02

Gerenciamento de dependências

Downloads

Caso queira usar o IVY imediatamente a partir desse vídeo, pode baixar o projeto aqui (http://s3.amazonaws.com/caelum-online-public/PM-77/ivy-cap1-importar.zip). O projeto deve ser importado no Eclipse.

Introdução ao IVY

ANT como ferramenta de construção de software ajuda muito na automatização de tarefas comuns no build de software. Quando construímos uma aplicação também importa saber componentes usaremos e suas versões. No mundo Java esses componentes são as bibliotecas, ou seja, as dependências da aplicação. A aplicação só funciona com esses componentes em uma versão específica. Isso significa que o desenvolvedor precisa se preocupar com as dependências.

O IVY é uma ferramenta que ajuda no gerenciamento das dependências e seu uso com ANT é muito simples. Para um projeto que já utiliza ANT como ferramenta de build o IVY pode ser facilmente integrado. Vamos configurar ANT com IVY e usar tarefas no ANT para resolver as dependências do projeto.

O gerenciamento de dependências manual ocorre quando o próprio desenvolvedor acessa o site da biblioteca, baixa uma versão e a adiciona no classpath da aplicação. Além disso ser muito trabalhoso, nem sempre é fácil saber qual versão da dependência é necessária e onde encontrá-la.

Por exemplo, um projeto define a dependência `hibernate-core` na versão 3.3.4. Com certeza o Hibernate utiliza outras bibliotecas, possuindo suas próprias dependências. Essas são as dependências transitivas (ou dependências de dependências). Como desenvolvedores não queremos nos preocupar em saber, por exemplo, qual versão do `commons-collections` o Hibernate usa. O gerenciador de dependência deve resolver isso e assumir a responsabilidade de cuidar das dependências transitivas.

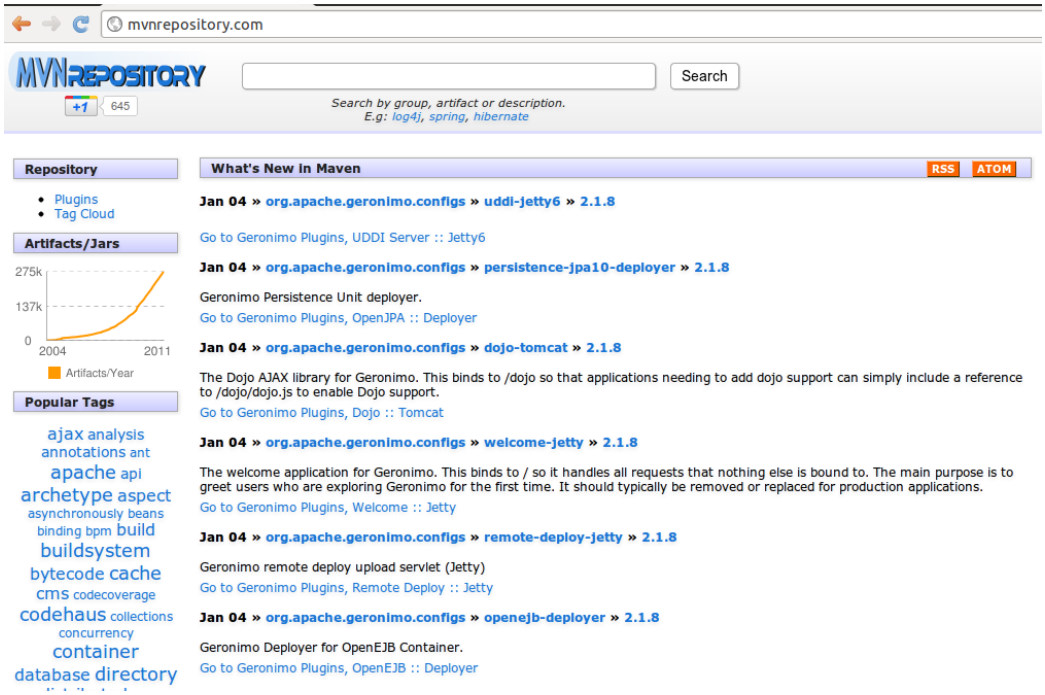
Com IVY podemos usar uma tarefa no ANT que automatiza esses passos. Isso deve ser tão fácil de usar quanto compilar ou empacotar o projeto, ou seja, apenas uma chamada no IDE ou na linha de comando. O IVY deve verificar e atualizar as dependências do projeto e avisar quando há conflitos entre JARs.

Encontrando as dependências

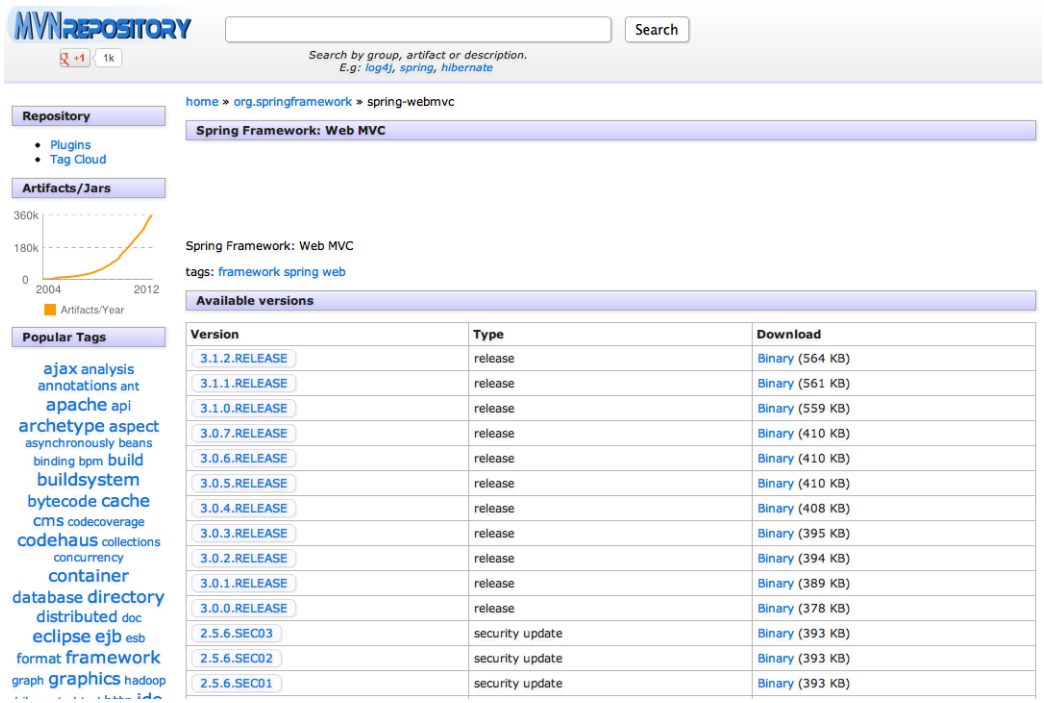
Para o primeiro exemplo, vamos definir as dependências de uma aplicação web que pretende usar Spring MVC como controlador MVC. Como já foi dito, o IVY baixará as dependências automaticamente, mas de onde ele as baixará?

Existem servidores na internet com a tarefa de fornecer JARs para aplicações. Esses servidores, ou melhor repositórios, foram criados para serem utilizados com outra ferramenta de construção de software, o Maven (http://maven.apache.org). O IVY utiliza os mesmos repositórios do Maven, aproveitando a ampla infraestrutura já existente.

Existem vários repositórios na internet. Um lugar central para pesquisar pelos artefatos é o site (http://mvnrepository.com/)http://mvnrepository.com (http://mvnrepository.com).



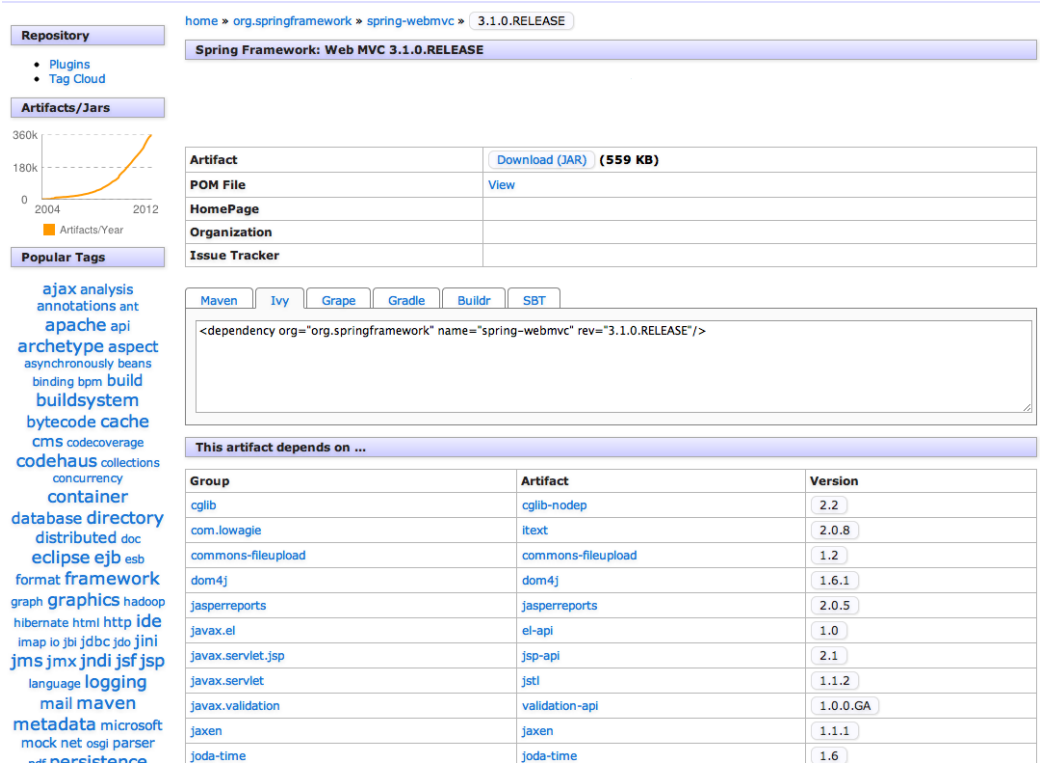
Procurando pelo "spring mvc" no [(http://mvnrepository.com/)http://mvnrepository.com (http://mvnrepository.com)](http://mvnrepository.com/) vamos encontrar o registro do framework `spring-mvc` [(http://mvnrepository.com/artifact/org.springframework/spring-webmvc)](http://mvnrepository.com/artifact/org.springframework/spring-webmvc). Nesse link podemos encontrar todas as versões disponíveis dentro do repositório.



Seguindo um link de uma versão aparecem mais detalhes sobre aquela dependência (ou artefato - segundo o Maven). Nessa página, escolhendo a aba `IVY` , encontramos a configuração para o arquivo `ivy.xml` que é utilizado para configurar as dependências do nosso projeto, por exemplo:

```
<dependency org="org.springframework" name="spring-webmvc" rev="3.1.0.RELEASE"/>
```

Na mesma página também estão listados os artefatos do Spring MVC, ou seja dependências desse framework. São essas as dependências transitivas.



Para ver mais detalhes sobre a versão 3.1.0 do Spring MVC acesse o site do repositório: [spring-webmvc-3.1.0 (http://mvnrepository.com/artifact/org.springframework/spring-webmvc/3.1.0.RELEASE)](http://mvnrepository.com/artifact/org.springframework/spring-webmvc/3.1.0.RELEASE)

Declaração das dependências

A configuração das dependências com IVY não costuma ser feita dentro do `build.xml` . O IVY possui um arquivo XML separado, o `ivy.xml` . Esse arquivo descreve um módulo e possui, além de algumas informações genéricas sobre o projeto, a definição das dependências.

A tag obrigatória `info` define a organização ou empresa (`br.com.caelum`) e nomeia o módulo, normalmente o nome da aplicação (aqui `agenda`). Após essas tags, seguem as dependências da aplicação. No exemplo abaixo há apenas uma. O módulo depende do framework `spring-mvc` na versão 3.1.0 que copiamos do site [(http://mvnrepository.com/)http://mvnrepository.com (http://mvnrepository.com)](http://mvnrepository.com):

```
<ivy-module version="2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://ant.apache.org/ivy/schemas/ivy.xsd">

    <info organisation="br.com.caelum" module="agenda" />

    <dependencies>
        <dependency org="org.springframework" name="spring-webmvc" rev="3.1.0.RELEASE" conf="de
    </dependencies>

</ivy-module>
```

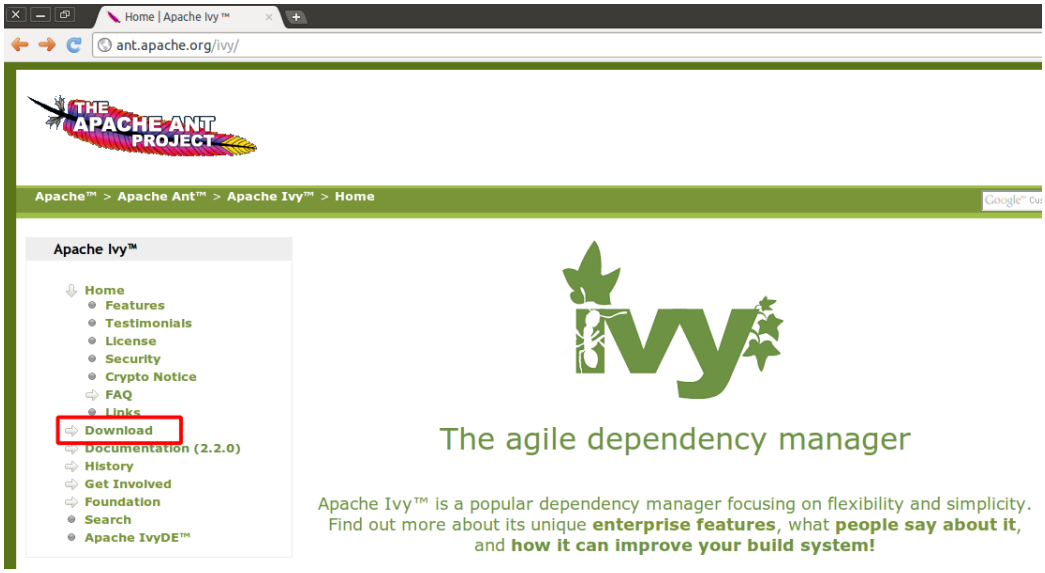
Task para resolver os JARs

Com as dependências definidas no `ivy.xml` , podemos utilizar as tarefas do IVY dentro do `build.xml` para baixá-las automaticamente. Vamos definir um `target` para resolver as dependências usando a tarefa específica do IVY `ivy:retrieve` :

```
<project name="agenda" xmlns:ivy="antlib:org.apache.ivy.ant">
    ...
    <target name="atualiza-dependencias" description="resolver as dependencias do projeto">
        <ivy:retrieve />
    </target>
    ...
</project>
```

Repare que foi preciso definir um namespace `xmlns:ivy=` na declaração do projeto.

Para o ANT achar a tarefa é preciso ter o JAR do IVY no classpath. O JAR vem da distribuição do Apache IVY e pode ser baixado no site do projeto: (http://ant.apache.org/ivy/)http://ant.apache.org/ivy/ (http://ant.apache.org/ivy/)



Então podemos disponibilizar o JAR do IVY (por exemplo `ivy-2.2.0.jar`) no classpath do ANT. No final é o ANT que vai usar IVY. Para adicionar o IVY no classpath podemos configurar o classpath pelo comando `ant` com o argumento `-lib`, por exemplo:

```
ant -lib ivy-2.2.0.jar
```

Alternativamente podemos declarar uma tag `path` com a biblioteca do IVY e definir na task no `build.xml`. Essa maneira tem a vantagem de não precisarmos lembrar de passar o JAR do IVY na linha de comando:

```
<path id="ivy.lib.path">
    <fileset dir="ivy-lib" includes="*.jar"/>
</path>

<taskdef resource="org/apache/ivy/ant/antlib.xml" uri="antlib:org.apache.ivy.ant" classpathref='
```

Pronto, a integração do ANT e IVY está feita. Foi preciso definir o classpath (JAR do IVY na pasta `lib`), declarar o `namespace` e usar a tarefa do IVY (`ivy:retrieve`) dentro do `build.xml`.

O arquivo `iyy.xml` deve estar na mesma pasta do `build.xml`. Assim podemos executar o target `ant atualiza-dependencias` gerando o seguinte resultado:

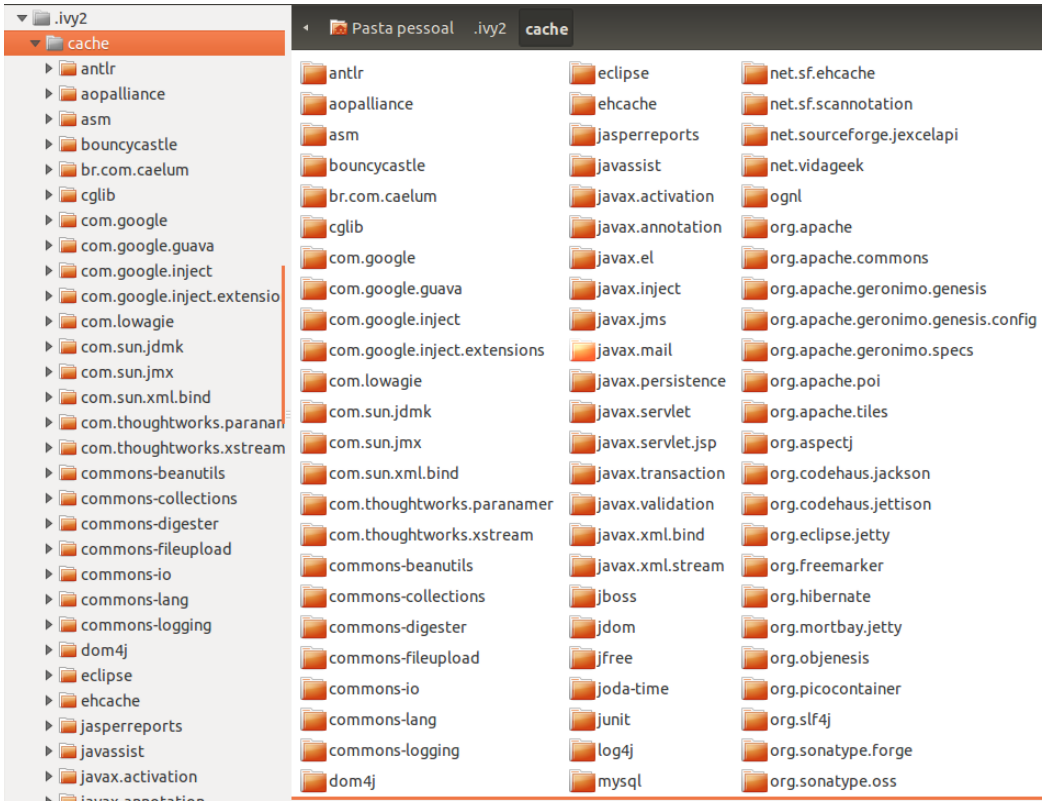
```
atualiza-dependencias:
[ivy:retrieve] :: Ivy 2.2.0 - 20100923230623 :: http://ant.apache.org/ivy/ ::
[ivy:retrieve] :: loading settings :: url = jar:file:/usr/share/ant/lib/ivy-2.2.0.jar!/org/apacl
[ivy:retrieve] :: resolving dependencies :: br.com.caelum#agenda;working@nico-vaio
[ivy:retrieve]     confs: [default]
[ivy:retrieve]     found org.springframework#spring-webmvc;3.1.0.RELEASE in public
[ivy:retrieve]     [3.1.0.RELEASE] org.springframework#spring-webmvc;3.1.0.RELEASE
....
[ivy:retrieve] downloading http://repo1.maven.org/maven2/org/springframework/spring-webmvc/3.1.(
[ivy:retrieve] ..........................................................................

....

[ivy:retrieve]     [SUCCESSFUL ] org.springframework#spring-aop;3.1.0.RELEASE!spring-aop.jar (3:
[ivy:retrieve] :: resolution report :: resolve 31969ms :: artifacts dl 152590ms
    ---------------------------------------------------------------------
    |                  |            modules            ||   artifacts   |
    |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
    ---------------------------------------------------------------------
    |      default     |   11  |   9   |   9   |   0   ||   11  |   9   |
    ---------------------------------------------------------------------
[ivy:retrieve] :: retrieving :: br.com.caelum#agenda
[ivy:retrieve]     confs: [default]
[ivy:retrieve]     11 artifacts copied, 0 already retrieved (3066kB/36ms)
```

A mensagem pode variar e na primeira execução demorar.

Cache do Ivy

A segunda execução será muito mais rápido. Mas por quê? O que acontece é que o IVY baixa os JARs do repositório, colocando-os dentro da pasta `lib` do projeto. Mas além de colocar as bibliotecas na pasta `lib` o IVY guarda os JARs em uma pasta independente do projeto. Normalmente essa pasta se encontra na raiz da pasta do usuário e se chama `.ivy2` :

```
/pastaDoUsuario/.ivy/cache/....
```



Na segunda execução os JARs vêm da pasta `.ivy2` e só serão copiadas localmente. No nosso exemplo, uma única dependência `spring-mvc` o IVY colocou 11 JARs dentro da pasta `lib` :

```
- aopalliance.jar
- commons-logging.jar
- spring-asm.jar
- spring-aop.jar
- spring-beans.jar
- spring-context.jar
- spring-context-support.jar
- spring-core.jar
- spring-expression.jar
- spring-web.jar
- spring-webmvc.jar
```

Resumo

Vimos como integrar o IVY com ANT, usando a tarefa `ivy:retrieve` . Como o IVY aproveita os mesmo repositório do Maven podemos procurar as dependências dentro desses repositórios. Para a configuração das dependências usamos um arquivo especifico do IVY, o `ivy.xml` .