02

Trabalhando com sessão

Transcrição

Bem vindo de volta! O objetivo agora é resolvermos nosso problema de autenticação e autorização. Mas que problema é esse mesmo?

Nós já criamos a tela de login, mas nossa aplicação ainda continua com acesso livre à `ListaEmpresas`. Porém, através da lógica por trás da nossa página de login, já conseguimos saber se o usuário existe ou não (ou seja, se o usuário se autenticou com as credenciais corretas).

Ainda gostaríamos de criar uma autorização. Dessa forma, somente o usuário que se autenticou poderá acessar as nossas ações. Para isso acontecer, a aplicação precisa, primeiramente, se lembrar de que o usuário que efetuou login.

Por exemplo, quando fazemos o login na Alura, as ações desse site não exigem mais o login. Isso pode parecer simples, mas não acontece tão automaticamente.

Na ação `Login`, nossa condição `if` permite reconhecermos se o usuário existe. Agora vamos lembrar desse usuário e pendurá-lo na requisição usando o método `request.setAttribute()`. Além disso, vamos nomeá-lo como `usuarioLogado` e colocar o objeto `usuario` lá dentro:

```
if(usuario !=null) {
    System.out.println("Usuario existe");
    request.setAttribute("usuarioLogado", usuario);
```

```
        return "redirect:entrada?acao=ListaEmpresas";
    } else {
        return "redirect:entrada?acao=LoginForm";

    }
```

COPIAR CÓDIGO

Quando o usuário faz o login corretamente, ele é redirecionado para `ListaEmpresas`. Porém, também queremos, nessa página, mostrar que o usuário está logado. Para isso vamos mexer um pouco no `listaEmpresas.jsp`. Abaixo do `<body>`, escreveremos `Usuario Logado:`, e vamos imprimir (através da linguagem de expressão `${}`) `usuarioLogado` acessando o `login`, que é o nome do atributo na classe `usuario`:

```
<%@ page language="java" contentType="text/html; charset=ISO-88
    pageEncoding="ISO-8859-1"%>
<%@ page import="java.util.List,br.com.alura.gerenciador.model
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Java Standard Taglib</title>
</head>
<body>

    Usuario Logado: ${usuarioLogado.login }

    <br>
    <br>
    <br>

    <c:if test="${not empty empresa}">
        Empresa ${ empresa } cadastrada com sucesso!
```

```
        </c:if>

        Lista de empresas: <br />

        <ul>
            <c:forEach items="${empresas}" var="empresa">

                <li>
                    ${empresa.nome } - <fmt:formatDate value="${em
                    <a href="/gerenciador/entrada?acao=MostraEmpre
                    <a href="/gerenciador/entrada?acao=RemoveEmpre
                </li>
            </c:forEach>
        </ul>

    </body>
    </html>
```

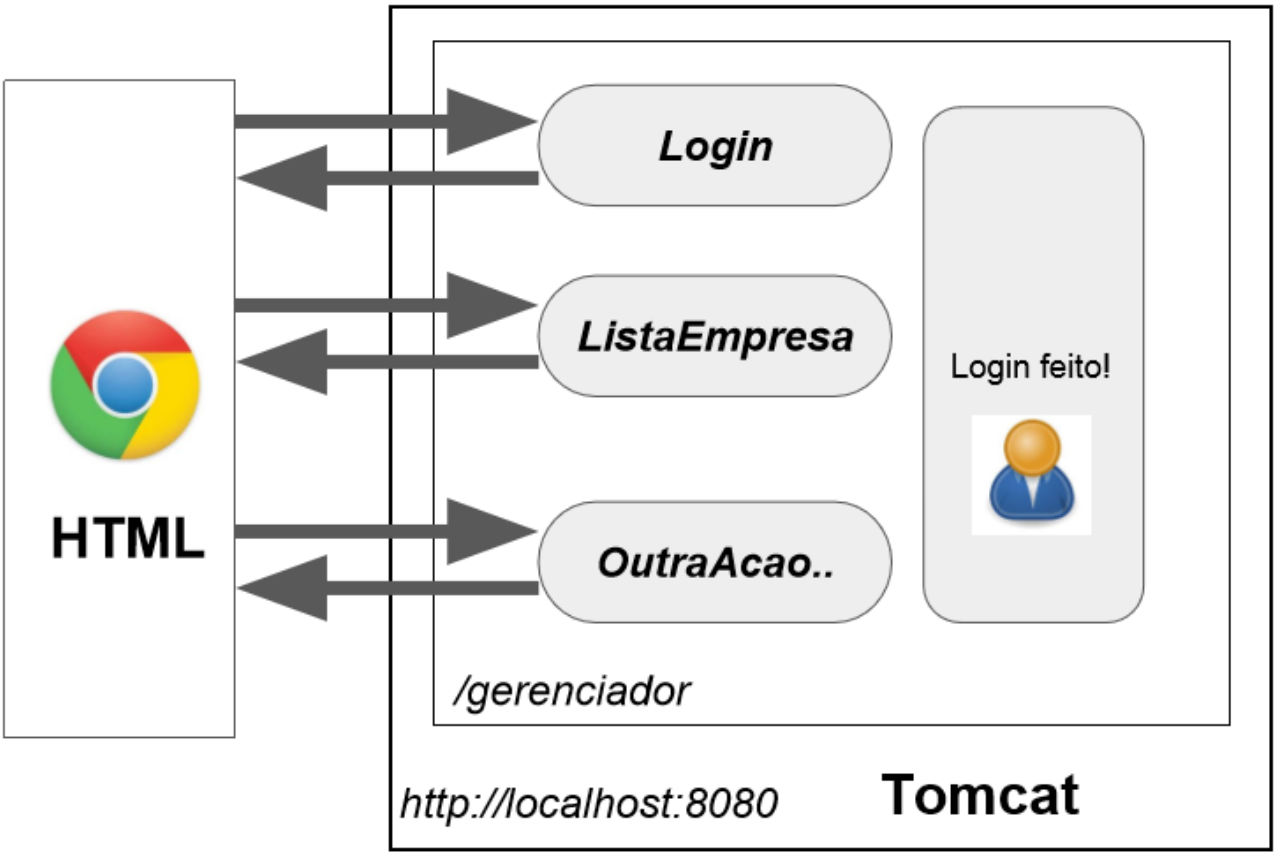
COPIAR CÓDIGO

Será que isso irá funcionar?

Vamos testar fazendo o login na página http://localhost:8080/gerenciador/entrada?acao=LoginForm (http://localhost:8080/gerenciador/entrada?acao=LoginForm). Seremos redirecinados para `ListaEmpresas` , e será exibido na tela:

> Usuario Logado:
>
> Lista de empresas:
>
> Alura - 25/10/2018 edita remove
>
> Caelum - 25/10/2018 edita remove

Por mais que a mensagem "Usuario Logado" apareça, nosso login não é mostrado, mesmo que nosso usuário exista e tenha sido colocado na ação. Mas por que não funcionou? Porque o objeto `request` (e consequentemente todos os atributos que são jogados dentro dele) só existe por uma requisição.



Vamos recapitular: a nossa requisição HTML chega na ação `Login`, e nessa ação sabemos que o usuário fez o login. Porém, após essa requisição, é feito um redirecionamento para o navegador e uma nova requisição é enviada para chamar `ListaEmpresas`. Portanto, o que era válido na requisição para a ação `Login` não é mais válido, pois temos um novo objeto.

Ou seja, no protocolo HTTP, cada requisição é tratada de forma totalmente isolada e é necessário repetir todas as informações. Como resolveremos esse problema?

Quando trabalhamos com HTTP, é criada automaticamente uma identificação do navegador - que na verdade é uma identificação do usuário -, que é chamada de **session ID**.

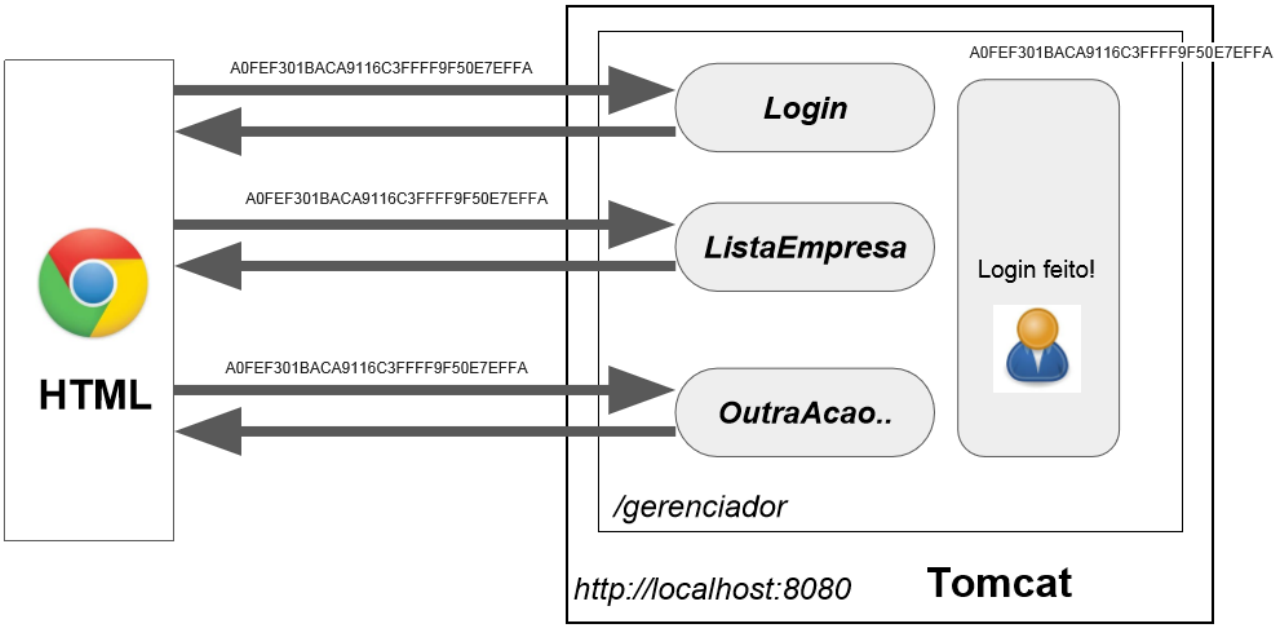
Ou seja, quando o Tomcat recebe uma requisição para a qual ele não tem uma identificação associada, ele percebe que não conhece essa requisição e automaticamente cria e devolve uma identificação para o navegador. Dessa forma, nas próximas requisições, ele sempre enviará essa mesma identificação. A partir dela, o Tomcat sabe que está se comunicando sempre com o mesmo navegador e com o mesmo usuário.

Para testarmos isso, vamos abrir uma aba anônima do navegador e acessar a URL http://localhost:8080/gerenciador/entrada?acao=ListaEmpresas (http://localhost:8080/gerenciador/entrada?acao=ListaEmpresas). No menu à direita, abriremos a opção "Mais ferramentas > Ferramentas do desenvolvedor". Agora enviaremos qualquer ação, como "editar" (que redireciona para http://localhost:8080/gerenciador/entrada?acao=MostraEmpresa&id=2 (http://localhost:8080/gerenciador/entrada?acao=MostraEmpresa&id=2)) e clicaremos na requisiçao que aparece na caixa "Name".

A aba "Header" vai nos mostrar mais informações sobre essa requisição, como os cabeçalhos e a resposta. Entre essas informações está algo parecido com "Cookie: JSESSIONID=22F22711D4837281782435BD4C891C0F" - ou seja, uma **JSESSION ID** (identificação de sessão java) a qual é atribuído um número hexadecimal que foi criado e devolvido automaticamente pelo Tomcat na resposta. Se fizermos outras requisições, esse número não irá mudar.

> Um cookie na verdade é um arquivo de texto que o servidor pode criar e associar na resposta. Nas próximas requisições, o navegador poderá enviar esse arquivo de texto automaticamente.

Como isso nos ajuda a resolver o problema de autenticação?



Quando o Tomcat recebe uma requisição "crua" (sem cookie), ele cria um cookie e um objeto chamado **HTTP session**, que são enviados na resposta e associados entre si. O número hexadecimal que compõe o cookie, e que é enviado novamente nas próximas requisições, é como uma chave de entrada que permite o acesso a esse objeto.

O melhor de tudo é que essa é uma funcionalidade que já existe no Tomcat, o que torna o nosso trabalho muito simples: só precisamos importar o Servlet `HttpSession` e usar o objeto `request` para obtermos o `HttpSession` que está sendo devolvido pelo navegador.

```
if(usuario !=null) {
    System.out.println("Usuario existe");
    HttpSession sessao = request.getSession();
```

COPIAR CÓDIGO

Se o usuário estiver usando o mesmo navegador, o objeto `HttpSession` será sempre o mesmo. Porém, se ele trocar de navegador, esse objeto será diferente.

Agora podemos usar o próprio `HttpSession` no método `setAttribute()`:

```
if(usuario !=null) {
    System.out.println("Usuario existe");
    HttpSession sessao = request.getSession();
    sessao.setAttribute("usuarioLogado", usuario);
    return "redirect:entrada?acao=ListaEmpresas";
```

COPIAR CÓDIGO

Dessa forma, enquanto o usuário estiver usando a aplicação, esse objeto `HttpSession` continuará válido na memória do servidor, guardando informações desse usuário (permissões, credenciais, produtos que ele está selecionando, etc.).

Por enquanto não precisaremos alterar mais nada no nosso código, já que nossa expressão de linguagem `${usuariologado.login}` é muito inteligente: ela primeiro procura o `usuariologado` dentro da requisição e, se não encontrar, o procura na `sessao`.

Agora reiniciaremos nosso servidor para testarmos a aplicação. Para garantirmos que não temos nenhum cookie guardado, abriremos uma nova janela anônima e então acessaremos http://localhost:8080/gerenciador/entrada?acao=ListaEmpresas (http://localhost:8080/gerenciador/entrada?acao=ListaEmpresas).

Nas "Ferramentas de desenvolvedor", poderemos perceber que o JSESSIONID agora é diferente. Se chamarmos a URL http://localhost:8080/gerenciador/entrada?acao=LoginForm (http://localhost:8080/gerenciador/entrada?acao=LoginForm), o JSESSIONID continuará o mesmo - ou seja, o Tomcat sabe que é o mesmo usuário.

Agora vamos fazer um login válido, por exemplo "nico". Dessa vez, teremos:

Usuario Logado: nico

Lista de empresas:

Alura - 25/10/2018 edita remove

Caelum - 25/10/2018 edita remove

Se chamarmos novas equisições, como "editar" ou "remove", o usuário "nico" permanecerá logado. Isso acontece porque, enquanto as requisições estiverem enviando o mesmo JSESSIONID, o Tomcat conseguirá recuperar o objeto `HttpSession`, dentro do qual temos o usuário com seus dados e credenciais.

Ou seja, se acessarmos a mesma página por outro navegador, esse usuário não estará logado.

Agora que já sabemos guardar um valor dentro do `HttpSession`, precisamos verificar em outras requisições se já existe um atributo dentro do `HttpSession`, pois, se ele existe, saberemos que o usuário fez login. Veremos isso no próximo vídeo. Até lá!