Performing Automatic Application Transformation using the RAFDA Tools

Installation

The RAFDA tools are distributed in the JAR file *rafda.jar* which must be present in the classpath along with the following dependencies:

- The *Java 2 Platform, Standard Edition*. We use v1.4.2 available at (http://java.sun.com/j2se/1.4.2/download.html).
- The *Byte Code Engineering Library (BCEL)*. We use v5.1 available at (http://jakarta.apache.org/site/binindex.cgi).
- The *Java Uuid Generator (JUG)*. We use v1.1.1 available at (http://www.doomdark.org/doomdark/proj/jug/curr/jug.jar).
- The *University of St Andrews Dynamic Java Compiler*. We use v1.4 available at (http://www-ppg.dcs.st-and.ac.uk/Java/DynamicCompilation/javacompiler.jar).
    - The Dynamic Compiler requires *tools.jar*, which is distributed with the J2SE. It can usually be found in the JAVA_HOME/lib folder but is not in the classpath by default.

Compile

Unpack the source distribution to the directory of your choice, which will be referred to as `<RAFDA>` in the rest of this section.

Copy the requisite JAR files listed above (BCEL, JUG, compiler.jar) to the `<RAFDA>/lib` directory. In addition you must copy the `tools.jar` from your Java distribution (found at `<JAVA_HOME>/lib/tools.jar`) to the same `<RAFDA>/lib` directory.

There are two ways of building the system. Included in the source distribution are an Ant build file and an Eclipse project. Following are the instructions for each alternative.

Using Ant

A build file for Ant is included in the source distribution. The command:

```
$ ant -f <RAFDA>/build.xml jar
```

will build the `rafda.jar` file in the `<RAFDA>/BUILD/lib` directory. Use Ant's project-help option:

```
$ ant -f <RAFDA>/build.xml -projecthelp
```

to know more about the various build targets available.

Using Eclipse

The `<RAFDA>` directory contains the Eclipse `.project` file. Import this project into your workspace ("Import..." under the "File" menu). The project should automatically build. Create the JAR file using the supplied `rafda.jardesc`.

Install

Run scripts included in this distribution are: `rafdap`, `rafdat-rio`, `rafda-rt` and `rafda-run`.

Edit the configuration section of the run scripts supplied with the distribution (found in the `misc` directory in the source distribution) to indicate where the RAFDA JAR file and requisite JAR files are located.

If you are using Eclipse then run configurations for the RAFDA run time can be found in the project. However, in order to use the transformation tools you must the run scripts.

Transform

The `rafdat-rio` script runs the transformation tool. You must supply **all** the classes used, directly or indirectly, by the application to be transformed. This **always include**s:

- `charsets.jar`
- `jce.jar`
- `jsse.jar`
- `rt.jar`

all of which can be found at `<JAVA_HOME>/jre/lib`. Use the help option:

```
$ rafdat-rio -help
```
to know more about the various configuration options available.

Running a Transformed Application

Before you run your transformed application you must define a distribution policy and start one or more instances of the RRT over which the application will be distributed.

Distribution Policy

A distribution policy must be specified. Policies are instances of classes that implement a special policy interface. Two implementations of policy are supplied:

```
uk.ac.stand.dcs.rafda.rrt.policy.distribution.RootRRTPolicy
```
which places all instantiated components on the root RRT in the system; and:

```
uk.ac.stand.dcs.rafda.rrt.policy.distribution.SpecificRRTPolicy
```

which places all instantiated components on an RRT running on a specifed host and port. The current RRT is used by default. Constructor takes a string representing RRT host interface and a port representing RRT host port, e.g.:

```
new SpecificRRTPolicy("ocumare", 5002)
```

Policy is set for each RRT individually. A call into the RRT specifies the policy. If a policy class is specified its default constructor is used to instantiate it.

Specify by policy class:

```
RafdaRunTime.setDistributionPolicyClass(Class c);
```
e.g.:

```
RafdaRunTime.setDistributionPolicyClass(RootRRTPolicy.class);
```

Specify by policy object:

```
RafdaRunTime.setDistributionPolicyObject(Policy p);
```
e.g.:

```
Policy srp = new SpecificRRTPolicy("panda", 5003);
RafdaRunTime.setDistributionPolicyObject(srp);
```

RRT Server

The `rafdat-rt` script starts an instance of the RRT. The first RRT you start must always be a special “root” RRT started by specifying the network interface, port number and the word ‘root’ as arguments to the `rafda-rt` script. Every RRT must have a distribution policy associated with it that indicates how newly instantiated objects are distributed. Several policies are supplied with the RRT.

To create a root RRT running on the network interface associated with the hostname panda, listening on port 5001 and using the distribution policy defined in class `uk.ac.stand.dcs.rafda.rrt.policy.distribution.RootRRTPolicy` you should write:

```
rafda-rt panda 5001 root
uk.ac.stand.dcs.rafda.rrt.policy.distribution.RootRRTPolicy
```

All other RRTs are started by specifying the network interface and port number on which this RRT is to listen and the network interface and port number of the single root RRT in the system. For example, to create a second RRT running on the network interface associated with the hostname ocumare, listening on port 5002 and using the same distribution policy as in the example above you should write:

```
rafda-rt ocumare 5002 panda 5001
uk.ac.stand.dcs.rafda.rrt.policy.distribution.RootRRTPolicy
```

Transformed Application

The `rafda-run` script runs the transformed application. The transformed application runs inside another instance of the RRT. In order to run the application you must specify the network interface and port number of the application's RRT and the network interface and port number of the single root RRT in the system. Finally, the name of the class containing the application's main method and optionally some arguments to the application are specified.

For example, if you used to run your application as follows:

```
application.Main arg0 arg1...
```

and you want to run the transformed application using an RRT running on the network interface associated with the hostname snarf, listening on port 5003 and using the distribution policy defined in class `uk.ac.stand.dcs.rafda.rrt.policy.distribution.RootRRTPolicy` you should write:

```
rafda-run snarf 5003 panda 5001
uk.ac.stand.dcs.rafda.rrt.policy.distribution.RootRRTPolicy
application.Main arg0 arg1...
```