Plug-in Development 201
Rich Client Applications

Chris Aniszczyk IBM Lotus, Austin

October 14, 2009

Tutorial Outline

- The Rich Client Platform
- Exercise Two: The Eclipse Browser Product
- Fragments
- Exercise Three: Internationalization
- Q&A

Tutorial Outline

The Rich Client Platform

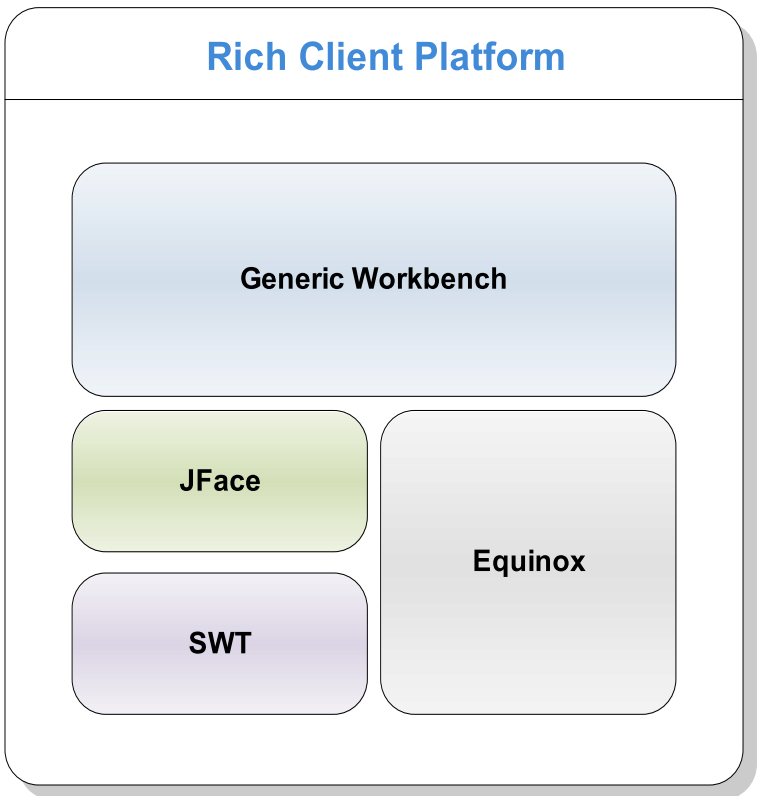
Exercise Two: The Eclipse Browser Product

Fragments

Exercise Three: Internationalization

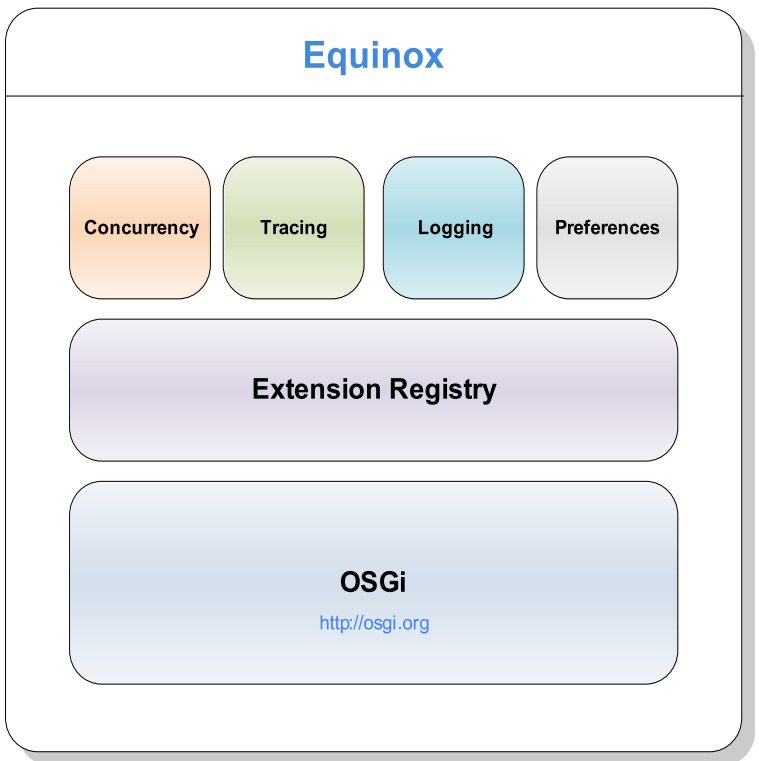
Q&A

Rich Client Platform (RCP)



- Equinox is the runtime
- Standard Widget Toolkit (SWT) is a portable and native widget toolkit for Java
- JFace is a framework for common UI programming tasks
- Generic Workbench provides the UI personality of the Eclipse platform

Equinox

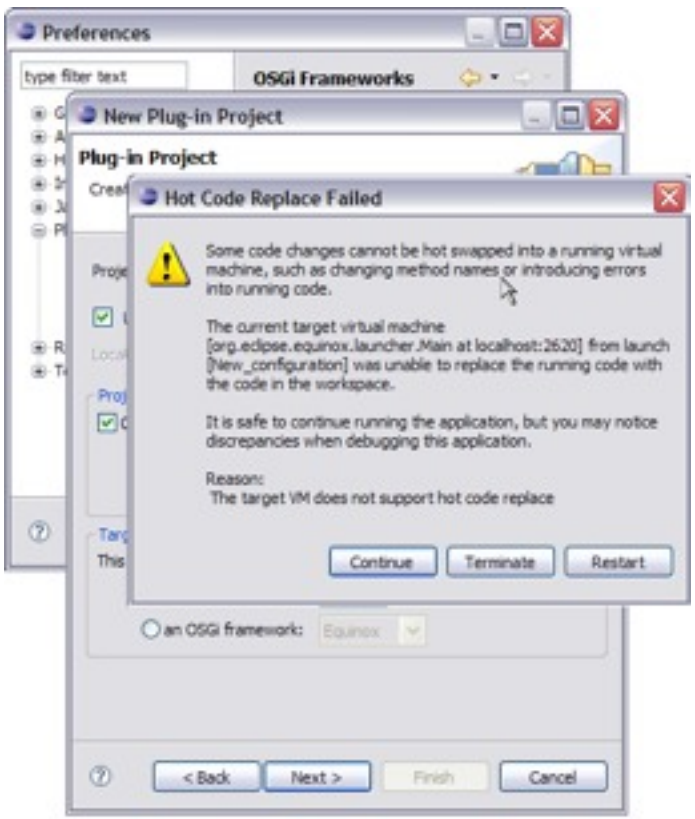


- OSGi is a framework that manages bundles
- Plug-in == Bundle
- The extension registry manages extensions and extension points
- The concurrency infrastructure allows for running background jobs
- Other runtime facilities include tracing, logging and preferences

Standard Widget Toolkit (SWT)

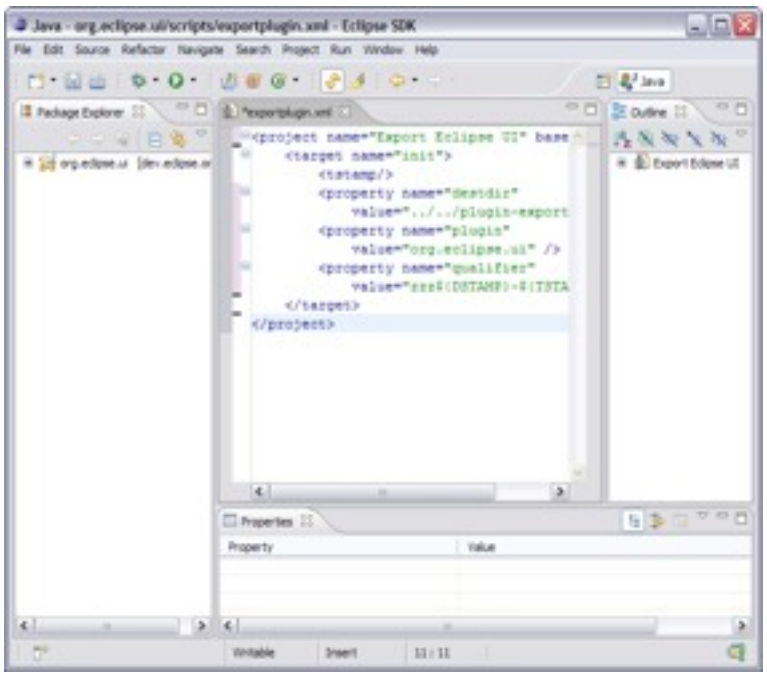
- SWT is a low-level graphics library that provides UI controls such as buttons, trees, combo boxes,…
- SWT uses native widgets as much as possible
- SWT has OS-independent API and is thus portable
- SWT is independent of the Eclipse runtime

JFace



- Built on top of SWT, JFace adds the model layer to the SWT widgets, e.g. tree viewers
- JFace provides common UI constructs such as wizards and dialogs
- JFace can be used standalone without the need for the Eclipse runtime

Generic Workbench



- The Workbench defines common user-defined paradigms:
  - Views, e.g. the Package Explorer
  - Editors, e.g. the Java and plug-in manifest editors
  - Perspectives: arrangement of views and editor

Contribution-Based Extensibility

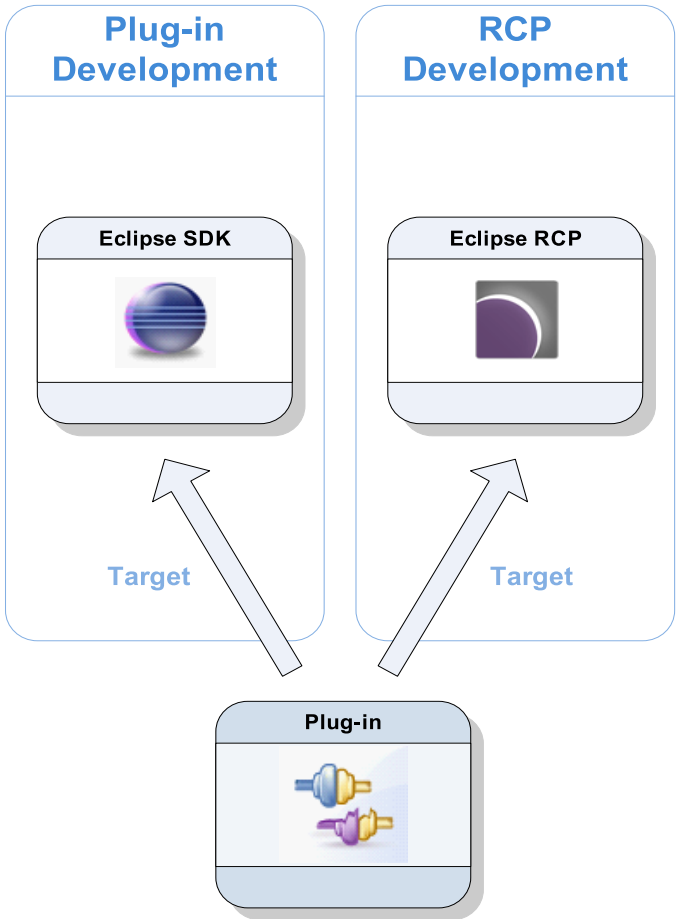
Why Use Eclipse RCP?

- An elegant plug-in architecture
- Eclipse RCP does the middleware. You do your job.
- From servers to embedded devices, RCP applications are portable
- RCP applications provide a native user experience
- JDT and PDE provide a first-class development environment

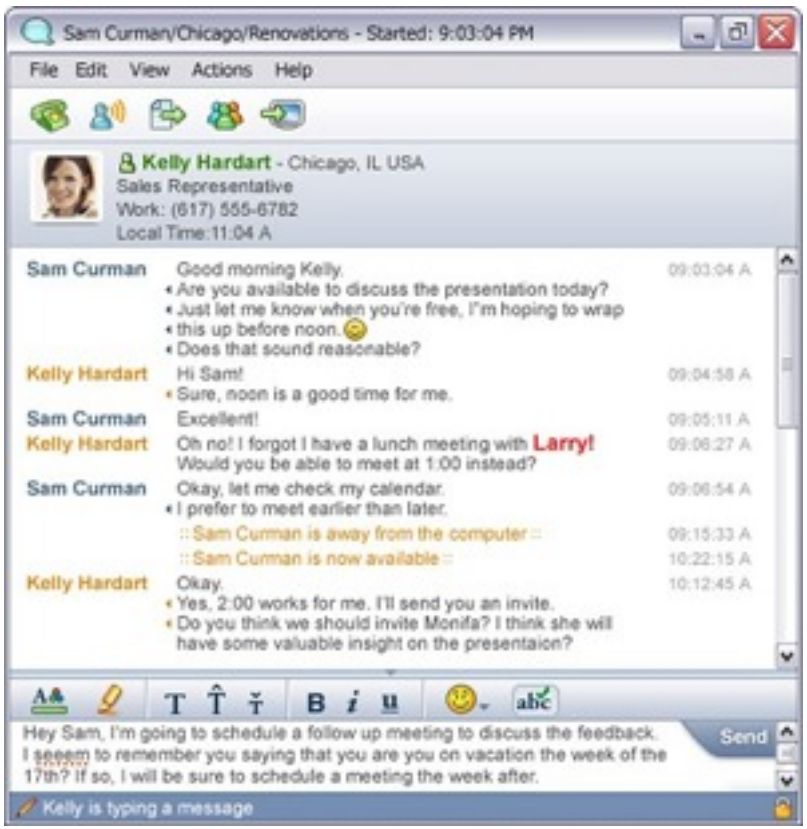
A Plug-in is a Plug-in is a Plug-in

- Developing plug-ins for a rich client application is identical to writing plug-ins for the Eclipse SDK
- Notable differences include:
  - Target Platform
  - Workbench configuration
  - Defining an application
  - Defining a product

A Smaller Target

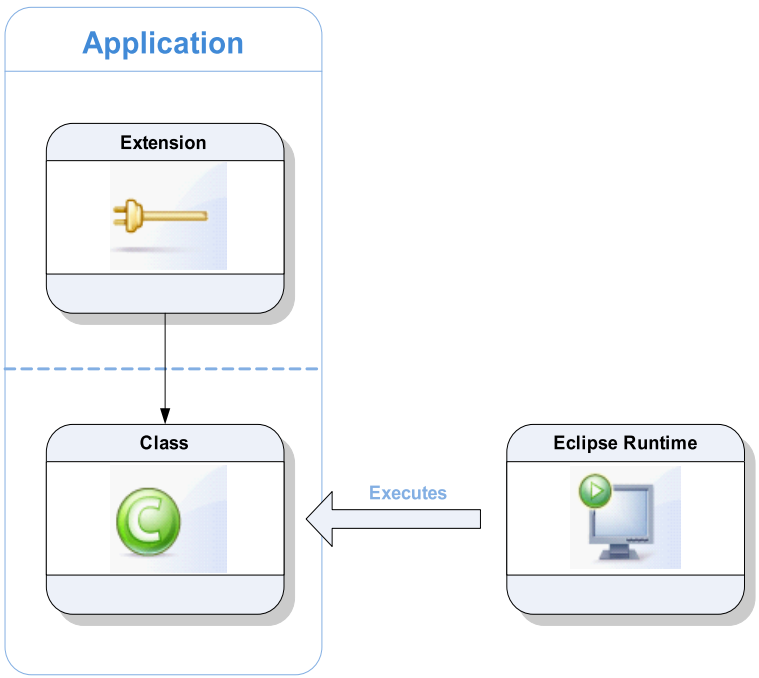


Customizing the Generic Workbench



- When writing a plug-in for the SDK, you can extend (i.e. add) to the workbench, but you cannot remove or override
- When writing an RCP application, you can configure every aspect of the workbench

Applications



- An application is to Eclipse what to the `main()` method is to a regular Java program
- To run Eclipse, an application has to be specified
- When you launch Eclipse, an application supplied by the IDE is run
- In an RCP scenario, you supply your own application

Products

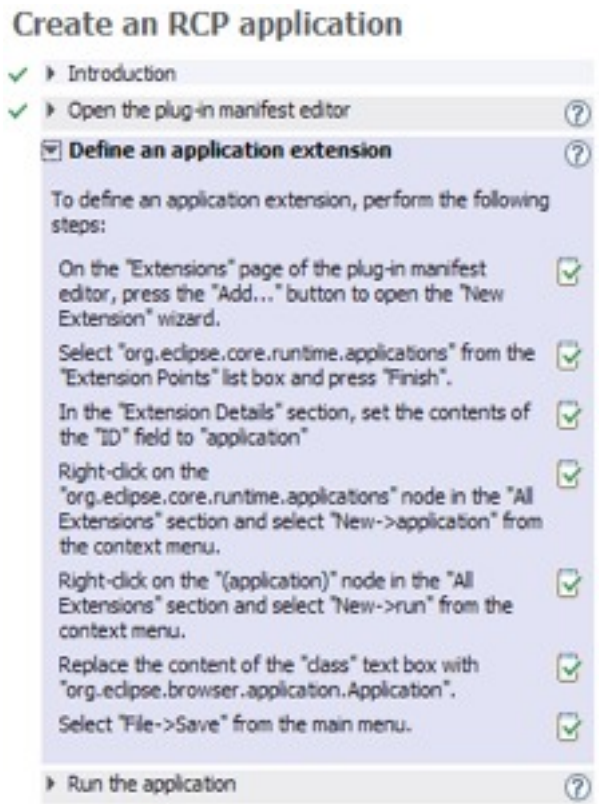
- A product is the Eclipse unit of branding
- Branding gives the rich client application a personality
- Branding encompasses window images, splash screen, a custom launcher
- A product is defined declaratively as an Eclipse extension

Tutorial Outline

- The Rich Client Platform
- **Exercise Two: The Eclipse Browser Product**
- Fragments
- Exercise Three: Internationalization
- Q&A

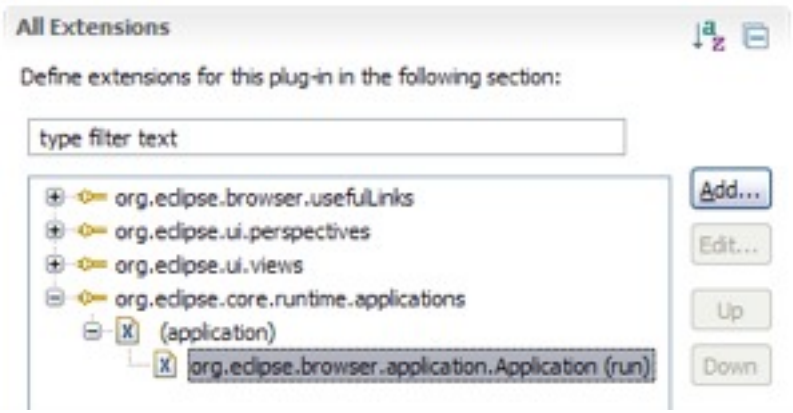
Exercise Two: The Eclipse Browser Product

Part I: Create an Eclipse Application



- This exercise is structured as a 3-step cheat sheet
- You use the plug-in manifest editor to create an application extension
- You launch the new application and you see that the same Browser view has seamlessly integrated into a different application

Define an Application Extension

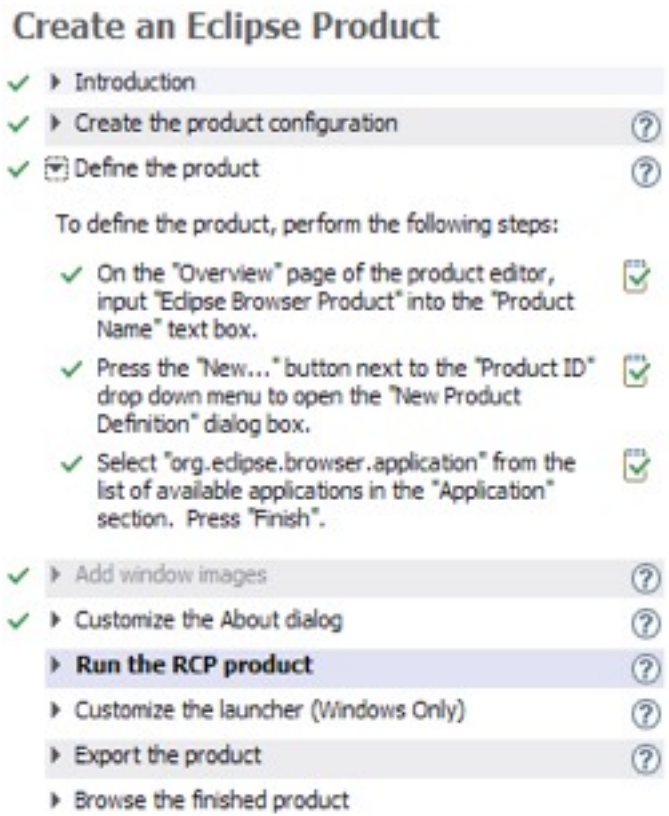


- An application is an `org.eclipse.core.runtime.applications` extension
- It specifies a class which serves as the entry point to the application
- For a typical UI application, the application creates, configures and runs a workbench
- The application exits when the workbench exits

Eclipse Browser Application

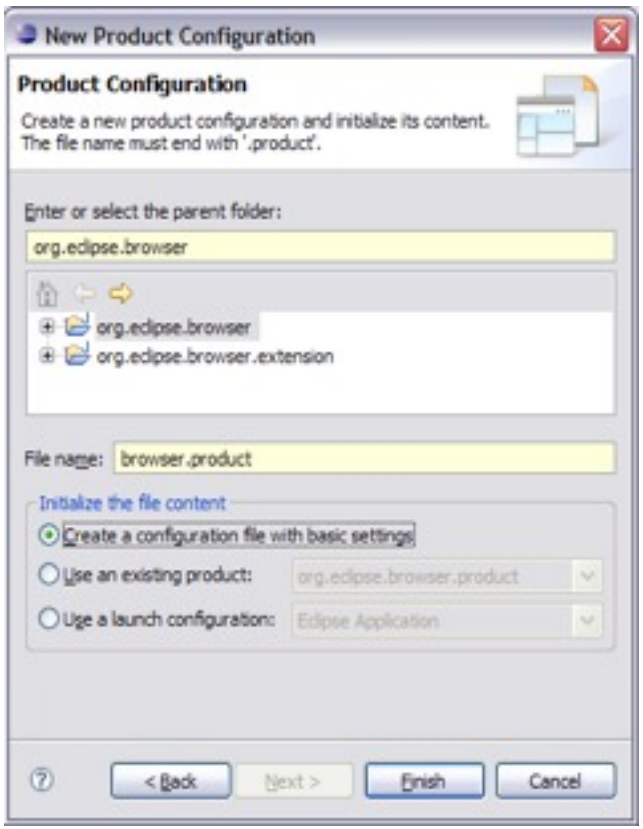
- The RCP application provides a standalone window with File and Help menus
- The Browser plug-in integrates seamlessly with the application without any code changes
- Don't underestimate this minimalistic application. It can go toe-to-toe with any other RCP application. It is configurable, extensible, …

Part II: Create an Eclipse Product



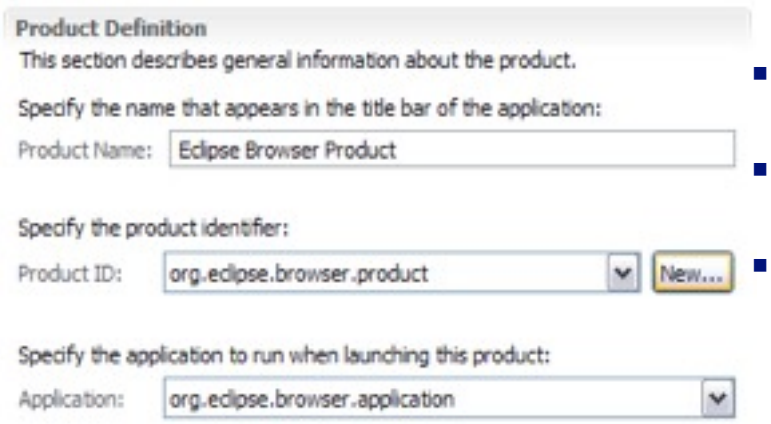
- This exercise is structured as a 8-step cheat sheet
- You use the product editor to define every aspect of your product: launcher, window images, About Dialog
- You export and run a fully-branded standalone product

A New Product Configuration



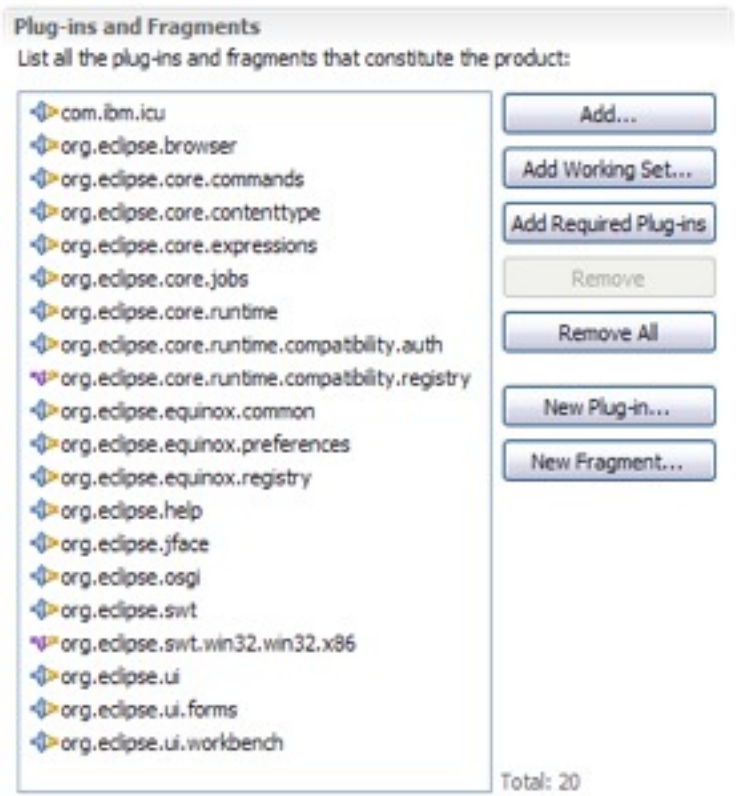
- A product configuration is the central place to manage all aspects of your product
- A product configuration is used by PDE to define and assemble a product
- A product configuration is neither read nor interpreted by the runtime

Product Definition



- A product is associated with an application
- A product provides branding and customization for the application
- A product name appears in the title bar of the application
- PDE uses this data to create an `org.eclipse.core.runtime.products` extension in the plug-in's manifest file

Building Blocks



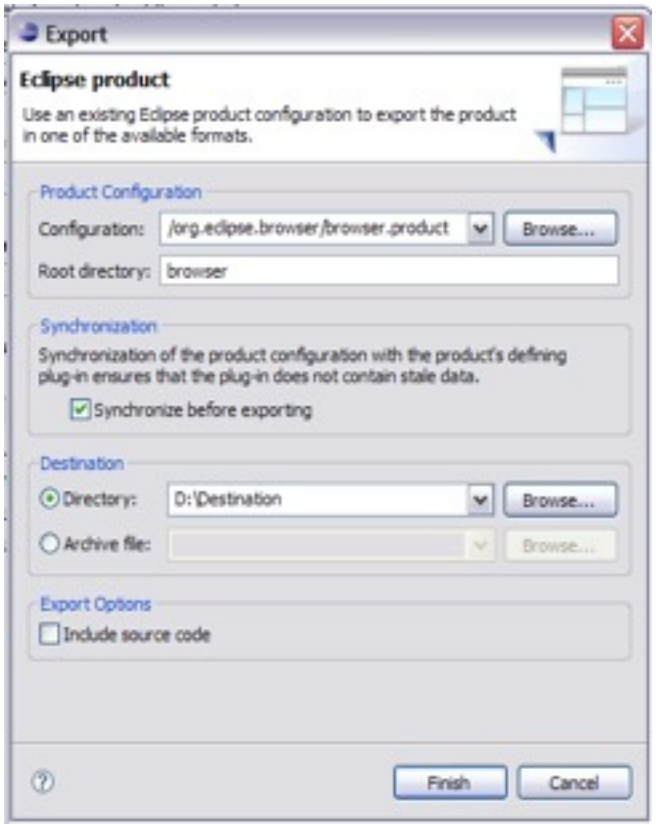
- A product must list all its constituent plug-ins. This list is only used by PDE to determine what to build and package
- Plug-ins that are in source form in the workspace are compiled and packaged
- Plug-ins that are already built (e.g. target plug-ins) are assembled as-is into the final product

Window Images

- Window Images are shown in the application window, task bar, … depending on the windowing system
- On Windows, the 16x16 GIF image is used for the task bar and the 32x32 GIF is used in the Alt-Tab application switcher

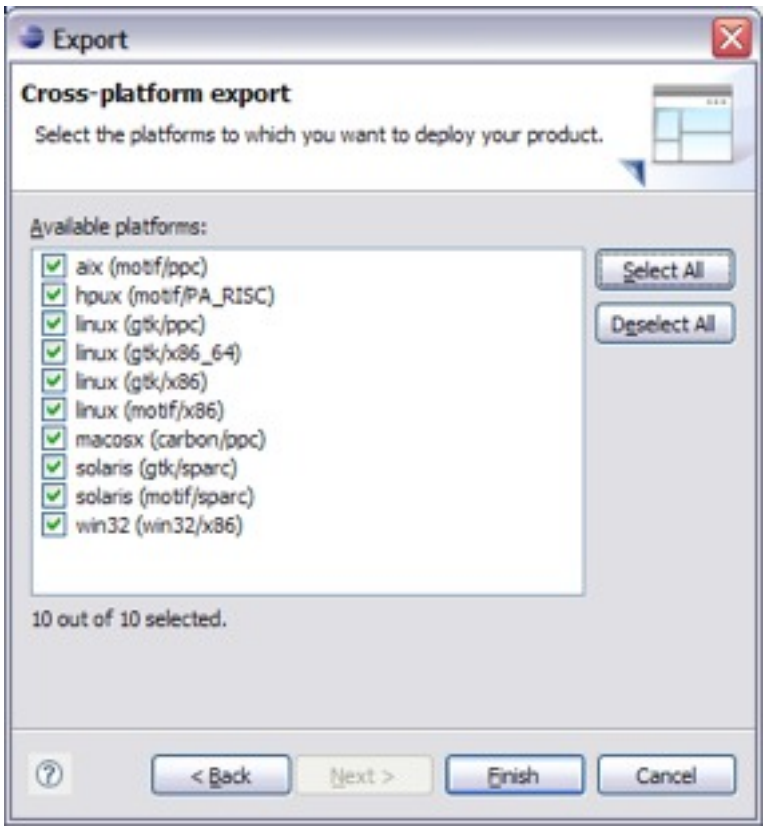
Customizing the About Dialog

One-click Export



- PDE provides a Product Export wizard that takes a product configuration file as input
- The name of the root directory of the product is customizable
- Options to export the product as a directory structure or an archive

One-click Export to Multiple Platforms



- The RCP delta pack is a separately downloaded archive that contains all OS-specific fragments and executables
- When present, you are able to export your plug-ins to all supported platforms in a single step


browser


eclipse
Browser


Eclipse Browser Product
File Help
Eclipse Browser
Eclipse Browser
Project Links
Double-click on a node to open the associated web site in a browser.
Eclipse Project
Platform
Equinox
JDT
PDE
Open source industry platform for the development of highly integrated tools
Useful Links
Click on a link to open the associated web site in a browser.
Eclipse Downloads
Eclipse Bugs
Eclipsepedia
Search
Enter a query into the text field and press enter to perform a Google search.
Search

Tutorial Outline

- The Rich Client Platform
- Exercise Two: The Eclipse Browser Product
- **Fragments**
- Exercise Three: Internationalization
- Q&A

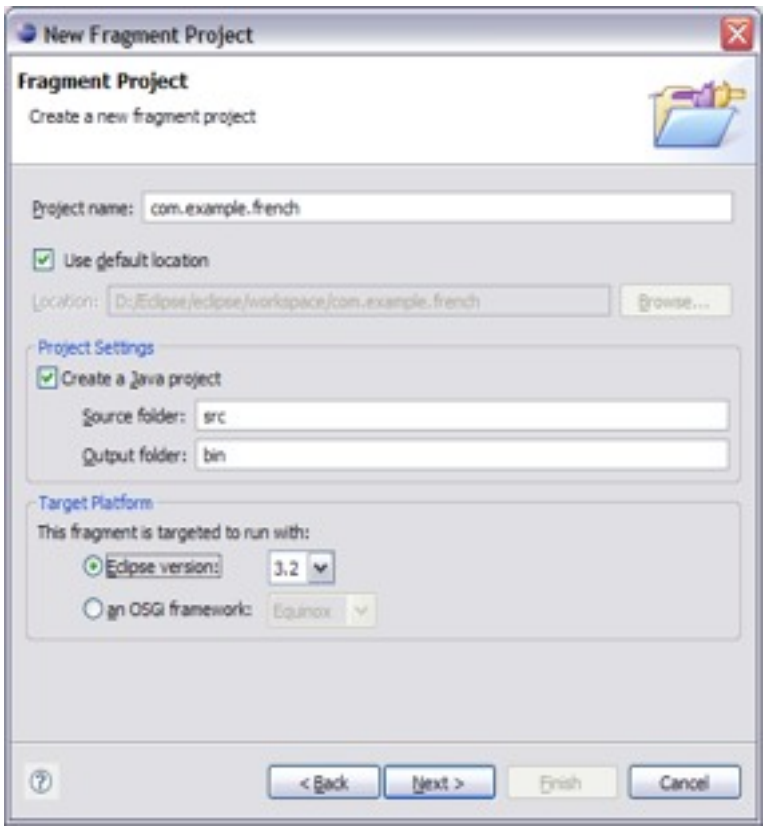
Fragments

- A fragment adds functionality to an existing plug-in (host)
  - Content (extensions, code) is merged with the host content at runtime
  - Fragments augment their host content. Fragments do NOT override.
- Fragments typically house:
  - Platform-specific content
  - **Locale-specific content: translation files**
- PDE provides a **New Fragment Project** creation wizard

How is a Fragment different from a Plug-in?

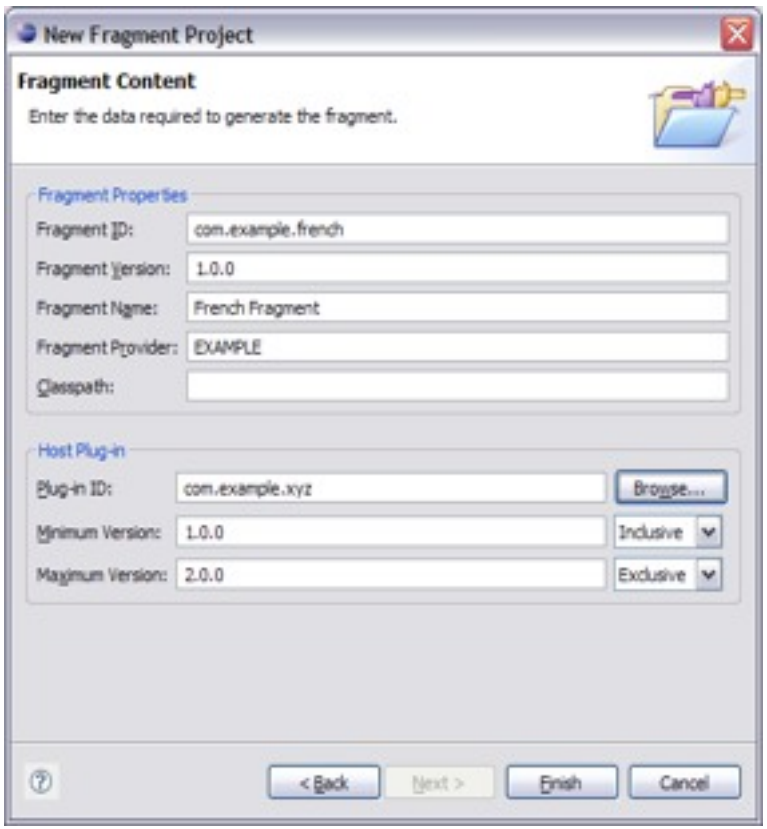
- Fragments need a host to attach to
- Fragments have **no activator and no lifecycle**
- Dependencies cannot be expressed on a fragment
- Fragments do not expose API
- Extensions and extension points are contained in **fragment.xml**

First Fragment – Step 1



- The first page looks very familiar.
- Same project settings as the one for a New Plug-in Project
- Same naming conventions, etc.

First Fragment – Step 2

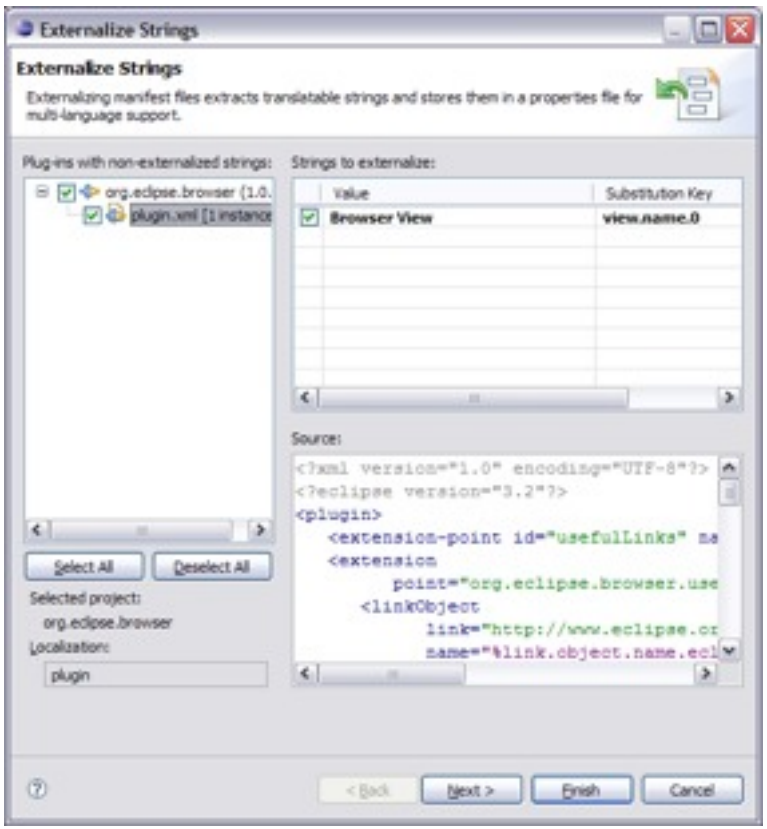


- We need the usual fragment ID, version, name, …
- No Activator. Fragments have no lifecycle.
- Specify the host to which the fragment will attach
- The fragment will only be matched up with a host that matches the version range constraint

Tutorial Outline

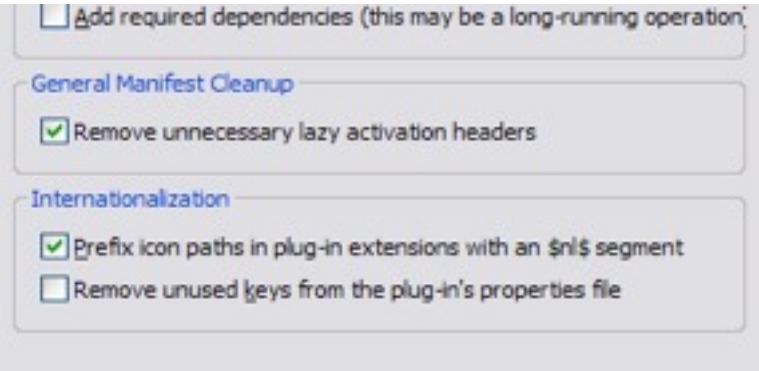
- The Rich Client Platform
- Exercise Two: The Eclipse Browser Product
- Fragments
- **Exercise Three: Internationalization**
- Q&A

Externalizing Strings



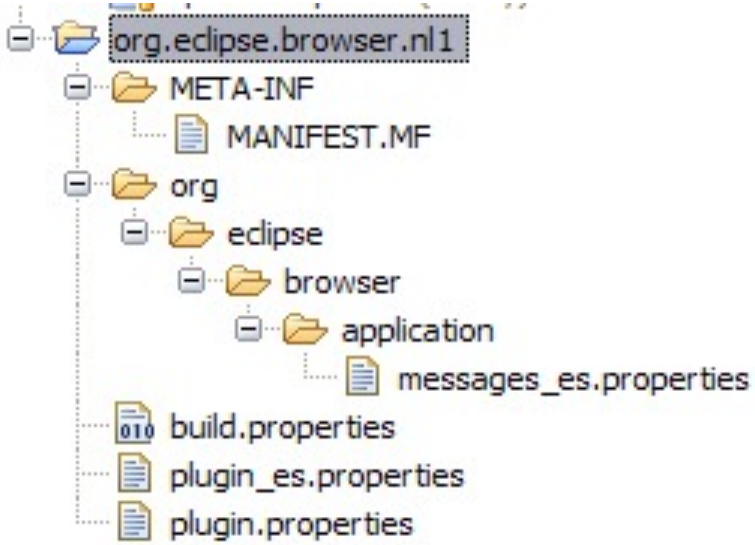
- Externalize strings so locale specific values can be substituted at runtime
- JDT provides an externalization wizard for java code while PDE provides an externalization wizard for plug-in data

Add locale specific icons



- The Organize Manifest wizard helps you to add the “nl” prefix to your icon paths which allows you to substitute custom icons based on locale

Language fragments



- Language fragments contain locale specific values in properties files.
- The properties files must follow the naming convention of “<host’s filename>_<locale>.properties”

Tutorial Outline

- The Rich Client Platform
- Exercise Two: The Eclipse Browser Product
- Fragments
- Exercise Three: Internationalization
- Q&A

Questions and Answers?

Legal Notices

- Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- IBM, Lotus, Rational are copyrights of IBM Corporation in the United States, other countries, or both.
- Other company, product, or service names may be trademarks or service marks of others.