03

Configurando OpenEntityManagerInView com Spring

Transcrição

Mas, o próprio SpringMVC possui um filtro especial que podemos utilizar para resolver esse problema: os interceptors. Eles permitem executar um código antes (e depois) de chamarem o método Controller. Para registrar o `OpenEntityManagerInViewInterceptor` na aplicação para sobrescrever o método `addInterceptors` que ganhamos por herança de `WebMvcConfigurerAdapter` na classe `Configurador`:

```
@Override
public void addInterceptors(InterceptorRegistry registry) {

}
```

Esse método permite adicionar interceptadores ao contexto do SpringMVC. Para tanto, vamos usar o método `addInterceptors` do `InterceptorRegistry`:

```
@Override
public void addInterceptors(InterceptorRegistry registry) {
    registry.addWebRequestInterceptor(???);
}
```

Mas, o que passar como argumento? Precisamos passar uma instância do interceptor que será registrada. Porém, o container de IoC do Spring precisa conhecer essa instância, sendo assim, é necessário criar um método que retorne uma instância do *interceptor* e anotá-lo com `@Bean`:

```
@Bean
public OpenEntityManagerInViewInterceptor getOpenEntityManagerInViewInterceptor() {
    return new OpenEntityManagerInViewInterceptor();
}
```

E, agora, podemos passar como argumento uma instância do Interceptor, a partir desse método:

```
@Override
public void addInterceptors(InterceptorRegistry registry) {
    registry.addWebRequestInterceptor(getOpenEntityManagerInViewInterceptor());
}
```

Após isso, retornamos na página de edição e verificamos que tudo continuará funcionando!