Plug-in Development Tips and Tricks

“In PDE we do tooling, but our business is people”

- Chris Aniszczyk <zx@code9.com>
- Principal Consultant at Code 9
- PDE Technical Lead
Agenda

- Plug-in Development with PDE
- Tips and Tricks
- Q&A
Seamless Integration of Components
PDE

- PDE = Plug-in Development Environment

- Tools to develop Eclipse plug-ins
  - Wizards to create, import and export plug-ins and features
  - Specialized editors for plug-in manifest files
  - Templates for new plug-ins
  - Launchers to run, debug and test plug-ins
  - NLS tools to internationalize plug-ins
  - Automated class path management
PDE Details

- PDE is implemented as a set of plug-ins
- PDE is built on top of the Eclipse Platform and JDT
  - Uses Eclipse Platform and JDT extension points and APIs
- PDE is seamlessly integrated into Eclipse
- PDE gets no special treatment from the Platform or JDT
Plug-ins All the Way Down

- A plug-in is the fundamental building block of an Eclipse product
- Plug-ins build on top of and use other plug-ins
- To extend Eclipse, you must write plug-ins
- To write a rich client application, you must write plug-ins
- To write an OSGi-based application, you must write plug-ins (bundles)
Agenda

- Plug-in Development with PDE
- Tips and Tricks
- Q&A
Target Management

- Helps you to specify plug-in in which to build and run with.

- Includes tabs to set environment values, launching arguments, implicit dependencies, and source code locations.

- You can add plug-ins to the current target platform by using target provisioners. Current provisioners allow you to specify locations on your file system and the locations of update sites.

- The plug-ins can be viewed as a list or a tree (separated by locations).
Plug-ins View

- A view into all the plug-ins you’re working with
- Has the ability to add things to java search which can help you find classes via Ctrl+Shift +T
Error Log

- More than meets the eye…
- Group log entries by
  - Session
  - Plug-in
Execution Environments

- Execution Environments are symbolic representations of JREs
- Bundle-RequiredExecutionEnvironment (BREE) manifest header
- PDE Build uses BREE to determine compile settings
- http://wiki.eclipse.org/Execution_Environments
OSGi Launch Configurations

- Provides a way to easily run and test your bundle in an OSGi environment.

- Extensible framework that allows other OSGi runtimes to provide implementations to let users test on runtimes other than Equinox

- Gives users more advanced control, including the option to specify start levels for individual bundles.
OSGi Console

- -console

- Integrate with the console that drives Eclipse

- Common commands
  - Status
  - Start/stop
  - Install/uninstall
  - diag

- Custom Commands

Display View

- The Display View allows you to execute/inspect code while you are stopped at a break point. This can be very helpful when trying to find specific objects or debug their contents.
Conditional Breakpoints

- You can enable any breakpoint to stop only when it meets certain criteria. This helps if you are trying to iterate through a collection or if you are debugging a function which is called multiple times.

![Conditional Breakpoints in Eclipse IDE]
Plug-in Dependencies View

- The **Plug-in Dependencies** view allows you to see all the dependencies for any plug-in project.

- You not only see what plug-ins a project depends on, but also what plug-ins depend on that project (callers and callees).

- It also can display the current state, including dependency wiring, of the plug-ins in the workspace and target platform. This will aid in finding resolution problems.
Graph Plug-in Dependencies View

- PDE Incubator Project
- Visualize your dependencies
- Pictures are worth a thousand words

* http://www.eclipse.org/pde/incubator/dependency-visualization/*
Organize Manifests Wizard

- Helps you modify and format your MANIFEST.MFs
- Accessible by right clicking on a MANIFEST.MF and selecting PDE Tools > Organize Manifest

- Helps you:
  - Export all packages in a project
  - Remove unresolved packages
  - Mark exported packages as internal
  - Modify unresolved dependencies (removing or making them optional)
  - Remove unused dependencies
  - Calculate dependencies (using Dependency Management)
  - Prefix icon paths with $nl$
  - Remove unused NLS keys
  - Calculate 'uses' directives
Product Editor

- A product definition helps you to easily customize, test and export an RCP/Eclipse application
- Customizations include:
  - Modifying which plug-ins are included
  - Create a splash screen
  - Bundling a JVM
  - Name for the launcher executable
  - Specify program and launching arguments
  - Define a welcome page and About dialog
Externalization Strings Wizard

- PDE’s Externalize Strings wizard allows you to quickly locate and painlessly externalize values in a plug-in’s MANIFEST.MF and plugin.xml.

- Accessible by right clicking on a MANIFEST.MF or plugin.xml and selecting **PDE Tools > Externalize Strings**…

- Externalized values are put in a file specified by the Bundle-Localization header. The default value for this file is “plugin.properties”
The “.qualifier” marker allows you to easily substitute a value for the micro segment of a plug-in or feature’s version.

The date is the default value, but you substitute any value when exporting your project using the PDE export wizards.
Plug-in Registry View

- The **Plug-in Registry** view is your eye into the runtime

- Show Advanced Operations
  - start/stop bundles

- Show Extension Content Only
  - quickly browse extensions
Plug-in Project from existing jars

- The **Plug-in Project from Existing Jars** wizard enables you to quickly convert jar files to plug-ins.

- Helpful when an application is being converted to OSGi and it depends on certain library jars.

- Can be very useful for utility jars, as they can be shared across multiple plug-ins instead of requiring the jars be included in each plug-in.

- Embedded JARs are evil.
Plug-in Spy (3.4M3)

- ALT+SHIFT+F1
- Allows you to introspect what you’re looking at…
- Hyperlinking
- Shows contributing plug-ins
Open Plug-in Artifact (3.4M4)

- Ctrl+Shift+A
- Quickly browse plug-ins, extensions and extension points
API Tooling (3.4M6)

- API tooling will assist developers in API maintenance by reporting...
  - API defects such as binary incompatibilities
  - incorrect plug-in version numbers
  - missing or incorrect @since tags
  - usage of non-API code between plug-ins
Target Editor

- A target definition is a file that helps to configure your PDE development environment.

- They can be created in the workspace or loaded from plug-ins who define them in your platform.
Embedded Rich Client Platform (RCP)

embedded Rich Client Platform: RCP meets device!
Rich Ajax Platform (RAP)
Agenda

Plug-in Development with PDE

Tips and Tricks

Q&A
Conclusion

- http://www.eclipse.org/pde

- Want to contribute?
  - PDE Bug Day

- Thank you!
Questions?