07

Implementando o logout

Transcrição

Bem vindo de volta! Antes de revisarmos a localização da nossa autenticação, vamos rapidamente implementar uma funcionalidade de **logout**. Como logout também é uma ação, precisaremos criar uma nova classe (`Logout`) no nosso pacote de ações, da mesma forma que fizemos anteriormente (lembrando de implementar a interface `acao`):

```
public class Logout implements Acao {

    @Override
    public String executa(HttpServletRequest request, HttpServ
            throws ServletException, IOException {

        return null;
    }

}
```

COPIAR CÓDIGO

Como o logout funciona? Normalmente, quando o usuário clica em um botão de sair, ele é redirecionado para uma página de login ou para alguma mensagem. Nesse caso, faremos um redirecionamento para `LoginForm` .

Existem duas maneiras de implementarmos o logout, mas de qualquer forma precisaremos, antes de tudo, trabalhar com o objeto `HttpSession` que é criado especificamente para o acesso daquele usuário:

```
public class Logout implements Acao {

    @Override
    public String executa(HttpServletRequest request, HttpServl
            throws ServletException, IOException {

        HttpSession sessao = request.getSession();

        return "redirect:entrada?acao=LoginForm";
    }

}
```

COPIAR CÓDIGO

Na implementação que fizemos para o login, tudo é decidido com base no atributo `usuarioLogado` : se esse atributo existe dentro da sessão, o usuário fez login; do contrário ele precisa se autenticar.

Portanto, a primeira ideia (e talvez a mais simples) é removermos esse atributo da `sessao` :

```
public String executa(HttpServletRequest request, HttpServletR
        throws ServletException, IOException {

    HttpSession sessao = request.getSession();

    sessao.removeAttribute("usuarioLogado");
    return "redirect:entrada?acao=LoginForm";
}
```

COPIAR CÓDIGO

Quando removemos o atributo, nosso Servlet não irá mais encontrá-lo em uma próxima requisição. Se o usuário tentar acessar alguma ação protegida, será redirecionado de volta para `LoginForm` .

A ação já está pronta, mas faz sentido oferecermos, no nosso HTML, um link para que o usuário execute essa ação. Portanto, na nossa página central ("WEB-INF > view > listaEmpresas.jsp"), criaremos um novo link logo abaixo do elemento `<body>` :

```
<body>
    <a href="entrada?acao=Logout">Sair</a>
    <br>
    <br>
    <br>
    Usuario Logado: ${usuarioLogado.login }
```

COPIAR CÓDIGO

Ou seja, temos um botão **Sair** que chama a ação `Logout` por meio do nosso controlador. Vamos testar? Clicando no botão "Sair", seremos redirecionados para `LoginForm` . Consequentemente, se tentarmos acessar uma ação protegida diretamente, também seremos redirecionados para `LoginForm` , o que significa que o atributo `usuarioLogado` foi removido com sucesso.

Agora veremos outra forma de implementarmos o logout. Na nossa `HttpSession` , não só guardamos o atributo `usuarioLogado` , mas também outros atributos, como permissões específicas ou um carrinho de compras (no caso de uma loja).

Portanto, deveríamos chamar o método `removeAttribute()` para cada um deles. Existe uma forma mais elegante de fazermos isso: por meio do método `invalidate()` , que remove o objeto `HttpSession` e todos os objetos associados a ele, ao mesmo tempo em que destrói o cookie. Com o método `removeAttribute()` , nós estávamos removendo o atributo, mas o objeto `HttpSession` continuava na memória.

```
public String executa(HttpServletRequest request, HttpServle
        throws ServletException, IOException {
```

```
        HttpSession sessao = request.getSession();

        //sessao.removeAttribute("usuarioLogado");
        sessao.invalidate();
        return "redirect:entrada?acao=LoginForm";
    }
```

COPIAR CÓDIGO

Podemos testar isso no navegador observando o comportamento dos cookies com as "Ferramentas do desenvolvedor".

Quando enviamos uma ação, consequentemente é criado um cookie `JSESSIONID` com um valor associado. Ao fazermos o login, continuamos com o mesmo valor para `JSSESIONID` . Porém, quando usamos o link "Sair", o navegador envia esse mesmo cookie na requisição, mas na resposta é enviado um novo cookie - ou seja, foi criada uma nova sessão Http.

Por fim, precisamos mostrar o link "Sair" em todas as páginas. Para não simplesmente copiarmos e colarmos o código HTML, criaremos uma nova página dedicada ao logout, que chamaremos de `logout-parcial.jsp` - parcial pois essa não será uma página HTML completa, mas apenas um pedaço de HTML. Nessa página teremos somente o código relacionado ao logout:

```
<a href="entrada?acao=Logout">Sair</a>
<br>
<br>
<br>
```

COPIAR CÓDIGO

Para que as outras páginas da nossa aplicação utilizem essa página parcial, usaremos uma tag da biblioteca *core* do JSTL, que é `c:import url="` :

```
<c:import url="logout-parcial.jsp"/>

Usuario Logado: ${usuarioLogado.login }
```

COPIAR CÓDIGO

Repetiremos essa tag nas outras páginas (exceto `LoginForm`) para que o botão "Sair" apareça em todas elas. Pronto! Conseguimos implementar o logout na nossa aplicação.

Na próxima aula, discutiremos a questão do nosso código de autorização e também veremos um novo recurso do mundo Servlet. Até lá!