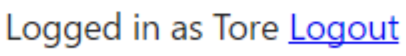
Log-in and Cookies exercise

1. In this exercise we will learn more about **cookies** and **session** by creating a simple web based **login system.** In real world applications, we would probably do this in a more secure way but it is still fundamental to understand the principles for logging in and session management.

The theory

2. On this page http://demo.edument.se/csrf you find a very simple login site. After logging in with your name, the private area of the site will display your name at the top and a logout link, like:



Logged in as Tore Logout

3. Explore using Fiddler what happens In the request and response headers and try to understand:
    a. What happens when we login?
    b. When is the login cookie set?
    c. If you see many cookies, which cookie is the session cookie?
    d. What happens if you delete the cookie from the browser?
    (Explore how you can find and remove an individual cookie for edument.se, depends on which browser you are using)
    e. Experiment with while logged in, what happens if you use the Fiddler composer to make a web-request to http://demo.edument.se/CSRF that:
        i. Have a different cookie value?
        ii. Have the same cookie value?
        Having the same cookie value, should let you see the "Logged in as…" text in the response of the request.
        iii. Have no cookie?

    You can drag existing request to the composer tab in Fiddler, then modify it and then press execute to send the request. Read about the composer here:

    http://docs.telerik.com/fiddler/Modify-Traffic/Tasks/CustomizeRequest
    http://docs.telerik.com/fiddler/Generate-Traffic/Tasks/CreateNewRequest

    f. What happens when we log out from the site?
    Can you make a request to the site with the session cookie? Try it!

In practice:

Let's implement our own simple but fundamental login system

1. Read about the Java HttpSession object:
   http://web.cs.ucla.edu/classes/winter15/cs144/projects/java/session/index.html

2. Keep Fiddler running all the time
3. Create a new Spring-Boot web project
4. Create a new default start page that contains a link named "**Access secret area"** that links to the **/secret** page
5. Create a new /**secret** page that should be password protected. In the corresponding action method add a check like:

```
if (session.getAttribute("user") != null)
     //logged in, Display the logged in username
else
    //nog logged in, redirect user to /login page
```

   The page should display the "logged in" username on the screen, take the username from the Session object.

   Tip: In a Spring Boot app, the Session object can be obtained by just specifying it as an input argument to a method. Spring Boot will inject a Session object for you. Just specify HttpSession session as an input argument.

6. Create a **/login** page with a typical log-in form, with the following fields:
   User name
   Password

   Make the password field so that it does not show the text entered and also add a login button. Make sure the form make a HTTP post when it submits the form data back.

7. In the controller for the login page, make sure the username and password matches some pre-defined login, like login "admin" and password "123".

   If the username/password does not match, then display the login form again and the form should remember the username and the password field should be blank.

   If it matches, then:
   - Save the username in the session object
   - redirect the user to the **/secret** page

Also make sure in Fiddler that the session cookie is set when logging in.

8. On the secret page add a **logout link** and the corresponding method should just delete the cookie, using:

```
Cookie cookie = new Cookie("jsessionid", null);
cookie.setMaxAge(0);
res.addCookie(cookie);
```

   After logout you are redirected to the standard home/start-page

9. Test your application and you should now be able to login and logout from the secret area via the links and verify:
   a. That the session cookie is set when you login
   b. The cookie is removed when you logout

Terminating the session

1. When we logout from the application, we delete the cookie from the browser! But is that really enough?
2. No, try to go back in the fiddler request list and find a request to **/secret** with the **cookie** in it
   a. Drag it to the Composer tab
   b. Execute it again
   Can you still access the secret page if you provide the cookie?

3. As the session is still valid on the server, you should still be able to access the secret area and this is a fairly common mistake that you kill the login cookie, but not the session on the server.

   To properly kill the session you can use the `session.invalidate();` method, add it to the logout code and try step 1 again.

   Now even if you have a valid cookie, you can’t login to the server.

More advanced login

1. Many web-sites like Svenska Dagbladet (http://www.svd.se/)lets you browse most of the without having to log-in. But every page also contains a login link that allows you to login and after login you get redirected back to the page you visited before logging in.
2. Visit any news article on SVD, like http://www.svd.se/naringsliv/motor and then inspect the login icon at the top right part of the page:

   As you see the link of the current page is part of the query string, so that the login page can know where to redirect to after a successful login.
3. Your assignment is to replicate this behavior. Implement this by:
   a. Create a few normal web pages
   b. On each page have a login link that contains the current page URL as a query-string parameter. The link should link to the /login page.
   c. After successful login, **redirect** the user back to the link passed into the query string. If the string is missing, then redirect to the start page.
   d. If you are logged in, **replace the login link** with the **username** at the top of the page together with a **logout** link. The logout link should also contain the current URL.
   e. Verify that it works, that you can login and logout from any of your pages without losing the current page.
4. Passing in a URL to redirect the user to via the query string have some security issues, read this article about the problems:
   Unvalidated Redirects and Forwards Cheat Sheet
   https://www.owasp.org/index.php/Unvalidated_Redirects_and_Forwards_Cheat_Sheet

Further reading:

Creating a secure login/session management system is hard, read more about the issues here:

- Session Management Cheat Sheet
  https://www.owasp.org/index.php/Session_Management_Cheat_Sheet
- Broken Authentication and Session Management
  https://www.owasp.org/index.php/Top_10_2013-A2-Broken_Authentication_and_Session_Management