12

Mãos na massa: Ajustes finais

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1) Adicione o XML do **Java Mail** dentro do seu arquivo **pom.xml**, dentro da *tag* `<dependencies>` :

```
<dependency>
    <groupId>javax.mail</groupId>
    <artifactId>mail</artifactId>
    <version>1.4.7</version>
</dependency>
```

2) Na classe `PagamentoController` , crie o método `enviaEmailCompraProduto` e injete um `MailSender` :

```
@RequestMapping("/pagamento")
@Controller
public class PagamentoController {

    @Autowired
    private MailSender sender;

    private void enviaEmailCompraProduto(Usuario usuario) {
        SimpleMailMessage email = new SimpleMailMessage();
        email.setSubject("Compra finalizada com sucesso");
        email.setTo(usuario.getEmail());
        email.setText("Compra aprovada com sucesso no valor de "
             + carrinho.getTotal());
        email.setFrom("compras@casadocodigo.com.br");

        sender.send(email);
    }

    // restante do código da classe omitido

}
```

3) Ainda na classe `PagamentoController` , no método `finalizar` , receba um `Usuario` por parâmetro e anote-o com `@AuthenticationPrincipal` . Além disso, caso o pagamento seja feito com sucesso, envie um e-mail para o usuário:

```
@RequestMapping(value="/finalizar", method=RequestMethod.POST)
public Callable<ModelAndView> finalizar(@AuthenticationPrincipal Usuario usuario,
        RedirectAttributes model)

    return () -> {
        String uri = "http://book-payment.herokuapp.com/payment";
        try {
            String response = restTemplate
                .postForObject(uri,
                    new DadosPagamento(carrinho.getTotal()),
                        String.class);
```

```
            System.out.println(response);

            // envia e-mail para o usuário
                  enviaEmailCompraProduto(usuario);

            model.addFlashAttribute("sucesso", response);

            return new ModelAndView("redirect:/produtos");

        } catch (HttpClientErrorException e) {
            e.printStackTrace();
            model.addFlashAttribute("falha",
                "Valor maior que o permitido");
            return new ModelAndView("redirect:/produtos");
        }
    };
}
```

4) Na classe `AppWebConfiguration`, crie o método **`mailSender`** com o seguinte conteúdo de exemplo:

```
@Bean
public MailSender mailSender() {

    JavaMailSenderImpl mailSender = new JavaMailSenderImpl();

    mailSender.setHost("smtp.gmail.com");
    mailSender.setUsername("alura.springmvc@gmail.com");
    mailSender.setPassword("alura2015");
    mailSender.setPort(587);

    Properties mailProperties = new Properties();
    mailProperties.put("mail.smtp.auth", true);
    mailProperties.put("mail.smtp.starttls.enable", true);

    mailSender.setJavaMailProperties(mailProperties);

    return mailSender;
}
```

5) Para a exibição dos preços dos produtos, acesse o atributo `precos` do objeto `produto` na **lista.jsp**, adicionando mais uma coluna:

```
<table class="table table-bordered table-striped table-hover">
    <tr>
        <th>Título</th>
        <th>Descrição</th>
        <th>Preços</th>
        <th>Páginas</th>
    </tr>
    <c:forEach items="${produtos}" var="produto">
        <tr>
            <td>
                <a href="${s:mvcUrl('PC#detalhe').arg(0, produto.id).build()}">
                    ${produto.titulo}
```

```
                </a>
            </td>
            <td>${produto.descricao}</td>
            <td>${produto.precos}</td>
            <td>${produto.paginas}</td>
        </tr>
    </c:forEach>
</table>
```

6) Na classe `Preco`, sobrescreva o seu método `toString`:

```
public String toString() {
    return this.tipo.name() + " - " + this.valor;
}
```

7) Altere a consulta no banco de dados que carrega os produtos, para que utilize o `distinct`. Na classe `ProdutoDAO`, no método `listar`, faça:

```
public List<Produto> listar(){
    return manager.createQuery("select distinct(p) from Produto p",
        Produto.class).getResultList();
}
```

8) Mantenha a sessão com o banco de dados aberta até que a visualização da página seja carregada. Para isso, adicione o filtro **`OpenEntityManagerInViewFilter`** na cadeia de filtros carregados no método `getServletFilters`, da classe `ServletSpringMVC`:

```
@Override
protected Filter[] getServletFilters() {
    CharacterEncodingFilter encodingFilter = new CharacterEncodingFilter();
    encodingFilter.setEncoding("UTF-8");

    return new Filter[] {encodingFilter, new OpenEntityManagerInViewFilter()};
}
```

9) Crie a classe **`ExceptionHandlerController`** no pacote `br.com.casadocodigo.loja.controllers` com o método `trataExceptionGenerica`, que redirecionará o usuário para uma página de erro, além de capturar o objeto gerado com a mensagem de erro, imprimir sua *stack trace* (pilha de erros) no console e enviar este objeto para a página, onde nos comentários do HTML você imprimirá toda a mensagem de erro para que o usuário ver o que está acontecendo:

```
@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;
    }
}
```

10) Crie a página **error.jsp** em **src/main/webapp/WEB-INF/views/** com o seguinte conteúdo:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib tagdir="/WEB-INF/tags" prefix="tags" %>

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

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

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

</tags:pageTemplate>
```