04. HttpSession

Date	@23/11/2022
Categoria	Java para Web
Curso	Java Servlet: autenticação autorização e o padrão MVC

Tópicos

- Projeto da aula anterior
- Trabalhando com sessão
- Detectando o problema
- A solução dos nossos problemas
- Testando o login
- Autorizando o acesso
- Implementando o logout
- Sabendo as diferenças
- Para saber mais: Timeout da sessão
- Faça como eu fiz na aula
- O que aprendemos?

Detectando o problema

No início da aula, tentamos salvar as informações do usuário, utilizando o `setAttribute`, mas vimos que isso não é possível. Por qual motivo não conseguimos fazer dessa forma?

- Cada requisição é tratada isoladamente, fazendo com que na próxima, já não tenhamos os dados do usuário.

A solução dos nossos problemas

Vimos que não conseguimos manter as informações do usuário disponíveis durante o uso da aplicação web, já que cada requisição é tratada de maneira isolada. Para conseguirmos controlar e saber qual navegador/usuário está acessando e se ele já realizou o login, precisamos encontrar outra estratégia. Qual foi essa estratégia?

- Precisamos utilizar o `session id` gerado pelo Tomcat para identificar a sessão do usuário e controlar o acesso.

Sabendo as diferenças

Quando implementamos o logout, fizemos de duas formas, removendo todos os atributos com o `removeAtribute` ou invalidando a sessão com o `invalidate`. Qual a diferença entre essas duas abordagens?

- Quando utilizamos o `removeAtribute`, nosso objeto `HttpSession` ainda continua em memória (o cookie também continua lá).
- Já quando usamos o `invalidate`, ele remove o objeto `HttpSession`, todos os objetos associados e também remove o cookie!

Para saber mais: Timeout da sessão

Imagina que você acessou hoje a nossa aplicação em 3 computadores diferentes. Isso significa que o servidor Tomcat criou no mínimo 3 objetos `HttpSession` para você só nesse dia, pois cada navegador é identificado por um ID de sessão diferente.

Pior ainda, quando você fecha o navegador, esse cookie com ID será automaticamente removido, pois trata-se de um ***session cookie*** que só vive enquanto o navegador está aberto (é um cookie transiente).

Vamos pensar que você adora fechar e abrir o navegador e fez isso 5 vezes em cada computador. Ou seja, fazendo o cálculo:

```
3 computadores x 5 vezes navegador reaberto = 15 objetos HttpSession
```

Foram então 15 objetos `HttpSession` só para você nesse dia. Imagine isso para uma aplicação com muitos acessos, por exemplo 1000 usuários por dia! Seriam 15000 objetos `HttpSession` em memória, em um dia!

Realmente, seriam milhares de objetos em memória, mas saiba que o objeto `HttpSession` tem um ciclo de vida. Isso significa que ele será criado mas também será destruído.

Acontece que a `HttpSession` tem um *timeout* associado. Se você não usa a nossa aplicação por um determinado tempo, o Tomcat automaticamente remove o objeto `HttpSession` da memória. O padrão do Tomcat 9 é de 30 minutos, ou seja, se você não usar a aplicação por 30 min, você será deslogado!!

Talvez você ache os 30 minutos pouco ou muito tempo, mas saiba que isso é configurável através do nosso **web.xml**, basta colocar o seguinte trecho:

```
<session-config>
    <!-- 10 min -->
    <session-timeout>10</session-timeout>
</session-config>
```

Só reforçando: é um *timeout de desuso*. No caso acima, o Tomcat só removerá a sessão se o usuário não ficar ativo por 10 minutos.

Obs: Você já deve ter percebido que a Alura não perde o seu login quando você reabre o navegador. Isto acontece pois a Alura não usa o cookie padrão e sim cria o seu próprio cookie persistente (e não transiente).

O que aprendemos?

Essa aula foi dedicada à autenticação e autorização dentro de uma aplicação web.

Aprendemos que:

- Por padrão, o navegador não envia nenhuma identificação sobre o usuário
- Quando o Tomcat recebe uma nova requisição (sem identificação), gerará um ID
- O ID fica salvo no cookie de nome `JSessionID`
- O ID é um *hash* (número aleatório)
- O cookie é anexado à resposta HTTP
- O navegador reenvia o cookie automaticamente nas próximas requisições
- O Tomcat gera, além do ID, um objeto chamado `HttpSession`
- A vida do objeto `HttpSession` fica atrelado ao ID
- Para ter acesso à `HttpSession`, basta chamar `request.getSession()`
- Usamos a `HttpSession` para guardar dados sobre o usuário (login, permissões, carrinho de compra)
- A `HttpSession` tem um ciclo de vida e será automaticamente invalidada

Com esse conhecimento, conseguimos proteger a nossa aplicação e criar um login, logout e autorizar o acesso.
