Java Services Reference Guide

You can run Java services on the **Scheer PAS** *BRIDGE*® as described in Managing Java Services. Find in this chapter some hints concerning the development of Java services.

On this Page:

- Developing Java Services
  - Structure of a Java Service Repository
    - Adding Information to META-INF /MANIFEST.MF
    - Building the JAR File
  - Writing to Bridge Log Files
  - Implementing a Service Shutdown Activity

Related Pages:

- Oracle Documentation of addShutdownHook

Related Documentation:

- Managing Java Services
- Deployment of Java Services

Developing Java Services

Structure of a Java Service Repository

A Java service repository has to be a self-contained JAR file containing also a folder **META-INF /MANIFEST.MF** holding the meta information of the Java Service.

You need to add the following name/values to META-INF/MANIFEST.MF.

Name	Mandatory	Description	Example
E2E-Service-Name	✔	Name of the service. Must be unique for each Bridge you deploy to.	E2E-Service-Name=helloworld
E2E-Service-Version		Version of the service. Can be any string. It is not parsed by the Bridge.	E2E-Service-Version=1.2.3
E2E-Service-Description		Short description of the service.	E2E-Service-Description=Hello World Service

Adding Information to META-INF/MANIFEST.MF

You cannot directly create or edit the **MANIFEST.MF**. There are two ways to add information to this file:

- Use the command line JAR tool as described in the Java tutorial Modifying a Manifest File.
- Do it within Java code as described in Add Manifest into JAR File Using Java.

Building the JAR File

The Java service repository has to be a self-contained JAR file. With the open-source tool One-Jar you can create such a file.

An easy way to start developing a new Java service with One-Jar is the **Application Generator** approach. This approach provides you with a complete Eclipse/Ant application directory, that you can use as a starting point for your own One-JAR application. The application generator is driven by a template built into the one-jar-appgen.jar file (see one-jar-appgen).

1. Download one-jar-appgen-0.97.jar.
2. Generate the application, build, and run it.

```
$ java -jar one-jar-appgen-0.97.jar
Enter project path (project name is last segment): c:/tmp/test-one-jar
Enter java package name: com.example.onejar
$ cd c:/tmp/test-one-jar
$ ant
$ cd build
$ java -jar test-one-jar.jar
test_one_jar main entry point, args=[]
test_one_jar main is running
test_one_jar OK.
```

Add source code to the **src** directory, library jars to the **lib** directory, and rebuild.

If you are using IntelliJ IDEA, you can import the **project template** BridgeJavaServiceTemplate.jar which uses the Gradle plugin gradle-one-jar.
Use

- Gradle task **jar** for building repositories without dependencies
- task **selfContainedJar** for building repositories which are depending on the Java libraries you added to the service.

Writing to Bridge Log Files

If you are writing messages to log files that will be displayed on the Bridge (start, stdout, stderr, custom logfiles), you must use **UTF-8 encoding**. If you do not, special characters may be displayed wrongly in the log view.

Implementing a Service Shutdown Activity

Bridge 7.2.0 Upon stopping a Java service, the Bridge will send an operating system signal (`SIGINT`) to the service to stop it. If you want to do some clean-up actions before stopping, you have to implement a signal handler for `SIGINT` in your Java service.
For more information, refer to the Oracle Documentation of addShutdownHook.