



Eclipse 4 Migration

Bangalore 2016

Table des matières



I - Migration to E4	5
A. Migration to E4.....	5
B. Migration tooling.....	8
C. Context concerns.....	12
D. Model Fragments and Processors.....	15
E. Extensions Migration.....	19
F. Resources.....	24

Migration to E4



Migration to E4	5
Migration tooling	8
Context concerns	12
Model Fragments and Processors	15
Extensions Migration	19
Resources	24

This part of the talk will explain :

- some general issues about migration
- tooling that could be used
- some injection issues
- the model fragments and processors
- how to migrate some standard extensions

A. Migration to E4

The technical reasons for using E4 application platform

- Application model is dynamic and platform agnostic (SWT, Java FX...)
- Injection is pretty cool, reduces the amount of code and simplifies testing (thanks to POJOs)
- Eclipse 4 event notification system (**IEventBroker**) is very concise and easy to use with injection
- You want to use the CSS styling capability and change element renderers of Eclipse 4
- You want to use the E4 spies to help to develop your application
- Your application will still live several years and it will provide an opportunity to refactor and decouple your components

The global prerequisites

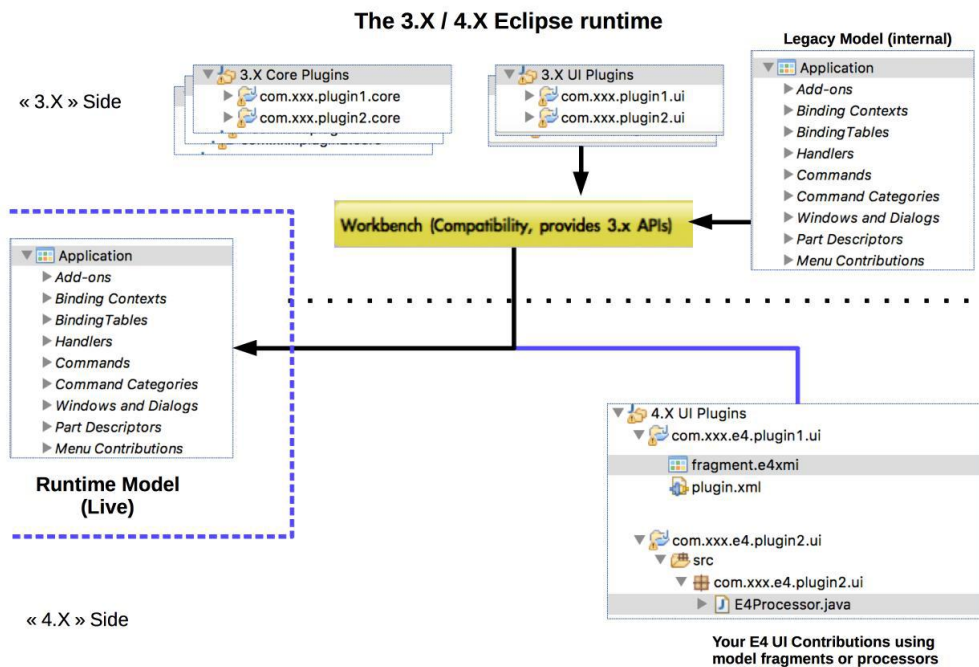
Be sure of your team's knowledge :

- do they know Eclipse 3 and Eclipse 4 ?
- do they know the application !! ?
- do they know how to migrate ?

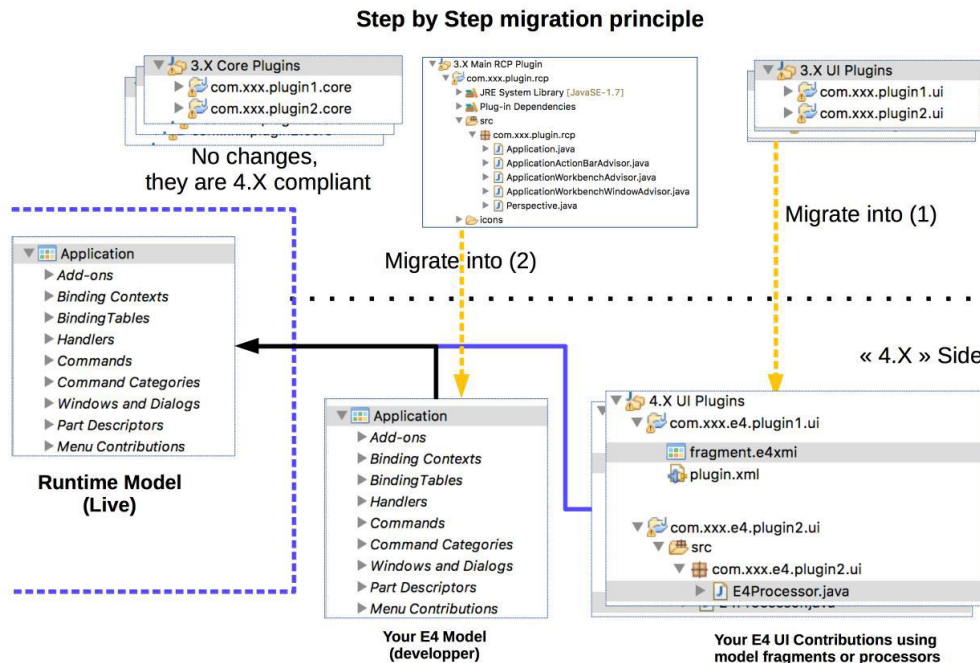
Create a migration strategy

- Identify the features you want to migrate and the reasons why
- Be aware that you will may not be able to migrate the entire application !

Big picture of 3.X application with 4.X runtime



Big picture of what we should do



The technical prerequisites

To prepare your E3 plugin/application migration you have to :

- remove the `org.eclipse.ui` internal package uses and imports
- ensure the application can be launched using the compatibility layer
 - `org.eclipse.equinox.ds`
 - `org.eclipse.equinox.event`
 - `org.eclipse.equinox.util`
 - `org.eclipse.e4.ui.workbench.addons.swt`
- clearly separate core and ui plug-ins
- have packages for each entities to migrate : views, handlers, etc...

Migration steps

To migrate a core plugin (without dependency to `org.eclipse.ui`), you must :

- do nothing !

To migrate an UI plug-in, you must :

- move the ui E3 extensions to a model fragment (or to the application model)
- migrate the relevant code
- remove all E3 extensions
- remove the `org.eclipse.ui` dependency when it is not used anymore
- add the `jface` dependency and others instead

Then, once all the plug-ins have been migrated, it is possible to remove the compatibility layer.

Practical advices

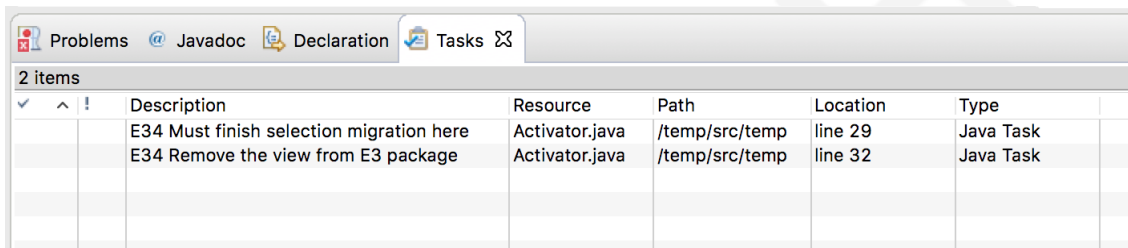
- Create a xxx.e4.xxx package to put the migrated class, **in current migrated plug-in**
 - for instance : xxx.e4.handlers or xxx.e4.parts
- Copy the E3 class and its dependencies in this package and keep the names
- Set the E3 classes as 'deprecated'
- Annotate with a **//E34** comment the current migrated areas when they are not finished
- Bind your new E4 class using a model fragment (or the main application model)
- Remove the old E3 packages when the migration is finished

These tips help maintain existing plugins and the build process

Displaying the //E34 tasks

It is possible to display the **//E34** comments in the task view :

- open the 'Tasks' view
- add a E34 tag in the preference page of Java->Compiler->Task



	Description	Resource	Path	Location	Type
✓	E34 Must finish selection migration here	Activator.java	/temp/src/temp	line 29	Java Task
	E34 Remove the view from E3 package	Activator.java	/temp/src/temp	line 32	Java Task

E34 tasks

B. Migration tooling

E4 Spies

- The E4 spies are useful to develop an E4 application
- They help in browsing the application model, injection contexts, events, css....
- It is possible to write its own spy for any specific data
- Eclipse Mars does not include the E4 spies
- They will be soon delivered by default
- To install them, upload the update site from :
 - <http://download.eclipse.org/e4/downloads>¹

Download the zipped update site and install it :

- Menu Help -> Install New Software..
- 'Add..', 'Archive..'

1 - <http://download.eclipse.org/e4/downloads>

Then select 'All Spies' :

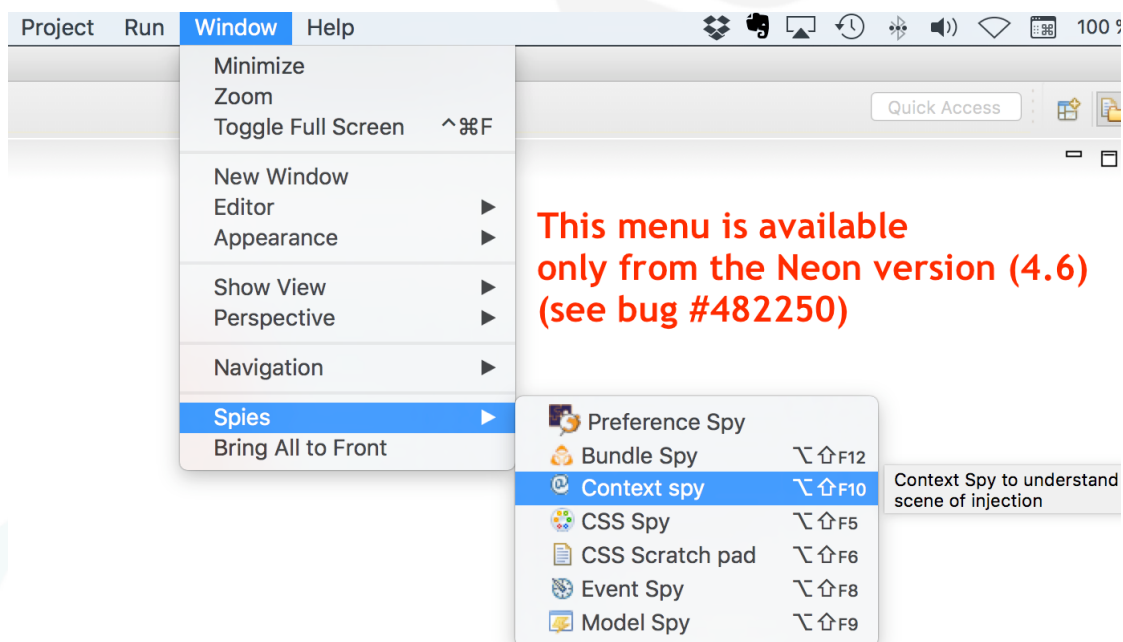
Name	Version
<input checked="" type="checkbox"/> Eclipse 4 - All Spies	
<input checked="" type="checkbox"/> Eclipse 4 - All Spies (Incubation)	0.1.0.v20150508-0800
<input type="checkbox"/> Eclipse 4 - Bundle Spy	
<input type="checkbox"/> Eclipse 4 - Context Spy	
<input type="checkbox"/> Eclipse 4 - CSS Spy	
<input type="checkbox"/> Eclipse 4 - Event Spy	
<input type="checkbox"/> Eclipse 4 - Model Spy	
<input type="checkbox"/> Eclipse 4 - Preference Spy	

Image 1 E4 tooling

Using the spies

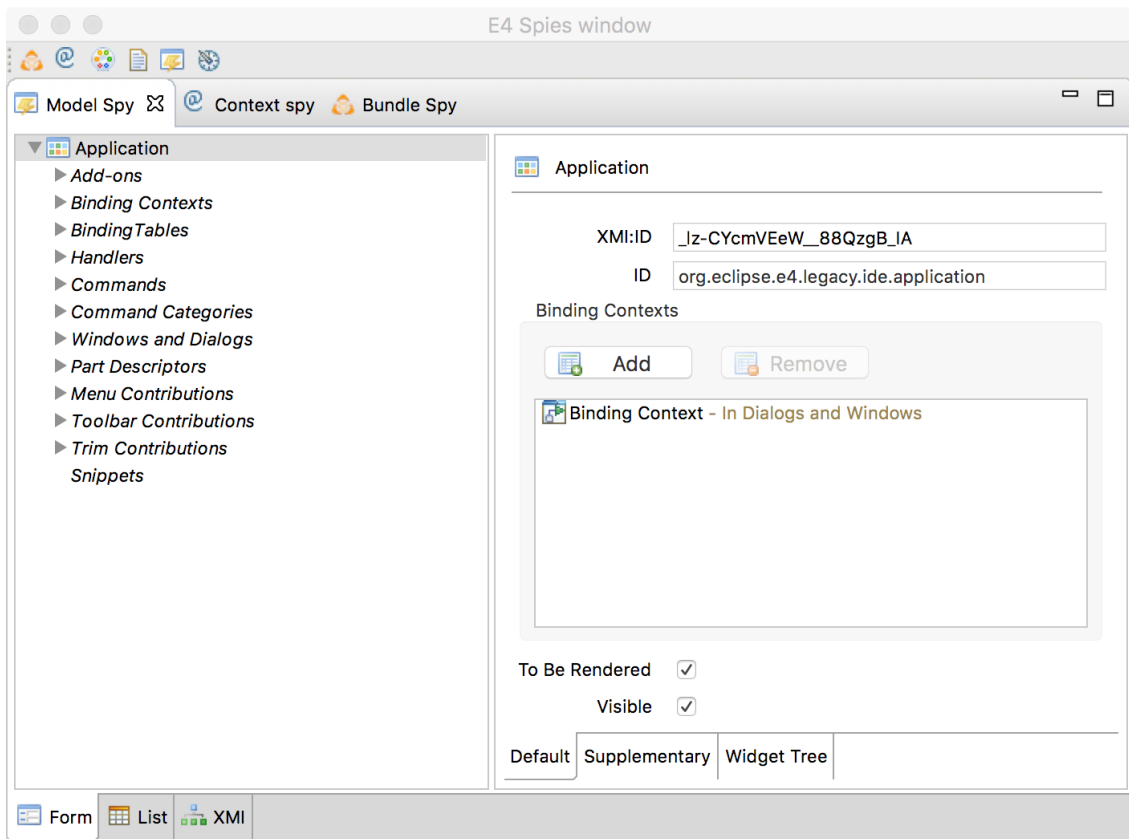
There are 3 different ways to open the spy window :

- use one of the shortcut (**Alt Shift F4** to **Alt Shift F10** for instance) depending on the installed spies
- look for "spy" in the quick access field
- use the Window->Spies menu (only in Neon) :



It will open a specific E4 Spies Window with a toolbar to display each spy.

For instance the **Model Spy** :



e4 Spies Window

A tooling to help to evaluate the migration cost

- OPCoach developed a specific statistic view dedicated to migration
- This plugin is available on github : <http://opcoach.github.io/E34MigrationTooling/>
- It is delivered under EPL license and it is free

- Select the projects in the workspace and get some statistics about used ui extension points :

Select the projects and check the stats

Usual Extension points

- views/view : 17
- editors/editor : 6
- propertyPages/page : 45
- commands/command : 256
- handlers/handler : 3
- menus/menuContribution : 1
- newWizards/wizard : 1
- importWizards/wizard : 4
- exportWizards/wizard : 6

Deprecated Extension points

- acceleratorConfigurations/acceleratorConfiguration : 0
- acceleratorSets/acceleratorSet : 0
- actionSetPartAssociations/actionSetPartAssociation : 6
- commands/keyBinding : 0
- commands/context : 0
- commands/activeKeyConfiguration : 0
- menus/group : 0
- popupMenus/objectContribution : 5
- viewActions/viewContribution : 0
- acceleratorScopes/acceleratorScope : 0
- actionDefinitions/actionDefinition : 0
- actionSets/actionSet : 11
- commands/keyConfiguration : 0
- commands/scope : 0
- editorActions/editorContribution : 3
- menus/widget : 0
- popupMenus/viewerContribution : 2

Use the toolbar to filter the results

Extension Points	org.eclipse.jdt.ui	org.eclipse.ui	org.eclipse.ui.ide	org.eclipse.ui.ide.application	Count
org.eclipse.jdt.ui.classpathAttributeConfiguration	1	0	0	0	1
org.eclipse.jdt.ui.classpathContainerPage	2	0	0	0	2
org.eclipse.jdt.ui.classpathFixProcessors	1	0	0	0	1
org.eclipse.jdt.ui.cleanUps	1	0	0	0	1
org.eclipse.jdt.ui.foldingStructureProviders	1	0	0	0	1
org.eclipse.jdt.ui.javaCompletionProposalComputer	19	0	0	0	19
org.eclipse.jdt.ui.javaCompletionProposalSorters	1	0	0	0	1
org.eclipse.jdt.ui.javaEditorTextHovers	1	0	0	0	1
org.eclipse.jdt.ui.javaElementFilters	1	0	0	0	1
org.eclipse.jdt.ui.quickAssistProcessors	1	0	0	0	1
org.eclipse.jdt.ui.quickFixProcessors	1	0	0	0	1
org.eclipse.ui.actionSetPartAssociations	5	0	0	0	5
org.eclipse.ui.actionSets	6	0	1	0	7
org.eclipse.ui.activitySupport	0	1	1	0	2
org.eclipse.ui.bindings	1	1	1	0	3
org.eclipse.ui.commandImages	0	1	1	0	2
org.eclipse.ui.commands	3	1	1	0	5
org.eclipse.ui.contexts	1	1	0	0	2

Migration Stat View

An evaluation form to check your migration

- OPCoach provides a form to help you to evaluate the work
- <http://www.opcoach.com/en/migration-evaluation/>

www.opcoach.com/en/migration-evaluation/

OPCOACH
ECLIPSE TRAINING AND CONSULTING

Home Courses Consulting Development References Jobs Blog Contact Us

Migration Evaluation Form

Home » Migration Evaluation Form

Migration Evaluation

This form will ask you some questions to help you to make your decision so as to migrate an Eclipse 3 application totally or partially to the Eclipse 4 development model.

I will analyze your description and give you an advice to migrate or not, totally or partially, your application to E4.

Step 1 of 5: Architecture
20%

Description of the project

Recent comments

- Coen Bijlsma on Testimonials
- Olaf Donk on Testimonials
- Frank van der Kruijsen on Testimonials
- Ralf Heydenreich on Managing preference pages with Eclipse 4
- OPCoach on Eclipse 4 workshop
- Vanina Savelli on Eclipse 4 workshop

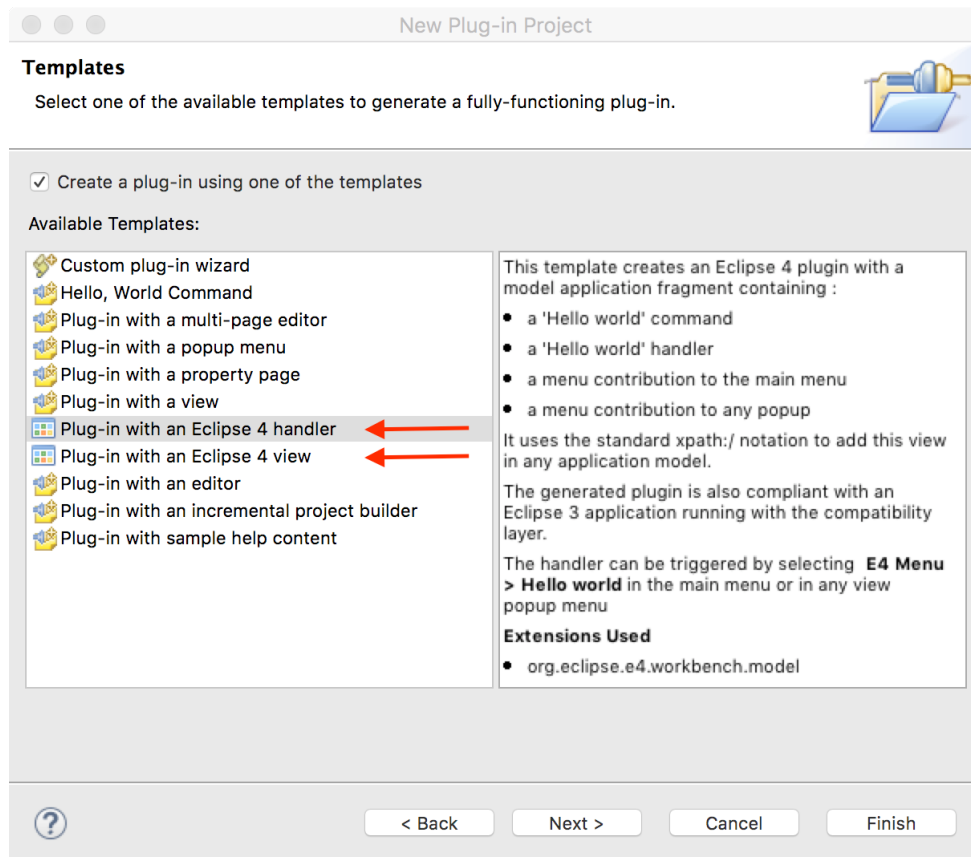
<http://www.opcoach.com/en/migration-evaluation/>

The plugin templates with model fragments

From Neon M4, it is now possible to create pure E4 plugins using model fragments.

- a plugin with a pure SWT E4 view

- a plugin with pure E4 handlers



C. Context concerns

Introduction

This presentation is not a course about injection.

It has already been presented in different talks and articles

Refer to this articles to be aware of this powerful mechanism

- Talk in Boston :
 - https://www.eclipsecon.org/2013/sites/eclipsecon.org.2013/files/E4_Injection_OPCoach_talk_0.pdf²
- Tutorial about injection :
 - <http://eclipsesource.com/blogs/tutorials/eclipse-4-e4-tutorial-part-4-dependency-injection-basics/>³
- Eclipse 4 context usage :
 - <http://www.vogella.com/tutorials/Eclipse4ContextUsage/article.html>⁴
- Eclipse Wiki

2 - https://www.eclipsecon.org/2013/sites/eclipsecon.org.2013/files/E4_Injection_OPCoach_talk_0.pdf

3 - <http://eclipsesource.com/blogs/tutorials/eclipse-4-e4-tutorial-part-4-dependency-injection-basics/>

4 - <http://www.vogella.com/tutorials/Eclipse4ContextUsage/article.html>

- https://wiki.eclipse.org/Eclipse4/RCP/Dependency_Injection⁵

E4 Injection

Principles of E4 injection

- The injector gathers hierarchically all common objects
- Listeners and initializations are simplified :
 - Methods annotated with **@Inject** are called automatically if a parameter changes in the context
 - Fields annotated with **@Inject** are automatically initialized if the value changes in the context
- Allows to have a framework independant of an external library (UI Agnostic)
- Simplify unit tests
- Example for the selection management :

Usage of injection for the selection

Just receive the selection object in the expected type and you will be notified !

```

11
12      /** This method will be invoked only if current selection is a Rental instance */
13      @Inject @Optional
14      public void receiveSelection(@Named(IServiceConstants.ACTIVE_SELECTION) Rental r)
15      {
16          setRental(r);
17      }
18

```

Get the selection



Attention: Object instantiation

A class containing injection annotations:

- must be instantiated using the **ContextInjectionFactory**
- **can not be instantiated with a call to new**

```

@Inject
public void defineMyService(IEclipseContext context)
{
    // Create an instance of MyService and inject its values
    MyService ms = ContextInjectionFactory.make(MyService.class, context);

    // Set the value in the context
    context.set(MyService.class, ms);
}

```

All POJOs bound in the application model are injected.



Attention: Memory management

- the POJOs instantiated and injected are deallocated when they are unused
- But, if you instantiate an object with make, YOU MUST uninject it somewhere !

5 - https://wiki.eclipse.org/Eclipse4/RCP/Dependency_Injection

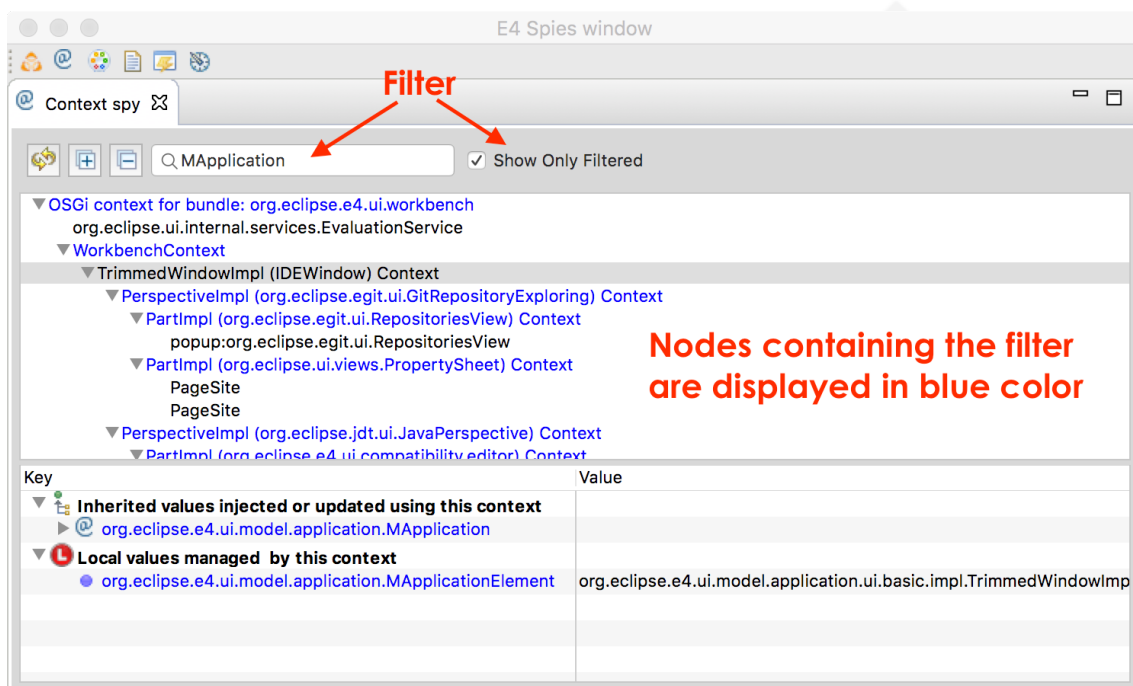
```
// Object containing @Inject annotations, created with make
MyService serv = ContextInjectionFactory.make(MyService.class, ctx);

// Somewhere in the code, don't forget to uninject it !
ContextInjectionFactory.uninject(serv, ctx);
```

Uninject

The context spy to explore your contexts

Open the spy window with the shortcut **Alt Shift F10** :



E3 and E4 context sharing

During a migration different use cases are possible :

- you define a value in E3 code and you want to publish it for E4 code
- you define a value in E4 code and you need to reuse it in E3 code

Sharing data from you E3 code

This use case happens when :

- you have still E3 plugins that are not migrated or will not migrate
- these plugins defines instances that should be stored in the context

It is possible to get the different E4 contexts:

- the OSGi context (this is the root context)
- the Application context

- the window context

Use this code to fill the context from E3 code :

```
public void getContextFromE3Code()
{
    //.....
    // Get the OSGI global E4 context :
    //.....
    Bundle e4Bundle = Platform.getBundle("org.eclipse.e4.ui.workbench");
    if (e4Bundle != null)
    {
        BundleContext e4BundleContext = e4Bundle.getBundleContext();
        IEclipseContext osgiCtx = EclipseContextFactory.getServiceContext(e4BundleContext);
        osgiCtx.set("myKeyInOsgi", "value");
    }

    //.....
    // Get the application E4 context
    //.....
    IWorkbench workbench = PlatformUI.getWorkbench();
    IEclipseContext appliCtx = workbench.getService(IEclipseContext.class);
    appliCtx.set("myKeyInAppli", "value");

    //.....
    // Get the main window E4 context
    //.....
    IEclipseContext windowCtx = workbench.getActiveWorkbenchWindow().getService(IEclipseContext.class);
    windowCtx.set("myKeyInWindow", "value");
}
```

Getting data from your E3 code in E4 code

Just inject it as any other E4 value !

Putting E4 data in context and reuse it in E3 code

In this use case, E4 code fills the context like usual

The E3 code can :

- extract the value from the context (get it with previous code)
- be notified automatically only if the E3 class instance has been created using **ContextInjectionFactory.make**

Manage the injected selection in a E3/E4 compliant code

In mixed mode, the selection can have different types :

- from the E3 code it is still an **ISelection**
- from the E4 code it is directly the selected type

Be aware to receive the both types in the E4 code

A full example is provided in the : 'E4 plugin template with a view'.

D. Model Fragments and Processors

Introduction

You can contribute to an application model by using two mechanisms :

- a **model fragment** : with the ID or xpath of model objects

- a **processor** : with a piece of code modifying the injected application

Model fragment

- The model fragment adds content to an existing application model
- To create a model fragment,
 - use the model fragment wizard (Ctrl N + fragment)
 - extract a piece of model into a fragment (contextual menu on application model editor)

Application fragment

It is possible to add any contribution to any object

- just select the ID of the object
- then select the feature to be populated
- then add a content

If you contribute on the top level application, you can use:

- the ID of the application
- the ID of the legacy E4 application : [org.eclipse.e4.legacy.ide.application](#)
- the 'xpath:/' to get any application whatever its ID (see bug #437958)
- This is the best practice for the top level contributions

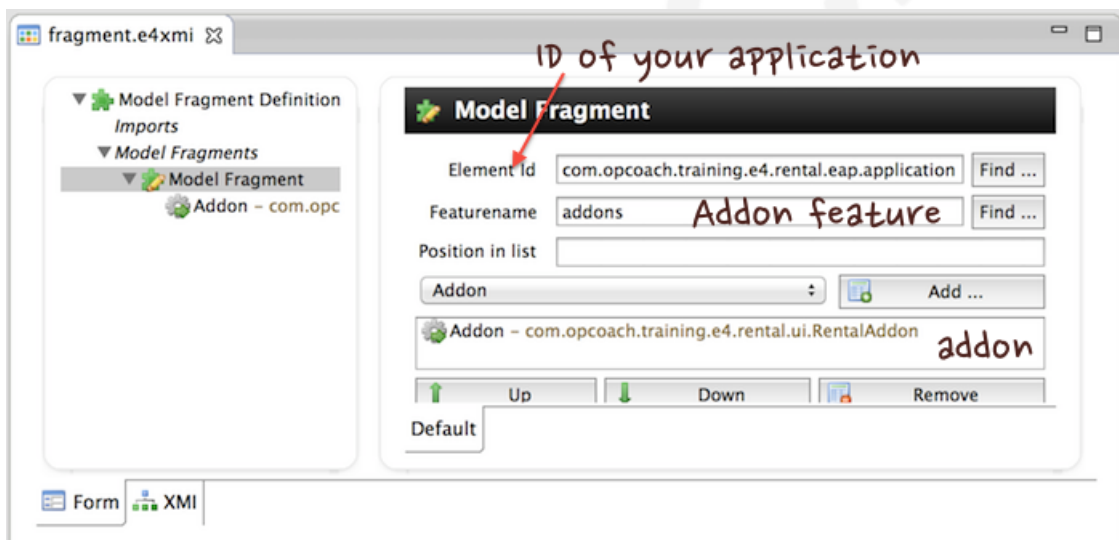


Image 2 Addon in fragment

Model fragment

Don't forget to declare the fragment in an extension ([org.eclipse.e4.workbench.model](#))

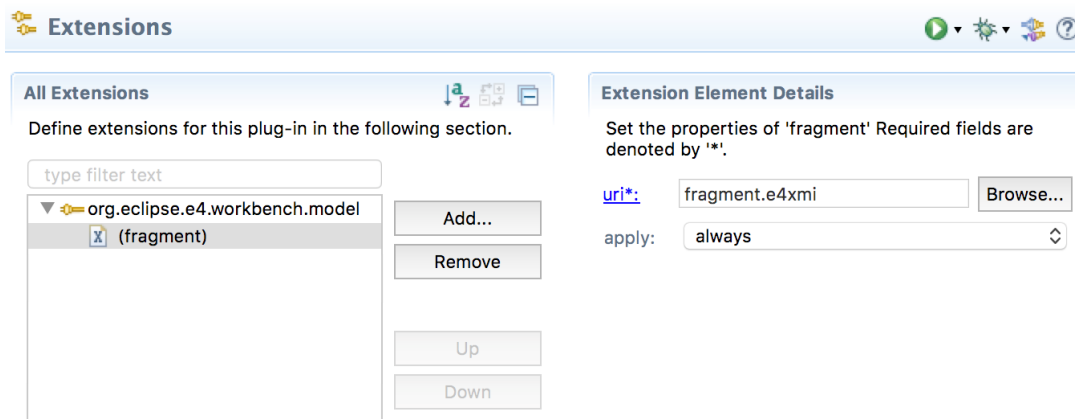
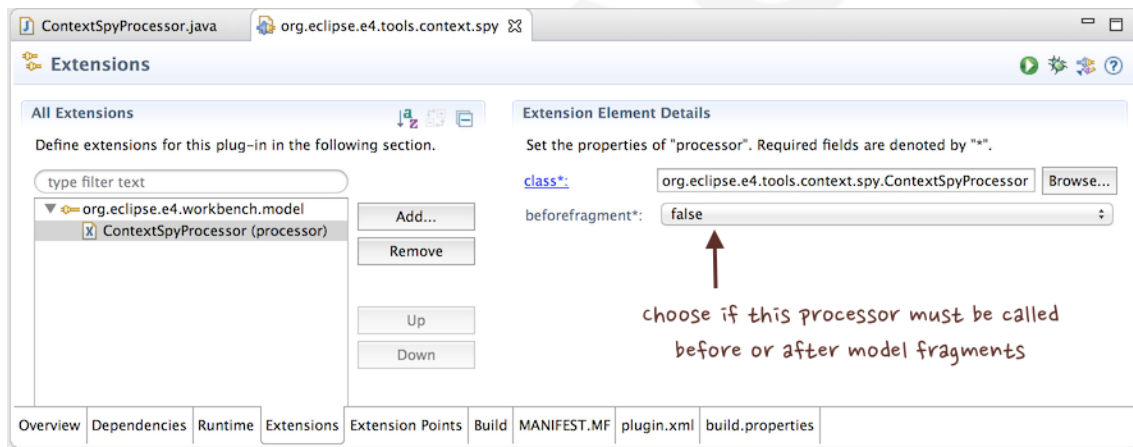


Image 3 Model Fragment

Processor declaration

- The processor is used when the object's ID is not known (application for instance)
- The application is received using injection so as to be modified
- It must be declared in the **org.eclipse.e4.workbench.model** extension using a processor parameter :



Extension for a processor

Processor code

- The processor code is a POJO with a **@Execute** annotation
- The method receives the application and needed services as fields or parameters
- Use the **modelService.createElement** method to create instances

```

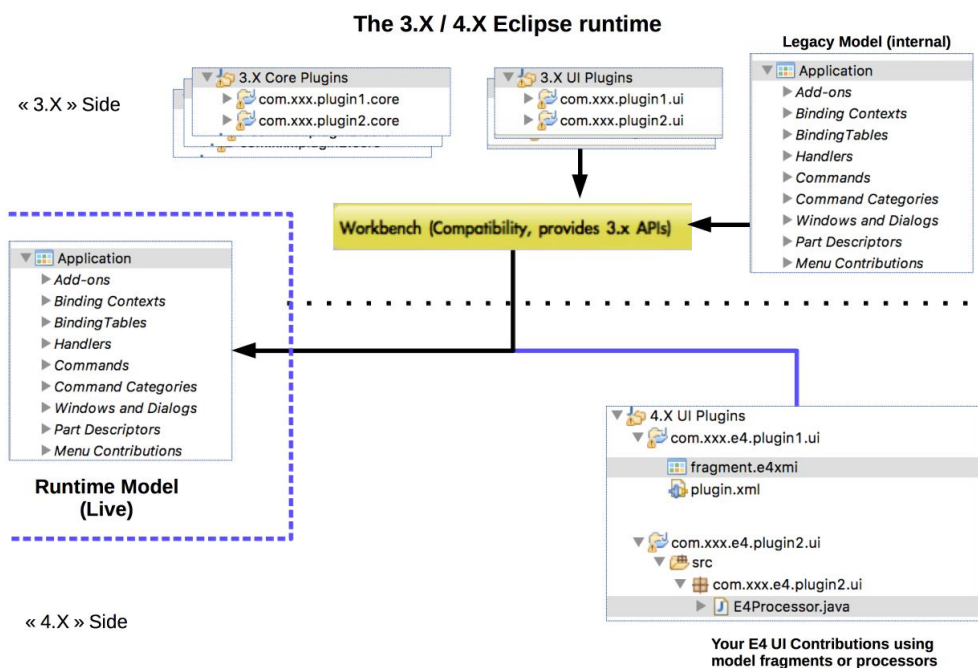
20
21 /** A sample E4 processor adding a command in application */
22 public class SampleE4Processor
23 {
24
25     @Execute
26     public void process(MApplication application, EModelService modelService)
27     {
28         // Just create a command and add it in the application
29         MCommand command = modelService.createModelElement(MCommand.class);
30         command.setElementId("id.of.my.command");
31         command.setCommandName("Launch My Command");
32         String contributorURI = "platform:/plugin/" + FrameworkUtil.getBundle(getClass()).getSymbolicName();
33         command.setContributorURI(contributorURI);
34         command.setDescription("A sample command added in application");
35
36         application.getCommands().add(command);
37     }
38 }
39
40

```

Code for a processor

Fragments / Processors and Migration

Remind that fragments and processors are the key mechanism to do the migration :



E. Extensions Migration

Content

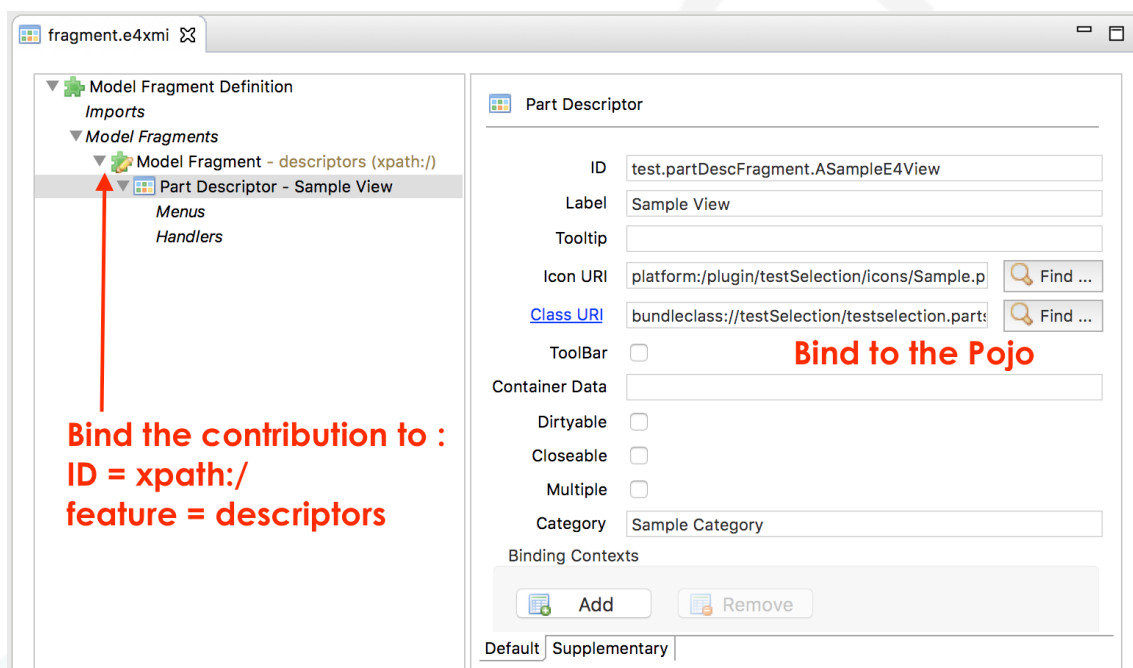
- This part will give some advices to migrate the main **org.eclipse.ui** extensions
- To find how to migrate an element, you can launch your application using the model spy and check what the compatibility layer has generated in the model.

View migration

An **org.eclipse.ui.views** extension is actually a **PartDescriptor** in the application model

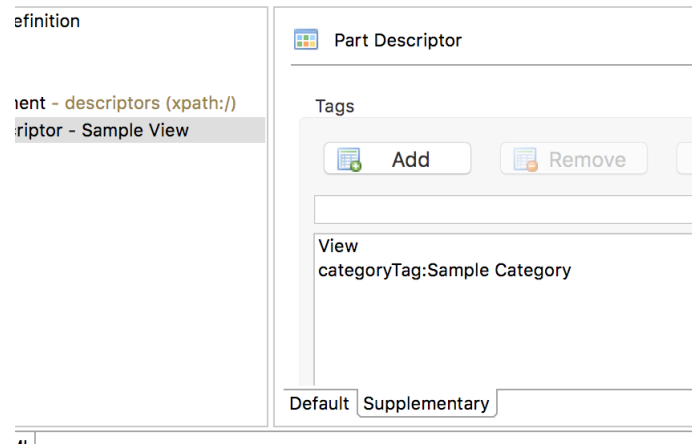
To migrate a view :

- Copy your ViewPart code in the xxx.e4.parts package
- Transform the code into a POJO :
 - remove inheritance to ViewPart
 - add **@PostConstruct** before the **createPartControl** method
 - add **@Focus** before the **setFocus** method
 - update the code to manage the selection using injection
 - remove the extension and the E3 code
- Bind this pojo in a model fragment :



To make the view appear in the 'Window -> Show view' menu :

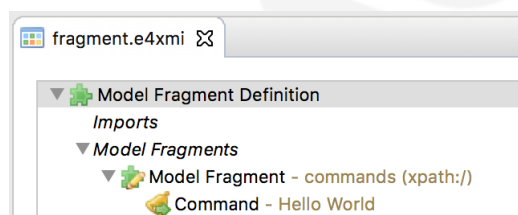
- add this tags in the supplementary tab



Command Migration

An **org.eclipse.ui.command** extension can be defined in the 'commands' feature of the application model

- keep the same ID
- add the command in the fragment :



Handler Migration

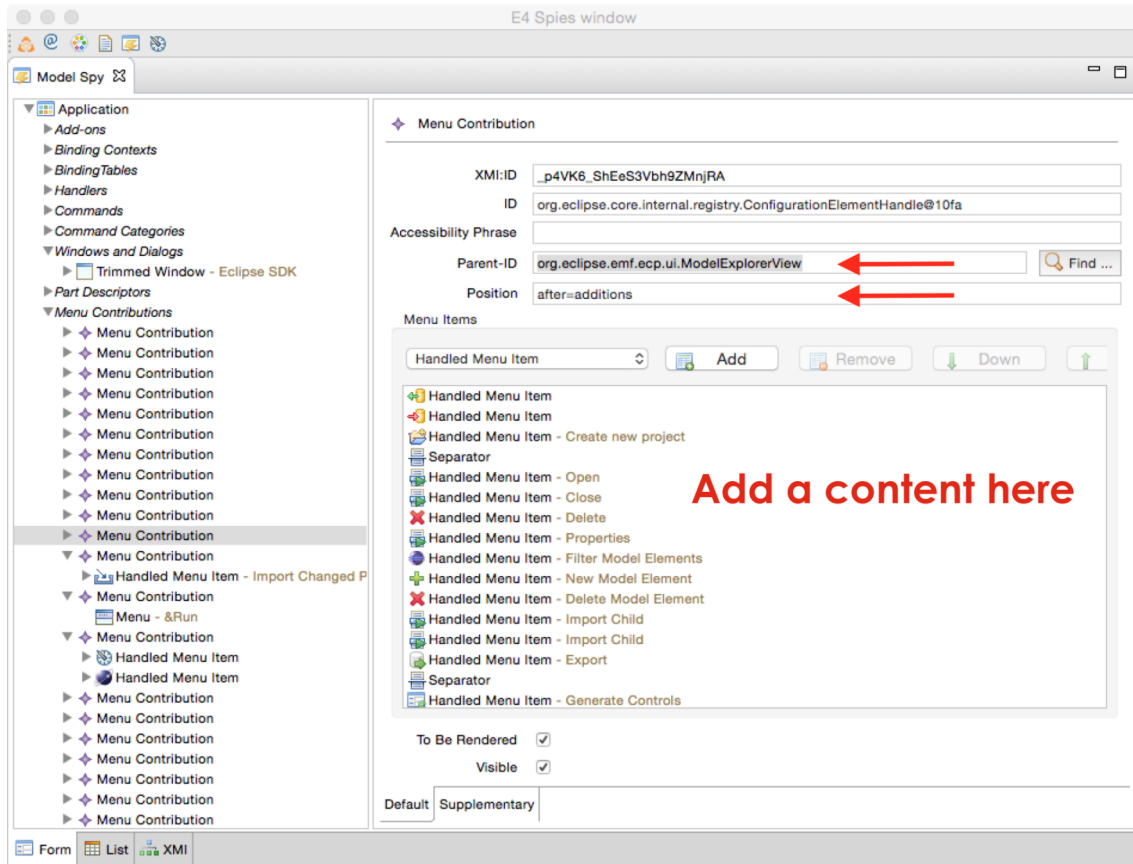
To migrate an **org.eclipse.ui.handlers** extension :

- Copy the E3 handler code in the xxx.e4.handlers package
- Transform the code into a POJO :
 - remove inheritance to AbstractHandler
 - add **@Execute** before the **execute** method
 - add **@CanExecute** annotated method if needed
 - receive needed values as parameters (will be injected)
- Bind this pojo in a model fragment (**xpath:/** and **handlers**)

MenuContribution Migration

An **org.eclipse.ui.menus** extension must be redefined in the model fragment

- use '**xpath:/**' and '**menuContributions**' feature
- The link is done using the parent ID



Menu Contribution

MenuContribution / Parameters

The following parent ID can be used :

- ID of an existing view (it must have been registered using the **EMenuService**)
- ID of an existing menu
- **org.eclipse.ui.main.menu** : used for the main menu
- **popup** : used to be located in any part
- **org.eclipse.ui.main.toolbar** : used to be located in the main toolbar.

For the position :

- an ID of any existing object (command, menu, etc...)
- **after=additions** : the default location

It is possible to open the model Spy so as to check the values used by the IDE

Wizard migration

- **org.eclipse.ui.[???]Wizards**
- Wizards are not defined in the application model
- There is also no extension point outside of org.eclipse.ui
- Therefore, the main dialog to choose a wizard is not available in a pure E4 application
- Nevertheless it is possible to open a specific wizard in a pure E4 code
- Wizards are only JFace code and can be adapted to deal with injected selection

- They must not implement **INewWizard**, **IImportWizard** or **IExportWizard** anymore
- A command must be created to open the wizard, using the **WizardDialog** of Jface

Sample wizard

```

1 package com.opcoach.training.e4.codesamples;
2
3 import javax.inject.Inject;
4
5 import org.eclipse.e4.core.contexts.ContextInjectionFactory;
6 import org.eclipse.e4.core.contexts.IEclipseContext;
7 import org.eclipse.jface.wizard.Wizard;
8
9 public class SampleWizard extends Wizard
10 {
11     private SampleWizardPage firstPage = null;
12     private IEclipseContext context;
13
14     @Inject
15     public SampleWizard(IEclipseContext ctx)
16     {
17         setWindowTitle("New Wizard");
18         context = ctx;
19     }
20
21     @Override
22     public void addPages()
23     {
24         firstPage = ContextInjectionFactory.make(SampleWizardPage.class, context);
25         addPage(firstPage);
26     }
27
28     @Override
29     public boolean performFinish()
30     {
31         // Do your stuff here by asking the pages...
32         return true;
33     }
34 }
35
36 }
37

```

can use the context
to create the pages

Sample wizard

Sample wizard page

```

1 package com.opcoach.training.e4.codesamples;
2
3 import java.io.File;
4
13
14 public class SampleWizardPage extends WizardPage
15 {
16     private Object selection;
17     private Label filename;
18
19     @Inject
20     public SampleWizardPage(@Named(IServiceConstants.ACTIVE_SELECTION) Object currentSelection)
21     {
22         super("wizardPage");
23         setTitle("Wizard Page title");
24         setDescription("Wizard Page description");
25         selection = currentSelection;
26     }
27
28     @Override
29     public void createControl(Composite parent)
30     {
31         Composite container = new Composite(parent, SWT.NULL);
32
33         filename = new Label(container, SWT.BORDER);
34         if (selection instanceof File)
35             filename.setText(((File)selection).getName());
36
37         setControl(container);
38     }
39
40     @Override
41     public boolean isPageComplete()
42     {
43         return filename.getText().length() > 0;
44     }
45
46 }
47

```

Inject current selection for content init

Sample wizard page

Opening wizard

```

11 public class OpenSampleWizard {
12     @Execute
13     public void execute(IEclipseContext ctx, Shell s)
14     {
15         Wizard w = ContextInjectionFactory.make(MyWizard.class, ctx);
16         WizardDialog wd = new WizardDialog(s, w);
17         wd.open();
18     }
19 }
20

```

Open wizard

Preference pages Migration

- Like wizards, preference pages are not defined in the application model
- It is possible to use the plugin : <https://github.com/opcoach/e4Preferences>⁶

6 - <https://github.com/opcoach/e4Preferences>

- You need to :
 - ensure that your preference pages are extending `FieldEditorPreferencePage`
 - change the extension `org.eclipse.ui.preferencePages` to `com.opcoach.e4.preferences.e4PreferencePages`
 - add the handler and the command in your model

For the default values, you can keep the `org.eclipse.core.runtime.preferences` extensions.

Other migrations

- There are still plenty tips for your migration
- Try to put it in your model fragment
- If you can not describe your contribution in a model fragment, use a model processor

F. Resources

Articles about migration

- Eclipse magazin about migration (german) :
 - <https://jaxenter.de/ausgaben/eclipse-magazin-6-15>⁷
- Recipes for your Eclipse 4 migration (english)
 - will be published on jaxenter.com
- OPCoach's article in eclipse magazin (german)
 - http://www.opcoach.com/wp-content/uploads/2015/09/Migration34_EclipseMagazine_final.pdf⁸
- Comment migrer vers eclipse 4 (french)
 - <http://opcoach.developpez.com/tutoriels/eclipse/migration-e3-e4>⁹

Ask your questions

Feel free to ask your questions

- in E4 forum
- using the form on the OPCoach's web site
- by email :
 - olivier@opcoach.com
- Now after this talk or during the conference !

7 - <https://jaxenter.de/ausgaben/eclipse-magazin-6-15>

8 - http://www.opcoach.com/wp-content/uploads/2015/09/Migration34_EclipseMagazine_final.pdf

9 - <http://opcoach.developpez.com/tutoriels/eclipse/migration-e3-e4>