**Dependency Scope**

Dependency scope is used to limit the transitivity of a dependency, and also to affect the classpath used for various build tasks.

There are 6 scopes available:

* **compile**  
  This is the default scope, used if none is specified. Compile dependencies are available in all classpaths of a project. Furthermore, those dependencies are propagated to dependent projects.
* **provided**  
  This is much like compile, but indicates you expect the JDK or a container to provide the dependency at runtime. For example, when building a web application for the Java Enterprise Edition, you would set the dependency on the Servlet API and related Java EE APIs to scope provided because the web container provides those classes. This scope is only available on the compilation and test classpath, and is not transitive.
* **runtime**  
  This scope indicates that the dependency is not required for compilation, but is for execution. It is in the runtime and test classpaths, but not the compile classpath.
* **test**  
  This scope indicates that the dependency is not required for normal use of the application, and is only available for the test compilation and execution phases. This scope is not transitive.
* **system**  
  This scope is similar to provided except that you have to provide the JAR which contains it explicitly. The artifact is always available and is not looked up in a repository.
* **import** *(only available in Maven 2.0.9 or later)*  
  This scope is only supported on a dependency of type pom in the <dependencyManagement> section. It indicates the dependency to be replaced with the effective list of dependencies in the specified POM's <dependencyManagement> section. Since they are replaced, dependencies with a scope of import do not actually participate in limiting the transitivity of a dependency.

Each of the scopes (except for import) affects transitive dependencies in different ways, as is demonstrated in the table below. If a dependency is set to the scope in the left column, transitive dependencies of that dependency with the scope across the top row will result in a dependency in the main project with the scope listed at the intersection. If no scope is listed, it means the dependency will be omitted.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | compile | provided | runtime | test |
| compile | compile(\*) | - | runtime | - |
| provided | provided | - | provided | - |
| runtime | runtime | - | runtime | - |
| test | test | - | test | - |

**(\*) Note:** it is intended that this should be runtime scope instead, so that all compile dependencies must be explicitly listed - however, there is the case where the library you depend on extends a class from another library, forcing you to have available at compile time. For this reason, compile time dependencies remain as compile scope even when they are transitive.

<https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html>