03. Formulário de login

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

Tópicos

- Projeto da aula anterior
- Preparando modelo e view
- Verificando o login
- O que mudou?
- Login funciona?
- Autenticação e autorização
- Para saber mais: O padrão JAAS
- Faça como eu fiz na aula
- O que aprendemos?

Autenticação e autorização

No nosso contexto de uma aplicação web, o que é *Autenticação* e o que é *Autorização*?

- A autenticação é sobre verificar a identidade. Isso pode acontecer através de um login, *token*, impressão digital, RG, entre várias outras formas e combinações.
- Na autorização, é verificada a permissão de acesso. No nosso contexto, queremos que apenas usuários autenticados tenham acesso, mas poderíamos criar permissões e papeis específicos que detalhem o acesso.

Para saber mais: O padrão JAAS

O Tomcat e mundo de Servlet já possuem uma forma padrão para trabalhar com login, senha, permissões e os recursos protegidos. Tudo isso pode ser configurado através do arquivo **web.xml** e uma pequena configuração no servidor Tomcat.

A ideia é que aplicação web defina que deve ter um login, quais são as permissões e os recursos (URLs) protegidos. Tudo isso fica no **web.xml**. Segue um exemplo de um **web.xml** que protege a URL `/entrada` e as páginas `.html`:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.

    <display-name>gerenciador</display-name>

    <welcome-file-list>
        <welcome-file>bem-vindo.html</welcome-file>
    </welcome-file-list>

    <login-config>
      <auth-method>BASIC</auth-method>
    </login-config>
```

```
    <security-role>
        <role-name>ADMINISTRADOR</role-name>
    </security-role>

    <security-role>
        <role-name>USUARIO</role-name>
    </security-role>

    <security-constraint>
        <web-resource-collection>
            <web-resource-name>acesso controlador</web-resource-name>
            <url-pattern>/entrada</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>ADMINISTRADOR</role-name>
        </auth-constraint>
    </security-constraint>

    <security-constraint>
        <web-resource-collection>
            <web-resource-name>acesso a paginas html</web-resource-name>
            <url-pattern>*.html</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>USUARIO</role-name>
        </auth-constraint>
    </security-constraint>
</web-app>
```

Nesse exemplo, temos duas permissões (`ROLE`s): `ADMINISTRADOR` e `USUARIO`. Para acessar `/entrada`, precisa ser `ADMINISTRADOR`, e para acessar `*.html`, basta ser `USUARIO`.

No outro lado, o Tomcat fica com a responsabilidade de carregar os usuários e as permissões. Uma forma simples de fazer isso é usar o arquivo **tomcat-users.xml** dentro do projeto `Servers`, na pasta Tomcat. Basta substituir com o conteúdo abaixo, para definir dois usuários (`admin` e `user`):

```
<tomcat-users>
  <role rolename="ADMINISTRADOR"/>
  <role rolename="USUARIO"/>
  <user username="admin" password="123" roles="ADMINISTRADOR, USUARIO"/>
  <user username="user" password="123" roles="USUARIO"/>
</tomcat-users>
```

Repare que o `admin` possui as permissões `ADMINISTRADOR` e `USUARIO`, o `user` apenas `USUARIO`.

Tudo isso foi definido dentro de um outro padrão, chamado *Java Autenthication and Authorization Service* (**JAAS - API padrão do Java para segurança**), no entanto, ele não é tão utilizado em aplicações web Java.

O que aprendemos?

Nesta aula, aprendemos:

- A representar o usuário através de uma classe `Usuario`
- A criar um formulário de login
- A criar a ação para chamar o formulário
- A criar a ação verificar o login e a senha
