Tabla de Contenidos

El concepto de Maven Module es uno de los conceptos más importantes de Maven pero que en muchos casos los desarrolladores no usan demasiado. Sin embargo cada día es más importante el saber como usarlos ya que cada día modularizamos más nuestro software. Vamos a crear un par de ejemplos sencillos con Maven y explicar para que sirven un Maven Module. Lo primero que me voy a hacer es crear un proyecto de Maven que disponga de una clase de dominio Persona:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.arquitecturajava</groupId>
  <artifactId>dominio</artifactId>
  <version>0.0.1-SNAPSHOT</version>

   <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
```

```
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

</project>
```

En este caso el fichero POM solo dispone de las propiedades básicas para que se trate de un proyecto de Java 8 .

El siguiente paso es crear una clase :

```
package com.arquitecturajava.dominio;


public class Persona {

  private String nombre;
  private String apellidos;
  private int edad;
  public String getNombre() {
    return nombre;
  }
  public void setNombre(String nombre) {
    this.nombre = nombre;
  }
  public String getApellidos() {
    return apellidos;
  }
  public void setApellidos(String apellidos) {
    this.apellidos = apellidos;
  }
  public int getEdad() {
```

```
    return edad;
  }
  public void setEdad(int edad) {
    this.edad = edad;
  }
  public Persona(String nombre, String apellidos, int edad) {
    super();
    this.nombre = nombre;
    this.apellidos = apellidos;
    this.edad = edad;
  }
}
```

Ya disponemos de nuestro primer Proyecto Maven :

Maven Module y dependencias

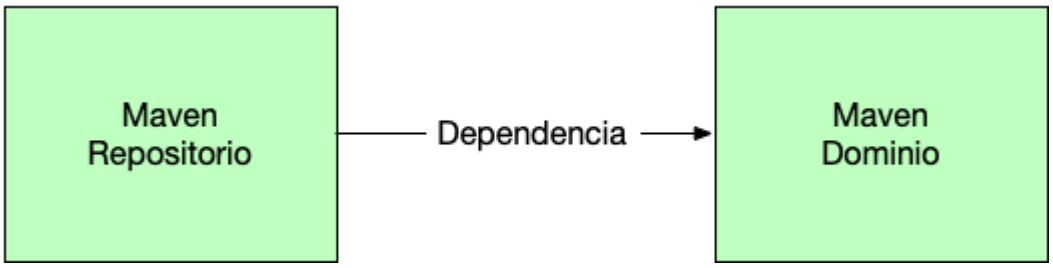
Es momento de construir otro proyecto que por ejemplo defina los repositorios y se encargue de tener un repositorio que nos devuelve una lista de Personas. El primer paso es ver el contenido del fichero Pom.xml que evidentemente tendrá una dependencia con el proyecto de dominio Maven

Aqui tenemos su código :

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
```

```
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.arquitecturajava</groupId>
  <artifactId>repository</artifactId>
  <version>0.0.1-SNAPSHOT</version>

   <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  <dependencies>
        <dependency>
                <groupId>com.arquitecturajava</groupId>
                <artifactId>dominio</artifactId>
                <version>0.0.1-SNAPSHOT</version>
        </dependency>
  </dependencies>
</project>
```

Ya disponemos de los dos proyectos

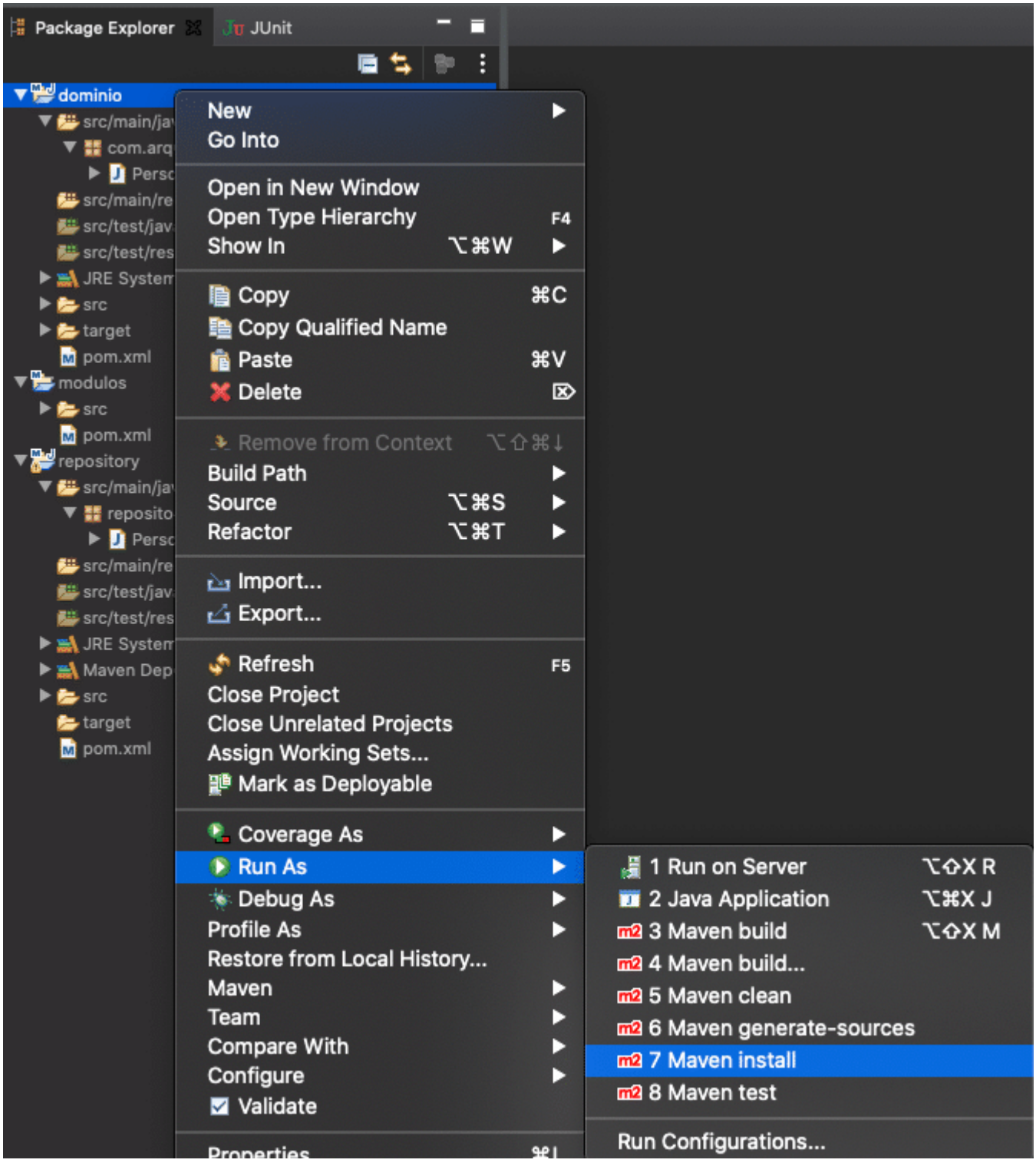


Es momento de ver el código de este segundo:

```
package repository;

import java.util.Arrays;
import java.util.List;
```

```
import com.arquitecturajava.dominio.Persona;

public class PersonaRepository {

  public List<Persona> buscarTodos() {
    Persona p= new Persona("pepe","perez",20);
    Persona p1= new Persona("ana","sanchez",30);
    return Arrays.asList(p,p1);
  }
 }
```

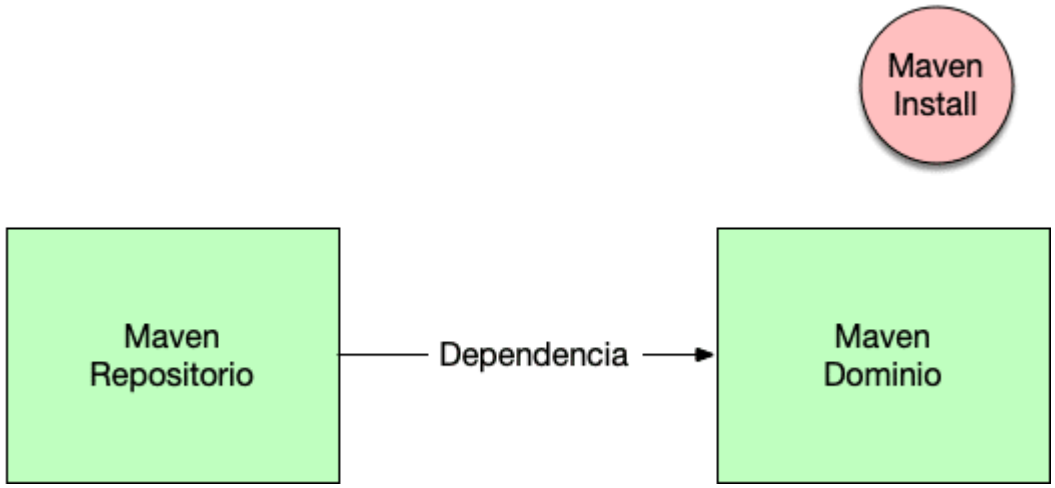
Cómo podemos observar es muy muy básico simplemente devuelve una lista de Personas, no hace más. Ahora bien para que este proyecto nos funcione correctamente tendremos que tener correctamente compilado ,empaquetado e instalado el proyecto de dominio. .Eso lo podemos hacer con mvn install o en el eclipse run as maven install



Al realizar esta operación se instalara el artefacto

```
[INFO] --- maven-install-plugin:2.4:install (default-install) @ dominio ---
[INFO] Installing /Users/cecilioalvarezcaules/basiconuevo/dominio/target/dominio-0.0.1
[INFO] Installing /Users/cecilioalvarezcaules/basiconuevo/dominio/pom.xml to /Users/ce
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.981 s
[INFO] Finished at: 2021-03-04T10:44:31+01:00
```

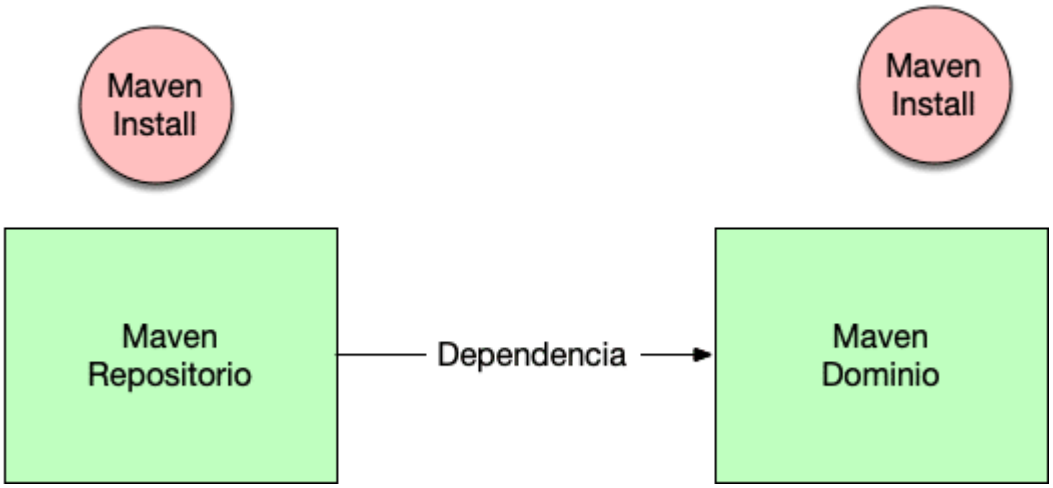
Ya tenemos instalado el primer artefacto en el repositorio:



Tenemos la primera parte construida es momento de realizar la misma instalación para el repositorio.

```
NFO] --- maven-install-plugin:2.4:install (default-install) @ repository ---
NFO] Installing /Users/cecilioalvarezcaules/basiconuevo/repository/target/repository-
NFO] Installing /Users/cecilioalvarezcaules/basiconuevo/repository/pom.xml to /Users/
NFO] ------------------------------------------------------------------------
NFO] BUILD SUCCESS
NFO] ------------------------------------------------------------------------
NFO] Total time:  0.924 s
NFO] Finished at: 2021-03-04T12:00:10+01:00
```

Con esto pasaremos a tener instalados los dos artefactos



Maven Modules e integración

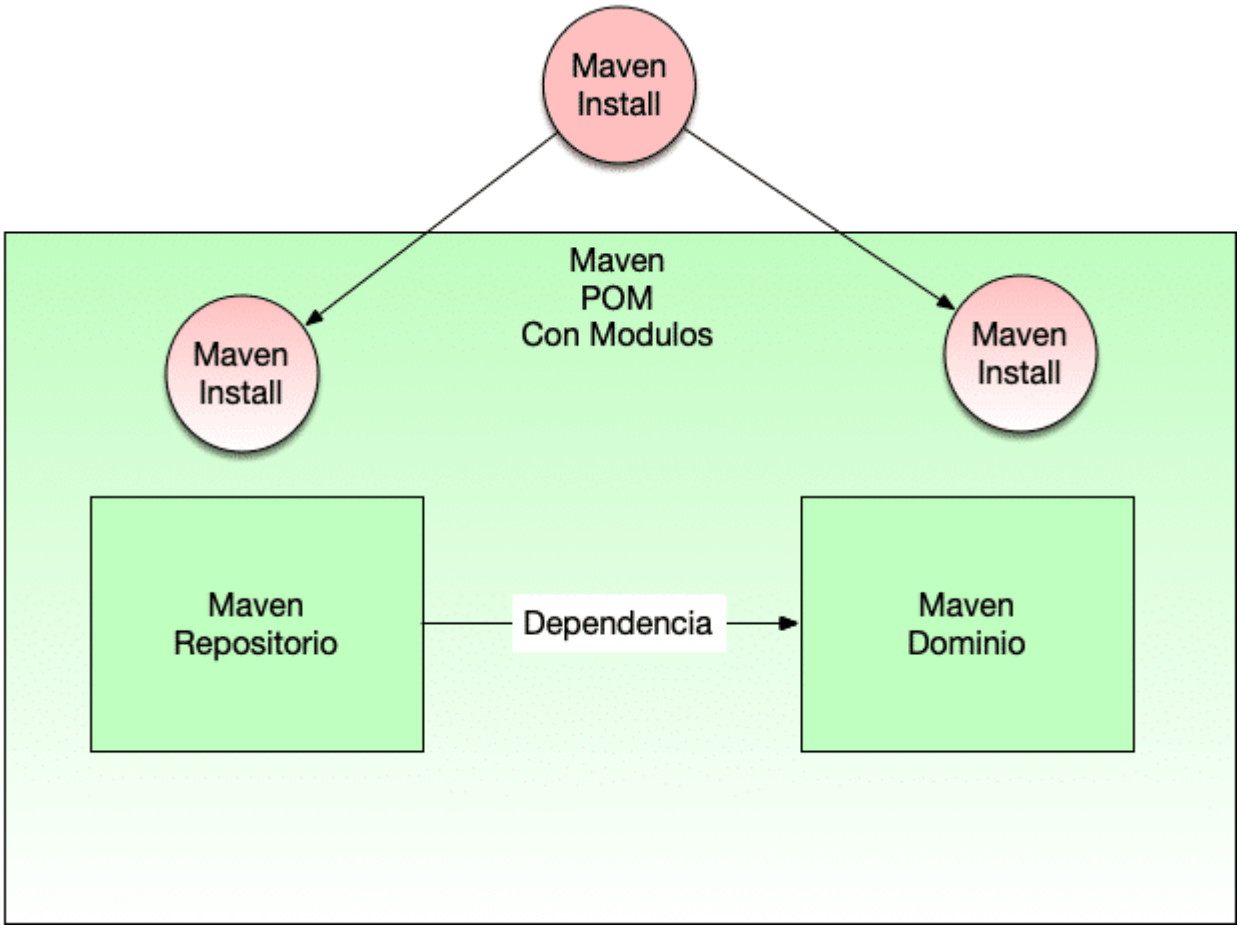
El problema que tenemos es que hemos tenido que instalar dos artefactos en el repositorio de Maven , cada uno de forma independiente , algo que claramente no es lo óptimo ya que son dos artefactos ligados intimamente al mismo Proyecto. Sería mucho mejor poder instalar los dos de forma simultánea . Para eso nos pueden ayudar los Maven Modules vamos a ver cómo usarlos.

Lo primero que tenemos que hacer es generarnos otro proyecto que denominaremos módulos y será un proyecto Maven de tipo POM.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.arquitecturajava</groupId>
  <artifactId>modulos</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>pom</packaging>
  <modules>
        <module>../dominio</module>
```

```
        <module>../repository</module>
  </modules>
</project>
```

Este tipo de proyectos no incluye código alguno sino que simplemente se usan para organizar el resto . En este caso hemos dado de alta la etiqueta modules y dentro de ella dos módulos el de dominio y el de repositorio.



Es momento de ejecutar maven install con este proyecto (modulos) . Este proyecto como tal esta vacío pero al incluir dos módulos realizará la instalación de forma automática de cada uno de ellos apoyandose en las dependencias para generar un orden.

```
[INFO] --- maven-install-plugin:2.4:install (default-install) @ modulos ---
[INFO] Installing /Users/cecilioalvarezcaules/basiconuevo/modulos/pom.xml to /User
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for modulos 0.0.1-SNAPSHOT:
[INFO]
[INFO] dominio ............................................ SUCCESS [  0.816 s]
[INFO] repository ......................................... SUCCESS [  0.033 s]
[INFO] modulos ............................................ SUCCESS [  0.006 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

Acabamos de realizar nuestro primer proyecto con maven módules

Otros artículos relacionados

1. ¿Que es un Maven LifeCycle y como funciona?
2. ¿Que es un Maven Goal ?
3. ¿Qué es un Java Maven Artifact ?
4. Curso Maven