Nested JARs

Java does not provide any standard way to load nested jar files (that is, jar files that are themselves contained within a jar). This can be problematic if you need to distribute a self-contained application that can be run from the command line without unpacking.

To solve this problem, many developers use “shaded” jars. A shaded jar packages all classes, from all jars, into a single “uber jar”. The problem with

Spring Boot
3.5.0
Search CTRL + k
Overview
Documentation
Community
System Requirements
Installing Spring Boot
Upgrading Spring Boot
Tutorials
Reference
How-to Guides
Build Tool Plugins
Spring Boot CLI
Rest APIs
Java APIs
Kotlin APIs
Specifications
Configuration Metadata
The Executable Jar Format
Nested JARs
Spring Boot’s “NestedJarFile” Class
Launching Executable Jars
PropertiesLauncher Features
Executable Jar Restrictions
Alternative Single Jar Solutions
Appendix

Edit this Page
GitHub Project
Stack Overflow

The Executable Jar File Structure

Spring Boot Loader-compatible jar files should be structured in the following way:

```
example.jar
 |
 +-META-INF
 |  +-MANIFEST.MF
 +-org
 |  +-springframework
 |     +-boot
 |        +-loader
 |           +-<spring boot loader classes>
 +-BOOT-INF
    +-classes
    |  +-mycompany
    |     +-project
    |        +-YourClasses.class
    +-lib
       +-dependency1.jar
       +-dependency2.jar
```

Application classes should be placed in a nested `BOOT-INF/classes` directory. Dependencies should be placed in a nested `BOOT-INF/lib` directory.

The Executable War File Structure

Spring Boot Loader-compatible war files should be structured in the following way:

```
example.war
 |
 +-META-INF
 |  +-MANIFEST.MF
 +-org
 |  +-springframework
 |     +-boot
 |        +-loader
 |           +-<spring boot loader classes>
 +-WEB-INF
    +-classes
    |  +-com
    |     +-mycompany
    |        +-project
    |           +-YourClasses.class
    +-lib
    |  +-dependency1.jar
    |  +-dependency2.jar
    +-lib-provided
       +-servlet-api.jar
       +-dependency3.jar
```

Dependencies should be placed in a nested `WEB-INF/lib` directory. Any dependencies that are required when running embedded but are not required when deploying to a traditional web container should be placed in `WEB-INF/lib-provided`.

Index Files

Spring Boot Loader-compatible jar and war archives can include additional index files under the `BOOT-INF/` directory. A `classpath.idx` file can be provided for both jars and wars, and it provides the ordering that jars should be added to the classpath. The `layers.idx` file can be used only for jars, and it allows a jar to be split into logical layers for Docker/OCI image creation.

Index files follow a YAML compatible syntax so that they can be easily parsed by third-party tools. These files, however, are *not* parsed internally as YAML and they must be written in exactly the formats described below in order to be used.

Classpath Index

The classpath index file can be provided in `BOOT-INF/classpath.idx`. Typically, it is generated automatically by Spring Boot’s Maven and Gradle build plugins. It provides a list of jar names (including the directory) in the order that they should be added to the classpath. When generated by the build plugins, this classpath ordering matches that used by the build system for running and testing the application. Each line must start with dash space (`"-·"`) and names must be in double quotes.

For example, given the following jar:

```
example.jar
 |
 +-META-INF
 |  +-...
 +-BOOT-INF
    +-classes
    |  +...
    +-lib
       +-dependency1.jar
       +-dependency2.jar
```

The index file would look like this:

```
- "BOOT-INF/lib/dependency2.jar"
- "BOOT-INF/lib/dependency1.jar"
```

NOTE

Spring Boot only uses the classpath index file when the jar or war file is executed with `java -jar`. It is not used when running the application from the IDE or when using Maven's `spring-boot:run` or Gradle's `bootRun`.

NOTE

When enabling reproducible builds, the entries in the classpath index file are sorted alphabetically.

Layer Index

The layers index file can be provided in `BOOT-INF/layers.idx`. It provides a list of layers and the parts of the jar that should be contained within them. Layers are written in the order that they should be added to the Docker/OCI image. Layers names are written as quoted strings prefixed with dash space (`"-·"`) and with a colon (`":"`) suffix. Layer content is either a file or directory name written as a quoted string prefixed by space space dash space (`"··-·"`). A directory name ends with `/`, a file name does not. When a directory name is used it means that all files inside that directory are in the same layer.

A typical example of a layers index would be:

```
- "dependencies":
  - "BOOT-INF/lib/dependency1.jar"
  - "BOOT-INF/lib/dependency2.jar"
- "application":
  - "BOOT-INF/classes/"
  - "META-INF/"
```

Prev
The Executable Jar Format

Next
Spring Boot's "NestedJarFile" Class