01

Repositório remoto e local

Transcrição

Nosso projeto está no Eclipse, sendo executado perfeitamente. Sabemos que quando executamos um comando novo, o Maven averigua se existe algum conteúdo que precisa ser baixado para que ele seja executado com sucesso.

Temos uma questão: todas as vezes em que executamos um comando, o Maven verifica a atualização de alguns plugins. Quando não especificamos a versão, é verificada a disponibilidade de uma nova versão e, caso haja, o download será realizado. Com isso, no entanto, corremos o risco desse processo ocorrer todas as vezes em que inciarmos uma execução, o que torna nosso trabalho mais lento.

Sendo assim, podemos solicitar que o Maven **execute uma ação de maneira offline** por meio do comando `-o` no terminal:

```
mvn -o test
```

Contudo, se o Maven necessitar de algum arquivo novo para executar a ação solicitada, ele não poderá realizar o download de dependências, uma vez que estamos trabalhando no modo offline. Por isso, é importante que tenhamos **certeza** de que temos todos os elementos necessários antes de operarmos nesse modo.

Quando dizemos que o Maven busca na internet as dependências necessárias, estamos nos referindo ao repositório central da ferramenta. Considerando a imensidão da internet, uma busca completa seria praticamente impossível.

Atualmente, o repositório central do Maven está hospedado em `repo.maven.apache.org/maven2/`. Vamos acessá-lo e clicar em "br/ > com/ > caelum/ > vraptor/", e encontraremos diferentes versões de *VRaptor*. Clicaremos sobre a versão `4.2.0-RC3/`, e encontraremos códigos fonte em formato `.jar`, Javadocs que podem ser descompactados, e assim por diante.

> No repositório central armazenamos as bibliotecas a serem compartilhadas com o mundo do Maven.

Não é comum acessarmos o repositório manualmente, para isso utilizamos o *Maven Repository*, como aprendemos nas últimas aulas. O repositório central é o que chamamos de **repositório remoto**, e lá coletamos os elementos necessários para a execução de tarefas em um projeto.

Imaginemos a criação de um projeto chamado `blog`. Nós já baixamos todos esses plugins do Maven, portanto ele não irá realizar o download novamente, e criará o projeto pulando essa etapa.

```
mvn archetype:generate -DartifactId=blog -DgroupId=br.com.alura.maven -Dinteractive
Mode=false -DarchetypeArtifactId=maven-archetype-quickstart
```

Se compilarmos esse projeto, ele não irá baixar as ferramentas de compilação, afinal elas já foram adquiridas no momento em que estávamos criando o projeto `produtos`. Quando formos executar testes, também não será necessário fazer o download da biblioteca JUnit. **Isso quer dizer que, além do repositório remoto, temos um diretório compartilhado entre todos os nossos projetos, isto é, um repositório local preenchido gradativamente.**

Vamos descobrir em que local o Maven está guardando os arquivos. Primeiramente, observaremos os diretórios do usuário por meio do comando `ls`, após `cd`. No entanto, não localizamos o diretório, pois, na verdade, o repositório local é **invisível por padrão**, `.m2`, que possui um diretório chamado `repository`.

Então, dentro do diretório do usuário (`/Users/alura`), acessaremos `.m2` e em `repository`:

```
pwd
cd .m2/
cd repository/
pwd
```

Dentro dela, temos vários projetos que já baixamos, dos quais se encontra o `xstream`, que veremos a partir dos seguintes comandos:

```
cd com
cd thoughtworks/
cd xstream/
ls
```

Teremos, então, o projeto `xstream`, a versão `1.4.8`, com `xstream-1.4.8.jar` e uma confirmação `.sha1` indicando que o `xstream` baixado é exatamente o `.jar` necessário. Além disso, temos o `.pom`, que poderemos analisar por meio de `cat xstream-1.4.8.pom`.

Nos depararemos com várias configurações do *build* do `xstream` e suas dependências, necessárias para o Maven baixar. A versão `1.4.8`, que estamos utilizando no momento, cria `.m2/repository/com/thoughtworks/xstream/xstream/1.4.8`, baixa todos os `.jar` do `xstream`, do repositório remoto para o local. O mesmo ocorre com o `junit`, de que foram baixadas quatro versões: `3.8.1`, `3.8.2`, `4.11` e `4.12`.

Isso porque outros plugins do Maven podem ter utilizado versões distintas do JUnit. Não precisamos ficar baixando-as todas as vezes em que tivermos projetos novos, pois um repositório local é utilizado em todos os projetos da máquina, o *M2 Repository*.

Vimos como o Maven funciona como repositório remoto, possível de ser configurado no `.pom`. Caso queiramos outro, basta usar as tags `<repositories>` para criá-las e usá-las em outras máquinas, por exemplo. Neste caso estamos usando apenas o repositório remoto padrão, que é suficiente para o que faremos.

O `.pom` pode ter uma quantidade enorme de configurações, e aprenderemos diversas delas ao longo do curso, lembrando que sua documentação está disponível no site Maven — POM Reference (https://maven.apache.org/pom.html).