Exercise 7: OSGi

TECHNISCHE UNIVERSITÄT DARMSTADT

Concepts and Technologies for Distributed Systems and Big Data Processing – SS 2016

Task 1 Implementing an OSGi service

Implement an OSGi service that computes the *n*-th Fibonacci number.

1. Create a *service* bundle which contains the service interface. The interface definition is given by the following:

   ```
   public interface FibonacciService {
     public int fib(int n);
   }
   ```

   The `fib` method computes the Fibonacci number for the given `n`. Both the service implementation and the bundles using the service import this *service* bundle.

2. Create a *service.host* bundle which contains the implementation for the service. Thus, this bundle needs to import the *service* bundle. The service is registered using `BundleContext::registerService`. When the service is registered, other bundles can lookup and use the service.

3. Create a *consumer* bundle which makes use of the service. Thus, this bundle needs to import the *service* bundle. However, note that this bundle does not need to import the *service.host* bundle, thus decoupling the service implementation from the consumer. As soon as the `FibonacciService` becomes available, the consumer should invoke the service to compute the first ten Fibonacci numbers and print them to standard output. In order to get informed when a service becomes available, a `ServiceTracker` can be used. Note that, besides the `org.osgi.framework` dependency, you also need to specify the `org.osgi.util.tracker` dependency in the *MANIFEST.MF* file, to make the `ServiceTracker` class available.

The Eclipse IDE has an embedded Equinox OSGi container which can be used to develop OSGi bundles. To create a new bundle, open the *New Project* dialog via *File* → *New* → *Project*. In the dialog, choose *Plug-in Project* and click *Next*. In the *Plug-in Project* dialog, enter the *Project name* and select *OSGi framework*: *standard* as *Target Platform*. You can leave the default values on the next pages and click *Finish* at the end to generate the project.

When you have implemented all three bundles, you can right-click on the *consumer* bundle in the Eclipse *Package Explorer* and select *Run As* → *OSGi Framework* to load and start the bundles.

You can also export each bundle as JAR files using *File* → *Export* and select *JAR file*. Follow the steps of the *JAR Export* dialog and make sure that you check *Use existing manifest from workspace* and select the correct *MANIFEST.MF* file. The JAR files can then be loaded as bundle into other OSGi containers. For instance, you can download the *Knopflerfish* OSGi container implementation from `http://www.knopflerfish.org/`. It comes with a graphical user interface to load, unload, start and stop bundles.