Maven Flexibilidad y el manejo de Artefactos . ¿Cómo podemos aumentar la flexibilidad de nuestros proyectos Maven utilizando Artefactos? .Vamos a ver un ejemplo sencillo apoyándoos en la creación de un interface de Servicio y su implementación en Java.

Un servicio es un conjunto de funcionalidad agrupada que nos sirve como unidad reutilizable de código. Ahora bien cada día tenemos más necesidad de diseñar Servicios que sean fácilmente reutilizables.

Maven y Flexibilidad

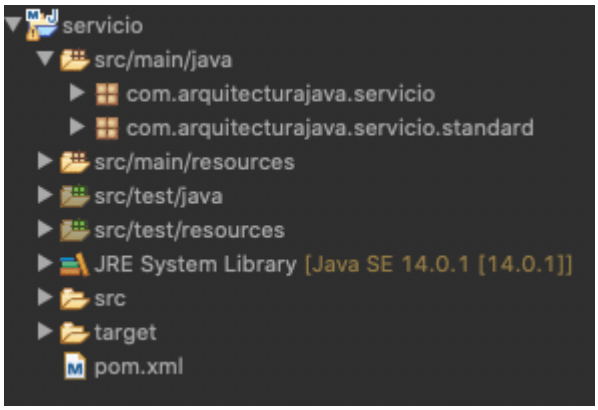
Para ello lo más sencillo es construir un proyecto de Maven que disponga de un interface de Servicio y su implementación y ver como configurarlo

El código sería:

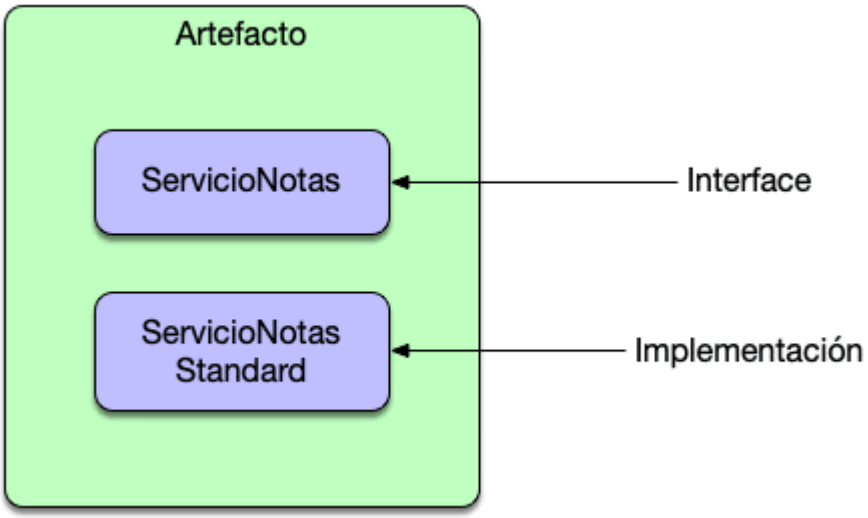
```
package com.arquitecturajava.servicio;

public interface ServicioNotas {
        public void obtenerNotas();
}
```

```
package com.arquitecturajava.servicio.standard;

import com.arquitecturajava.servicio.ServicioNotas;

public class ServicioNotasStandard  implements ServicioNotas {

        public void obtenerNotas() {
                System.out.println("notas standard del servicio");
        }

}
```

Acabamos de definir un servicio y su implementación . Ambos pertenecen a un proyecto Maven y se encuentran ubicados en packages diferentes.

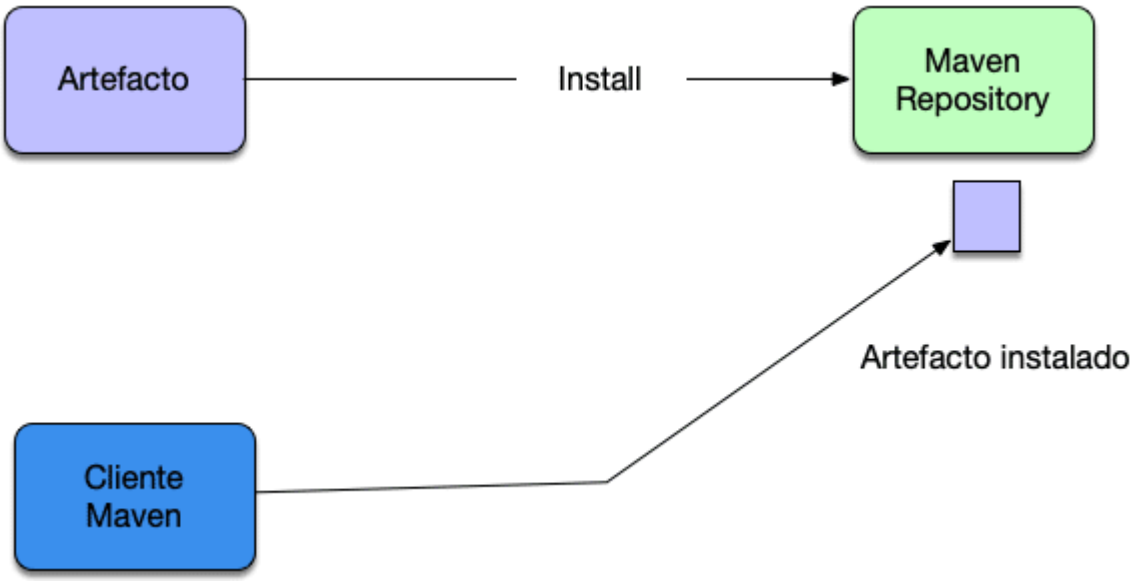


Estamos ante un ejemplo muy elemental ,el cual hemos construido como proyecto Maven .

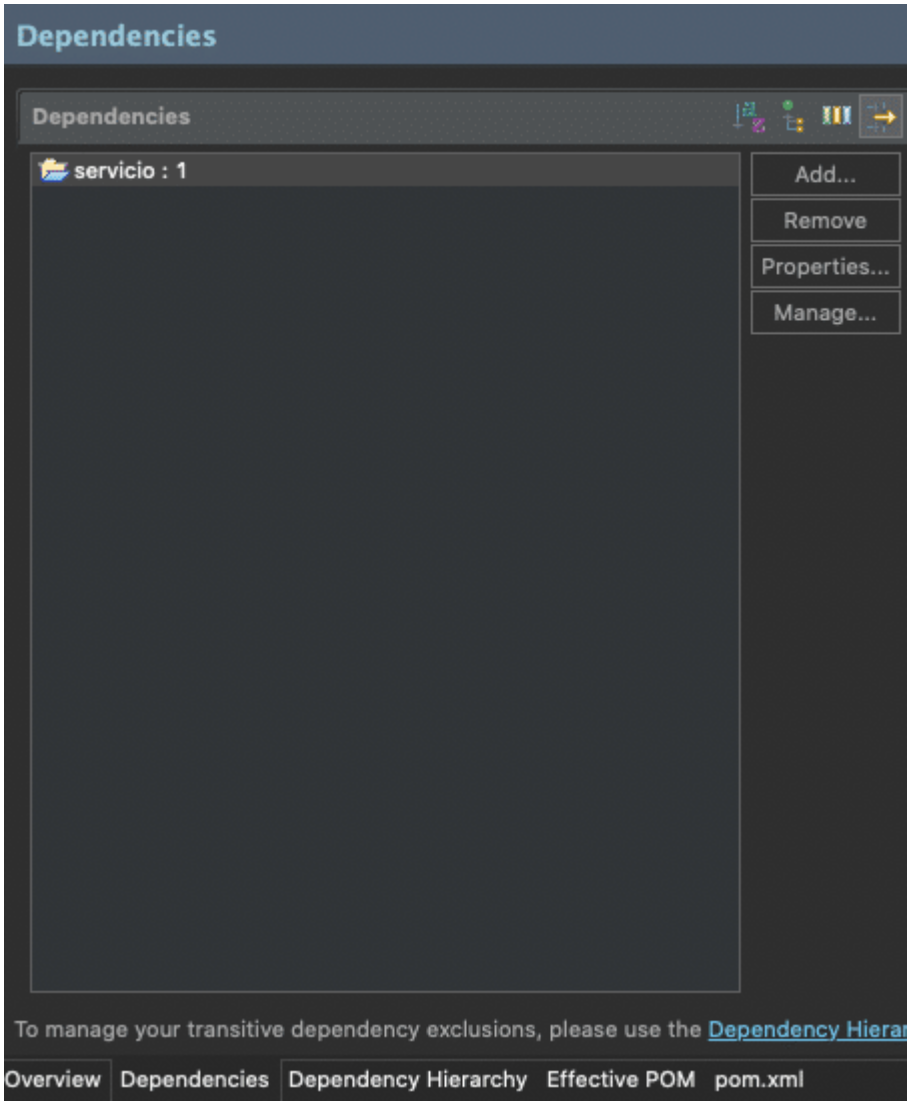


Tanto el interface como su implementación se encuentran dentro del mismo Artefacto. Podemos utilizar Eclipse y realizar un Maven install para instalar en el repositorio local este artefacto y que otros proyectos se puedan apoyar en él.

Una vez instalado podemos construir otro proyecto Maven (ClienteMaven) que se encargará de importar esta dependencia a nivel de POM.xml para utilizarla.



A nivel de Eclipse el nuevo proyecto tendrá que incluir la dependencia en el fichero pom.xml



El fichero XML quedaría:

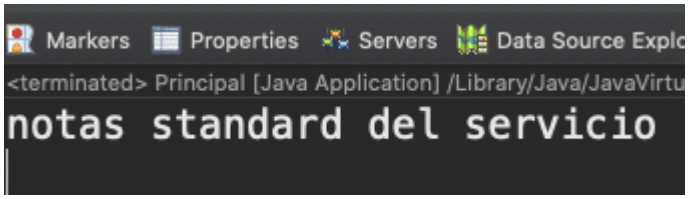
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.arquitecturajava.cliente</groupId>
  <artifactId>ClienteMaven</artifactId>
  <version>1</version>
```

```
  <dependencies>
        <dependency>
                <groupId>com.arquitecturajava</groupId>
                <artifactId>servicio</artifactId>
                <version>1</version>
        </dependency>
  </dependencies>
</project>
```

Una vez importada la dependencia usarla es algo tan sencillo como crear un programa principal e invocar al servicio a través de su implementación.

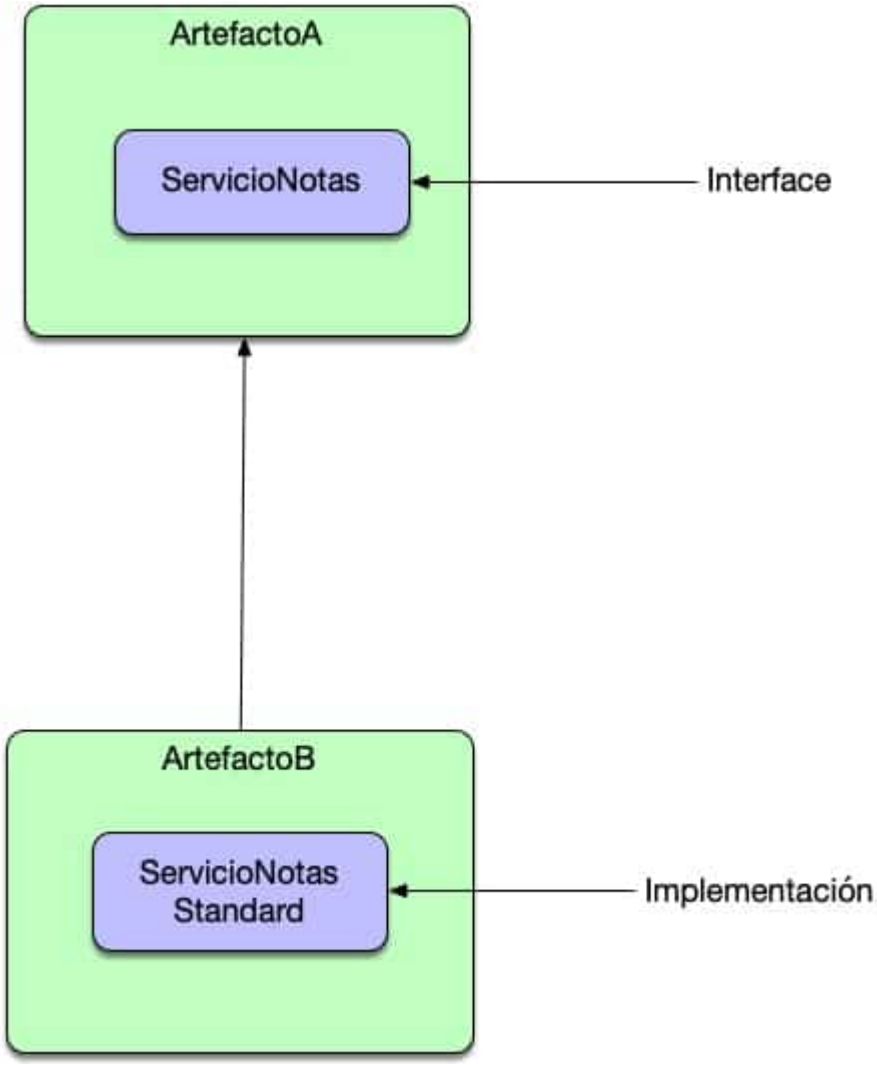
```
1  package com.arquitecturajava.cliente;
2
3  import com.arquitecturajava.servicio.ServicioNotas;
4  import com.arquitecturajava.servicio.standard.ServicioNotasStandard;
5
6  public class Principal {
7
8      public static void main(String[] args) {
9
10
11         ServicioNotas miservicio= new ServicioNotasStandard();
12         miservicio.obtenerNotas();
13
14     }
15
16 }
17
```

El resultado se imprime en la consola.

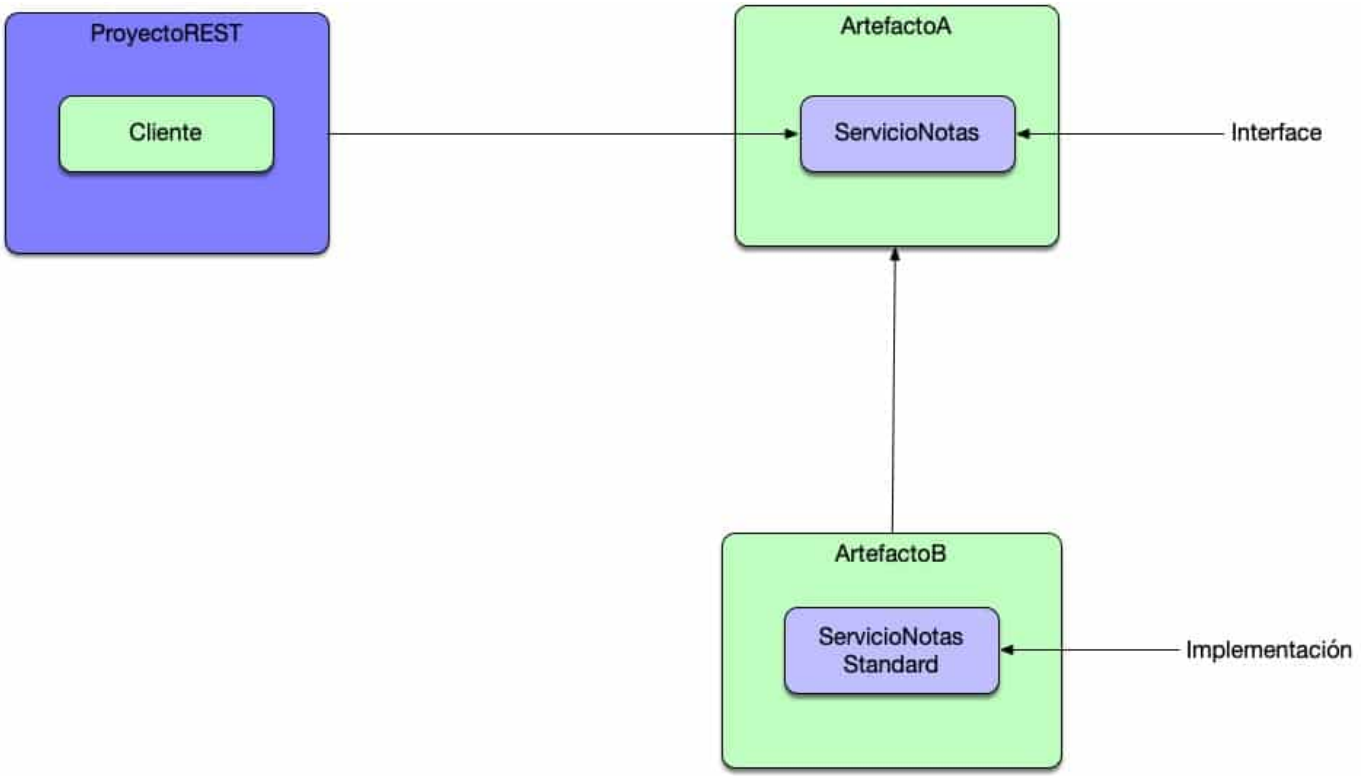


Java Service y flexibilidad

¿Es esta la solución más correcta? . La realidad es que es una solución razonable , pero quizás no sea la más correcta a día de hoy . Si quisieramos tener a nivel de Maven flexibilidad hubiera sido mucho mas razonable dividir los interfaces y la implementación en dos artefactos diferentes de tal forma que cada proyecto que importe los interfaces decida si necesita utilizar una implementación y en el caso de necesitarla poder elegir libremente que dependencia usar.



De esta manera podemos tener un proyecto que necesite por ejemplo acceder vía REST al servicio de Notas . Con el nuevo diseño de Artefactos nos valdrá con referenciar únicamente al artefacto que incluye el interface no necesitaremos para nada usar la implementación Standard ya que nuestra aplicación cliente tendrá su API para conectarse al servicio REST que nada tiene que ver con la implementación Standard que nosotros tenemos y que probablemente se ubique en el servidor.



Maven y división de responsabilidades

Acostumbremonos a poco a poco dividir de forma más clara a nivel de Maven los interfaces de sus implementaciónes.

Otros Artículos Relacionados

- Maven Artifact
- Maven Parent POM y uso de librerias
- Utilizando Maven Profiles
- Apache Maven