

Javaddin prototype

Introduction

This document provides information about the Javaddin prototype and its overall architecture including how to install it and run the samples. It also provides information about the source code for each of its components.

Note: it is assumed that you have unzipped the javaddin_prototype.zip in a directory of your choice. This zip file contains the following subdirectories: sources, build.

Javaddin is an experimental generic domino addin task that lets users create Domino server tasks using Java OSGi plugins. There are many benefits in using a java framework like OSGi to write Domino Server tasks:

- Declarative task scheduling
- Built-in mechanism to access server task user arguments.
- List of tell commands to query the OSGi framework.
- Multi-platform support
- Extensibility
- Easy deployment
- Better tooling with Eclipse IDE and its first class Java and plugins editors and Java debugger.
- ...

OSGi itself is a java based framework for running unit of code called bundles. It is usually associated with Eclipse, so a lot people might be surprised that it is used as a server technology. However at its core, OSGi is only a thin runtime that governs how bundles are loaded, their dependencies, how their services are exposed, etc... You can find more information about OSGi at <http://www.osgi.org/Main/HomePage>.

Overall architecture

As explained in the diagram below, the javaddin generic server task launches a JVM and invokes code located in launcher.jar to launch the OSGi container.

The OSGi framework located in {notesbin}/javaddin/rcp/eclipse contains the core plugins to run a OSGi framework. The minimum set is comprised of the following list:

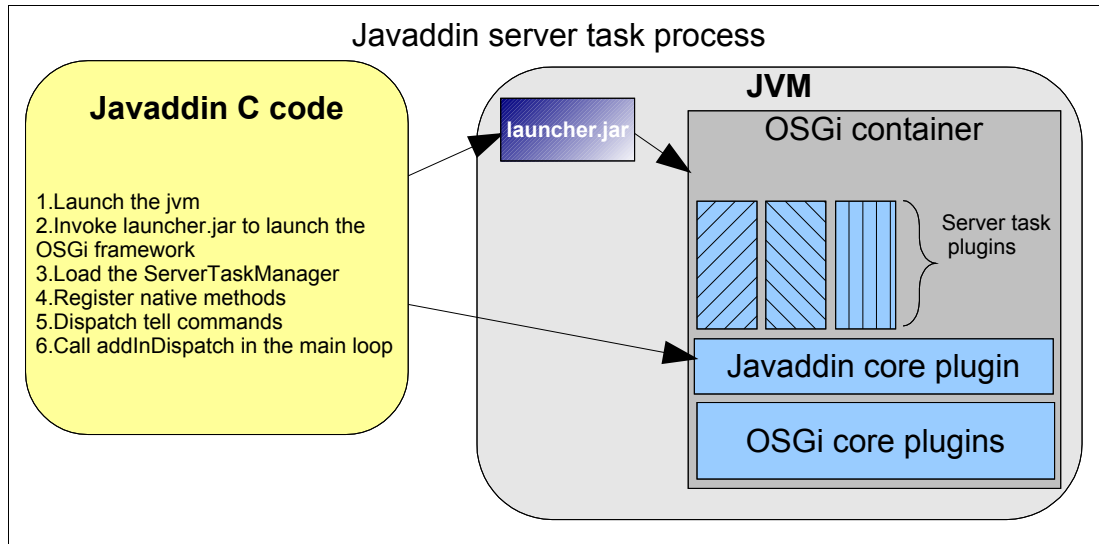
1. org.eclipse.update.configurator
2. org.eclipse.osgi
3. org.eclipse.equinox.preferences
4. org.eclipse.equinox.registry
5. org.eclipse.equinox.app
6. org.eclipse.equinox.common
7. org.eclipse.core.runtime.compatibility.auth
8. org.eclipse.core.jobs
9. org.eclipse.core.runtime
10. org.eclipse.core.contenttype

You should be able to simply copy these 10 plugins in the {notesbin}/javaddin/rcp/eclipse directory from any installed version of Eclipse SDK. If you do not have an eclipse SDK installed on your machine, you can download it from <http://www.eclipse.org/download> (there are many flavors of the eclipse IDE in this page, if not sure pick the "Eclipse IDE for Java Developers").

The "org.openntf.domino.javaddin" plugin located in {notesbin}/javaddin/shared/eclipse/ plugins, is part of this prototype. This plugin is essential to communicate between the javaddin server task and the server tasks plugins. Among other things, it provides:

- The extension point that allows a user to register the server task
- The ServerTaskManager instance
- Tell commands implementation

A sample plugin called "org.openntf.javaddin.samples" is included in this prototype. This sample contains a few server tasks to illustrate the different configurations available e.g. Scheduled, runonce, run on startup, etc...



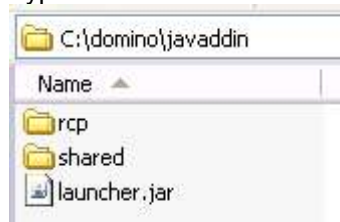
Source code:

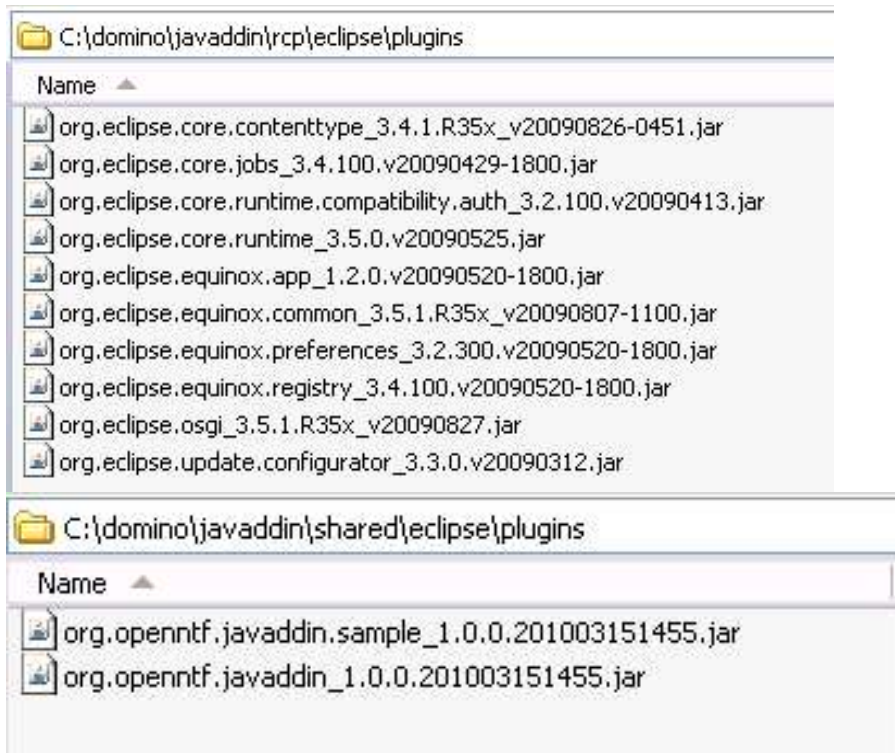
- The source code is included in the “**sources**” directory and contains the following files:
1. javaddin.cpp: contains the source code for the server task. To recompile it, you will need to install the Notes/Domino C Api Toolkit available here: <http://www.ibm.com/developerworks/lotus/downloads/toolkits.html>. Please refer to the Notes/Domino C Api toolkit documentation for instructions on how to compile javaddin.cpp. Since javaddin.cpp makes use of the jni API, you will also need to install a JDK. For example, you can download one at <http://java.sun.com/javase/downloads/index.jsp>.
 2. org.openntf.osgi.launcher: Eclipse project containing the source code for launcher.jar which is responsible for launching the OSGi framework. To compile it, it is recommended that you install an Eclipse IDE and load the project in your workspace. Eclipse IDEs can be install from <http://www.eclipse.org/download>.
 3. org.openntf.javaddin: Eclipse project containing the source code the core javaddin plugin
 4. org.openntf.javaddin.sample: Eclipse project containing the source code for sample server tasks.

Built code:

- The built code is included in the “**build**” directory. In contains the following files:
1. javaddin.exe: executable that must be installed in {dominobin} directory
 2. launcher.jar: jar file that must be installed in {dominobin}/javaddin directory
 3. org.openntf.javaddin_1.0.0.XXXX.jar: jar file that must be installed in {dominobin}/javaddin/shared/eclipse/plugins directory
 4. org.openntf.javaddin.sample_1.0.0.XXXX.jar: jar file that must be installed in {dominobin}/shared/eclipse/plugins directory

Typical install:





To launch javaddin, use the following command from the domino server console: "load javaddin"

```
> load javaddin
> Listening on port 1234 ...
03/15/2010 03:05:16 PM Javaddin started
03/15/2010 03:05:17 PM (org.openntf.sample.scheduled) Scheduled run: Runs every 1 - day. Last Modified Date for db OSGiSample.nsf: 3/15/10 3:04 PM
```

Javaddin tell commands:

help:	Display this help
list:	Display list of OSGi server tasks
run [taskid]:	Run the task once specified by the task id
info [taskid]:	Display detailed information about the task

Examples:

```
> tell javaddin list
>
   id:          org.openntf.sample.sendMail
  description: Sample task that send a summary email for all the databases available on
the server. Options: [emailaddress]
   class Name:  org.openntf.javaddin.sample.SendMailExample

   id:          org.openntf.sample.scheduled
  description:  Scheduled Task
   class Name:  org.openntf.javaddin.sample.ScheduledTask

   id:          org.openntf.sample.listViewes
  description:  Sample task that list the views of a database. Options: [dbPath]
   class Name:  org.openntf.javaddin.sample.ListViewsExample
>
```

```
> tell javaddin info org.openntf.sample.scheduled
>
   id:          org.openntf.sample.scheduled
  description:  Scheduled Task
   class Name:  org.openntf.javaddin.sample.ScheduledTask
  Run on Start: false
                Runs every 5 - minute
                Runs every 1 - day
>
```

Description of the org.openntf.javaddin.task extension point:

A detailed description of the “org.openntf.javaddin.task” extension point is available in the `pde_schema_org-openntf-javaddin-task_preview.html` file included in the distribution zip file.

Server Addin Task

Identifier: org.openntf.javaddin.task

Since: [Enter the first release in which this extension point appears.]

Description: [Enter description of this extension point.]

Configuration Markup:

```
<!ELEMENT extension (task+)>
<!ATTLIST extension
  point CDATA #REQUIRED
  id CDATA #IMPLIED
  name CDATA #IMPLIED
>
```

```
<!ELEMENT task (run)*>
<!ATTLIST task
  class CDATA #REQUIRED
  id IDREF #REQUIRED
  description CDATA #IMPLIED
  runOnStart (true | false)
>
```

- **class** - Fully qualified class name, must extend org.openntf.javaddin.task.AbstractServerTask.
- **id** - Unique id for this task
- **description** - Task description
- **runOnStart** - Indicates if the task should be run once when the javaddin task starts. True by default

```
<!ELEMENT run EMPTY>
<!ATTLIST run
  every CDATA #REQUIRED
  unit (second|minute|day)
>
```

- **every** - Schedule Frequency for the server task
- **unit** - Frequency unit for the Task Can be second, minute or day. If day, then "every" attribute is not used and the task will run once a day.

The “org.openntf.javaddin.task” extension point requires a class that must extends org.openntf.javaddin.task.AbstractServerTask. In this class, the user must override the following methods:

```
/**
 * @param serverTaskInfo
 * Called before the task is ready to run
 * Note: a default implementation is provided in
AbstractServerTask, subclasses must call super if overriding this method
 */
public void init( ServerTaskInfo serverTaskInfo );

/**
 * @param runWhen: Type of run
 * @param args: Optional arguments
 * @throws NotesException
 */
public void run(RunWhen runWhen, String[] args) throws
NotesException;

/**
 * Called when the task instance is being discarded
 * Task should recycle any created backend objects
 */
```

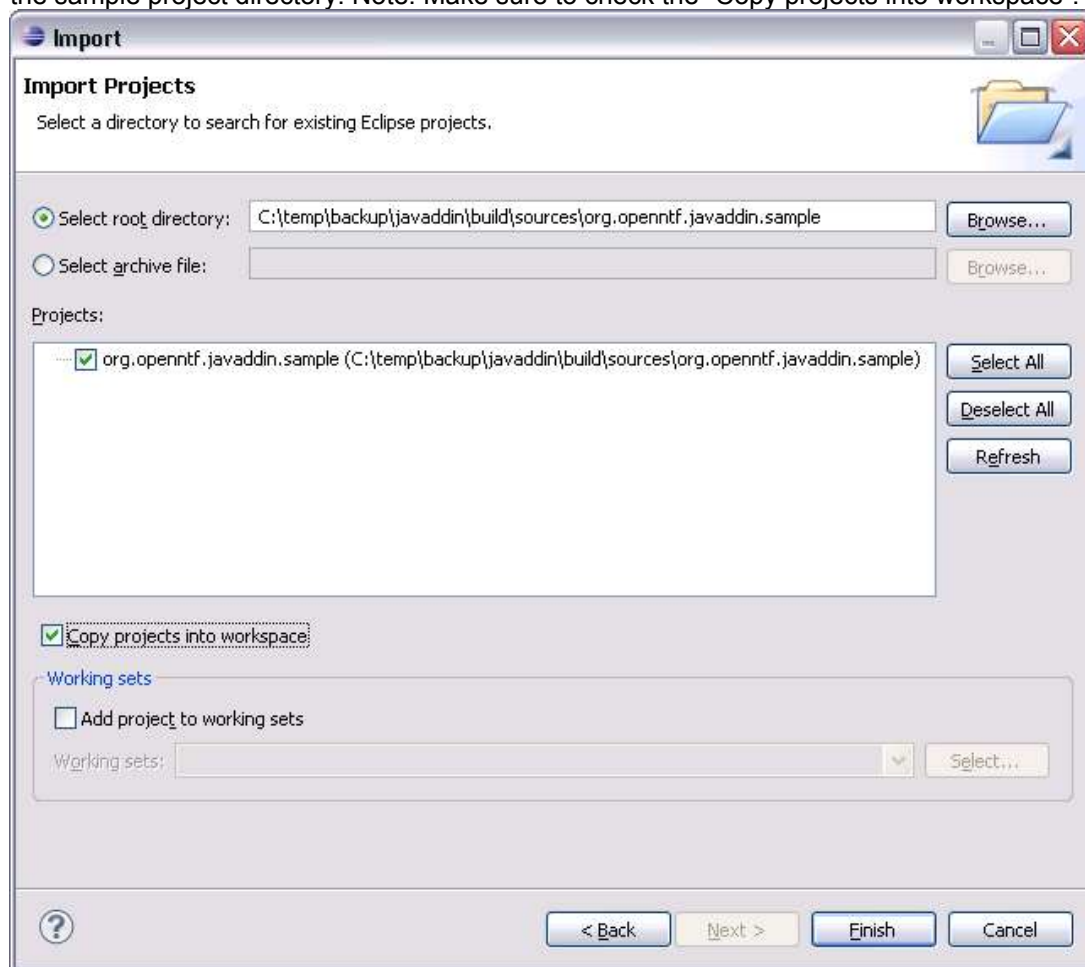
```
public void dispose() throws NotesException;
```

For more information, please look at the sample tasks provided in “org.openntf.javaddin.sample” plugin.

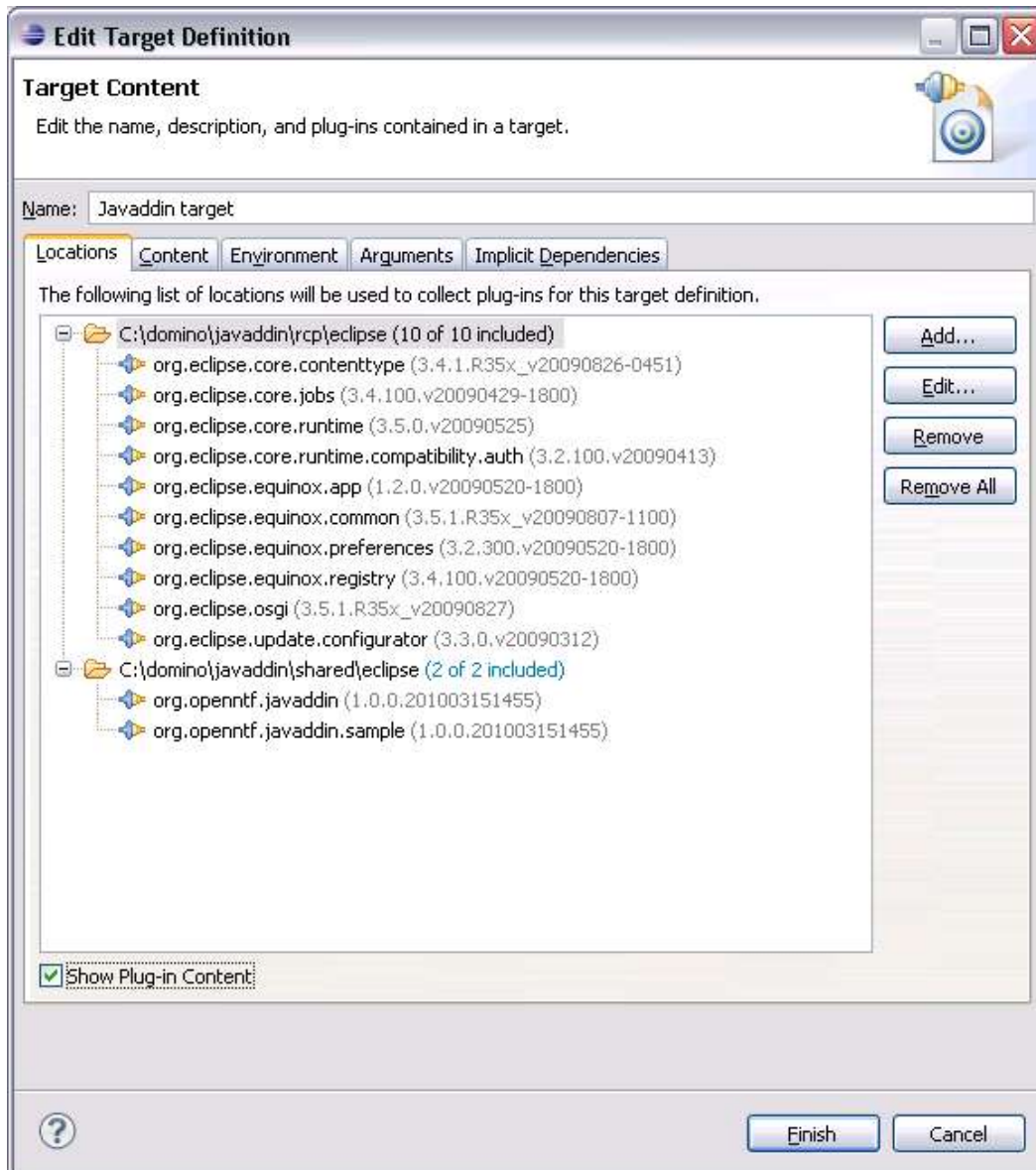
Setting up the dev environment to work with the sample plugin:

The recommended version for the Eclipse sdk : 3.5.1

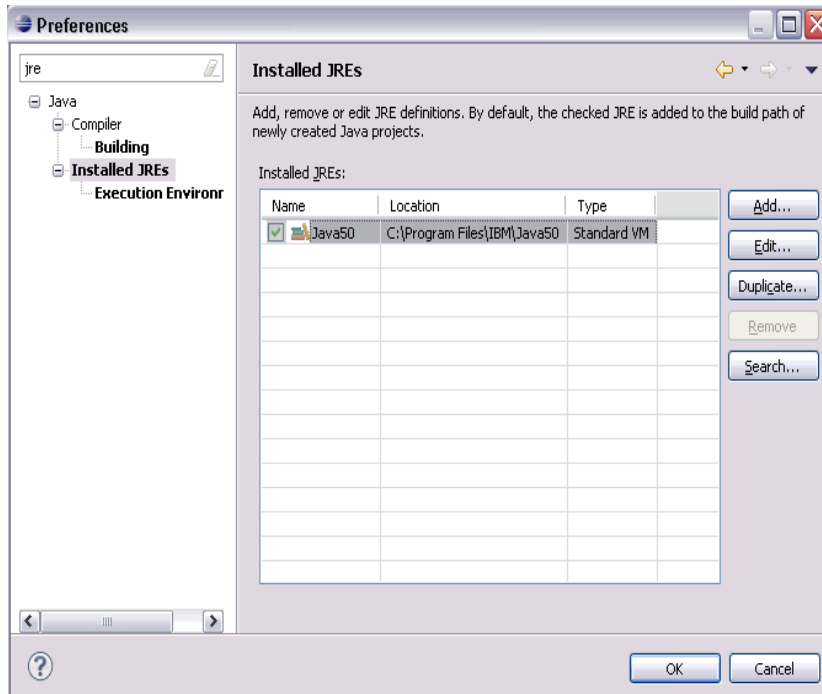
1. Import the source code for the sample project org.openntf.javaddin.sample located in the sources directory: use file\import menu, and select “General\Existing projects into Workspace”. In the next page use “select root directory:” radio button, then use the browser button to load the sample project directory. Note: Make sure to check the “Copy projects into workspace”.



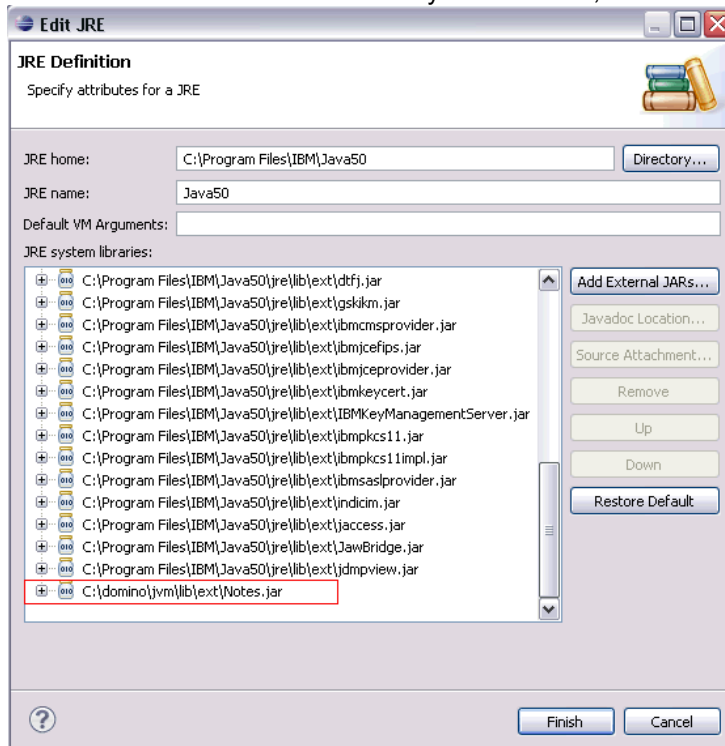
2. Set up the target platform: Use Windows\Preferences\Plug-in development\Target Platform and add the following 2 locations {notesbin}/javaddin/rcp/eclipse and {notesbin}/javaddin/shared/eclipse. The following screen shot shows how your target platform should look like:



- Note: Notes.jar is automatically made available via a fragment to the system bundle at runtime, but not at compile time. To fix this problem, you need to add the notes.jar to the JRE and add a dummy plugin to the target platform that exports the lotus.domino package.
 - Add Note.jar the JRE settings: Use Windows/Preferences and type JRE in the filter text



Select the Edit button and use the "Add External Jars" button. Browse to the Domino JVM directory under lib\ext, and select the Notes.jar:

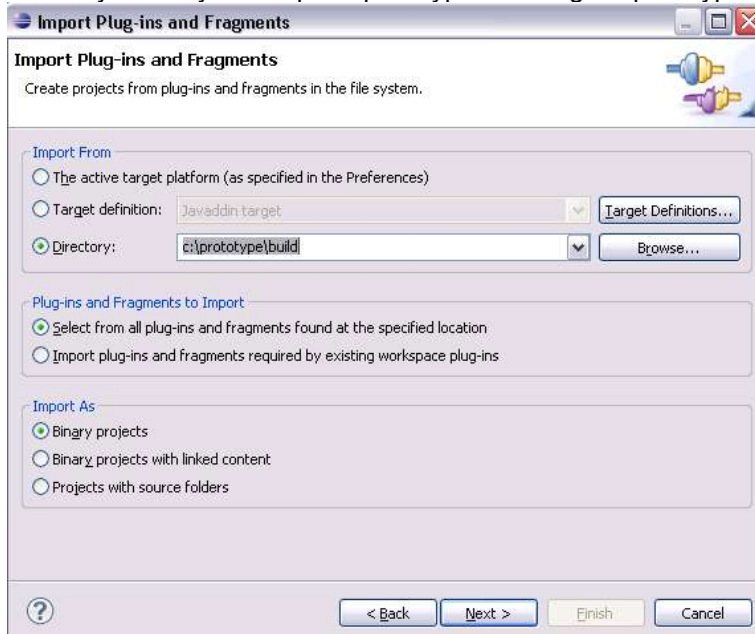


Note: In some cases, you may have Access Restriction errors on lotus.domino packages, this is because some plugin specify an Execution-Environment of 1.5 in the manifest. To fix this problem, you can do the following: use Windows\preferences\Java\Compiler\Errors-Warnings and select Ignore in the Forbidden references (access rules) field

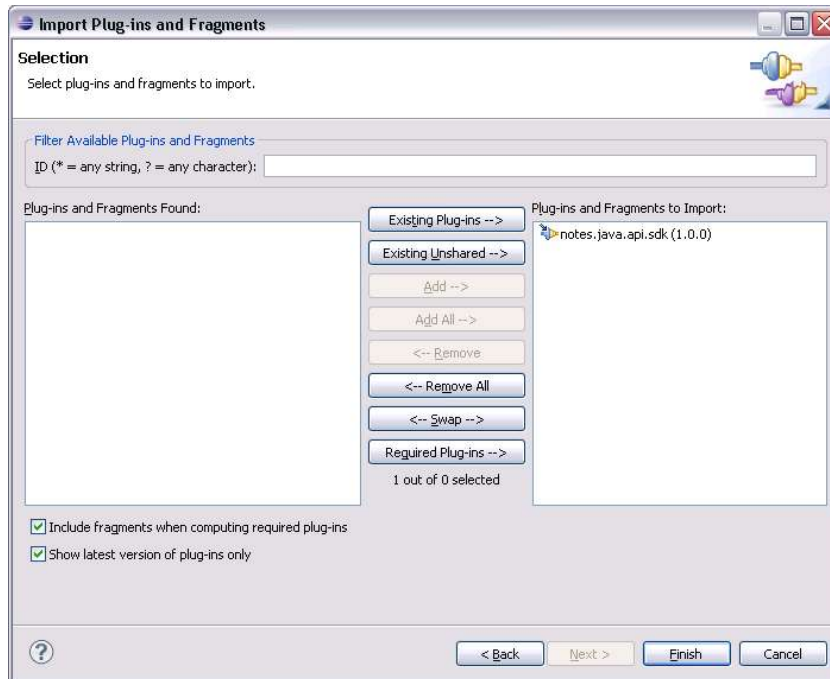
- Import the "notes.java.api.sdk" plugin located in the build directory: use file\import menu and select "Plug-ins and Fragments":



Click next. In the Import from section, use “Directory” radio button and select the build directory where you unzip the prototype code e.g. C:\prototype\build:

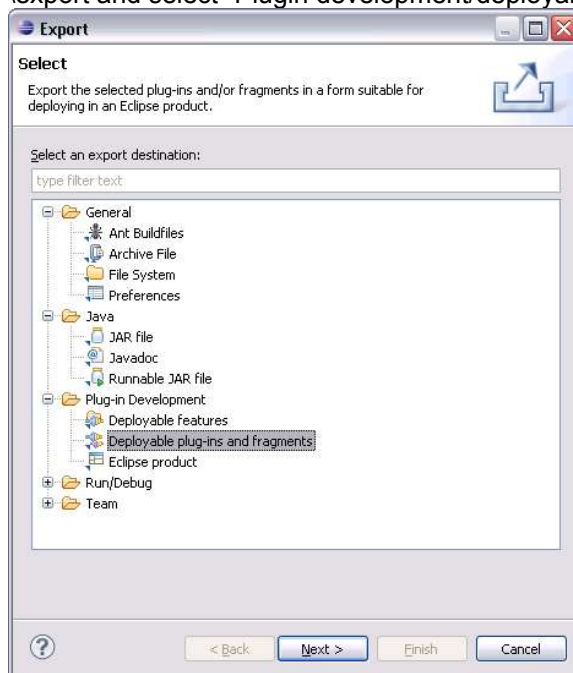


click next, double click on notes.java.api.sdk to add it to the list on the right:



click finish. You should have a new project called notes.java.api.sdk on your workspace.

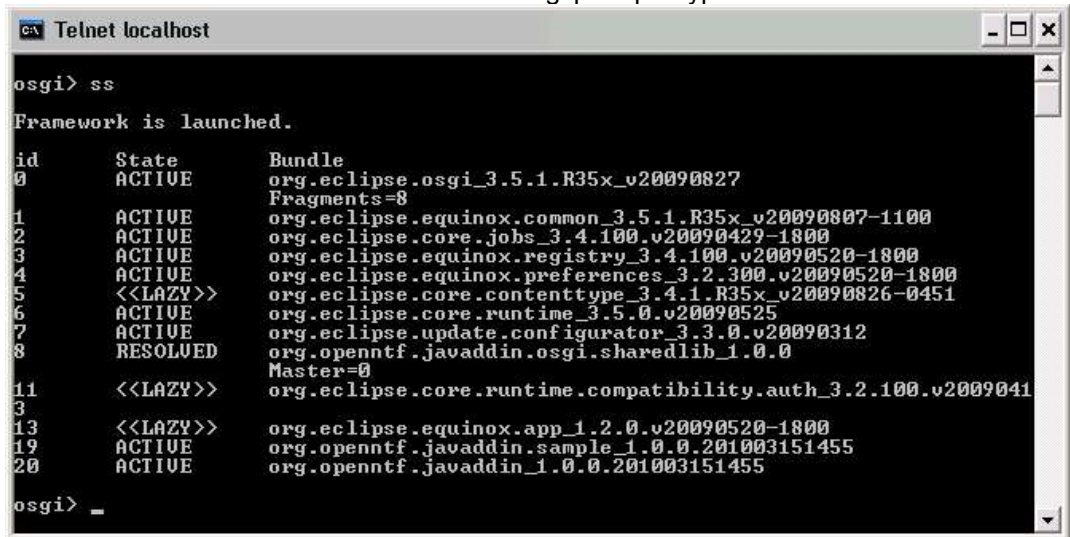
3. Click OK. You should now compile the sample project without error. The next step is to deploy the project as a jar (using file\export deployable plugins and fragments). Deploy the jar in {notesdata}\domino\workspace-javaddin\applications\ eclipse\plugins: Use file \export and select "Plugin development/deployable plug-ins and fragments"



click next, select the plugin you want to deploy. Use the "directory" radio button and specify "c:/domino/javaddin/shared/eclipse" as the directory. (Note: the export dialog box will automatically add the plugins directory so you don't have to specify it.

4. At this point, the sample plugin is deployed, you must restart the javaddin task to be loaded again.
5. You can verify that it is loaded correctly by using the osgi console. In a dos prompt, type

"telnet localhost 1234". You should see the osgi prompt. Type the ss command



```
CA Telnet localhost

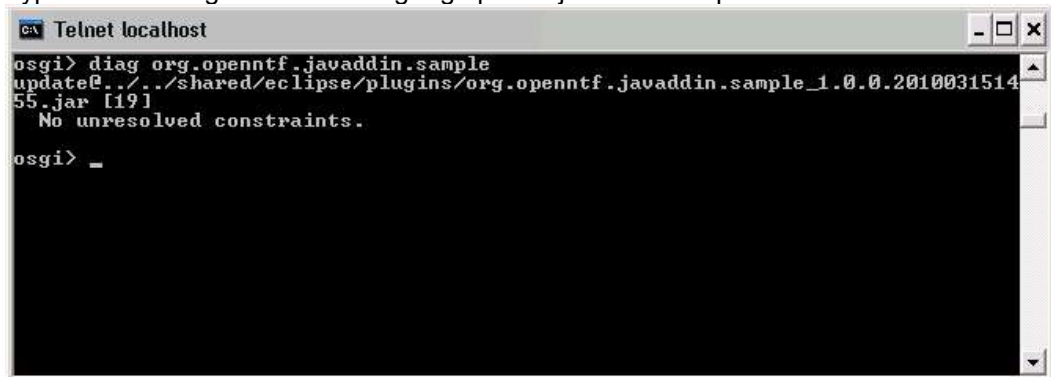
osgi> ss

Framework is launched.

id      State      Bundle
0       ACTIVE    org.eclipse.osgi_3.5.1.R35x_v20090827
        Fragments=8
1       ACTIVE    org.eclipse.equinox.common_3.5.1.R35x_v20090807-1100
2       ACTIVE    org.eclipse.core.jobs_3.4.100.v20090429-1800
3       ACTIVE    org.eclipse.equinox.registry_3.4.100.v20090520-1800
4       ACTIVE    org.eclipse.equinox.preferences_3.2.300.v20090520-1800
5       <<LAZY>>   org.eclipse.core.contenttype_3.4.1.R35x_v20090826-0451
6       ACTIVE    org.eclipse.core.runtime_3.5.0.v20090525
7       ACTIVE    org.eclipse.update.configurator_3.3.0.v20090312
8       RESOLVED  org.openntf.javaddin.osgi.sharedlib_1.0.0
        Master=0
11      <<LAZY>>   org.eclipse.core.runtime.compatibility.auth_3.2.100.v20090413
13      <<LAZY>>   org.eclipse.equinox.app_1.2.0.v20090520-1800
19      ACTIVE    org.openntf.javaddin.sample_1.0.0.201003151455
20      ACTIVE    org.openntf.javaddin_1.0.0.201003151455

osgi> _
```

If all goes well, you should see the org.openntf.javaddin.sample plugin in the list. Type the following command: diag org.openntf.javaddin.sample



```
CA Telnet localhost

osgi> diag org.openntf.javaddin.sample
update@./../shared/eclipse/plugins/org.openntf.javaddin.sample_1.0.0.201003151455.jar [19]
No unresolved constraints.

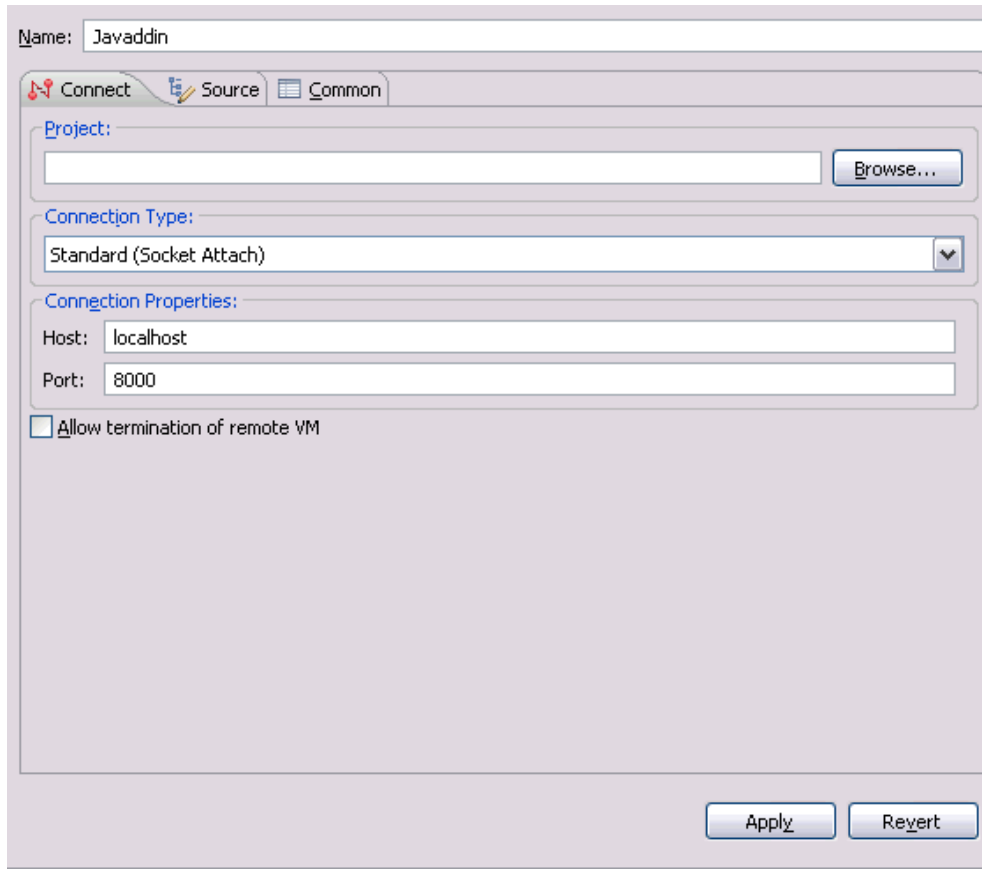
osgi> _
```

Debugging your server task plugin

Start the jvm in debug mode by using the following parameters:
load javaddin -debugaddress=<port> -debugsuspend=<y/n>

- debugaddress: connection port for the java debugger (usually 8000)
- debugsuspend: if y, then the jvm is start in suspend mode and resumes only when a debugger attaches to it. This is useful to debug starting code. Default is n

In eclipse dev environment, create a debug configuration (use run\debug configurations... menu) for javaddin using the same port used to start the jvm in debug mode, i.e. if port is 8000:



Once the debugger is attached, you can set a breakpoint in your server task and run it manually on the server using: `tell javaddin run <taskid>`.

Caveats:

1. This prototype has only been tested on windows 32
2. It is very likely that it will not work on IBM i platforms as the JRE is provided by the OS and the code currently assumes that it is located under {dominobin} directory

Possible future enhancements:

1. Allow multiple instances of javaddin
2. Allow plugin server tasks to run in their own thread.
3. Allow plugin to provide credentials for the domino session used to run the task
4. Ability to programmatically change the schedule of a task