02

Produção x Desenvolvimento

Transcrição

Legal, já estou sabendo o poder do Vagrant com o Puppet. Consigo dar um `vagrant up` e dar um `sudo puppet apply web.pp` e eu tenho uma máquina igual toda vez que eu executo esse comando. Toda vez ela nasce e é criada igualzinho. Os softwares são provisionados como eu imaginava. O MySQL, inclusive.

Só que a minha aplicação, isto é, o meu arquivo .war do `vraptor-musicjungle` ainda está apontando pro banco em memória. Isso por quê? Porque eu quero mudar o meu ambiente de desenvolvimento para produção. É claro, meu código tem que, de alguma maneira, ler essa configuração. De alguma maneira, ele tinha que saber que tinha uma configuração: ambiente de desenvolvimento, ambiente de produção.

Eu quero mudar o meu ambiente de produção. Como é que eu faço isso? Eu mudo! Se a gente der uma olhada lá no `vraptor-musicjungle`, a gente vai ver que ele usa o JPA. E o JPA tem o conceito de *persistence-unit*: configuração do banco de dados.

A configuração padrão, isso é, `<persistence-unit name="default">` do banco de dados que está lá no `musicjungle` é a configuração pra usar um banco em memória, o `hsqldb` em memória. Que, toda vez que eu desligo e ligo, ele dá um problema de limpar a memória, limpar o banco. Não é um problema, pra desenvolvimento é tranquilo. Mas pra produção, não é o que eu gostaria. Eu gostaria de usar o MySQL.

Então, da mesma maneira que eu criei um `persistence-unit` pro `hsqldb`, eu vou criar um `persistence-unit` pro `mysql`. Dentro do mesmo arquivo, eu coloco dois `persistence-unit`s, um que chama `default`, e um que chama `mysql`.

O que chama `mysql` tem lá o driver do MySQL, o dialeto do MySQL, o usuário *root* do MySQL, o usuário e senha do MySQL etc. Toda a configuração do MySQL. Eu passo a ter 2 `persistence-unit`s. Só que, é claro, a gente está usando aqui o Vraptor, está usando o Vraptor JPA, que é o plugin. E esse plugin lê esses `persistence-unit`s. Por padrão, ele usa o `persistence-unit` com o nome `default`.

É claro, minha aplicação, uma das maneiras de configurá-la é ter as duas configurações dentro da aplicação. E ela, de alguma maneira, fazer o *switch*: escolher o arquivo `mysql` ou o `default`. Pra isso, eu vou criar um arquivo de propriedade. Um arquivo chamado `production.properties`. No `production.properties`, eu coloco a linha que fala:

- Olha, é pra usar o `persistence-unit` do `mysql`.

Quer dizer, eu estou na minha aplicação, seja ela Java, seja ela qual for, meio que falando assim: se o ambiente for `production`, isso é, de `production.properties`, então o `persistence-unit` que é pra usar é o do `mysql`.

Se você usa outra linguagem, outro servidor, outro serviço, outro controle MVC etc., cada um pode ter a sua maneira de fazer isso. Uma maneira de você configurar que no ambiente de produção, isso é, no `production.properties` é pra usar um `persistence-unit`; no outro ambiente, no ambiente de `development.properties`, é pra usar o `default`. Como o `default` é default, a gente não precisa criar o `development.properties`.

Legal, no meu Vraptor eu criei o meu `production.properties` dentro do projeto e configurei um segundo `persistence-unit`. Então quer dizer que, quando for production, ele vai ler o `production.properties` e vai conectar com o MySQL.

Mas como é que ele fica sabendo que está no `production`, e que não está em `dev`? Eu ainda tenho que fazer alguma coisa: eu tenho que configurar o meu Tomcat. É uma das maneiras, claro. Eu vou na configuração do meu Tomcat, isso é, no arquivo `/etc/default/tomcat7`. Nesse meu arquivo, que tem algumas coisas de *startup* do meu Tomcat. Qualquer lugar do Tomcat serviria pra isso, posso fazer de diversas maneiras.

Nesse arquivo, eu quero colocar uma opção. Eu quero colocar `JAVA_OPTS`, isto é, as opções do Java são as opções do Java e `br.com.caelum.vraptor.environment=production`.

Nesse momento, eu estou falando pro Tomcat assim:

- Olha, coloca uma variável aí de sistema que fala que o `environment` é `production`.

Aí o Tomcat passa isso pra frente. A minha aplicação web percebe: "Opa, o environment é production, deixa eu ler o `production.properties`. O que ele encontra no `production.properties` mesmo? Ele encontra: é pra ler o persistence do MySQL. E aí ele lê o `persistence-unit` do MySQL, conecta no MySQL, e tudo vai pra frente bonito e lindão.

Eu quero editar esse arquivo e colocar essa linha no final do arquivo. Como é que eu faço isso? Vou no meu Puppet e falo:

- Olha, eu gostaria de adicionar uma linha, `file_line`.

Esse `file_line` eu vou chamar de `production`, lembra, é só um nome que eu estou dando. O que ele vai fazer? Ele vai falar: no arquivo `/etc/default/tomcat7` é para colocar esta linha. Estou só dando o *escape* das aspas duplas e do cifrão pra tomar um cuidado pra escapar string dentro de string.

```
file_line { "production":
    file => "/etc/default/tomcat7",
    line => "JAVA_OPTS=\"\$JAVA_OPTS -Dbr.com.caelum.vraptor.environment=production\"",
    require => Package["tomcat7"],
    notify => Service["tomcat7"]
}
```

Estou chamando o `file_line` pra colocar uma linha dentro de um arquivo. Legal, ele vai colocar. Só que o `file_line` é uma extensão. É um módulo que é uma extensão do Puppet. Então, em vez de usar a extensão do Puppet, eu gostaria de mostrar pra vocês como é que eu posso criar funções minhas dentro do Puppet. Da mesma maneira que o `file_line` foi criado por alguém, eu posso criar o meu próprio `file_line`. Assim a gente aprende a criar algo que vai ser utilizado no nosso projeto. Por exemplo, agora eu quero uma função, uma macro, algo que se assemelha a uma função, que vai se chamar `file_line`.

Como é que eu faço isso? Eu falo que eu quero definir o `file_line`. E eu defino o `file_line` recebendo `file` e recebendo `line`. Genial! Quando ele recebe o `file` e o `line`, o que ele quer fazer? Ele quer executar alguma coisa no bash. Ele quer pegar essa linha e colocar dentro do arquivo. Então, estou dando um `echo` dessa linha praquele arquivo.

Eu estou definindo a função chamada `file_line`, que recebe o `file` e recebe o `line`. E ela executa um `echo`, isto é, ela pega a linha e enfia dentro do arquivo. Legal, funcionaria. Posso rodar uma vez que funciona.

Só que tem um perigo: se eu rodar essa linha duas vezes, só com `exec`, ele vai adicionar a mesma linha diversas vezes seguidas dentro do arquivo. E eu não quero ficar adicionando essa linha 300 vezes. Você se lembra que um script do Puppet tem que ser **idempotente**! Quando o executo diversas vezes, o resultado devia ser o que eu queria, e não cada vez ficar concatenando `file_line` s lá embaixo, dizendo que as options são diferentes.

O que eu vou falar? Vou falar pra ele executar isso somente se (unless), como a gente já tinha visto, essa linha não existe. Como é que eu verifico se uma linha ainda não existe no Linux, no bash? Eu verifico através do *grep*.

Eu dou um `grep` no arquivo, verificando se essa linha existe. Se essa linha já existe, não precisa adicionar. Se essa linha não existe, adiciona. Essa é a minha definição do `file_line`.

```
define file_line($file, $line) {
    exec { "/bin/echo '${line}' >> '${file}'":
        unless => "/bin/grep -qFx '${line}' '${file}'"
    }
}
```

E aí eu executo `sudo puppet apply` e ele coloca a linha lá dentro. Depois, eu executo de novo `sudo puppet apply` e ele não coloca a linha lá dentro. Por que ele não coloca? Porque ele já colocou, já está a linha lá dentro.

Legal. O que eu faço agora? Quero restartar. Por que eu quero restartar? Porque o meu Tomcat tem que reler essas variáveis de ambiente pra passar pro nosso Vraptor, pra passar pro plugin etc.

Eu vou restartar agora na mão, na unha. Como é que eu restarto um serviço no Linux? `sudo service tomcat7 restart`. E ele restarta o Tomcat pra gente.

Aí a gente pode ir lá no nosso navegador, se cadastrar bonito com "Guilherme", me cadastrei; vou no MySQL e ele está usando MySQL agora. Por quê? Porque eu fiz com que o Tomcat configurasse uma variável de ambiente. Essa variável de ambiente fala "O environment é production". O Vraptor já é esperto, ele lê o `production.properties`. E esse `producion.properties` está falando pra ele ler o `persistence-unit` chamado `mysql`. E não o `default`.

Isto é, se você está em qualquer outro `environment` que não é `production`, ele vai ver o `default`. E se você está no `environment` chamado `production`, ele vai ler o `mysql`. E aí ele se conecta no MySQL, funciona tudo no MySQL. Eu vou lá, edito o meu arquivo e confiro:

- Olha, é aqui que eu gostaria de adicionar essa linha.

Mas eu não vou adicionar! Não salvei o arquivo, não sou bobo. Eu vou no Puppet falar que eu quero um `file_line`. Digito o nome do meu arquivo e digito a linha. Mas eu preciso definir a função. Definir a função `file_line`.

A função `file_line` vai executar o quê? O `echo`. `echo` + essa linha pra dentro do arquivo.