CORSO DI LAUREA IN INFORMATICA

Tecnologie Software per il Web

SECURITY

a.a. 2022-2023

Introduction

There are two major aspects to securing Web applications:

- **Preventing unauthorized users from accessing sensitive data.**
  This process involves *access restriction* (identifying which resources need protection and who should have access to them) and *authentication* (identifying users to determine if they are one of the authorized ones)
- **Preventing attackers from stealing network data while it is in transit.**
  This process involves the use of Secure Sockets Layer (SSL) to encrypt the traffic between the browser and the server

Access restriction

- The approaches to access restriction are the same regardless of whether or not you use SSL. They are:
- **Declarative security.** With declarative security none of the individual servlets or JSP pages need any security-aware code. Instead, both of the major security aspects are handled by the server (use web.xml)
- **Programmatic security.** With programmatic security, protected servlets and JSP pages at least partially manage their own security:
  - To prevent unauthorized access, each servlet or JSP page must either authenticate the user or verify that the user has been authenticated previously
  - *To safeguard network data, each servlet or JSP page has to check the network protocol used to access it* (request.isSecure())*. If users try to use a regular HTTP connection to access one of these URLs, the servlet or JSP page must manually redirect them to the HTTPS (SSL) equivalent*

Restricting Access to Web Pages

- **Main approach: "*declarative*" security via web.xml/annotation settings**
- **Alternative: programmatic HTTP**

1. Check whether there is Authorization header. If not, go to Step 2. If so, skip over word “basic” and reverse the **base64 encoding** of the remaining part. This results in a string of the form **username:password**. Check the username and password against some stored set. If it matches, return the page. If not, go to Step 2

2. Return a 401 (Unauthorized) response code and a header of the following form:
   WWW-Authenticate: BASIC realm="some-name"
   This instructs browser to pop up a dialog box telling the user to enter a name and password for some-name, then to reconnect with that username and password embedded in a single base64 string inside the Authorization header

Programmatic HTTP Security

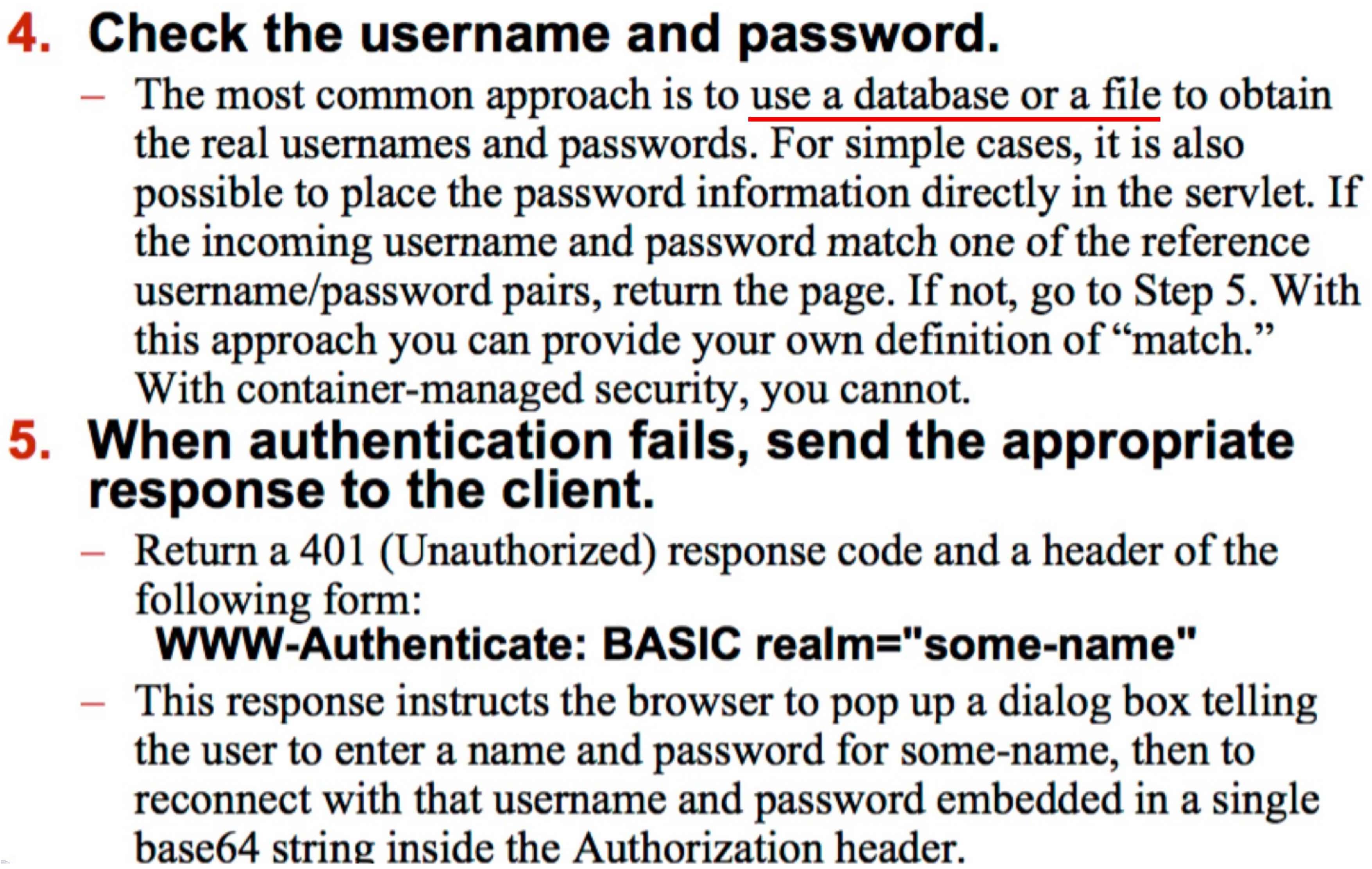
- **Idea**
  - Each protected resource authenticates users and decides what (if any) access to grant
- **Advantages**
  - Totally portable
    - No server-specific component
  - Permits custom password-matching strategies
  - No **web.xml** entries needed (except maybe url-pattern)
- **Disadvantages**
  - Much harder to write and maintain
  - Each and every resource has to use the code
    - You can build reusable infrastructure (e.g., servlets that inherit from certain classes or custom JSP tags), but it is still a lot of work

Programmatic HTTP Security (2)

1. **Check whether there is an Authorization request header.**
   – If there is no such header, go to Step 5.
2. **Get the encoded username/password string.**
   – If there is an Authorization header, it should have the following form:
      • Authorization: Basic encodedData
   – Skip over the word Basic and the space—the remaining part is the username and password represented in base64 encoding.
3. **Reverse the base64 encoding of the username/password string.**
   – Use the decodeBuffer method of the BASE64Decoder class. This method call results in a string of the form username:password. The BASE64Decoder class is bundled with the JDK; in JDK 1.3+ it can be found in the sun.misc package in *jdk_install_dir/jre/lib/rt.jar*.

```
import java.util.Base64;
Base64.getDecoder().decode(...)
```

Programmatic HTTP Security (3)

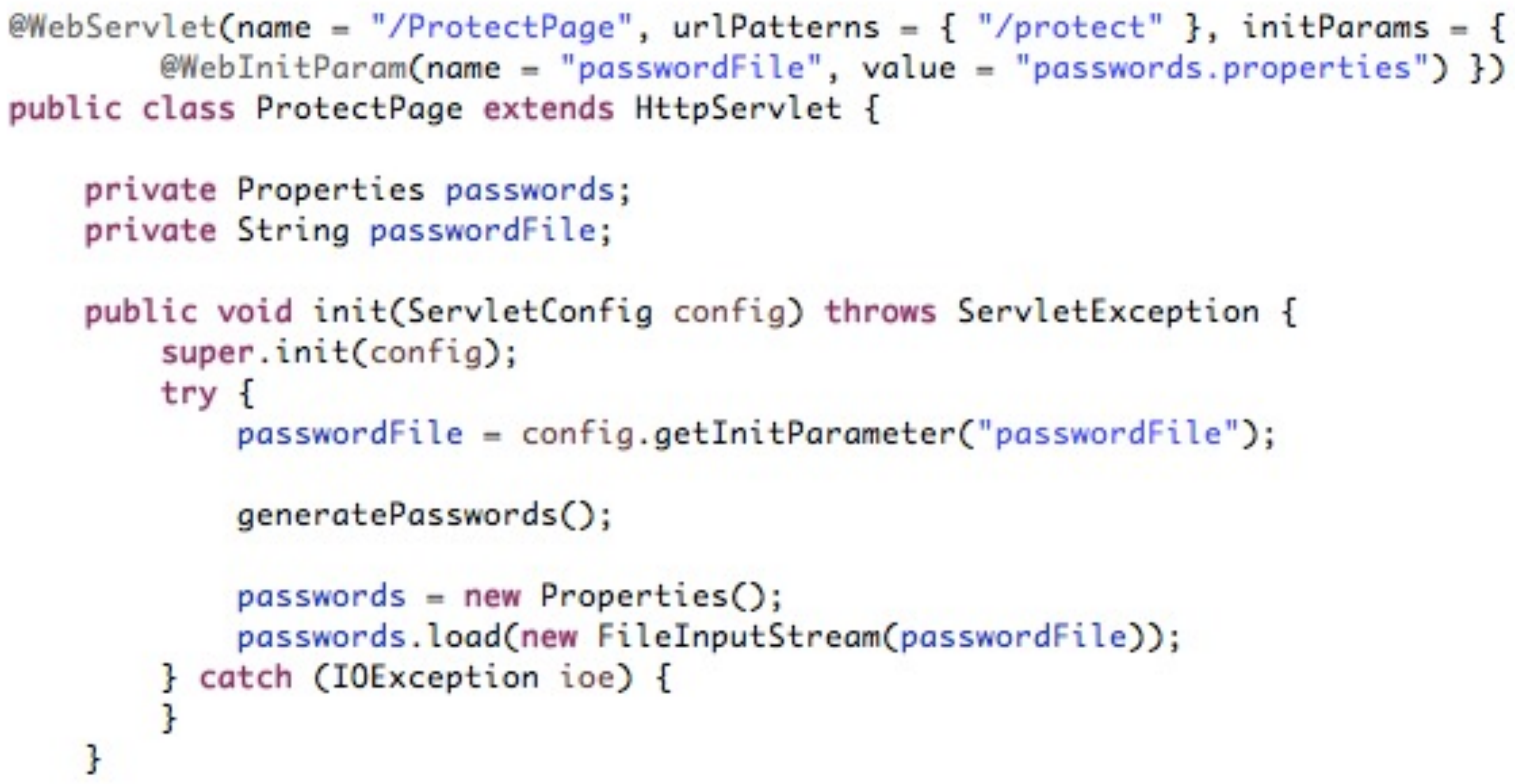


4. **Check the username and password.**
   – The most common approach is to use a database or a file to obtain the real usernames and passwords. For simple cases, it is also possible to place the password information directly in the servlet. If the incoming username and password match one of the reference username/password pairs, return the page. If not, go to Step 5. With this approach you can provide your own definition of "match." With container-managed security, you cannot.
5. **When authentication fails, send the appropriate response to the client.**
   – Return a 401 (Unauthorized) response code and a header of the following form:
     WWW-Authenticate: BASIC realm="some-name"
   – This response instructs the browser to pop up a dialog box telling the user to enter a name and password for some-name, then to reconnect with that username and password embedded in a single base64 string inside the Authorization header.

SecretServlet (Registered Name of ProtectedPage Servlet)

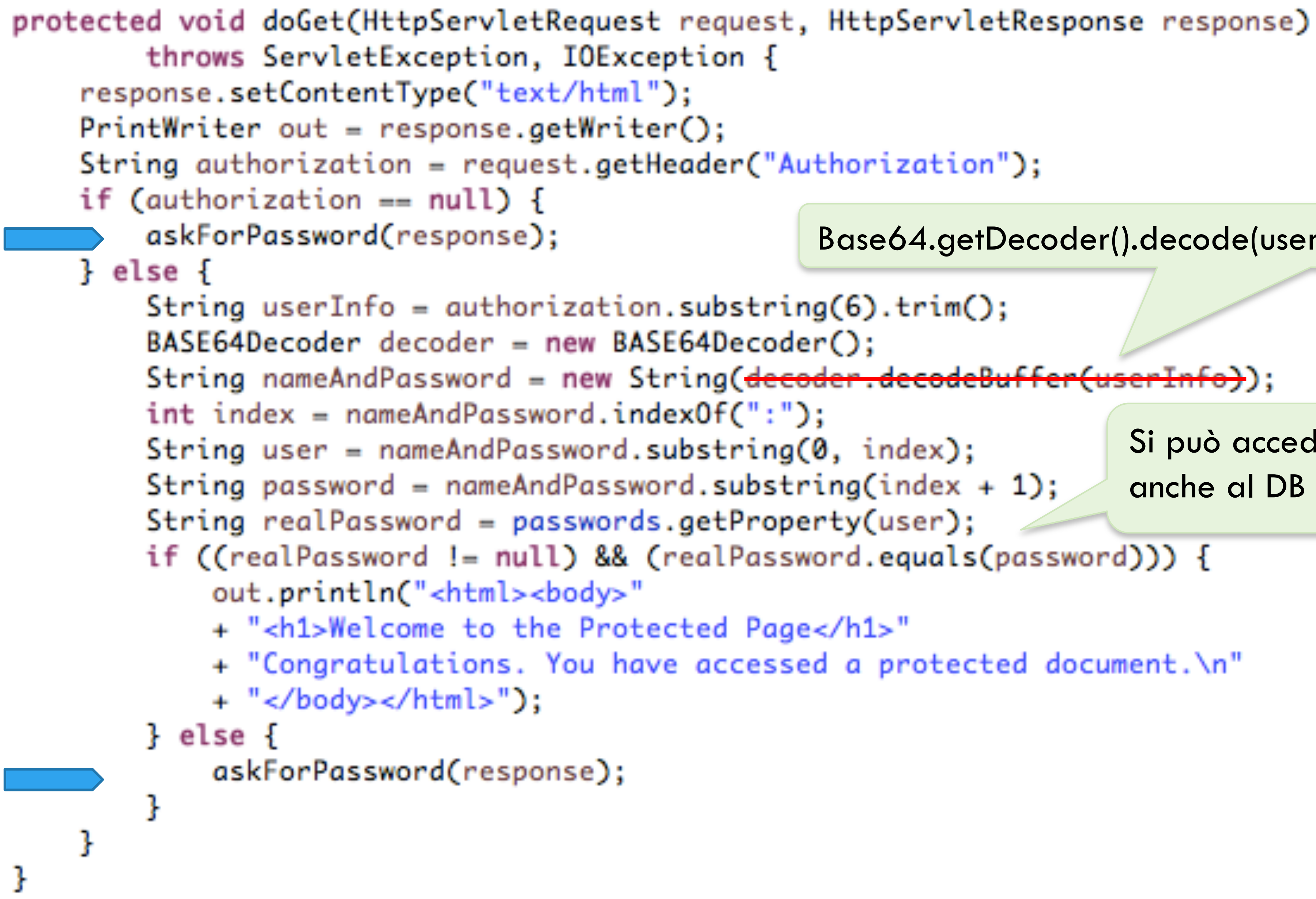
…

```
import java.util.Base64;
```



```
@WebServlet(name = "/ProtectPage", urlPatterns = { "/protect" }, initParams = {
        @WebInitParam(name = "passwordFile", value = "passwords.properties") })
public class ProtectPage extends HttpServlet {

    private Properties passwords;
    private String passwordFile;

    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        try {
            passwordFile = config.getInitParameter("passwordFile");

            generatePasswords();

            passwords = new Properties();
            passwords.load(new FileInputStream(passwordFile));
        } catch (IOException ioe) {
        }
    }
```

SecretServlet (2)



```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String authorization = request.getHeader("Authorization");
    if (authorization == null) {
        askForPassword(response);
    } else {
        String userInfo = authorization.substring(6).trim();
        BASE64Decoder decoder = new BASE64Decoder();
        String nameAndPassword = new String(~~decoder.decodeBuffer(userInfo)~~);
        int index = nameAndPassword.indexOf(":");
        String user = nameAndPassword.substring(0, index);
        String password = nameAndPassword.substring(index + 1);
        String realPassword = passwords.getProperty(user);
        if ((realPassword != null) && (realPassword.equals(password))) {
            out.println("<html><body>"
            + "<h1>Welcome to the Protected Page</h1>"
            + "Congratulations. You have accessed a protected document.\n"
            + "</body></html>");
        } else {
            askForPassword(response);
        }
    }
}
```

Base64.getDecoder().decode(userInfo)

Si può accedere anche al DB

SecretServlet (3)

Status Code 401

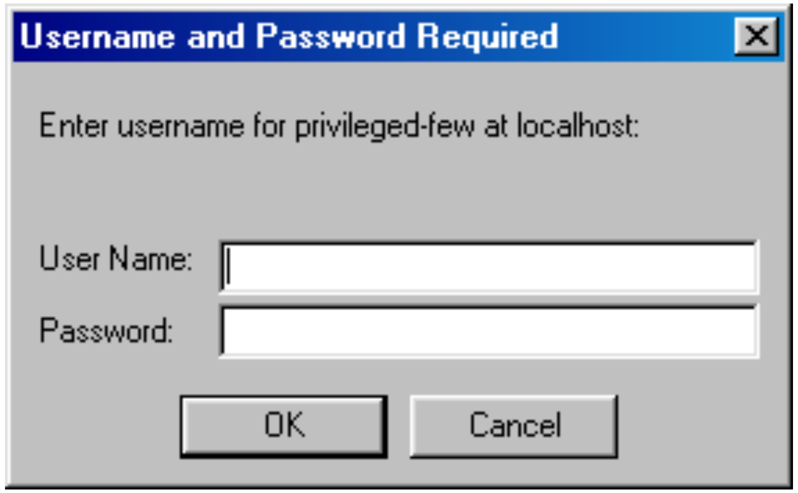
```
// If no Authorization header was supplied in the request.
private void askForPassword(HttpServletResponse response) {
    response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
    response.setHeader("WWW-Authenticate", "BASIC realm=\"privileged-few\"");
}

private void generatePasswords() {
    Properties passwords = new Properties();
    passwords.put("root", "admin");
    passwords.put("tiger", "tiger");
    try {
        FileOutputStream out = new FileOutputStream(passwordFile);
        passwords.store(out, "Passwords");
    } catch (FileNotFoundException e) {
    } catch (IOException ie) {
    }
}
```

How does it work? (Protection.zip)

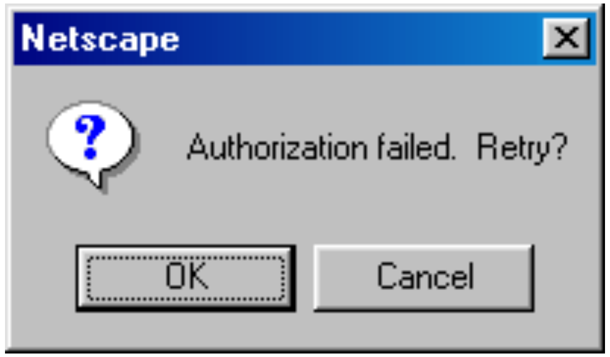
- Run: http://localhost:8080/Protection/protect

- Login: **root** pwd: **admin** (see generatePasswords() in ProtectPage.java)
- or Login: **tiger** pwd: **tiger**

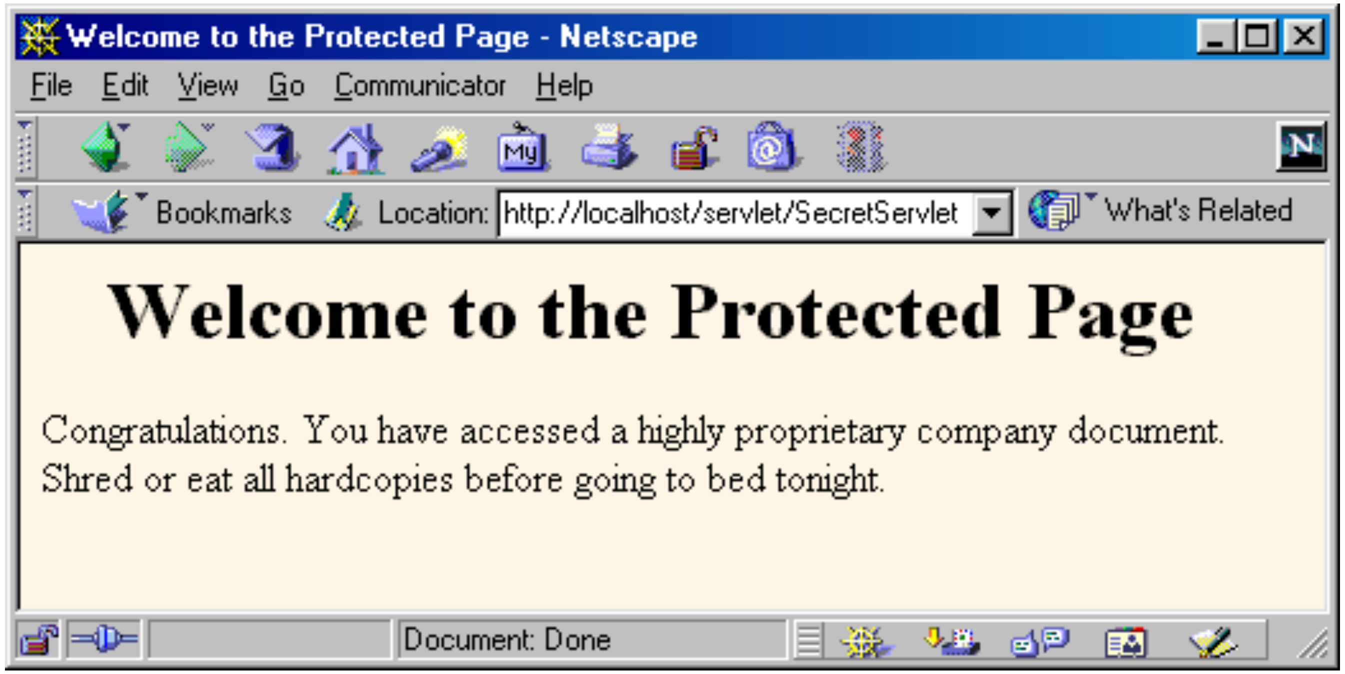
ProtectPage Servlet In Action



Enter username for privileged-few at localhost:



Authorization failed. Retry?


Welcome to the Protected Page - Netscape
File Edit View Go Communicator Help
Bookmarks
Location:
What's Related
Document: Done

http://localhost/servlet/SecretServlet

Welcome to the Protected Page

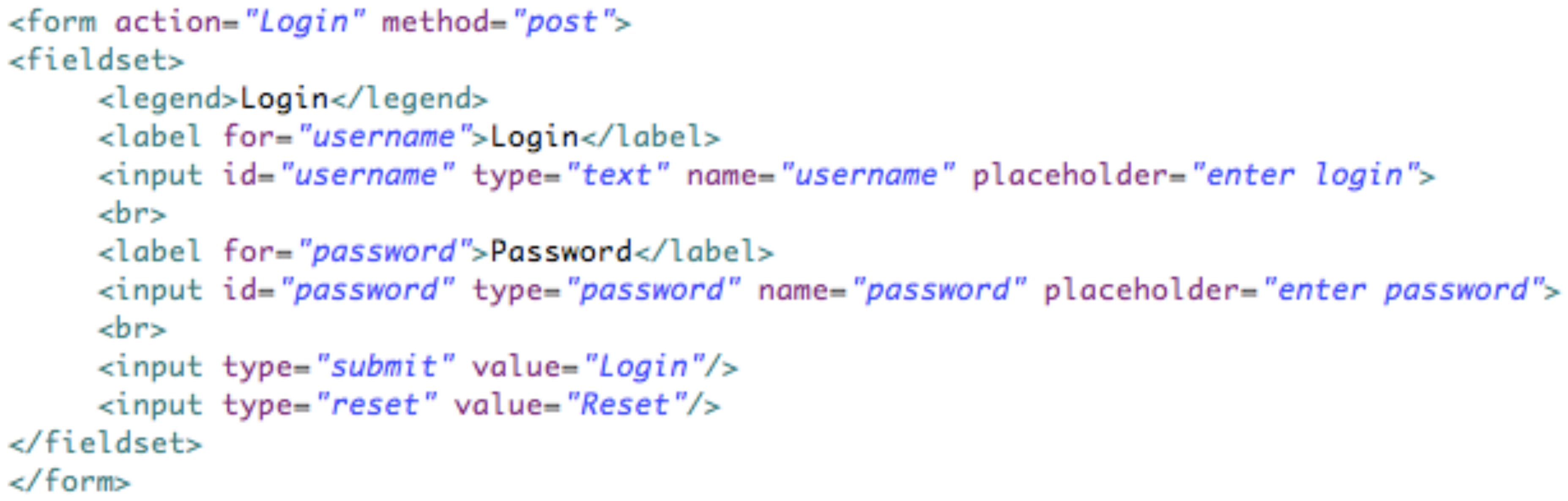
Congratulations. You have accessed a highly proprietary company document. Shred or eat all hardcopies before going to bed tonight.

SESSION AND TOKEN

MANUAL APPROACH

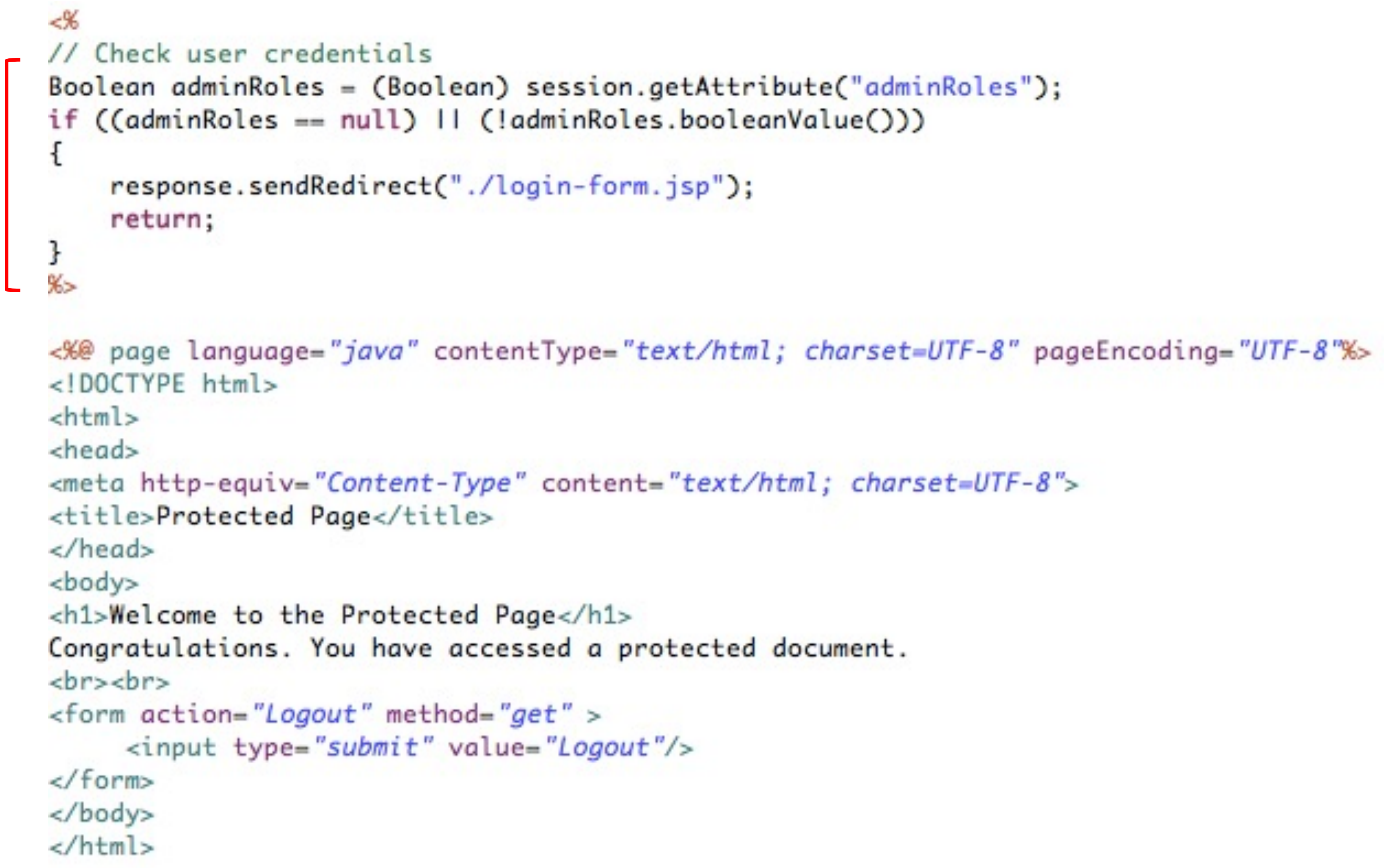
Example 1: Token in session (page login-form.jsp)

…



```
<form action="Login" method="post">
<fieldset>
     <legend>Login</legend>
     <label for="username">Login</label>
     <input id="username" type="text" name="username" placeholder="enter login">
     <br>
     <label for="password">Password</label>
     <input id="password" type="password" name="password" placeholder="enter password">
     <br>
     <input type="submit" value="Login"/>
     <input type="reset" value="Reset"/>
</fieldset>
</form>
```

Token in session (page protected.jsp)



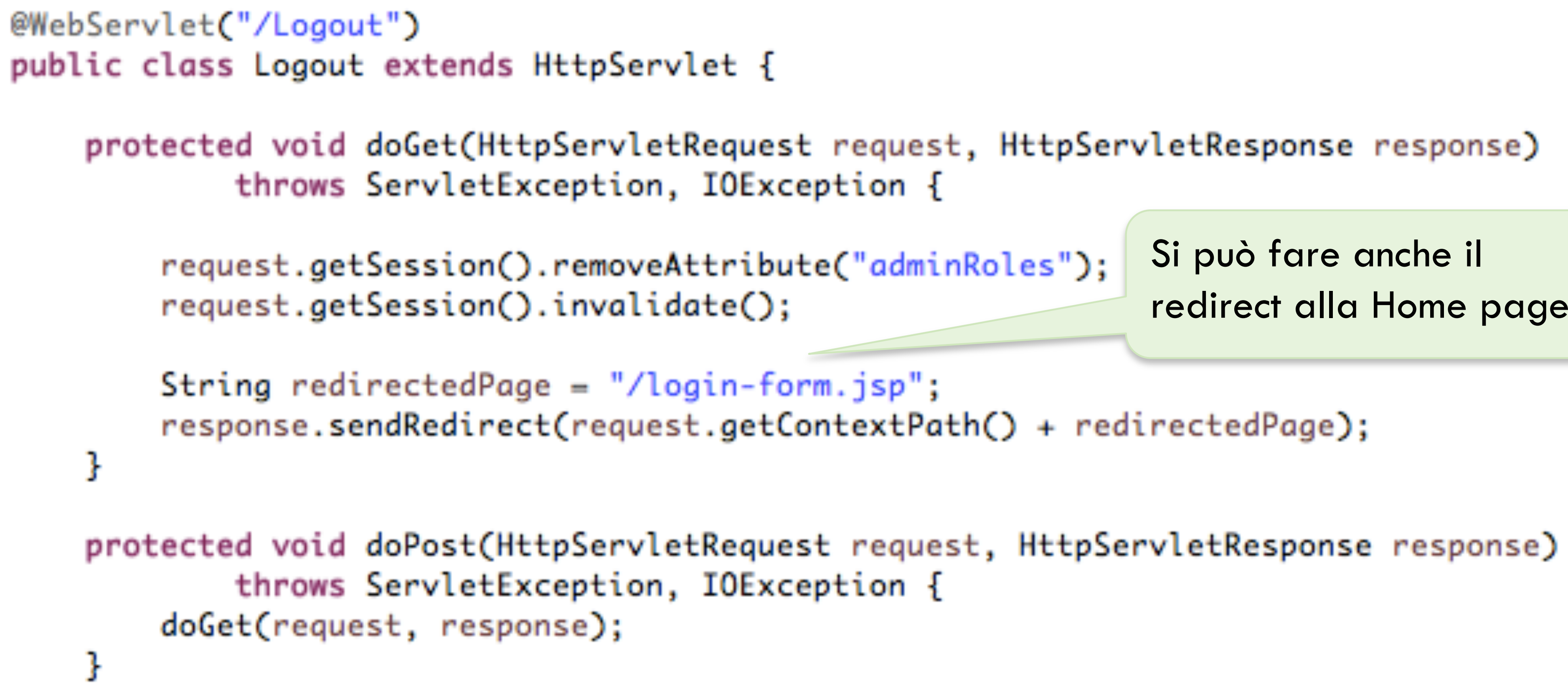
```
<%
// Check user credentials
Boolean adminRoles = (Boolean) session.getAttribute("adminRoles");
if ((adminRoles == null) || (!adminRoles.booleanValue()))
{
    response.sendRedirect("./login-form.jsp");
    return;
}
%>

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Protected Page</title>
</head>
<body>
<h1>Welcome to the Protected Page</h1>
Congratulations. You have accessed a protected document.
<br><br>
<form action="Logout" method="get" >
     <input type="submit" value="Logout"/>
</form>
</body>
</html>
```

Token in session (servlet Login)

```
@WebServlet("/Login")
public class Login extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        {
            String username = request.getParameter("username");
            String password = request.getParameter("password");
            String redirectedPage;
            try {
                checkLogin(username, password);
                request.getSession().setAttribute("adminRoles", new Boolean(true));
                redirectedPage = "/protected.jsp";
            } catch (Exception e) {
                request.getSession().setAttribute("adminRoles", new Boolean(false));
                redirectedPage = "/login-form.jsp";
            }
            response.sendRedirect(request.getContextPath() + redirectedPage);
        }
    }

    private void checkLogin(String username, String password) throws Exception {
        if ("root".equals(username) && "admin".equals(password)) {
            //
        } else
            throw new Exception("Invalid login and password");
    }
```

Si può accedere anche al DB

Token in session (servlet Logout)



```java
@WebServlet("/Logout")
public class Logout extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        request.getSession().removeAttribute("adminRoles");
        request.getSession().invalidate();

        String redirectedPage = "/login-form.jsp";
        response.sendRedirect(request.getContextPath() + redirectedPage);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
```

Si può fare anche il redirect alla Home page

- *Filter can be used to control the access to resources*