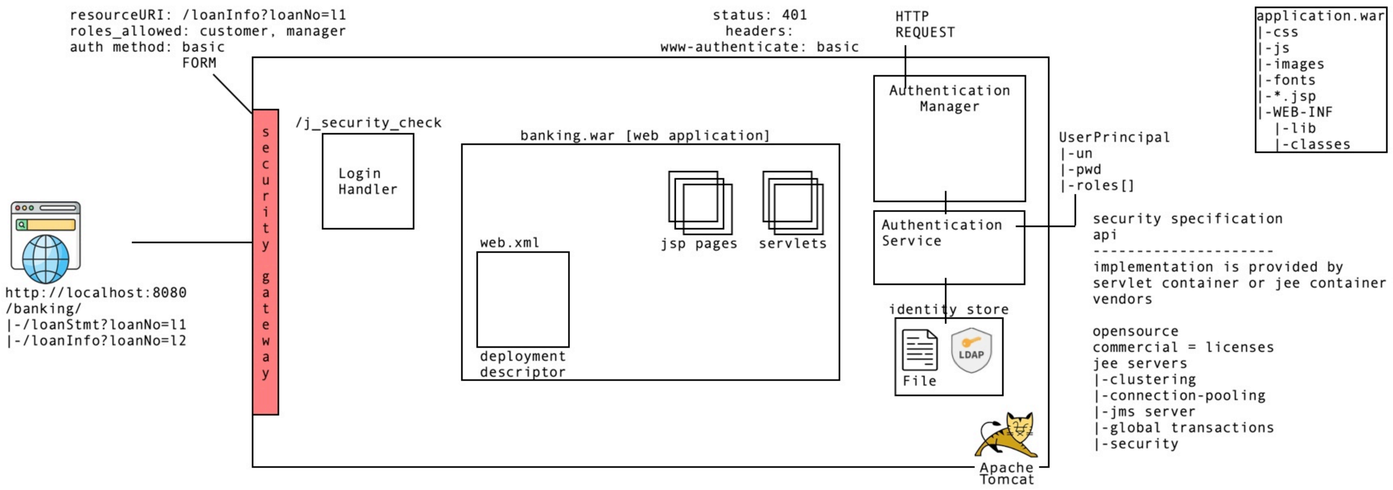

Securing an web application is a commmon requirement for any application irrespective of language or platform in which it has been built. Since every web application works over http protocol.
The http protocol itself has defined security standards, in exchanging the authentication or security data and response between the client/server. Based on these http security standards

every programming language has to provide an appropriate apis in implementing security for their applications that are being build in their language. So java language also has to provide an api to implement security for the web applications that are being built on java platform. The java language has provided javax.security api insupport of implementing security based on http protocol standards.
The servlet containers and jee application servers has provided implementation for the jee security api.

The http protocol has defined #3 types of security mechanisms:
1. basic authentication
2. form-based authentication
3. digest

1. basic authentication
1.1 The client sends a request to the server asking for accessing an authenticated resource, upon receving the request the server responds to the client with 401 (unauthorized) response status. The server provides information on how to authorize with a www-Authenticate response header containing atleast one challenge.

1.2 The client that wants to authenticate itself with the server can send an Authorization header aspart of the request with credentials.
If the client is an web browser it will be presented a username & password prompt/dialog allowing the user to input the username/password. So that the browser will re-send the request to the server with Authorization header added to the request

@1
/loanStmt?loanNo=l1 = 401 (unathorized)

@2 browser (dialog/prompt)
-> /loanStmt?loanNo=l1 [req header: Authorization]

Header Name: Authorization
Header Value: Basic Base64.encoded(username:password)
For eg..
username: techsriman
password: welcome1
base64(techsriman:welcome1) = dGVjaHNyaW1hbjp3ZWxjb21lMQ==
Authorization: Basic dGVjaHNyaW1hbjp3ZWxjb21lMQ==

Note: since the username/password are just encoded not encrypted and exchanged aspart of the request headers, and even over SSL the request headers are not encrypted, the basic authentication is not safe, because any intruder can steal the username/password that is being exchanged over the request.

2. FORM based authentication
The FORM based authentication is not an HTTP authentication standard, rather it is a programming technic through which the authentication data is being exchanged between the client/server and is supported by most of the languages/technologies.

here the authentication data like username/password is being exchanged over an POST request within the request body. The server upon authenticating the user will stores the credentails within the HTTP Session object allowing/granting the user access to the resources of the application.
upon logging out the session object will be invalidated and redirected to the logout.

Note: J2EE security standard supports FORM based authentication.

3. Digest authentication
Digest is a more secured version of the basic authentication with the challenge-response procedure
3.1 The client sends a request to the server for accessing an resource
3.2 If the resource is an protected resource, the server sends back an 401 unauthorized access with a nonce back to the client
3.3 Client sends the another request to the server
generate_md5_key(nonce, username, realm, URI, password_given_by_user_to_browser)
note: md5 hash = from the hash we cannot compute the original value
3.4: The server takes username and realm plus it knows the URI the client is requesting. it lookup fro the password of that username in the database and it even owns the nonce it has send back to client
so by taking all these recomputes the generate_md5_hash(nonce, username, realm, URI, dbpassword_of_the_give_username)
and compares the md5 hash string send by the client to determine the password is valid or invalid.

```
<!— authorization roles —>
<security-constraint>
  <web-resource-collection>
    <web-resource-name>loanStatement</web-resource-name>
    <url-pattern>/loanStmt</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>customer</role-name>
    <role-name>manager</role-name>
  </auth-constraint>
</security-constraint>

<!— authentication mechanism —>
<login-config>
  <auth-method>basic</auth-method>
</login-config>

<!— total roles of our application —>
<security-role>
  <role-name>customer</role-name>
</security-role>
<security-role>
  <role-name>manager</role-name>
</security-role>
<security-role>
  <role-name>clerk</role-name>
</security-role>
```

```
[if the user is authenticated and allowed to access]

class LoanStatementServlet extends HttpServlet {
  public void service(HttpServletRequest req,
                      HttpServletResponse resp) {
    String loanNo=req.getParameter(“loanNo”);
    UserPrincipal principal = req.getUserPrincipal();

    if(principal.isUserRole(“customer”)) {
      String name = principal.getName();
      String loanOwner = dao.getLoanOwner(loanNo);
      if(name.equals(loanOwner)) {
        // generate loans statement
      } else {
        req.getRequestDispatcher(“/un-authorized.jsp”).forward(req, resp);
      }
    }else if(principal.isUserRole(“manager”)) {
      // generate statement for any loan and dispatch to client.
    }
  }
}
```

Form-Based Authentication
Form-Based authentication allows the developer to control the look and feel of the login pages and error pages that an HTTP Browser presents to the enduser.

How do we configure the FORM based authentication?
web.xml

```
<login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
    <form-login-page>/login.jsp</form-login-page>
    <form-error-page>/login-error.jsp</form-error-page>
  </form-login-config>
</login-config>
```
The login and error pages are configured relatative to the location of the deployment descriptor.

login.jsp

```
<form method=“POST” action=“j_security_check”>
   username: <input type=“text” name=“j_username”/><br/>
   password: <input type=“password” name=“j_password”/><br/>
   <button type=“submit” value=“login”/>
</form>
```