06

Cuidando dos detalhes

Transcrição

Aprendemos a criar interdependência de projetos, e entendemos que o Eclipse possui recursos para torná-la explícita. Assim, as mudanças que realizamos em um projeto serão refletidas em outro automaticamente. Porém, essa automação possui uma desvantagem: as classes de teste podem ser acessadas via `ContatoServlet.java`.

Vamos tentar acessar uma classe de teste no Eclipse (`ProdutoTest()`)?

Para o Eclipse, o acesso pode ser realizado sem problemas, pois todo conteúdo presente no *classpath* também está no projeto. Além disso, ao tentarmos executar o Jetty, não teremos qualquer questão, e o servidor estará em produção. Contudo, ao acessarmos a nossa página (`localhost:8080/contato`), teremos a seguinte mensagem de erro:

```
HTTP ERROR 500

Problem accessing/contato.Reason:

    Server Error

Caused by:

java.lang.NoClassDefFoundError: br/com/alura/maven/ProdutoTest
```

A classe `ProdutoTest` não foi encontrada, afinal as classes de teste não são enviadas para outros projetos, pois não são por padrão "empacotadas" — ocorre um acesso indevido pelo Eclipse. A compilação no programa será bem sucedida ao acessarmos uma classe de teste, mas em tempo de execução, teremos problemas. Essa é a desvantagem ao referenciarmos projetos.

Uma maneira de tentar contornar esse problema seria remover o projeto `produtos`. Selecionaremos o projeto na área "Project Explorer" e pressionaremos a tecla "Delete". Na caixa de diálogo que surgirá é importante que **não** marquemos a opção de exclusão de conteúdo, pois iremos deletar o projeto **sem excluir seu conteúdo**.

No arquivo `pom.xml` acionaremos o botão "Enter", apenas para que o projeto seja reconfigurado no Eclipse. Dessa forma, não encontraremos mais erros, e em `Libraries` veremos que o arquivo `.jar` do projeto `produtos` foi acessado.

O Eclipse possui mecanismos inteligentes para lidar com interdependências: se o projeto de que dependemos já está no Eclipse, é criado um link para ele, caso contrário é criado um link para o arquivo `.jar` do projeto. Se este arquivo `.jar` não estivesse em nosso repositório local, será que o programa não iria funcionar? Vamos testar acessando o repositório na linha de comando:

```
cd
cd .m2/
cd repository/
cd br/com/alura/maven/produtos/
ls
```

E então removeremos o arquivo `produtos` do repositório local:

```
cd ..
ls
rm -fr produtos/
```

Voltaremos ao arquivo `pom.xml`, e acionaremos a tecla "Enter". Imediatamente o Eclipse enviará uma mensagem de erro: `Missing artifact br.com.alura.maven:produtos:jar:1.0-SNAPSHOT`. Faremos a instalação do projeto no repositório local via linha de comando:

```
cd
cd Documents/
cd guilherme/
cd workspace/
cd produtos/
mvn install
```

Assim feito, no arquivo `pom.xml` pressionaremos a tecla "Enter", como habitual. Dessa vez, contudo, foi instalado apenas o `produtos.jar`, e deixamos de ter acesso às classes de teste. Sem o projeto, nos protegemos de erros e acessos indevidos, mas muitas vezes isto não ocorre, pois trabalhamos com o projeto no dia a dia. Sendo assim, precisamos tomar cuidado com os acessos indesejados ao longo do trabalho.