04

Gerando relatórios

Transcrição

Executamos o `mvn clean` em nosso terminal, o que faz com que todo o código de saída seja limpo, e fiquemos somente com o de fonte. Isso faz com que o diretório `target` seja removido, afinal, tudo foi gerado em seu interior. Assim, nosso projeto está organizado, e tudo o que precisamos está em `src` e no arquivo `pom.xml`.

Conheceremos mais algumas facilidades que o Maven nos fornece em tarefas rotineiras de quem está trabalhando em um projeto que fornece um arquivo `.jar`. Por exemplo, queremos gerar um relatório dos testes realizados. Por padrão, quando executamos `mvn test`, é gerado um arquivo TXT e uma versão XML, e ambos os formatos são desagradáveis para a leitura. Seria muito mais interessante que o relatório fosse exibido no navegador, e para isso utilizamos `mvn report`.

Contudo, de qual plugin seria esse relatório?

Alguns comandos, como `clean`, `test` ou `compile` são de plugins padrão do Maven, e a instalação é realizada automaticamente. Alguns dos objetivos que queremos alcançar, como o `report`, não fazem parte dos plugins padrão, portanto o Maven não encontrará esse recurso de forma automática, sendo necessário declarar qual estamos utilizando.

Primeiro declararemos **o nome do plugin**, em seguida o **nome do objetivo** no caso, `surefire-report:` e `report`.

```
mvn surefire-report:report
```

Será realizado o download do que for necessário, e o relatório será gerado. Entraremos no diretório "produtos > target > site > surefire-report.html", e desse modo o teste será exibido no navegador.

> Para descobrirmos o nome de todos esses plugins, digitaremos no buscador do Google algo como "maven plugins", e várias informações serão disponibilizadas. Caso seja necessário realizar uma busca mais direcionada, como o plugin de geração de relatório, utilize "maven plugins test report". Encontraremos instruções na documentação do Maven, sobre como utilizar o plugin mais adequado de acordo com a necessidade.

Usando a documentação e os guias de referência, conseguimos o nome do plugin de uma tarefa específica a ser executada, e ainda entender como ela pode ser executada. Obviamente muitas dúvidas vão surgindo ao longo do processo, para isso podemos visitar fóruns como o GUJ (http://www.guj.com.br), que possui muitos tópicos acerca do universo da programação, que podem nos ajudar a solucionar problemas do dia dia.

Aprendemos como gerar um relatório dos testes usando um plugin fora do corpo padrão do Maven, agora vamos seguir para o próximo caso. Queremos transformar todo o conteúdo da nossa aplicação em um arquivo `.jar`. No arquivo `pom.xml`, vimos que o `<packaging>` é do tipo JAR.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>br.com.alura.maven</groupId>
```

```
    <artifactId>produtos</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>produtos</name>
    <url>http://www.alura.com.br</url>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>jnuit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    <dependencies>
</project>
```

Para que o Maven "empacote" nosso programa, pediremos que o objetivo `package` seja executado.

```
mvn package
```

Serão baixados os conteúdos necessários, como de praxe. Em seguida, receberemos o endereço de onde nosso programa foi empacotado:

```
[INFO] Building jar: /Users/alura/Documents/guilerme/workspace/produtos/target/produtos-1.0-SNAI
```

A versão `SNAPSHOT` vem do arquivo `pom.xml`. À medida em que mudamos as versões, será gerado um pacote com arquivos novos. Acessaremos o diretório `target` e veremos o arquivo `produtos` a partir dos seguintes comandos:

```
cd target
pwd
ls
```

Para executarmos `produtos-1.0-SNAPSHOT.JAR`, escreveremos `java -cp produtos-1.0-SNAPSHOT.jar br.com.alura.maven.App`, isto é, incluiremos o arquivo em um *classpath* `-cp`, e executaremos a classe `br.com.alura.maven.App`.

Assim feito, teremos:

```
Hello World!
```

Todas as vezes em que usarmos `mvn package`, será executado o teste, e ao final teremos o `.jar` gerado. Dessa forma aprendemos mais dois objetivos:

- gerar relatórios HTML;
- gerar pacotes `.jar`.

Usar os *goals* do Maven é muito simples, o desafio é apenas entender como eles interagem entre si e quais são as fases do *build*, mas nós entenderemos esse processo ao longo do curso.

Veremos a seguir como podemos utilizar nossos conhecimentos no Eclipse.