Developing Eclipse Plug-ins that make use of CAL Workspaces

Edward Lam

Last modified: October 25, 2007

Overview:

Eclipse's plugin system imposes unique conditions on the functioning of the Quark environment. This document provides instructions on how to setup the Quark Platform to use CAL workspaces within the context of an Eclipse plugin, so that clients using CAL may continue to operate.

Note that this document applies only to deployment of Quark using CAL workspaces (eg. using Cars or Car-jars). If the client does not require runtime CAL compilation or access to meta-programming capabilities (such as the ability to query for gems with a given type or add new modules to the workspace), then deployment of Quark via standalone JARs is recommended instead.

Steps:

1. Include the CAL_Eclipse_Bridge project

The CAL_Eclipse_Bridge project contains Eclipse-specific code which allows the CAL Workspace to find resources within an Eclipse environment.

Since is is a plugin project, it can be included in an Eclipse workspace in its current form. Alternatively, it can be packaged together with other code in one or more .jars, and included in a separate project.

2. Set the workspace provider system property

The system property

org.openquark.cal.services.CALWorkspaceEnvironmentProvider.Factory

must be set to the name of the Eclipse workspace environment provider:

org.openquark.cal.eclipse.bridge.EclipseWorkspaceEnvironmentProvider$Factory

For instance, the following can be provided as a command-line argument to the Java VM:
-Dorg.openquark.cal.services.CALWorkspaceEnvironmentProvider.Factory=
org.openquark.cal.eclipse.bridge.EclipseWorkspaceEnvironmentProvider$Factory

Note that this step may not be necessary if the CAL_Eclipse_Core project is active since it sets this system property during plugin activation.

3. Ensure CAL resource visibility

For any projects with CAL resources, ensure that these are visible to the CAL_Platform plugin.

If CAL_Platform is included as a plugin project in its current form:

- for any other projects with CAL resources, add the following to the manifest:
  `Eclipse-RegisterBuddy: org.openquark.cal.platform`

- the CAL Platform manifest specifies "buddy-loading", with the "registered" policy. This means that CAL Platform can "see" code for any project which has registered CAL Platform as a buddy.

If CAL_Platform is packaged together with other code in one or more .jars:

- Typically, the CAL resources will exist in file system folders in the same project as the CAL Platform code. In this case, no further measures are necessary.
- If CAL resources exist in projects different from that which contains the code for CAL Platform, it will be necessary to make those resources visible to the CAL Platform code. This can be done by means of buddy-loading.

Example:

CAL_Eclipse_Bridge_Example project

This is an example of a project which calls a function in CAL_Platform.

To see this example in action:

- Add this project and its dependents to the Eclipse workspace.
- Create a launch configuration under "Eclipse Application".
- Within the launch configuration, set the system property under "VM Arguments".
- Execute the launch configuration.

The Eclipse runtime workbench will be executed. The installed action can be invoked via the top-level menu labelled "BridgeTest". This runs the EventLoop demo (based on code from the CAL_Samples project), which:

- Compiles the CAL_Platform test workspace.
- Executes the EventLoop demo, using the compiled program.

The output from this demo will be output to the console.