## JAR Index

### **Overview**

Since 1.3, JarIndex is introduced to optimize the class searching process of class loaders for network applications, especially applets. Originally, an applet class loader uses a simple linear search algorithm to search each element on its internal search path, which is constructed from the "ARCHIVE" tag or the "Class-Path" main attribute. The class loader downloads and opens each element in its search path, until the class or resource is found. If the class loader tries to find a nonexistent resource, then all the jar files within the application or applet will have to be downloaded. For large network applications and applets this could result in slow startup, sluggish response and wasted network bandwidth. The JarIndex mechanism collects the contents of all the jar files defined in an applet and stores the information in an index file in the first jar file on the applet's class path. After the first jar file is downloaded, the applet class loader will use the collected content information for efficient downloading of jar files.

The existing jar tool is enhanced to be able to examine a list of jar files and generate directory information as to which classes and resources reside in which jar file. This directory information is stored in a simple text file named INDEX.LIST in the META-INF directory of the root jar file. When the classloader loads the root jar file,  it reads the INDEX.LIST file and uses it to construct a hash table of mappings from file and package names to lists of jar file names. In order to find a class or a resource, the class loader queries the hashtable to find the proper jar file and then downloads it if necessary.

Once the class loader finds a INDEX.LIST file in a particular jar file, it always trusts the information listed in it. If a mapping is found for a particular class, but the class loader fails to find it by following the link, an InvalidJarIndexException is thrown. When this occurs, the application developer should rerun the jar tool on the extension to get the right information into the index file.

To prevent adding too much space overhead to the application and to speed up the construction of the in-memory hash table, the INDEX.LIST file is kept as small as possible. For classes with non-null package names, mappings are recorded at the package level. Normally one package name is mapped to one jar file, but if a particular package spans more than one jar file, then the mapped value of this package will be a list of jar files. For resource files with non-empty directory prefixes, mappings are also recorded at the directory level.  Only for classes with null package name, and resource files which reside in the root directory, will the mapping be recorded at the individual file level.

<https://docs.oracle.com/javase/8/docs/technotes/guides/jar/jar.html>