CETEJ35 - Java Web - JAVA_XXX (2024_01)

Listeners & Cookies

✓ **Feito:** Ver | **A fazer:** Gastar pelo menos 20 minutos na atividade | **A fazer:** Passar pela atividade até o fim

Fecha: segunda-feira, 2 dez. 2024, 00:00

Nesta aula vamos falar sobre dois tópicos complementares no contexto do aplicativo que estamos usando como exemplo, mas importantes no desenvolvimento de aplicações Web.

LISTENERS

Listeners são parte do gerenciamento de eventos do Spring. Um evento é algo que acontece, como o login na aplicação, o acesso ao banco de dados, ou a inicialização da aplicação. Esses eventos emitem notificações que são captadas por *listeners*.

Assim como quase tudo no Spring, eventos e *listeners* podem ser estendidos para atender necessidades específicas. Nesta seção, vamos ver como usar um evento e um *listener* para identificar o usuário atualmente logado na aplicação.

Um *listener* atua sob um método que você definiu. Por isso, vamos começar criando um método. Nosso método será criado na classe `crudcidades.SecurityConfig`, porque esse método está relacionado com a autenticação.

> Observe que o método poderia ser criado em qualquer classe gerenciada pelo Spring Boot, como a `cidade.CidadeController`, por exemplo.

Crie o método público `printUsuarioAtual()` na classe `crudcidades.SecurityConfig`. O método deve receber um parâmetro do tipo `org.springframework.security.authentication.event.InteractiveAuthenticationSuccessEvent`, que vamos mapear com a variável `event`. Esse objeto carrega valores e métodos que são entregues pelo *evento* que foi disparado ao *listener*. Podemos usar esse objeto para extrair o nome do usuário atualmente logado. Como de costume, o Spring Boot se encarrega de instanciar o objeto, por meio da *injeção de dependências*.

```
43      public void printUsuarioAtual(InteractiveAuthenticationSuccessEvent event) {
44
45      }
```

No corpo do método, vamos usar a variável `event` para ter acesso ao método `getAuthentication()`, do objeto `org.springframework.security.authentication.event.InteractiveAuthenticationSuccessEvent`. Esse método retorna o objeto que representa a autenticação. Por meio desse objeto, podemos ter acesso ao método `getName()`, que retorna o nome do usuário atual. Após ter identificado o usuário, vamos imprimir o nome do usuário na console do sistema.

```
43      public void printUsuarioAtual(InteractiveAuthenticationSuccessEvent event) {
44
45          var usuario = event.getAuthentication().getName();
46
47          System.out.println(usuario);
48      }
```

Como o esse método sabe o nome do usuário atual? Você já informou pra ele! Quando? Quando implementou a interface `UserDetails`. Esse é um dos grandes benefícios do uso de interfaces.

O último passo é identificar que esse método deve ser acionado por um *listener*. Para isso, vamos adicionar a anotação `org.springframework.context.event.EventListener` imediatamente antes do método. Adicionalmente, vamos usar a classe `org.springframework.security.authentication.event.InteractiveAuthenticationSuccessEvent.class` como argumento da anotação.

```
42      @EventListener(InteractiveAuthenticationSuccessEvent.class)
43      public void printUsuarioAtual(InteractiveAuthenticationSuccessEvent event) {
44
45          var usuario = event.getAuthentication().getName();
46
47          System.out.println(usuario);
48      }
```

A classe `org.springframework.security.authentication.event.InteractiveAuthenticationSuccessEvent.class` representa um evento que é disparado pelo Spring Boot quando o usuário loga na aplicação (por isso *AuthenticationSucess* no nome da classe). A anotação serve como um *inscrito*, que *ouve* esse tipo de evento. O próp framework cuida de tudo e, quando o evento é disparado, o método que definimos entra em ação, imprimindo o nome do usuário na console do sistema.

Contudo, imprimir algo na console do sistema costuma ser pouco útil para uma aplicação Web. Por isso, na próxima seção vamos salvar o nome do usuário na *sessão da aplicação* para podermos apresentar na tela principal.

O código desenvolvido nesta Seção está disponível no **Github**, na branch `semana08-10-misc-listener`.

Retroceder Avançar

◄ Verificação de aprendizado - Segurança

Seguir para...

Avaliação Oficial ►

Campo Mourão
Cornélio Procópio
Curitiba
Dois Vizinhos
Francisco Beltrão
Guarapuava
Londrina
Medianeira
Pato Branco
Ponta Grossa
Reitoria
Santa Helena
Toledo
UTFPR
Ajuda
Chat UTFPR
Calendário Acadêmico
Biblioteca
e-Mail
Nuvem (OwnCloud)
Produção Acadêmica
Secretaria Acadêmica
Sistemas Corporativos
Sistema Eletrônico de Informação - SEI
Suporte ao usuário
Criação de curso
Comunidade
Português - Brasil (pt_br)
Deutsch (de)
English (en)
Português - Brasil (pt_br)

Resumo de retenção de dados

Baixar o aplicativo móvel.

Dê um feedback sobre este software

Universidade Tecnológica Federal do Paraná - UTFPR

Suporte ao usuário