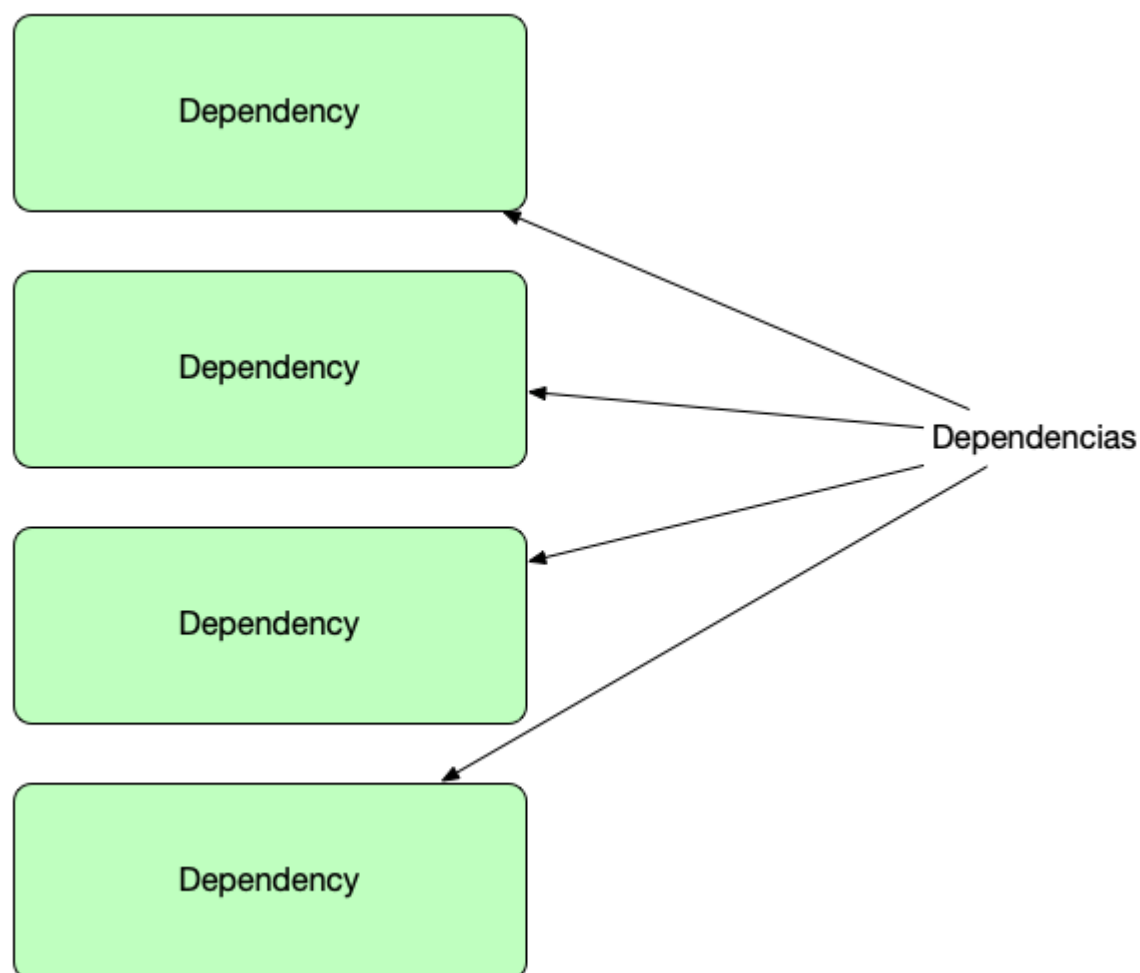


¿Qué es un Maven Property? .Cuando usamos Maven en muchas ocasiones disponemos de un montón de dependencias en el proyecto.



Sin embargo muchas de estas dependencias suelen compartir versión como se puede ver en el siguiente proyecto.

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>5.3.14</version>
</dependency>
<!--
```

<https://mvnrepository.com/artifact/org.springframework/spring-webmvc> -

```

->

<!--
https://mvnrepository.com/artifact/org.springframework/spring-context
-->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>5.3.14</version>
</dependency>
<!--
https://mvnrepository.com/artifact/org.thymeleaf/thymeleaf-spring5 -->
<dependency>
  <groupId>org.thymeleaf</groupId>
  <artifactId>thymeleaf-spring5</artifactId>
  <version>3.0.14.RELEASE</version>
</dependency>

<!--
https://mvnrepository.com/artifact/org.springframework/spring-webmvc -
->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>5.3.14</version>
</dependency>

```

Aquí nos podemos dar cuenta que todas comparten la versión 5.3.14 . No se trata de un gran problema , pero si que es cierto que en futuras evoluciones del proyecto nos podemos encontrar con la necesidad de cambiar de versión de Spring Framework y nos veremos obligados a cambiarlo en cada referencia. Estas referencias pueden pertenecer a un

proyecto o a varios . Por lo tanto es importante poder ubicar la versión de Spring Framework en un solo lugar para simplificar el trabajo de nuestro día a día. Para ello usaremos Maven Properties

Maven Properties y POMs

Vamos a ver un ejemplo muy sencillo del uso de Maven Properties refactorizando la versión de Spring Framework en el proyecto actual.

```
<properties>
<spring.version>5.3.14</spring.version>
</properties>

<dependencies>
  <!--
https://mvnrepository.com/artifact/org.springframework/spring-web -->
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>${spring.version}</version>
  </dependency>
  <!--
https://mvnrepository.com/artifact/org.springframework/spring-webmvc -
->

  <!--
https://mvnrepository.com/artifact/org.springframework/spring-context
-->
  <dependency>
    <groupId>org.springframework</groupId>
```

```

        <artifactId>spring-context</artifactId>
        <version>${spring.version}</version>
    </dependency>
    <!--
https://mvnrepository.com/artifact/org.thymeleaf/thymeleaf-spring5 -->
    <dependency>
        <groupId>org.thymeleaf</groupId>
        <artifactId>thymeleaf-spring5</artifactId>
        <version>3.0.14.RELEASE</version>
    </dependency>

    <!--
https://mvnrepository.com/artifact/org.springframework/spring-webmvc -
->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>${spring.version}</version>
    </dependency>

```

Como podemos ver en la parte superior del proyecto aparece una nueva etiqueta que pone `<properties>` en la cual nosotros podemos definir una propiedad con el nombre que deseemos ya que esta totalmente abierto no hace falta seguir ningún tipo de convención concreta. En este caso yo he decidido denominarla `<spring.version>`. Al hacerlo así cada una de las dependencias pasa a tener una referencia a la propiedad que acabamos de definir.

```

<groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>${spring.version}</version>
</dependency>

```

Maven Property y JDK

Muchas de las Maven properties vienen predefinidas y se pueden usar apoyándonos en documentación adicional. Uno de los ejemplos más clásicos es el cambio del compilador del JDK a nivel de Maven.

```
<properties>
  <spring.version>5.3.14</spring.version>
  <maven.compiler.target>1.8</maven.compiler.target>
  <maven.compiler.source>1.8</maven.compiler.source>
</properties>
```

De esta manera nuestro proyecto se ejecutará con Maven y un JDK 1.8. Siempre que podamos hagamos un uso fuerte del concepto de Maven Property para reforzar el concepto de Dont Repeat Yourself

Otros artículos relacionados:

1. [Maven Dependency Tree y librerías](#)
2. [¿Que es un Maven Module?](#)
3. [¿Que es un Maven Goal ?](#)
4. [Maven Flexibilidad y Artefactos](#)
5. [Maven Site](#)