02

Complementando a aplicação com os Plugins

Trabalhando com plugins do VRaptor

Vimos que criar um interceptor como o `ControleDeTransacaoInterceptor` é uma prática muito comum quando estamos usando o JPA, por exemplo. Além disso, tivemos que criar um **produtor do CDI** (`@Produces`) para a classe `EntityManager` , para tirar proveito da injeção de dependências. Essas são apenas algumas situações que os desenvolvedores que pretendem usar JPA com VRaptor precisam se preocupar para que tudo funcione.

A fim de otimizar esse processo e aumentar sua produtividade, o VRaptor possui diversos complementos criados pela comunidade ou pela própria caelum para facilitar ainda mais seu trabalho de desenvolvimento.

Um de seus complementos, ou **plugins** como são habitualmente chamados, é o **vraptor-jpa**. Ele te ajuda com todo esse trabalho de controle de transações e injeção de classes do JPA. Vamos usá-lo em nosso projeto!

Instalação e uso do vraptor-jpa

Como estamos usando um projeto *maven*, para que ele nos forneça o `vraptor-jpa` basta adicionar essa dependência no arquivo `pom.xml` de nosso projeto. Se você preferir não usar o maven, basta baixar o jar (http://search.maven.org/remotecontent?filepath=br/com/caelum/vraptor/vraptor-jpa/4.0.1/vraptor-jpa-4.0.1.jar) e adicionar ao seu `classpath` , como qualquer outra biblioteca de seu projeto java.

Vamos abrir o arquivo `pom.xml` , que se encontra na raiz do nosso projeto. Agora só precisamos adicionar as linhas abaixo dentro da tag `dependencies` :

```
<dependencies>
    <!-- outras dependências já declaradas -->

    <dependency>
        <groupId>br.com.caelum.vraptor</groupId>
        <artifactId>vraptor-jpa</artifactId>
        <version>4.0.1</version>
    </dependency>

</dependencies>
```

Para o maven baixar essa dependência, basta clicar com o botão direito do mouse no projeto `vraptor-produtos` , selecionar a opção `Maven` e depois `Update Project` . Ou você pode utilizar o atalho `ALT + F5` .

Como o plugin já faz todo o controle de transações e de produção de objetos, vamos apagar o nosso `ControleDeTransacaoInterceptor` e também o produtor de `EntityManager` , o `EntityManagerProducer` .

Pronto! Não precisamos de nenhuma configuração, basta restartar o servidor e testar para garantir que tudo está funcionando! Adicionar complementos ao VRaptor é um processo bem simples, não acha?

Enviando e-mail com VRaptor-simplemail

Para deixar nossa app de controle de estoque de produtos ainda mais completa, vamos criar uma nova funcionalidade que envolva o envio de e-mails notificando que precisamos de um estoque maior pra determinado produto. Mas como enviar um e-mail em uma aplicação Java Web? Podemos utilizar outro complemento do VRaptor, o plugin **vraptor-simplemail**!

Esse é outro complemento muito utilizado no dia a dia do desenvolvedor web com VRaptor, pois ele te provê uma forma muito simples e prática de configurar e fazer envio de e-mails.

Para adicionar esse plugin em nosso projeto, vamos abrir novamente o arquivo `pom.xml` e adicionar sua dependência:

```
<dependencies>
    <!-- outras dependências já declaradas -->

    <dependency>
        <groupId>br.com.caelum.vraptor</groupId>
        <artifactId>vraptor-simplemail</artifactId>
        <version>4.0.0</version>
    </dependency>

</dependencies>
```

Vamos pedir para o maven baixar essa nova dependência, assim como fizemos com o outro plugin: um clique com o botão direito do mouse no projeto `vraptor-produtos`, selecionar a opção `Maven` e depois `Update Project`.

Pronto! Já podemos começar a utilizar o **vraptor-simplemail**!

Vamos começar adicionando um link na `lista.jsp`, que o usuário usará para pedir mais itens de um determinado produto em estoque:

```
<c:forEach itens="${produtoList}" var="produto">
<tr>
    <td>${produto.nome}</td>
    <td>${produto.valor}</td>
    <td>${produto.quantidade}</td>
    <td>
        <a href="<c:url value='/produto/enviaPedidoDeNovosItens?
            produto.nome=${produto.nome}'/>">Pedir mais itens!</a>
    </td>
</tr>
</c:forEach>
```

Agora, em nosso `Controller` precisamos criar um método que vai receber essa requisição. Vamos chamá-lo de `enviaPedidoDeNovosItens`:

```
@Get
public void enviaPedidoDeNovosItens(Produto produto) {
    // agora só precisamos enviar um e-mail notificando!
}
```

O vraptor-simplemail trabalha internamente com a biblioteca Commons Email, da Apache (http://commons.apache.org/email/). Para criar um e-mail, precimos instanciar a classe `SimpleEmail` dessa biblioteca, e

adicionar suas informações, algo como:

```
@Get
public void enviaPedidoDeNovosItens(Produto produto) throws EmailException {
    Email email = new SimpleEmail();
    email.setSubject("[vraptor-produtos] Precisamos de mais estoque");
    email.addTo("rodrigo.turini@caelum.com.br");
    email.setMsg("Precisamos de mais itens do produto" + produto.getNome());
}
```

Excelente, agora precisamos efetuar o envio desse e-mail que criamos. Para isso precisamos pedir injetada a classe `Mailer` em nosso `ProdutoController`, pois essa é a classe do `vraptor-simplemail` responsável pelo envio de e-mails.

```
@Controller
public class ProdutoController {

    // outros atributos da classe
    private final Mailer mailer;

    @Inject
    public ProdutoController(Result result, ProdutoDao dao,
            Validator validator, Mailer mailer) {
        this.result = result;
        this.dao = dao;
        this.validator = validator;
        this.mailer = mailer;
    }
```

E em nosso método `enviaPedidoDeNovosItens` só precisamos invocar seu método `send` passando esse nosso e-mail como argumento, e em seguida podemos redirecionar o usuário para a listagem de produtos:

```
@Get
public void enviaPedidoDeNovosItens(Produto produto) throws EmailException {
    Email email = new SimpleEmail();
    email.setSubject("[vraptor-produtos] Precisamos de mais estoque");
    email.addTo("rodrigo.turini@caelum.com.br");
    email.setMsg("Precisamos de mais itens do produto" + produto.getNome());
    mailer.send(email);
    result.redirectTo(this).lista();
}
```

Mas se tentarmos enviar um e-mail agora, qual será a conta de e-mail utilizado para o envio? De qual servidor? Com qual senha? Ainda precisamos informar essas configurações! Só que note que queremos enviar e-mails reais apenas em produção, nunca em testes ou desenvolvimento. Ou talvez até podemos precisar enviar em testes ou desenvolvimento, só que nunca na mesma conta de e-mail que usaremos em produção. Para facilitar esse trabalho, o `vraptor-simplemail` é integrado com a função de **Environment** (diferentes ambientes) do VRaptor 4.

Em nosso `web.xml` podemos definir qual ambiente estamos utilizando:

```
<context-param>
    <param-name>br.com.caelum.vraptor.environment</param-name>
```

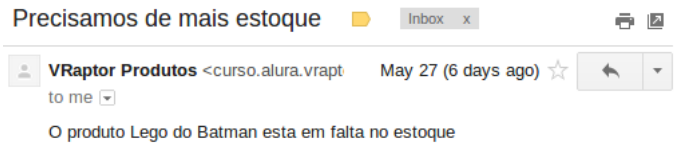
```
    <param-value>production</param-value>
</context-param>
```

E de acordo com essa informação, ele vai procurar o arquivo chamado: **nomeDoAmbiente.properties**. Neste caso, ele vai procurar o arquivo **production.properties**. Vamos adicionar nesse arquivo as configurações básicas para um envio de e-mail:

```
vraptor.simplemail.main.server=smtp.gmail.com
vraptor.simplemail.main.port=587
vraptor.simplemail.main.tls=true
vraptor.simplemail.main.from=curso-alura-vraptor4@gmail.com
vraptor.simplemail.main.from.name=VRaptor Produtos
vraptor.simplemail.main.username=curso-alura-vraptor4@gmail.com
vraptor.simplemail.main.password=xxxxxx
```

Pronto, agora basta testar! Vamos abrir a listagem de produtos e clicar em *Pedir mais itens!* em algum dos campos.

Veja que rapidamente recebemos o e-mail com as informações!



Existem diversos outros plugins para tornar seu desenvolvimento com VRaptor ainda mais produtivo e sua aplicação ainda mais completa! Você pode conhecer alguns outros em nossa documentação. (link)