Lab 6 - Session Timeout

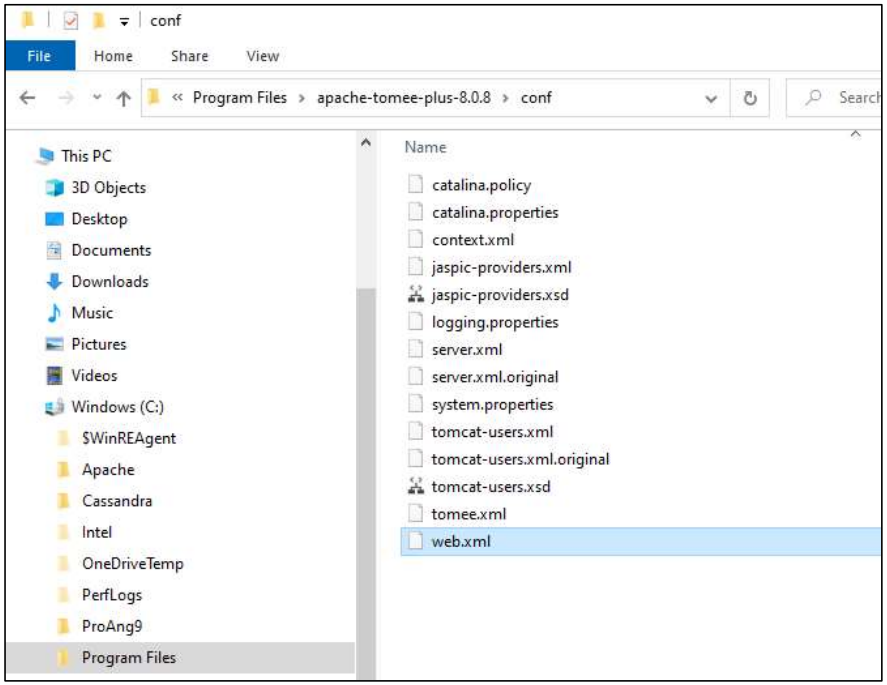
In this lab, we examine the following three ways to control session timeout

- Determine the container default session timeout value (using server config file, web.xml)
- Customize our session timeout for a particular servlet to be a low value and observe the timeout in action (using code in doXXX method)
- Customize our session timeout for our application to be a low value and observe the timeout in action (using a ServletContextListener)

Lab 6 - Session Timeout

Determine the container default session timeout value (using server config file, web.xml)

- Locate web.xml file
- <Tomcat install directory>/conf/web.xml
- Inside web.xml find session-timeout and checkout the value




```
<session-config>
  <session-timeout>30</session-timeout>
</session-config>
```

Just look at your session-timeout value, system admins will typically change this as needed but it's good for you to be aware

Lab 6 - Session Timeout

Customize our session timeout for a particular servlet to be a low value and observe the timeout in action (using code in doXXX method)

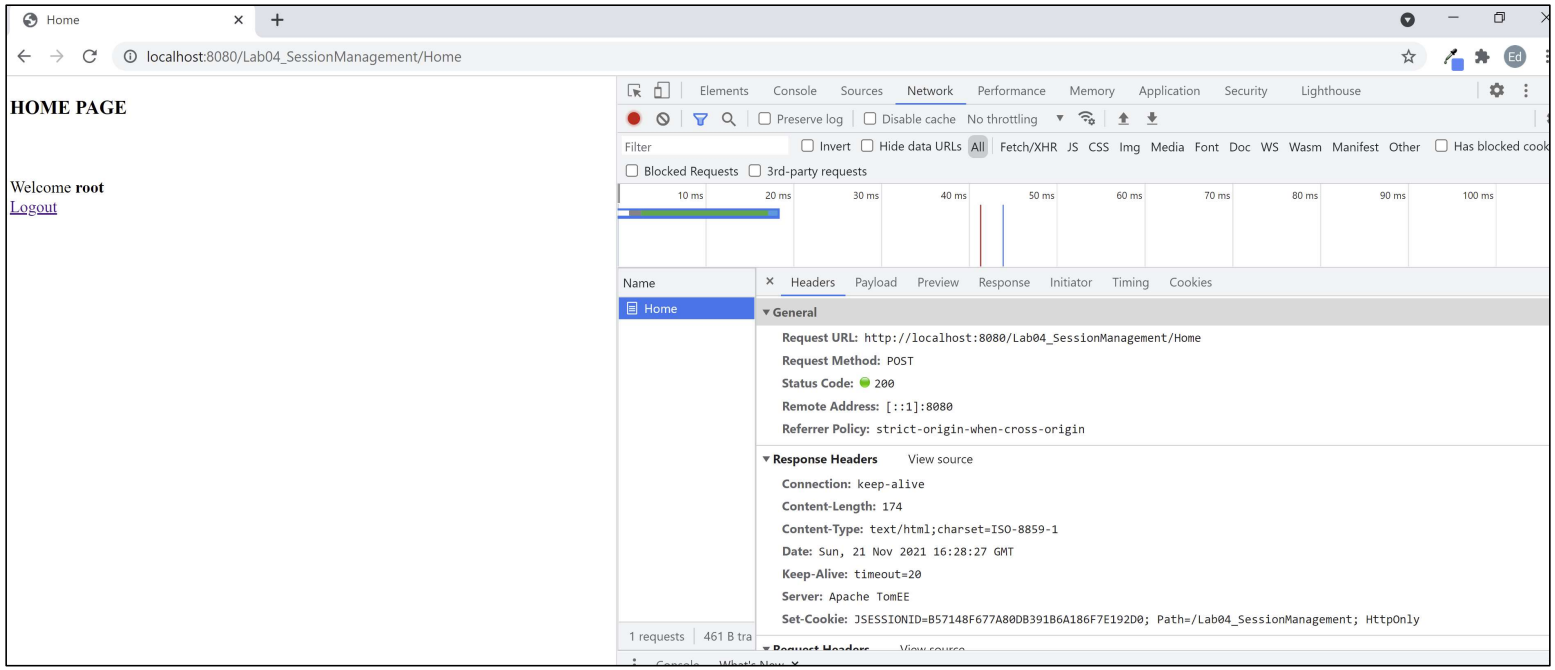
- In Home Servlet, doGet add the two highlighted lines of code below

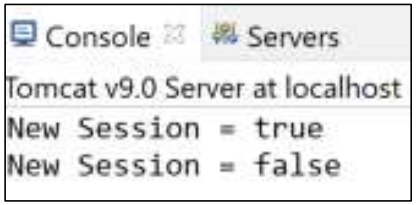
```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

// get session HttpSession session = request.getSession();

System.out.println("New Session = " + request.getSession().isNew());
request.getSession().setMaxInactiveInterval(60);
```

Lab 6 - Session Timeout

Customize our session timeout for a particular servlet

- Observe that there is only a JSESSIONID in the response
- Also, note that our New Session message returns true
- If you refresh the page in less than a minute, you will notice the request sends the same JSESSIONID and the New Session message returns false





Lab 6 - Session Timeout

Customize our session timeout for a particular servlet

- Wait at least 60 seconds and refresh the page
- This time we send our JSESSIONID to the server, but the session is gone
- A new session is created (see that New Session returns true)
- A new JSESSIONID is returned in the response

Response Headers View source

Connection: keep-alive
Content-Length: 174
Content-Type: text/html;charset=ISO-8859-1
Date: Sun, 21 Nov 2021 16:34:20 GMT
Keep-Alive: timeout=20
Server: Apache TomEE
Set-Cookie: JSESSIONID=65D6507ACAA3DD14E67BD26ADC07E6C6; Path=/Lab04_SessionManagement; HttpOnly

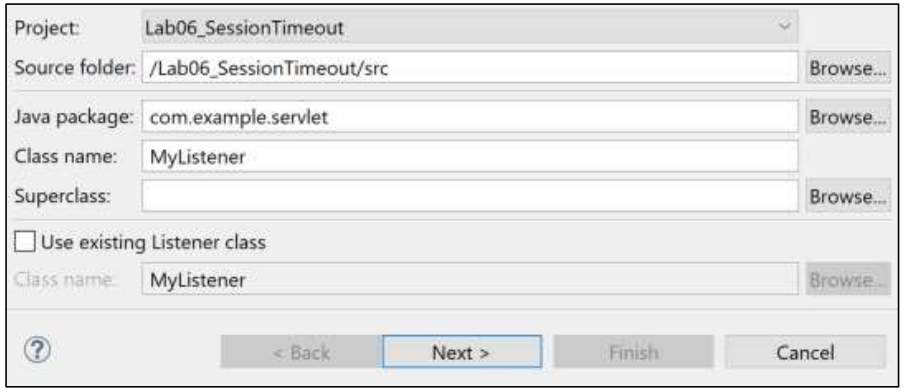
Request Headers View source

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cache-Control: max-age=0
Connection: keep-alive
Content-Length: 35
Content-Type: application/x-www-form-urlencoded
Cookie: JSESSIONID=B57148F677A80DB391B6A186F7E192D0

Lab 6 - Session Timeout

Customize our session timeout for our application to be a low value and observe the timeout in action (using a ServletContextListener)

- **Create a Web Listener**
  - Right click on on the com.example.servlet package > New > Other …
  - Search for "Listener" and select Listener under Web, and Select Next
  - Within the Create Listener window enter a Class name of "MyListener", and then select "Next"
  - Select servlet context events > Life cycle then select HTTP session events > Lifecycle
  - Select Finish



Lab 6 - Session Timeout

Edit the Web Listener file and modify the sessionCreated, sessionDestroyed, and contextIniitalized as shown below

```
public void sessionCreated(HttpSessionEvent se) {
    System.out.println("A new HttpSession was created");
}
```

```
public void sessionDestroyed(HttpSessionEvent se) {
   System.out.println("An HttpSession was destroyed");
}
```

```
public void contextInitialized(ServletContextEvent sce) {
    ServletContextListener.super.contextInitialized(sce);

    sce.getServletContext().setSessionTimeout(2); // 2 minutes
    System.out.println("contextInitialized() called");
}
```

Lab 6 - Session Timeout

Test

- Right click on your index.html file and run the app on the server
  - Note the contextInitialized method is called
- Login as before (root, pa$$word)
  - Note a new HttpSession was created
  - Our "New Session = true" message runs
- Wait at least 2 minutes without interacting with your browser
  - You will then see the session destroyed message