Maven依赖的范围

1.什么是依赖范围?

maven 项目不同的阶段引入到classpath中的依赖是不同的，例如，编译时，maven 会将与编译相关的依赖引入classpath中，测试时，maven会将测试相关的的依赖引入到classpath中，运行时，maven会将与运行相关的依赖引入classpath中，而依赖范围就是用来控制依赖于这三种classpath的关系。

2.依赖范围在pom.xml中如何体现？

pom文件如下配置：

```
<dependency>        <groupId>junit</groupId>       <artifactId>junit</artifactId>
<version>4.7</version>        <scope>test</scope>   </dependency>
```

其scope标签就是依赖范围的配置，默认是compile,可选配置有test、provided、runtime、system、import

2.有哪些依赖范围？

既<scope>标签的可选配置：compile、test、provided、runtime、system、import，下面一一介绍

1. 编译依赖范围（compile），该范围就是默认依赖范围，此依赖范围对 于编译、测试、运行三种classpath都有效，举个简单的例子，假如项目中有spring-core的依赖，那么spring-core不管是在编译，测试，还是运行都会被用到，因此spring-core必须是编译范围（构件默认的是编译范围，所以依赖范围是编译范围的无须显示指定）

```
<dependency>        <groupId>org.springframework</groupId>
<artifactId>spring-core</artifactId>          <version>2.5</version>
<scope>compile</scope> <!--默认为该依赖范围，无须显示指定-->       </dependency>
```

2)测试依赖范围(test)，顾名思义就是针对于测试的，使用此依赖范围的依赖，只对测试classpath有效，在编译主代码和项目运行时，都将无法使用该依赖，最典型的例子就是 Junit,构件在测试时才需要，所以它的依赖范围是测试，因此它的依赖范围需要显示指定为<scope>test</scope> ,当然不显示指定依赖范围也不会报错，但是该依赖会被加入到编译和运行的classpath中,造成不必要的浪费。

```
<dependency>        <groupId>junit</groupId>       <artifactId>junit</artifactId>
<version>4.7</version>        <scope>test</scope>   </dependency>
```

3)已提供依赖范围(provided),使用该依赖范围的maven依赖，只对编译和测试的classpath有效，对运行的classpath无效，典型的例子就是servlet-api， 编译和测试该项目的时候需要该依赖，

但是在运行时，web容器已经提供了该依赖，所以运行时就不再需要此依赖，如果不显示指定该依赖范围，并且容器依赖的版本和maven依赖的版本不一致的话，可能会引起版本冲突，造成不良影响。

```
<dependency>        <groupId>javax-servlet</groupId>        <artifactId>servlet-
api</artifactId>        <version>2.0</version>       <scope>provided</scope>
</dependency>
```

4)运行时依赖范围(runtime),使用该依赖范围的maven依赖，只对测试和运行的classpath有效，对编译的classpath无效，典型例子就是JDBC的驱动实现，项目主代码编译的时候只需要JDK提供的JDBC接口，只有在测试和运行的时候才需要实现上述接口的具体JDBC驱动。

5),系统依赖范围（system）,该依赖与classpath的关系与 provided依赖范围完全一致，但是系统依赖范围必须通过配置systemPath元素来显示指定依赖文件的路径，此类依赖不是由maven仓库解析的，而且往往与本机系统绑定，可能造成构件的不可移植，因此谨慎使用，systemPath元素可以引用环境变量：

```
<dependency>        <groupId>javax.sql</groupId>        <artifactId>jdbc-
stext</artifactId>        <version>2.0</version>        <scope>system</scope>
<systemPath>${java.home}/lib/rt.jar</systemPath>    </dependency>
```

6)导入依赖范围(import),该依赖范围不会对三种classpath产生影响，该依赖范围只能与dependencyManagement元素配合使用，其功能为将目标pom文件中dependencyManagement的配置导入合并到当前pom的dependencyManagement中。有关dependencyManagement的功能请了解maven继承特性。