
  - An error page is also available if the login fails from the user side.
  - After getting the username and password validated by the server, the access is provided to the user.
  - Here, a user can define how to protect the Web content for the URLs you are planning to protect.
- Figure depicts form-based authentication.



Configuring Authentication in web.xml 1-2

- The sub elements of `login-config` element are as follows:

`<auth-method>`	This sub element names the authentication method used in the element.
`<realm-name>`	This sub element names the Web resource where the `<login-config>` maps.
`<form-login-config>`	This sub element is optional, and is specified only when using form-based authentication that is the `<auth-method>` value is set to FORM.
`<form-login-page>`	This sub element specifies the form to display when a request is made to a protected Web resource in the Web application.

Configuring Authentication in web.xml 2-2

- The code snippet uses the form-based authentication mechanism.

```
<web-app>
. . .
<login-config>
   <auth-method>FORM</auth-method>
   <form-login-config>
          <form-login-page>/login.htm</form-login-page>
          <form-error-page>/loginerror.htm</form-error-page>
   </form-login-config>
</login-config>
. . .
</web-app>
```

Authorization Strategies

- The Java security model has two strategies for providing access control namely:
  - Declarative access control
  - Programmatic access control

Implementing Declarative Security 1-7

- The declarative security provides security to a resource with the help of the server configuration.
- Declarative security works as a different layer from the Web component with which it works.

Advantages of Declarative Security

- It gives scope to the programmers to ignore the constraints of the programming environment.
- Updating the mechanism does not require total change in security model.
- It is easily maintainable.

Disadvantages of Declarative Security

- Access is provided to all or denied.
- Access is provided by the server only if the password matches.
- It cannot use both form-based and basic authentication for the same page.

Implementing Declarative Security 2-7

- To implement the declarative security, perform the following eight steps:
  - **Set up usernames, passwords, and roles**
    - When a user tries to access a secured resource in a Web application, which uses form-based authentication, the system uses a form to ask for a username and a password.
    - After verifying the password, the system gives access to the user.
    - Then, it determines the role such as user, administrator or executive, which is defined by the user.
  - **Tell the Server that you are using Form-based authentication and designate locations of login and login failure pages**
    - After validating the user and the role of user, it uses a form-based authentication, which supplies a value for the `auth-method` sub element, and uses the `form-login-config` sub element to give the locations of the login and login-failure pages.
    - These pages can consists of either JSP or HTML.

Implementing Declarative Security 3-7

- **Create a Login page**
  - The `login-config` element informs the server to use form-based authentication for creating a login page.
  - The login page requires a form for security check, a text field named username and a password field named password.
  - Redirect unauthenticated users to a designated error page.
  - The code snippet indicates that the container is using form-based authentication.

```
. . .
<FORM ACTION=”securityServlet” METHOD=”POST”>
<TABLE>
  <TR><TD>User name: <INPUT TYPE=”TEXT” NAME=”j_username”>
  <TR><TD>Password: <INPUT TYPE=”PASSWORD”
NAME=”j_password”>
  <TR><TH><INPUT TYPE=”SUBMIT” VALUE=”Log In”>
</TABLE>
</FORM>
. . .
```

Implementing Declarative Security 4-7

- **Create a page to report failed login attempts**
  - The `login-failure` page contains a link to an unrestricted section of the Web application.
  - The link shows ‘login failed’ or ‘username and password not found’ message.
- **Specifying URLs that should be password protected**
  - Specifying the URLs and describing the protection they should have, are the purposes of the security-constraint element.
  - The `security-constraint` element should come before the `login-config` element in `web.xml`, and contains four sub elements, `display-name`, `web-resource-collection`, `auth-constraint`, and `user-data-constraint`.

Implementing Declarative Security 5-7

- The code snippet shows the use of `security-constraint`.

```
. . .
<web-app>
<!-- ... -->
<security-constraint>
  <web-resource-collection>
     <web-resource-name>Sensitive</web-resource-name>
     <url-pattern>/sensitive/*</url-pattern>
   </web-resource-collection>
  <auth-constraint>
    <role-name>administrator</role-name>
    <role-name>executive</role-name>
  </auth-constraint>
</security-constraint>
<login-config>...</login-config>
</web-app>
```

Implementing Declarative Security 6-7

- **Specifying URLs that should be available only with SSL**
  - If your server supports SSL, you can define that certain resources are available only through encrypted HTTPS (SSL) connections.
  - Use of SSL does not obstruct the basic way that form-based authentication works.
  - The `user-data-constraint` sub element of security-constraint can mandate that certain resources be accessed only with SSL.
  - The code snippet instructs the server to permit only https connections to the associated resource.

```
<security-constraint>
<!-- ... -->
<user-data-constraint>
  <transport-guarantee>CONFIDENTIAL</transport-
guarantee>
</user-data-constraint>
</security-constraint>
```

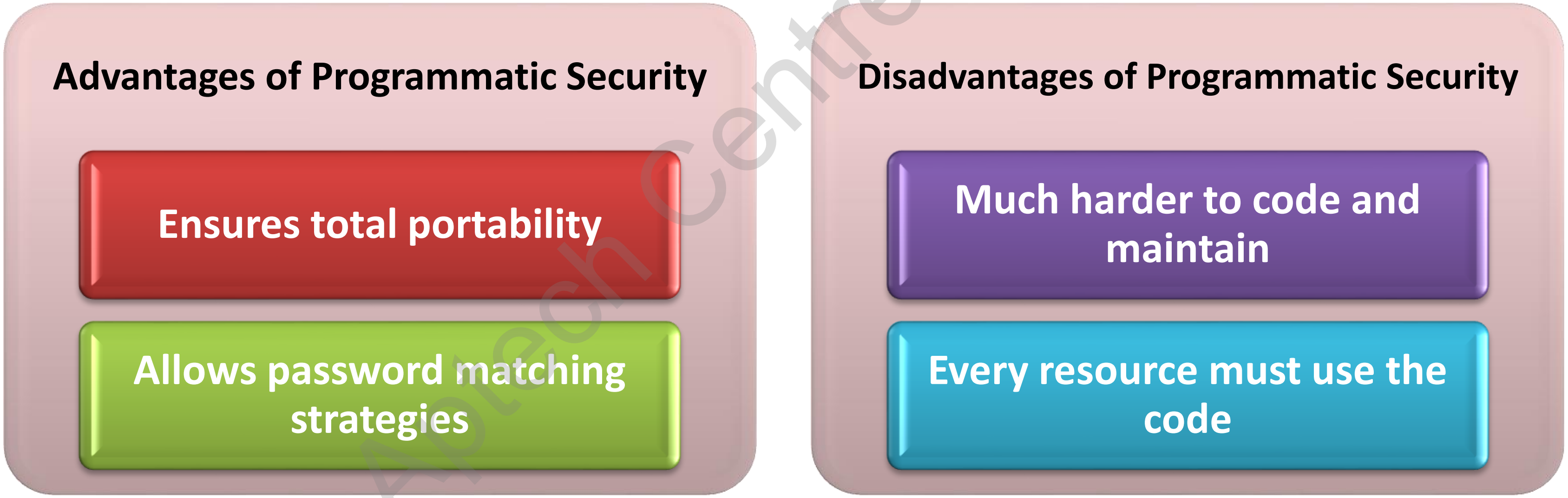
Implementing Declarative Security 7-7

- **Turning off these invoker servlet**
  - The most portable approach of turning off the invoker servlet is to remap the servlet pattern in the Web application, so that all requests that include the pattern are sent to the same servlet.
  - To remap the pattern, first create a simple servlet that prints an error message.
  - Then, use the `servlet` and `servlet-mapping` elements to send requests that include the servlet pattern to that servlet.

- **Write authentication filter**
  - Write the Servlet filter and mention in `web.xml`. This `Filter` class filters the HTTP requests based on its type.

Programmatic Security

- Programmatic security authenticates the users and grants access to the users.
- The servlet or JSP page either authenticates the user or verifies that the user has authenticated earlier, to check the unauthorized access.
- To ensure the safety of network data, the servlet or JSP page redirects the HTTP connection to HTTPS of the URLs.



Advantages of Programmatic Security

- Ensures total portability
- Allows password matching strategies

Disadvantages of Programmatic Security

- Much harder to code and maintain
- Every resource must use the code

‘HttpServletRequest’ Methods for Identifying User 1-4

- The methods in the `HttpServletRequest` interface used for identifying a user by the help of cookies are as follows:

getAuthType()

- □ This method returns the authentication scheme name.
- □ **Syntax:**

```
public java.lang.String getAuthType()
```

getCookies()

- □ It returns an array, which have all the information of the Cookie objects sent by the client.
- □ **Syntax:**

```
public Cookie[] getCookies()
```

‘HttpServletRequest’ Methods for Identifying User 2-4

`getHeader()`

- ❑ It returns the value of the header as a String, which is requested.
- ❑ **Syntax:**

```
public java.lang.String
getHeader(java.lang.String name)
```

`getRemoteUser()`

- ❑ If the user is authenticated, it returns the login name of the user, else it returns null.
- ❑ **Syntax:**

```
public java.lang.String getRemoteUser()
```

‘HttpServletRequest’ Methods for Identifying User 3-4

getRequestedSessionId()

❑ This method returns the session ID that is defined by the client.

❑ **Syntax:**

```
public java.lang.String getRequestedSessionId()
```

getSession()

❑ This method returns the current session or creates one if there is no session.

❑ **Syntax:**

```
public HttpSession get Session()
```

‘HttpServletRequest’ Methods for Identifying User 4-4

`isUserInRole()`

- ❑ It returns a boolean value, which indicates whether the authenticated user is included in the logical ‘role’.
- ❑ **Syntax:**

```
public boolean
isUserInRole(java.lang.String role)
```

`getUserPrincipal()`

- ❑ It checks the validity of the requested session ID.
- ❑ **Syntax:**

```
public java.security.Principal
getUserPrincipal()
```

`isRequestedSessionIdValid()`

- ❑ This method returns a `java.security.Principal` object.
- ❑ **Syntax:**

```
public boolean isRequestedSessionIdValid()
```

Implementing Programmatic Security 1-2

- The six steps to implement programmatic security are as follows:

Check whether there is an authorization request header

- It checks whether the header exists or not.

Get the string, which contains the encoded username/password

- If the authentication header exists it returns the string, which contains the username and password in base64 encoding format.

Reverse the base64 encoding of the username/password string

- Reverse the base64 encoding of username or password by using the `decodeBuffer` method of BASE64Decoder class.

Check the username and password

- The common approach of checking the username and password is to use database to get the original username and password.
- In other way, put the information of the password in the servlet.
- You will get access if the incoming username and password matches the reference username and password pairs.

Implementing Programmatic Security 2-2

If authentication fails, send the proper response to the client

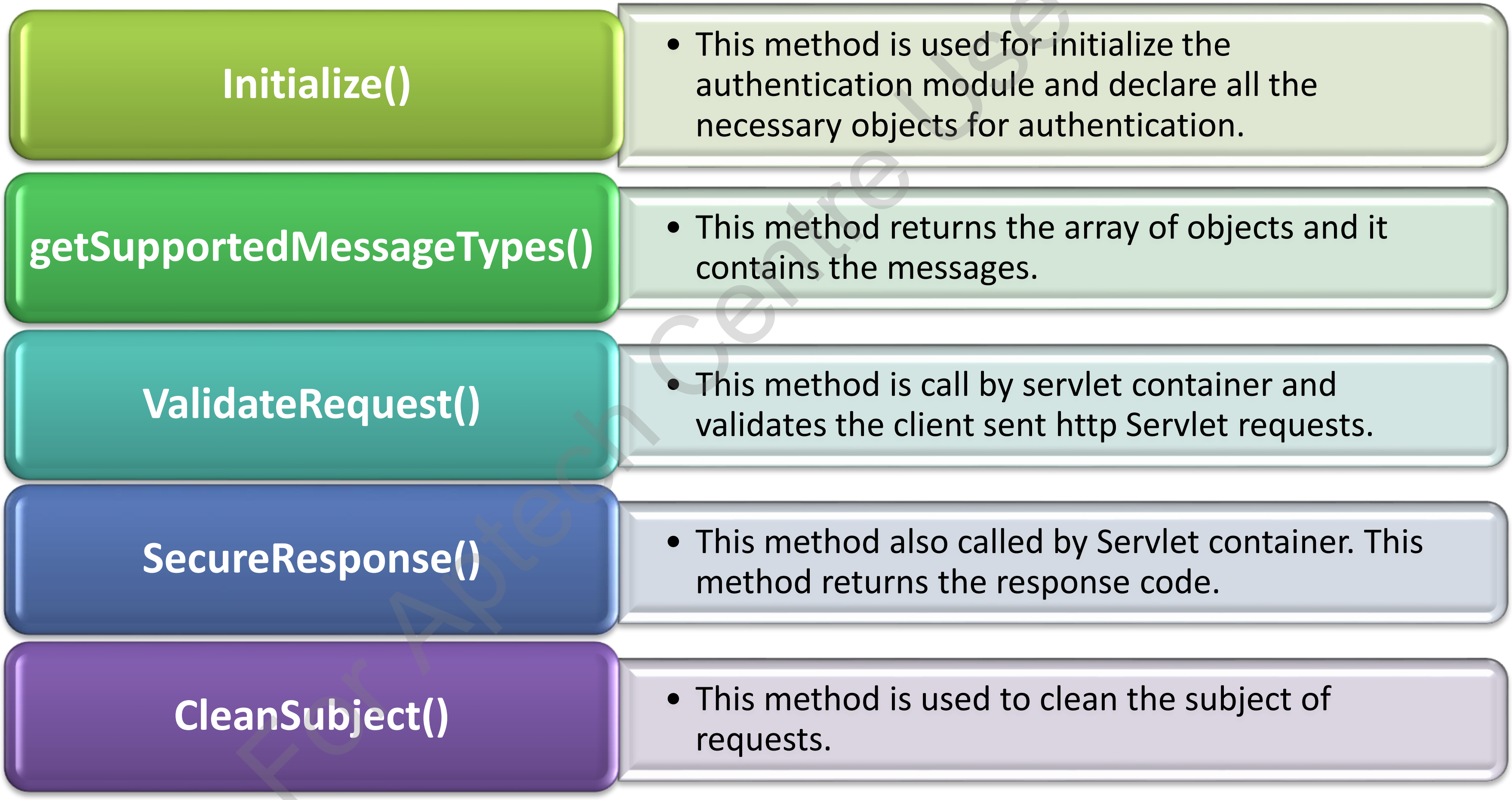
- It returns a 401 response code, which is an unauthorized one in the form of, WWW-Authenticate: `BASIC realm="some-name``.
- This response pops-up a dialog box requesting the user to enter a name and password for some-name.

Servlet Annotations

- Based on J2EE API, `javax.servlet.annotation` package provides the number of classes and interfaces.
- Using of these classes and interfaces declare the filters, servlets, and listeners to authenticate the http requests.

ServerAuthModule Interface

- `ServerAuthModule` interface helps to validate the client request.
- `ServerAuthModule` interface has five methods that are as follows:



Initialize()

- This method is used for initialize the authentication module and declare all the necessary objects for authentication.

getSupportedMessageTypes()

- This method returns the array of objects and it contains the messages.

ValidateRequest()

- This method is call by servlet container and validates the client sent http Servlet requests.

SecureResponse()

- This method also called by Servlet container. This method returns the response code.

CleanSubject()

- This method is used to clean the subject of requests.

Summary

- A Web application is an application, which is accessed with the help of a Web browser.
- The reason of popularity of Web application is the ability to maintain it without changing the client computers. When you access the Web, hackers can get your information. So to keep your information secret, it is necessary to secure Web applications.
- There are four authentication mechanisms available. These are HTTP Basic Authentication, HTTP Digest Authentication, HTTPS Client Authentication, and Form-Based Authentication.
- To configure a user in Tomcat, first include the Tomcat 6.0 from Apache Software Foundation. When browser loads a resource, which is secured by web.xml file, the browser responds in two ways.
- The browser challenges the user if you are using basic authentication or forwards the login page if you are using form-based authentication.
- The declarative security provides security to a resource with the help of server configuration.
- Programmatic security authenticates the users and grants access to the users.