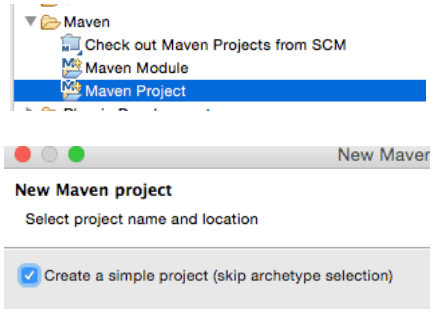
05

Mãos à obra: Maven

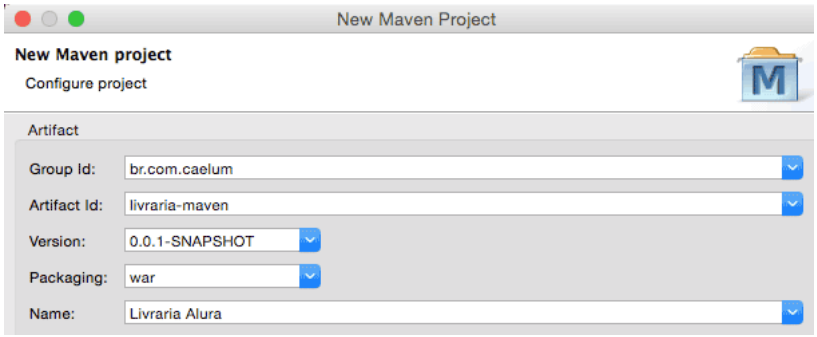
Que tal usarmos Maven em nosso projeto e delegarmos todas as nossas dependências (jars) para ele? Seguem passos gerais para essa tarefa. Se sentir dificuldade e quiser algo mais detalhado não deixe de recorrer ao vídeo do capítulo.

1 - Vamos criar um novo projeto, que terá a integração com o Maven. Para isso, iremos em `New -> Project -> Maven Project`. Selecionamos **Create a simple project (skip archetype selection)**, pois iremos criar tudo do zero.

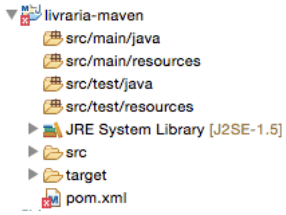


2 - Algumas configurações do projeto:

- **`Group Id`** : Esse é o nosso pacote, que no caso é `br.com.caelum` ;
- **`Artifact Id`** : Esse é o nome do projeto. Já temos `livraria` no `workspace` , então vamos colocar `livraria-maven` ;
- **`Version`** : `0.0.1-SNAPSHOT` ;
- **`Packaging`** : `war` ;
- **`Name`** : Livraria Alura;



3 - Após criação o seu projeto deve ter a estrutura seguinte:



Agora, iremos copiar todos os pacotes do projeto `livraria` **(apenas os pacotes, sem a pasta `META-INF`)** para dentro da pasta **`src/main/java`** do novo projeto. Nada compila ainda, mas não se preocupe. Agora sim iremos copiar a pasta `META-INF` , para dentro de **src/main/resources**. Por último, vamos copiar tudo que está **dentro** da pasta `WebContent` e colar na pasta **src/main/webapp**.

4 - O arquivo `pom.xml`

O `pom.xml` é o arquivo XML do Maven. Vamos lembrar que queremos que o Maven gerencie nossos `JARs` e acabe com o trabalho manual. Para isso, precisamos informar ao Maven quais dependências ele irá gerenciar. Dentro do XML, nós informamos isso ao Maven através da tag `<dependecies></dependencies>` . Porém fica difícil saber a versão, o nome da dependência, e por isso existe um site específico para nos ajudar, o mvnrepository.com (http://mvnrepository.com). Nele podemos procurar por dependências dentro do repositório do Maven na internet.

Já separamos aqui os as declarações da cada dependência do **mvnrepository** necessárias. Ou você pode já baixar o `pom.xml` completo aqui (https://s3.amazonaws.com/caelum-online-public/jsf-cdi/arquivos/pom.xml):

- weld-servlet (http://mvnrepository.com/artifact/org.jboss.weld.servlet/weld-servlet/2.3.3.Final)
- validation-api (http://mvnrepository.com/artifact/javax.validation/validation-api/1.1.0.Final)
- primefaces (http://mvnrepository.com/artifact/org.primefaces/primefaces/5.3)
- mysql-connector (http://mvnrepository.com/artifact/mysql/mysql-connector-java/5.1.38)
- hibernate (http://mvnrepository.com/artifact/org.hibernate/hibernate-entitymanager/5.1.0.Final)
- jsf-api (http://mvnrepository.com/artifact/com.sun.faces/jsf-api/2.2.13)
- jsf-impl (http://mvnrepository.com/artifact/com.sun.faces/jsf-impl/2.2.13)

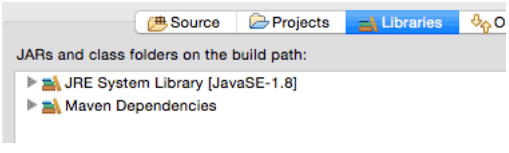
5 - Algumas dependências podem não existir no repositório padrão do Maven, como é o caso do JAR do tema do Primefaces. Para resolver isso, precisamos adicionar no `pom.xml` o repositório externo, para aí sim adicionar a dependência. Isso acontece com o *primefaces themes*. Você pode achar o repositório do PrimeFaces aqui (http://www.primefaces.org/downloads). Fora de `<dependencies>` , você adiciona o novo repositório:

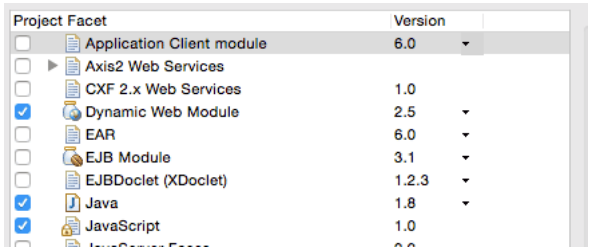
```
<repository>
    <id>prime-repo</id>
    <name>PrimeFaces Maven Repository</name>
    <url>http://repository.primefaces.org</url>
    <layout>default</layout>
</repository>
```

E agora podemos pegar aqui (http://www.primefaces.org/themes) as dependências que não existiam no mvnrepository, porém existem no repositório deles adicionado.

```
<dependency>
    <groupId>org.primefaces.themes</groupId>
    <artifactId>all-themes</artifactId>
    <version>1.0.10</version>
</dependency>
```

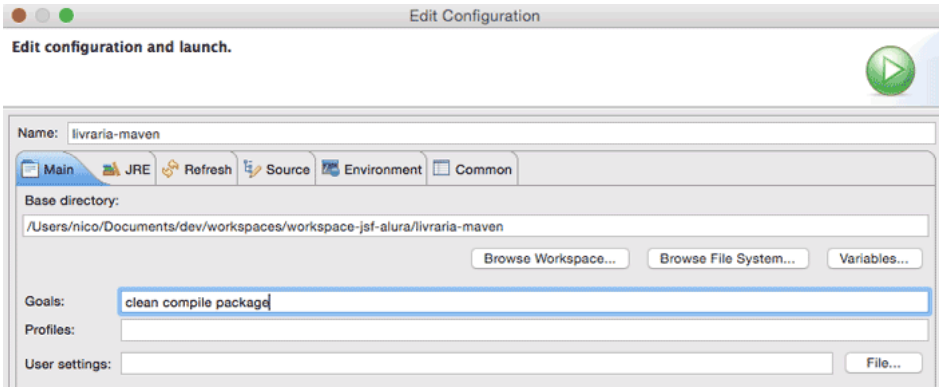
6 - Com as dependências todas gerenciadas pelo Maven, a maior parte dos erros deve sumir. Porém, ao criarmos o projeto no Eclipse como *Maven Project*, ele acabou por se utilizar de algumas configurações bem antigas. A versão da JRE usada é a 1.5, e precisamos da JRE 1.8. Por isso, precisamos ir em *properties* do projeto, e no *Java Compiler*, desmarcamos a opção *Enable project specific settings*. Agora em *Java Build Path*, apagamos a *Library* `JRE System Library [J2SE-1.5]` e adicionamos uma nova, com a JRE 1.8. Ainda sobra um erro, por causa da versão do `Java project facet` , basta clicarmos no erro, apertar `Ctrl+1` e mudar o `Java facet` para 1.8.





7 - Podemos também rodar a aplicação através do Maven, para assim não ficarmos tão ligados ao Eclipse. Para isso, **botão direito** no projeto, e `Run As... -> Maven Build` . Em **goals**, ou seja, quando executarmos com o Maven, queremos que ele, por exemplo, dê um *clean*, compile o projeto e gere um WAR para *deploy*, para isso, colocamos:

```
clean compile package
```



Isso já é suficiente para usarmos Maven com o nosso projeto.