

Porting Guide

Porting Phoenix 4.0.x applications

Loom supports applications running in Phoenix 4.0.x releases, provided that the appropriate libraries are bundled with the in the SAR file.

 See the list at the end of this page for a comparison about included libraries.

For users that have an Apache Ant based build, the Phoenix Sar task can still be used. Otherwise, it is possible to create a Loom Sar using Ant's jar task.

Creating a Loom Sar with Maven

Since the generation of metaclass descriptors is more generic in Loom compared to Phoenix, but still fully compatible with it, users can also deploy applications running in Phoenix 4.0.x by simply regenerating the descriptors from the Java source using the [Maven Loom Plugin](#).

To create a Loom SAR, using the Maven Loom Plugin simply add the dependency to your Maven project

```
<dependency>
  <groupId>loom</groupId>

  <artifactId>maven-loom-plugin</artifactId>
  <version>[plugin-version]</version>
  <type>plugin</type>
</dependency>
```

and execute

```
maven loom:sar
```

Creating a Loom Sar with Ant

Loom's new metaclass descriptors can still be created with Ant. To do so, use the following snippet in your Ant build file (Jar file versions may need to be adjusted to the latest releases) :

```
<taskdef
  name="metaclassGen"

  classname="org.codehaus.metaclass.tools.task
```

```
s.GenerateClassDescriptorsTask">
  <classpath>
    <pathelement
location="${path.to}/metaclass-runtime-1.1.jar"/>
    <pathelement
location="${path.to}/metaclass-tools-1.1.jar"/>
    <pathelement
location="${path.to}/qdox-1.3.jar"/>
    <pathelement
location="${path.to}/asm-1.4.jar"/>
  </classpath>
</taskdef>
```

```
<metaclassGen destDir="${dest.dir}"
namespaceTagsOnly="false">
  <fileset dir="${src.dir}"/>
  <interceptor
name="org.codehaus.loom.info.PhoenixAttributeInterceptor">
    <classpath>
      <pathelement
location="${path.to}/maven-loom-plugin-[version].jar"/>
```

```
</classpath>
</interceptor>
</metaclassGen>
```

For the time being, it is recommended to continue using the Ant-based Sar task to create a Sar. (instructions on how to create one without it are welcome!)

Comparison of libraries bundled

Compared to the Phoenix 4.0.x distribution, in the Loom 1.0 distribution some of the libraries in the lib/ directory have been removed as they are not used by the container. If your app uses any of the excluded libraries, you simply need to bundle them with the SAR, or put them in the lib/ directory.

The general migration tip is to make sure that what is in your old lib/ directory is either in the new lib/ directory, or bundled with your sar.

There were migration issues from Phoenix 4.0.3 to 4.0.4, but that should not be an issue upgrading to Loom, as the problem libraries are no longer in a shared classpath between Loom's core and the applications.

Loom 1.0 comes with a lean set of application libraries:

- asm
- avalon-framework
- avalon-phoenix-client
- dna-api
- excalibur-instrument
- logkit
- log4j
- metaclass-runtime
- mx4j-jmx
- mx4j-tools

Phoenix 4.0.x ships a bigger set of libraries:

- avalon-framework
- avalon-phoenix-client
- excalibur-baxter
- excalibur-cli
- excalibur-collections
- excalibur-concurrent
- excalibur-configuration
- excalibur-containerkit
- excalibur-extension
- excalibur-i18n
- excalibur-instrument
- excalibur-io
- excalibur-logger
- excalibur-pool
- excalibur-thread
- excalibur-threadcontext
- excalibur-util

- isorelax.jar
- jing.jar
- logkit
- xalan
- xercesImpl
- xml-apis