03

Autorizando acesso

Transcrição

Agora ao tentarmos acessar qualquer página no projeto, seremos redirecionados para esta tela de login e para realizar qualquer operação, teremos que nos autenticar. Isso não é interessante! Nós queremos que os usuários acessem os livros e os detalhes dos mesmos, o carrinho de compras e outras páginas sem impedimentos. Vamos voltar para o código e ver como podemos fazer este comportamento funcionar.

A classe `SecurityConfiguration` herda um método chamado `configure` que em sua implementação padrão, bloqueia e redireciona todas as requisições não autenticadas. Veja o código deste método abaixo:

```
protected void configure(HttpSecurity http) throws Exception {
    logger.debug("Using default configure(HttpSecurity). If subclass

    http
        .authorizeRequests()
            .anyRequest().authenticated()
            .and()
        .formLogin().and()
        .httpBasic();
}
```

O que podemos fazer é sobrescrever este método em nossa classe de configuração e descrever os padrões de **URLs** que queremos ou não bloquear. Estes padrões são definidos através do método `antMatchers` da classe `HttpSecurity`. Veja o código abaixo que também libera o acesso a pasta `resources` para carregar os CSS e imagens :

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.authorizeRequests()
    .antMatchers("/resources/**").permitAll()
    .antMatchers("/carrinho/**").permitAll()
    .antMatchers("/pagamento/**").permitAll()
    .antMatchers("/produtos/form").hasRole("ADMIN")
    .antMatchers(HttpMethod.POST, "/produtos").hasRole("ADMIN")
    .antMatchers(HttpMethod.GET, "/produtos").hasRole("ADMIN")
    .antMatchers("/produtos/**").permitAll()
    .antMatchers("/").permitAll()
    .anyRequest().authenticated()
    .and().formLogin();
}
```

Perceba que montamos uma série de padrões de caminhos definindo onde será permitido o acesso com autenticação ou não. Estamos bloqueando o acesso a `/produtos/form` para todos que não são **ADMIN**, assim como requisições do tipo **POST** para o caminho `/produtos/form`. Estamos permitindo o acesso ao **home** da nossa aplicação através do `/` e também ao **carrinho de compras** através do `/carrinho/**`. Por último, estamos sinalizando que estas verificações devem ser feitas para todas as requisições e que as bloqueadas através do `hasRole` devem ser autenticadas.

A cada requisição, a verificação é feita e caso uma das páginas bloqueadas tenha tentativas de acesso sem autenticação, estas serão redirecionadas para o formulário de login. Experimente testar quais páginas estão ou não bloqueadas agora. Lembre-se de reiniciar o servidor.

Se verificarmos a página inicial novamente, veremos que os links para a listagem e cadastro de produtos continuam a ser exibidos apesar de sermos direcionados para o formulário de login ao clicá-los.


localhost:8080/casadocodigo/
Casa do Código
Livros e Tecnologia
Lista de Produtos
Cadastro de Produtos
Seu carrinho
Sobre nós
Home
Agile
Front End
Games
Java
Mobile
Web
Outros
Test-Driven Development:
Java SE 8 Programmer I: O
Java 8 Prático: Lambdas,
PL/SQL
PL/SQL
PL/SQL