Create an archetype with a property file

使用“属性文件”创建“原型项目”

Creating an archetype using a property file is possible by defining a text file containing the properties that will lead the archetype creation, and by calling the Archetype Plugin with the command line property `archetype.properties` giving it the path to the property file.

属性文件将引导“原型项目”创建，并通过调用原型插件的命令行

The directory above the would be archetype project:

原型项目的目录

```
$ tree
.
|____archetype.properties
|____some-project
| |____pom.xml
| |____src
| | |____main
| | | |____groovy
| | | | |____com
| | | | | |____company
| | | | | | |____App.groovy
| | | |____java
| | | | |____com
| | | | | |____company
| | | | | | |____App.java
| | |____test
| | | |____java
| | | | |____com
| | | | | |____company
| | | | | | |____AppTest.java
| | | |____resources
| | | | |____App.properties
```

The content of the property file:

属性文件的内容

```
$ cat archetype.properties
archetype.groupId=my.group.id
archetype.artifactId=archetype-with-properties
archetype.version=2.0

archetype.filteredExtensions=java
archetype.languages=groovy

an_additional_property=my specific value
```

Creating the archetype:

创建“原型项目”

mvn archetype:create-from-project -Darchetype.properties=../archetype.properties

```
$ cd some-project
$ mvn archetype:create-from-project -Darchetype.properties=../archetype.properties
[INFO] Scanning for projects...
[INFO] Searching repository for plugin with prefix: 'archetype'.
[INFO] ------------------------------------------------------------------------
```

```
[INFO] Building some-project
[INFO]    task-segment: [archetype:create-from-project] (aggregator-style)
[INFO] ------------------------------------------------------------------------
[INFO] Preparing archetype:create-from-project
[INFO] ------------------------------------------------------------------------
[INFO] Building some-project
[INFO] ------------------------------------------------------------------------
[INFO] No goals needed for project - skipping
[INFO] Setting property: classpath.resource.loader.class => 'org.codehaus.plexus.velocity.Contex
tClassLoaderResourceLoader'.
[INFO] Setting property: velocimacro.messages.on => 'false'.
[INFO] Setting property: resource.loader => 'classpath'.
[INFO] Setting property: resource.manager.logwhenfound => 'false'.
[INFO] [archetype:create-from-project]
[INFO] Setting default groupId: com.company
[INFO] Setting default artifactId: some-project
[INFO] Setting default version: 1.0-SNAPSHOT
[INFO] Setting default package: com.company
[INFO] Archetype created in target/generated-sources/archetype
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2 seconds
[INFO] Finished at: Tue Sep 16 19:42:06 CEST 2008
[INFO] Final Memory: 8M/15M
[INFO] ------------------------------------------------------------------------

$ cd target/generated-sources/archetype/
$ mvn install
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building archetype-with-properties
[INFO]    task-segment: [install]
[INFO] ------------------------------------------------------------------------
[INFO] [resources:resources]
[INFO] Using default encoding to copy filtered resources.
[INFO] [resources:testResources]
[INFO] Using default encoding to copy filtered resources.
[INFO] Setting property: classpath.resource.loader.class => 'org.codehaus.plexus.velocity.Contex
tClassLoaderResourceLoader'.
[INFO] Setting property: velocimacro.messages.on => 'false'.
[INFO] Setting property: resource.loader => 'classpath'.
[INFO] Setting property: resource.manager.logwhenfound => 'false'.
[INFO] [archetype:jar]
[INFO] [archetype:add-archetype-metadata]
[INFO] [archetype:integration-test]
[INFO] [install:install]
[INFO] Installing /private/tmp/archetypes/some-project/target/generated-sources/archetype/target
/archetype-with-properties-2.0.jar to /Users/raphaelpieroni/.m2/repository/my/group/id/archetype
-with-properties/2.0/archetype-with-properties-2.0.jar
[INFO] [archetype:update-local-catalog]
```

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 5 seconds
[INFO] Finished at: Tue Sep 16 19:43:57 CEST 2008
[INFO] Final Memory: 10M/20M
[INFO] ------------------------------------------------------------------------
```

Using that archetype:

```
$ mvn archetype:generate -DarchetypeCatalog=local
$ mvn archetype:generate -DarchetypeCatalog=local
[INFO] Scanning for projects...
[INFO] Searching repository for plugin with prefix: 'archetype'.
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Default Project
[INFO]    task-segment: [archetype:generate] (aggregator-style)
[INFO] ------------------------------------------------------------------------
[INFO] Preparing archetype:generate
[INFO] No goals needed for project - skipping
[INFO] Setting property: classpath.resource.loader.class => 'org.codehaus.plexus.velocity.Contex
tClassLoaderResourceLoader'.
[INFO] Setting property: velocimacro.messages.on => 'false'.
[INFO] Setting property: resource.loader => 'classpath'.
[INFO] Setting property: resource.manager.logwhenfound => 'false'.
[INFO] [archetype:generate]
[INFO] Generating project in Interactive mode
[INFO] No archetype defined. Using maven-archetype-quickstart (org.apache.maven.archetypes:maven
-archetype-quickstart:1.0)
Choose archetype:
1: local -> archetype-with-properties (archetype-with-properties)
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 1
Define value for groupId: : foo
Define value for artifactId: : bar
Define value for version:  1.0-SNAPSHOT: :
Define value for package:  foo: :
Confirm properties configuration:
an_additional_property: my specific value
groupId: foo
artifactId: bar
version: 1.0-SNAPSHOT
package: foo
 Y: : N
Define value for an_additional_property:  my specific value: : my new value
Define value for groupId: : my.new.group
Define value for artifactId: : my-new-artifact
Define value for version:  1.0-SNAPSHOT: :
Define value for package:  my.new.group: :
Confirm properties configuration:
an_additional_property: my new value
groupId: my.new.group
```

```
artifactId: my-new-artifact
version: 1.0-SNAPSHOT
package: my.new.group
 Y: :
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 58 seconds
[INFO] Finished at: Tue Sep 16 19:51:09 CEST 2008
[INFO] Final Memory: 8M/15M
[INFO] ------------------------------------------------------------------------
```

As you can see, the additional property is provided by default. As one want to override the value, one just have to answer `N` to the first confirmation, then the whole bunch of properties is proposed.

The tree of the generated project show that the only language recognized as such is groovy. The property `archetype.languages` gives the list of the languages. The languages are the directories in `src/main` and `src/test` that have the capacity to contain a "packaged" directory tree. This means that the `package` value is used to generate the new directory for that language.

```
$ cd my-new-artifact/
 tree
.
|____pom.xml
|____src
| |____main
| | |____groovy
| | | |____my
| | | | |____new
| | | | | |____group
| | | | | | |____App.groovy
| | |____java
| | | |____com
| | | | |____company
| | | | | |____App.java
| |____test
| | |____java
| | | |____com
| | | | |____company
| | | | | |____AppTest.java
| | |____resources
| | | |____App.properties
```

The contents of both App.groovy and App.java shows that the only filtered files are java files. The property `archetype.filteredExtensions` gives the list of file extensions that permit the filtering. In the example, only the files whose names ending with `.java` are filtered.

```
$ cat src/main/groovy/my/new/group/App.groovy
package com.company

class App
```

```
{
  def value = 'my specific value'
}

$ cat src/main/java/com/company/App.java
package my.new.group;

/**
 * Hello world!
 *
 */
public class App
{
    String value = "my new value";
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
    }
}
```

The languages and the filtered extensions are used even if that don't make any sense. All the properties defined in the property file can be used as command line properties.

When undefined, the `archetype.languages` and `archetype.filteredExtensions` properties are given sensible default values:

- `archetype.languages` have: `java, xml, txt, groovy, cs, mdo, aj, jsp, gsp, vm, html, xhtml, properties, .classpath, .project`. Notice the dotted filtered extensions that contains the complete file names.
- `archetype.filteredExtensions` have: `java, groovy, csharp, aspectj`.