EclipseCon 2015

Advanced features of the Eclipse 4 API for RCP and IDE development
Lars Vogel

Eclipse Platform UI Co-Lead
e4 Committer
Java Champion

Founder of vogella GmbH which offers Eclipse and Android consulting and training support

Author of several Eclipse and Android books and
vogella.com website with more than 1,5 million visits per month.
Part 1 - extending the IDE with e4
The Goal

Extensions

Define extensions for this plug-in in the following section.

* org.eclipse.ui.perspectiveExtensions
* org.eclipse.ui.views
* org.eclipse.ui.menus

fragment.e4xmi

- Model Fragment Definition
  - Imports
  - Model Fragments
    - Model Fragment - children (topLeft)
    - Part
    - Model Fragment - menuContributions (org.eclipse.e4.legacy.ide.application)
      - Menu Contribution
        - HandleMenuitem
    - Model Fragment - commands (org.eclipse.e4.legacy.ide.application)
      - Command - IDE Command
    - Model Fragment - handlers (org.eclipse.e4.legacy.ide.application)
      - Handler - IDE Command
Eclipse application modularity with fragments and processors
Model fragments – xmi snippets contribute to the runtime model (declaratively)

Model processors – Java classes to change the model (programmatically)
Contribute to ID and Feature

ID in your main Application.e4xmi

Feature to contribute too
Prerequisites

Eclipse 4 IDE still uses a lot 3.x plug-ins

BUT

maps the 3.x Elements to E4 Application-Model
• Use Menu Contributions, Toolbar Contributions, Trim Contributions to extend the IDE with e4 menus and toolbars
• Works in the IDE and 3.x and 4.x RCP applications

The trick is to use the correct ID and model element
Steps

• Find the Element ID of the parent
  • Plugin-Spy / Plugin-Menu-Spy
  • Model-Spy

• Attach your E4 Elements to this parent via fragment.e4xmi
Exercises....
Samples for the usage of e4 API

- [https://github.com/vogellacompany/](https://github.com/vogellacompany/)
  - Saneclipse – Templates and Defaults for Coding
  - Codemodify – JUnit 4 migration tool
  - PreferenceSpy – See changed preferences in a Table
  - ...

- ....
Planned enhancement: Contribute via X-Path

- Bug 324954 - [Model] Allow model fragments to use XPath-Syntax to decide where to contribute
  - https://bugs.eclipse.org/324954
The Goal: Change the IDE behavior
Event service

Provider

Software Component

Event provider does not know anything about the receivers

Part activated

Label changed

Children saved

more events...

@Inject
private IEventBroker broker;
Dependency injection looks in the EclipseContext

Diagram:
- Application
  |- Window
  |  |- Perspective
  |     |- Part
  |     |  |- Part
Actually DI uses other sources as well, for example the OSGi service registry
@Inject @Optional @EventTopic("key1") Type t;
Receiving Events (types)

@Inject
@Optional
public void receive(@EventTopic("key1") Todo o) {
    // do something
}

@Inject
@Optional
public void receive(@EventTopic("key1") String o) {
    // do something
}
Eclipse communication

The Eclipse platform uses events for its complete user interface communication.

Events listed in
org.eclipse.e4.ui.workbench.UIEvents
Model add-ons are a way to react to events in the IDE
Part of the application model below application without user interface
Typically register for events and act upon them
Core IDE functionality controlled by model add-ons
Several classes are stored in the IEclipseContext which you can override
Demo and exercises.... (different save dialog)
Using e4 API in 3.X components
3.x part accessing 4.x API

// get the context of a part
IEclipseContext parentContext = (IEclipseContext) site.getService(IEclipseContext.class);
// or access directly a value in the context based on its key
EModelService service = (EModelService) site.getService(EModelService.class);
3.x handler accessing 4.x API

// get the context of a part
IEclipseContext parentContext = (IEclipseContext) site.getService(IEclipseContext.class);

// or access directly a value in the context based on its key
EModelService service = (EModelService) site.getService(EModelService.class);
Additional model data
Additional model data

Tags
Persisted State
Context Properties
Transistent data
Usage

Renderer can use the information to drive different behavior

Events are send out if this data changes, e.g. if a tag is set on a model element

Model data can be accessed via API, e.g., to configure the parts

Tags can be used as CSS classes.