10

Tratamento de exceções

Transcrição

Um comportamento que fizemos anteriormente foi permitir que a url da página de detalhes ficasse mais amigável. Isso foi possível usando um padrão diferente no endereço dos produtos que deixaram de exibir o identificador do produto de forma parametrizada (`produto?id=1`) para um novo formato (`produto/detalhe/1`).

Mas o que acontece se o usuário editar manualmente o endereço da página e por um identificador que não existe? Algo como 150 ou 300. Ao acessar `produto/detalhe/150` por exemplo, teremos um erro.

O que acontece é que ao buscar o produto no banco de dados, o método `getSingleResult` ao perceber que o resultado não existe, lança uma exceção do tipo `NoResultException` .

Para a solução deste tipo de problema, podemos criar um novo método na classe `ProdutosController` para que ao lançamento deste tipo de erro, este seja executado, fazendo com que ao invés de apresentar este erro para o usuário, simplesmente redirecione-o para uma página de erro mais amigável.

O método se chamará `trataDetalheNaoEncontrado` e retornará apenas uma *String* informando qual página deverá ser mostrada ao usuário. Este método precisa ser anotado com `@ExceptionHandler` para que ao lançamento da excessão, o *Spring* repasse a mesma para este método. Esta anotação precisa do identificador da exceção a ser capturada, sendo este o nome da classe da qual é a excessão, neste caso `NoResultException.class` . Na classe `ProdutosController` então teremos:

```
@ExceptionHandler(NoResultException.class)
public String trataDetalheNaoEcontrado(){
    return "error";
}
```

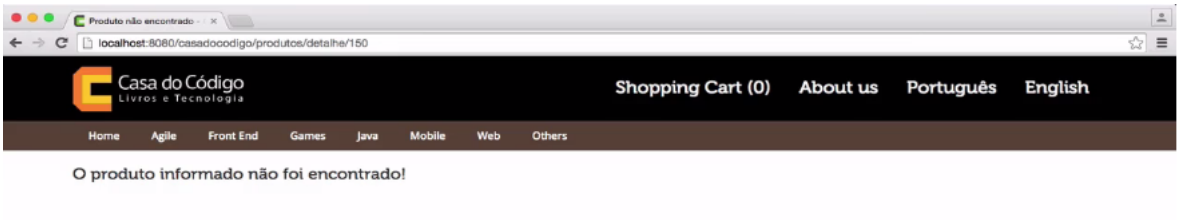
A página `error.jsp` não existe ainda, mas criaremos a mesma a partir da página `home.jsp` . Copiaremos todo o código da página `home.jsp` , criaremos o arquivo `error.jsp` no mesmo diretório e colaremos todo o código copiado.

Na página `error.jsp` removeremos todo o código que exibe produtos e escreveremos apenas uma mensagem envolvida em uma tag de título `h2` com o texto: `O Produto Informado não foi encontrado` . Ao final, o arquivo `error.jsp` terá o seguinte código:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib tagdir="/WEB-INF/tags" prefix="tags" %>

<tags:pageTemplate titulo="Produto não encontrado">

    <section id="index-section" class="container middle">
```

```
        <h2>O produto informado não foi encontrado</h2>
    </section>

</tags:pageTemplate>
```

Note que o título da página também foi alterado para `Produto não encontrado`. Agora ao tentarmos acessar a página de detalhes de um produto que não existe, teremos nossa página de erro exibida em vez da mensagem de erro do próprio *Java*.



O erro do tipo `NoResultFound` foi resolvido. Mas e se acontecer algum outro erro aleatório? O que acontece? Por questões de teste, no método `detalhe` da classe `ProdutosController` lançaremos um erro proposital para verificar o comportamento da aplicação nestes casos. O erro será do tipo `RuntimeException`.

```
@RequestMapping("/detalhe/{id}")
public ModelAndView detalhe(@PathVariable("id") Integer id){
    ModelAndView modelAndView = new ModelAndView("produtos/detalhe");
    Produto produto = produtoDao.find(id);

    if(true) throw new RuntimeException("Excessão Genérica Acontecendo!!!!");

    modelAndView.addObject("produto", produto);

    return modelAndView;

}
```

A página de erro do *Java* volta a ser exibida:

Podemos resolver este problema simplesmente mudando o tipo de excessão a ser tratada pelo método `trataDetalheNaoEncontrado`. O tipo mais genérico das exceções é a classe `Exception`. A anotação `@ExceeptionHandler` passa agora a tratar qualquer tipo de exceção lançada.

```
@ExceptionHandler(Exception.class)
public String trataDetalheNaoEcontrado(){
    return "error";
}
```

Note que agora, existindo ou não o livro, teremos o erro sendo lançado e que qualquer tipo de exceção será tratada pelo método `trataDetalheNaoEcontrado`. Mas será só isso mesmo? Resolvemos todos os problemas referente a exibição dos

erros ao usuário? Vamos adicionar o código que lança o erro para a classe `CarrinhoComprasController`, no inicío do método `itens`.

```
@RequestMapping(method=RequestMethod.GET)
public ModelAndView itens(){

        if(true) throw new RuntimeException("Excessão Genérica Acontecendo!!!!");

        return new ModelAndView("/carrinho/itens");
}
```

Com esta adição, simplesmente mudamos o problema de lugar. No caso da página de detalhe dos produtos, estamos tratando a ocorrência de qualquer tipo de exceção, mas na página de itens do carrinho não.

Acontece que o tratamento que estamos fazendo está isolado em um único *controller*. A solução ideal é que este fosse feito em todos os *controllers* da aplicação.

Como solução para o problema, criaremos um novo *controller* chamado `ExceptionHandlerController` e colocaremos neste o método que trata as *exceptions* que está no `ProdutosController`

```
@Controller
public class ExceptionHandlerController {
    @ExceptionHandler(Exception.class)
    public String trataDetalheNaoEcontrado(){
        return "error";
    }
}
```

Este passo ainda não resolve o problema por que este é apenas um *controller* comum. O que precisamos é este monitore todos os outros *controllers* da aplicação. Neste caso, a anotação muda de `@Controller` para `@ControllerAdvice`.

Agora, ao invés de simplesmente redirecionar o usuário para uma página de erro, vamos também capturar o objeto gerado com a mensagem de erro. Imprimir sua *Stack Trace* (Pilha de erros) no console e ainda enviar este objeto para a página, onde nos comentários do *HTML* imprimiremos toda a mensagem de erro para que possamos ver no *HTML* o que está acontecendo. Também renomearemos o método `trataDetalheNaoEcontrado` para `trataExceptionGenerica`. Assim teremos na classe `ExceptionHandlerController`:

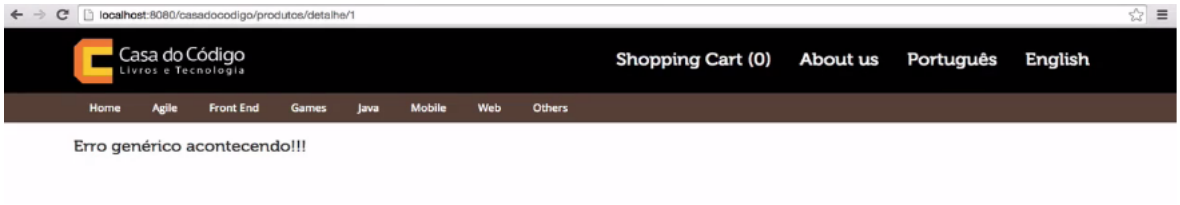
```
@ControllerAdvice
public class ExceptionHandlerController {
    @ExceptionHandler(Exception.class)
    public ModelAndView trataExceptionGenerica(Exception exception){
        System.out.println("Erro genérico acontecendo");
        exception.printStackTrace();

        ModelAndView modelAndView = new ModelAndView("error");
        modelAndView.addObject("exception", exception);

        return modelAndView;
    }
}
```

E na pagina de erro (`error.jps`):

```
<tags:pageTemplate titulo="Produto não encontrado">

    <section id="index-section" class="container middle">
        <h2>Erro genérico acontecedo!!!</h2>
    </section>

    <!--
        Mensagem: ${exception.message}
        <c:forEach items="${exception.stackTrace}" var="stk">
            ${stk}
        </c:forEach>
    -->

</tags:pageTemplate>
```

Observe que a mensagem envolvida na tag `h2` do *HTML* mudou! E agora como resultado teremos:

A página de erro HTML sendo exibida normalmente:



No código fonte da página *HTML* as mensagens de erro sendo impressas:

```
<section id="index-section" class="container middle">
    <h2>Erro genérico acontecendo!!!</h2>
    <!--
        Mensagem: Exceção Genérica acontecendo!!!!
            br.com.casadocodigo.loja.controllers.CarrinhoComprasController.itens(CarrinhoComprasController.java:40)
            sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
            sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
            sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
            java.lang.reflect.Method.invoke(Method.java:497)
```

E esta mesma mensagem impressa no código *HTML* também aparecendo no console do *Eclipse*:

```
java.lang.RuntimeException: Exceção Genérica acontecendo!!!!
Erro genérico acontecendo
        at br.com.casadocodigo.loja.controllers.CarrinhoComprasController.itens(CarrinhoComprasControll
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
```

Agora, com todos os cenários cobertos e resolvidos, podemos remover a linha de código que usamos para gerar o erro proposital da classe `ProdutosController` e também da classe `CarrinhoComprasController` .

```
// linha a ser removida
if(true) throw new RuntimeException("Excessão Genérica Acontecendo!!!!");
```

Note que estamos capturando qualquer tipo de exceção que aconteça em qualquer *controller* e por mais que resolva todos os problemas, as vezes é necessário tratar tipos específicos de exceção. Nestes casos ainda pode-se se utilizar da primeira forma que vimos nesta aula, onde definimos o método que trata a exceção no próprio *controller* ou criar diversos métodos na classe `ExceptionHandlerController` para tratar as especificidades das exceções.

Algo comum de se fazer também nos casos de exceções é armazená-las todas em um **Log**, com o objetivo de que possamos consultá-lo depois e obtermos mais informações. Imprimimos os erros no código *HTML* por questões didáticas, mas não recomendamos que você faça isso em projetos reais em produção. Utilize *logs* para isso.