



E4 Spies and tools

29 October 2014

OPCoach



OPCoach

- About me
 - **Olivier Prouvost**
 - Eclipse expert trainer and committer (E4 Tools)
 - olivier.prouvost@opcoach.com
 - [@OPCoach_Eclipse](#)
- About **OPCoach**
 - Company founded in June 2009
 - Member of the Eclipse Foundation (as Solution Member)
 - Web site : ¹<http://www.opcoach.com>
 - Provides Eclipse training and consulting

1 - <http://www.opcoach.com>

➤ Some references :



Image 1

E4 Spies



Agenda

- A sample application
- Using the spies
 - model spy
 - context spy
 - event spy
 - css spy
- Coding new spies :
 - bundle spy
- Other tools
 - The migration E3->E4 statistics view

Sample application

To have interesting cases to use the spies, we will use this application :

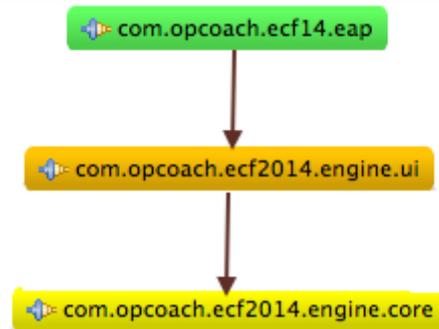
The screenshot shows a Java Swing application window titled "com.opcoach.ecf14.eap". The interface is divided into three main sections:

- Dashboard Part:** Contains two circular gauges. The left gauge is labeled "Rpm" and has a scale from 0 to 8000 with major ticks at 2000, 4000, 6000, and 8000. The right gauge is labeled "Speed" and has a scale from 0 to 220 with major ticks every 20 units. Both gauges feature a multi-colored arc (green, yellow, red) and a needle pointing to zero.
- EngineControl Part:** Features an "Acceleration slider" with a scale from -10 to 10. The slider has four vertical markers labeled "LOLO", "LO", "HI", and "HIHI" in red. Below the slider is a "Start/Stop Engine" button, which is a green circular button with the word "Start" in white.
- Alarm Part:** Contains a table with three columns: "Date", "Hour", and "What happened?". The table has four empty rows. Below the table is a "Clear alarms" button.

Application overview

Application architecture and location

This application is built using this architecture :



Architecture

- **ecf14.eap** plug-in contains :
 - the Application model
- **ecf2014.engine.ui** plug-in contains :
 - the 3 parts : Dashboard, Engine Control, Alarm Viewer
- **ecf2014.engine.core** plug-in contains :
 - the EngineSimulator
 - the Alarm model
 - the EngineLogger
 - the EngineWatcher
 - A context function to create the logger

Application Links

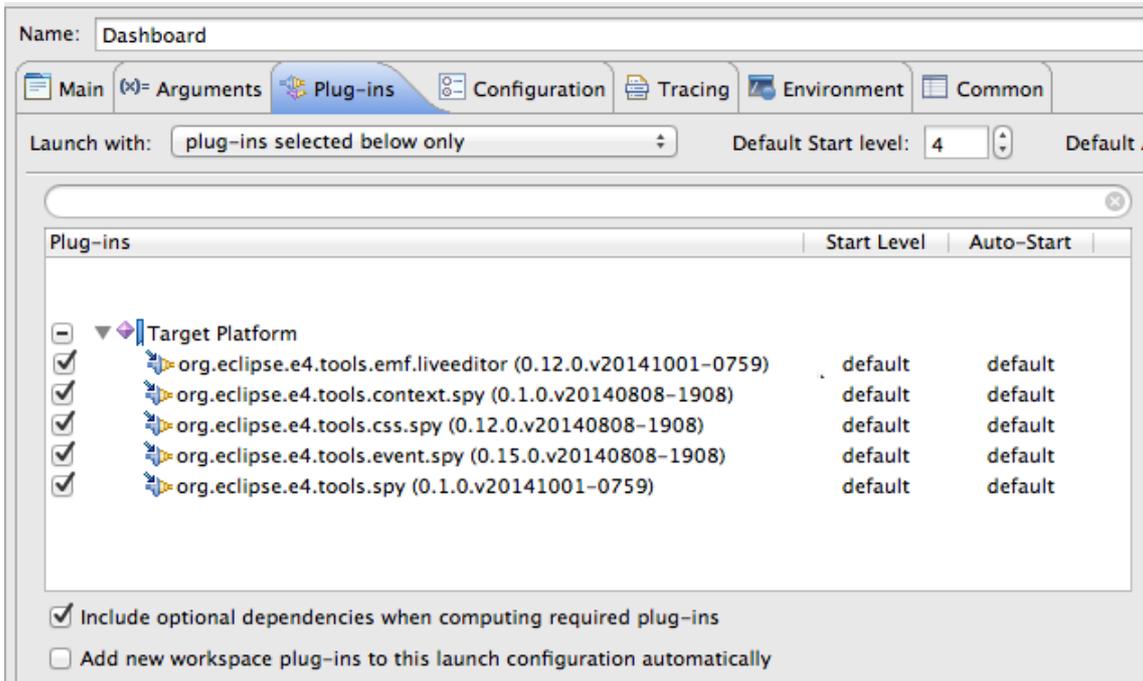
- This application can be downloaded from github :
 - <https://github.com/opcoach/Conferences>
 - Get it from the **ECF14** folder.
- The 'howto' make this application was explained during a workshop at Eclipse Con France 2014
 - get the pdf and explanations here : http://www.opcoach.com/en/eclipse4_workshop/

Spies concept

- Spies display dynamic information from a developer point of view
- They are not delivered to the end user
- E4 introduces new concepts like application model, injection, event, css...
- Each concept has its own spy
- You can add you own spies for you specific concepts (IOT, statistics, encode/decode operations...)

Launching your application

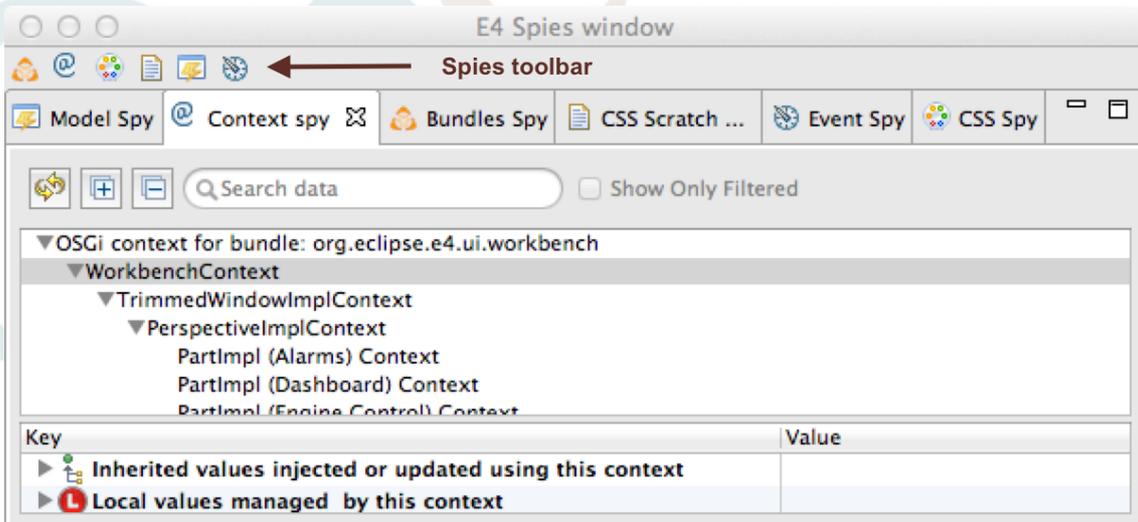
To have the spies in your application, you must set your launch configuration with :



Spy Launch Config

The E4 Spies window

- All the spies will open in the E4 spy window
- But you can move them in another window after

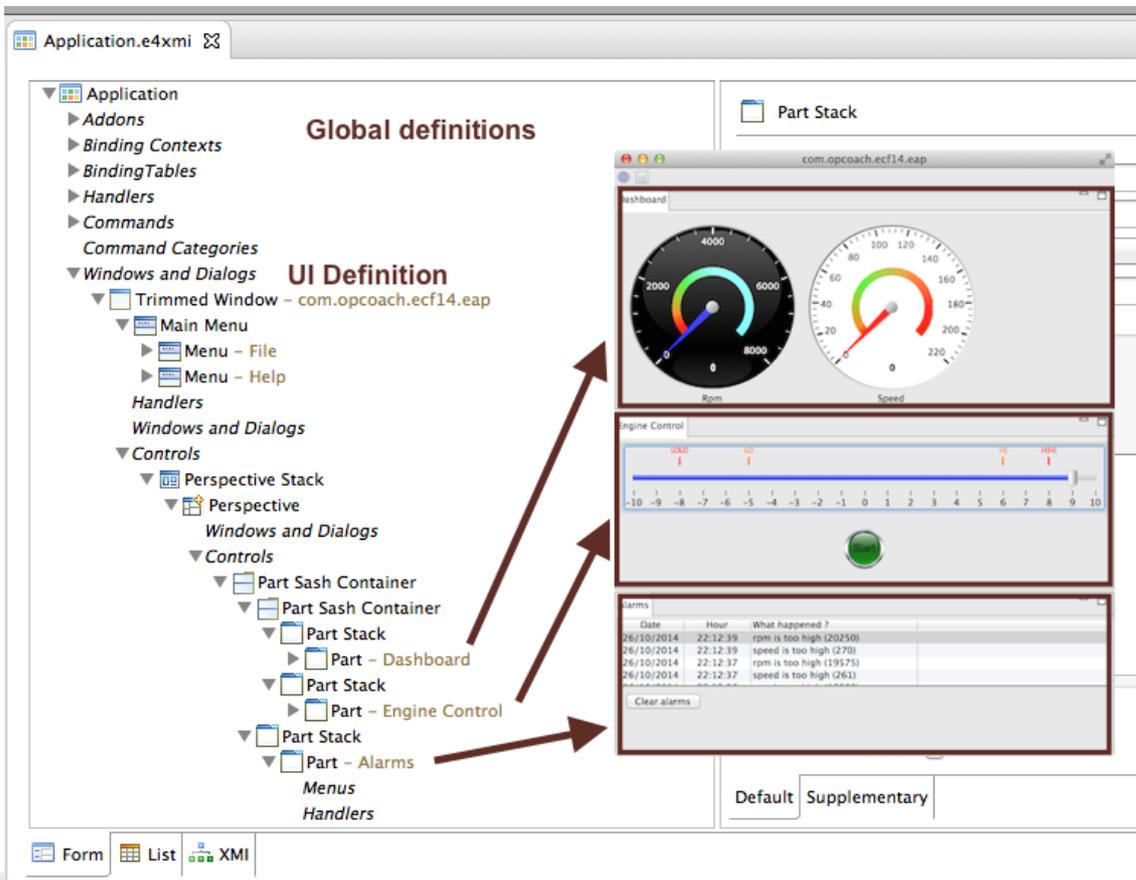


E4 spies window

Concept 1 : Application model

- The application model describes the content of the application
- It is a UI agnostic skeleton
- It is bound to Pojo classes (parts, handlers, ...)
- It is rendered by a specific renderer depending on the content of the Pojo (JavaFx, Swt..)

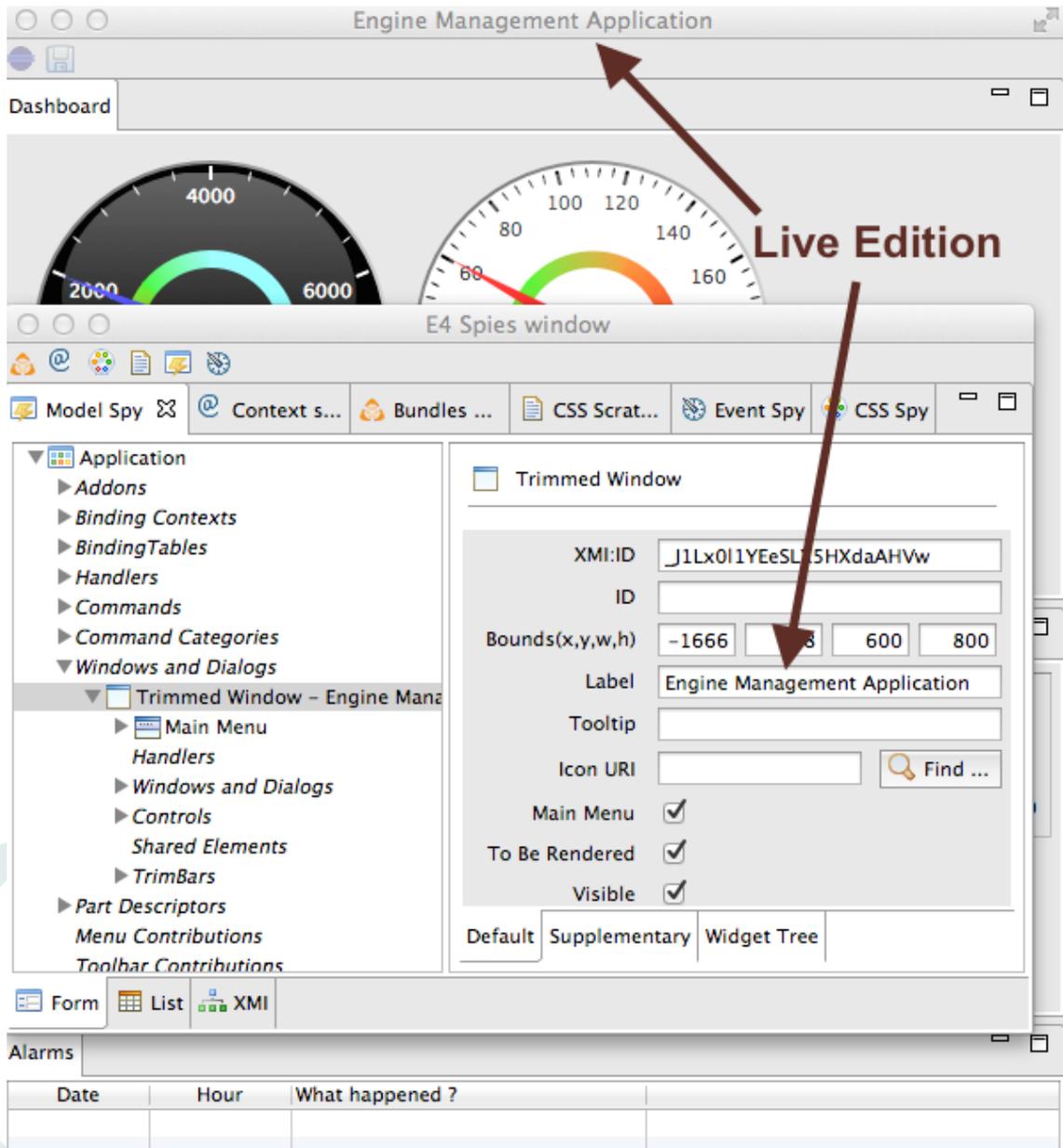
- This model is read at runtime and can be modified on live



Application model

Model Spy

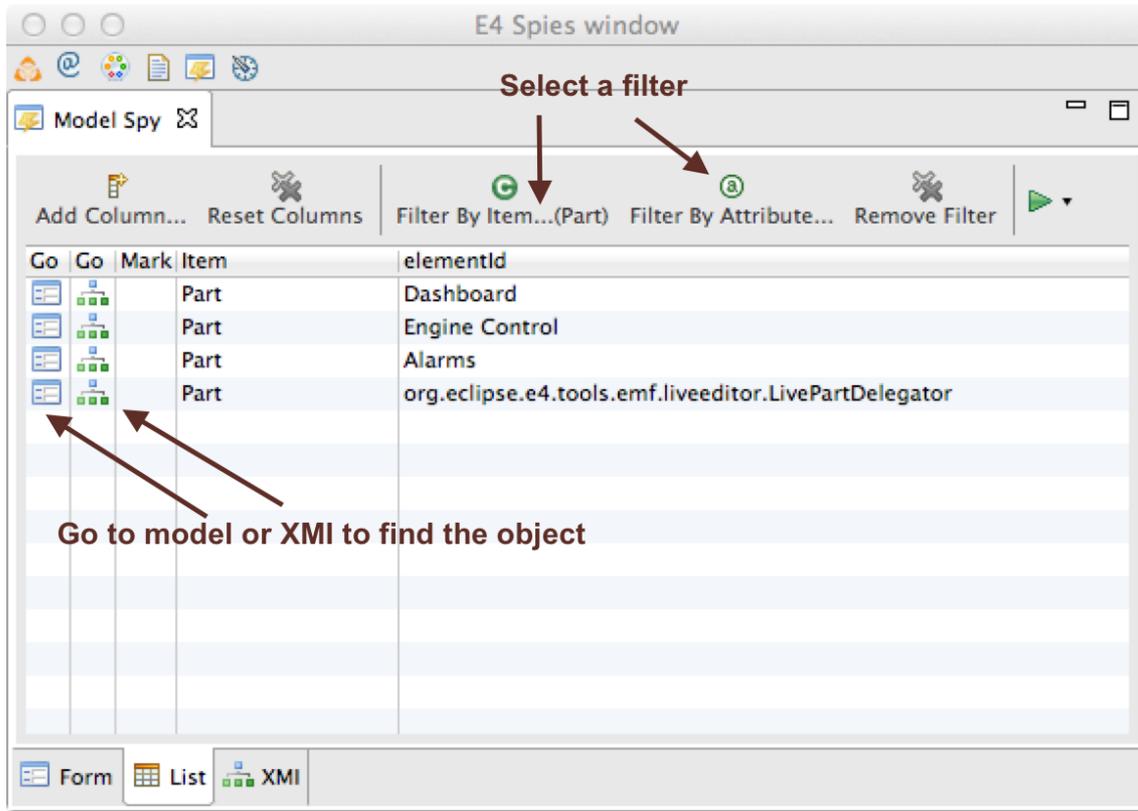
You can open it with the shortcut : Alt Shift F9
It displays the model and you can change it live



Model Spy

Model Spy, search

- The 'List' tab allows you to search an element
- Select the filter and navigate in the model



Model Spy filters

Model spy in action

Demo

Concept 2 : Injection

- The goal of injection is to delegate the field or parameters initializations to a framework
- Injection uses a context containing the values
- Use the annotation `@Inject` (javax.inject) to inject the values
- It can be applied to a constructor, a method or a field.
- In E4, the context is a tree of maps.

The context of this application

An E4 application can use the context to store its own data :

Key	Value
Inherited values injected or updated using this context	
Local values managed by this context	
com.opcoach.ecf2014.engine.core.EngineSimulator	com.opcoach.ecf2014.engine.core.EngineSimulator@11ce2e22
com.opcoach.ecf2014.engine.core.EngineWatcher	com.opcoach.ecf2014.engine.core.EngineWatcher@6826c41e
com.opcoach.ecf2014.engine.core.IEngineLogger	com.opcoach.ecf2014.engine.core.impl.DefaultEngineLogger@3051e0b2
@ engine.rpmValue	5625
@ engine.speedValue	75

The engine data stored in context

Engine data in the main context

Setting data using a class key

```

41
42 @Inject
43 public DashBoard(MApplication appli)
44 {
45     // We will use the application context to store and inject values.
46     IEclipseContext appliContext = appli.getContext(); Get application context
47
48     // We also need an ImageRegistry for the application
49     appliContext.set(ImageRegistry.class, new ImageRegistry()); Use class as key
50
51     // Step 5 : create and start Engine.
52     EngineSimulator simu = ContextInjectionFactory.make(EngineSimulator.class, appliContext);
53     appliContext.set(EngineSimulator.class, simu); Call make
54
55     // Step 8 : create the engine alarm watcher and keep a reference on it !
56     EngineWatcher watcher = ContextInjectionFactory.make(EngineWatcher.class, appliContext);
57     appliContext.set(EngineWatcher.class, watcher);
58 }
59

```

Use a class as a key

Setting data using a name

```

57 @Inject Inject context
58 IEclipseContext ctx; // The context where values will be injected
59
60 public void stop()
61 {
62     if (timer != null)
63     {
64         timer.cancel();
65         speed = 0;
66         rpm = 0;
67         ctx.set(ENGINE_SPEED_VALUE, 0);
68         ctx.set(ENGINE_RPM_VALUE, 0); Setting a named value in context
69     }
70     timer = null;
71 }
72

```

Setting context with a named value

How to inject values from the context ?

```

59
60 @Inject @Optional
61 public void listenToRpmValue(final @Named(EngineSimulator.ENGINE_RPM_VALUE) int value, UISynchronize sync)
62 {
63     if (rpmCounter != null)
64     {
65         sync.asyncExec(new Runnable()
66         {
67             @Override public void run() Inject the value
68             {
69                 rpmCounter.setValue(value);
70             }
71         });
72     }
73 }
74

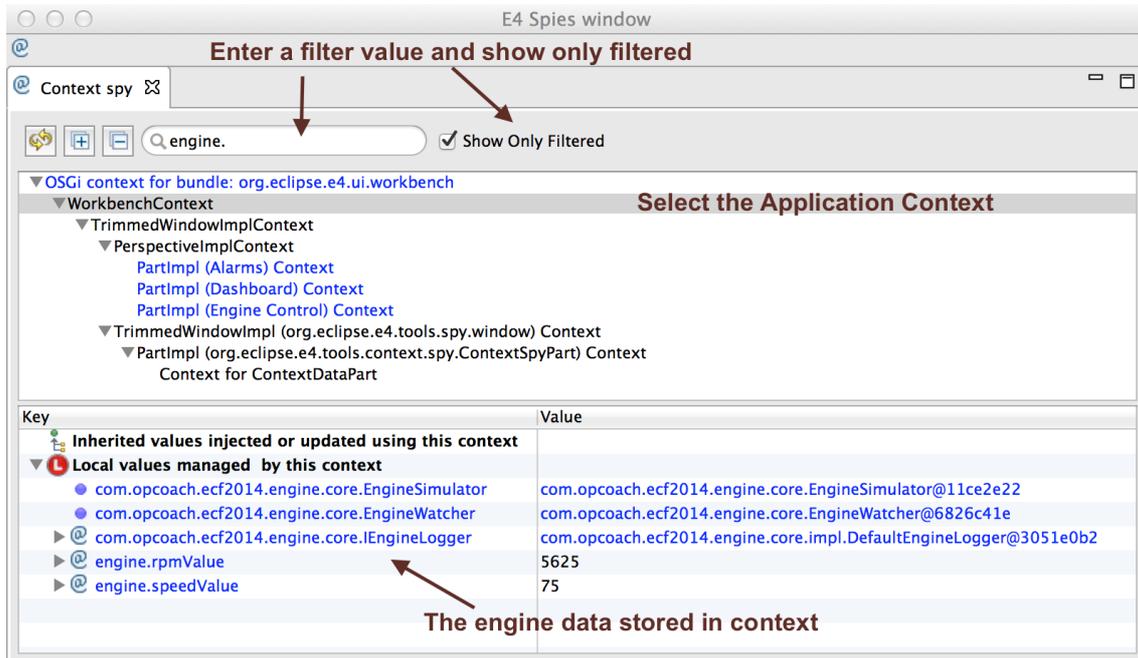
```

Will be invoked when value changes in context

Inject value

The context spy

- It can be opened with Alt Shift F10
- It displays the tree context and provide a filter to find data



Context content

Local values managed by this context ?

- This part of the tree contains for the selected context, all the values directly set in this context.
- Parent context can not see these values
- Only the current context and child context can access them

Inherited values injected or updated using this context ?

- This part of the tree displays only values defined in parent context(s)
- These values are injected using the current selected context
- It is possible to open it and to check where injection is used (method or field)
- The values injected with @PostConstruct are never displayed (because called once)

Context spy in action

Demo

Concept 3 : Event management

- A good framework must provide an event management mechanism
- Usually, to be notified of an event, a listener must be defined
- And for each case a specific method must be defined
- Example: if you want to listen to what is going on with xxx, we would have:
 - xxxListener with xxxCreated (xxxEvent), xxxModified (xxxEvent) ...
 - or you can use the EMF adapters.
- This is painful

IEventBroker

- In E4 there is a more simple mechanism: the **IEventBroker**.
- All events occurring on the Application model are sent (see **UIEvents** class)

- moving a window
- adding a part
- activating a part
- etc...
- You can define your own events

Sending events

```

9 public class EngineWatcher
10 {
11     // Define the sent topics          Define event keys
12     public static final String ALARM_TOPIC = "Alarm/*";
13     public static final String ALARM_RPM_TOO_HIGH = "Alarm/RpmTooHigh";
14     public static final String ALARM_SPEED_TOO_HIGH = "Alarm/SpeedTooHigh";
15
16     // Get the event broker by injection
17     @Inject
18     IEventBroker ebroker;    Receive the Event Broker using injection
19
20     @Optional
21     @Inject
22     public void checkRpmValue(final @Named(EngineSimulator.ENGINE_RPM_VALUE) int value)
23     {
24         // Inject the RPM value from context
25         if (value > 5000)
26         {
27             // Send an alarm
28             Alarm a = new Alarm("rpm is too high (" + value + ")", value);
29             ebroker.send(ALARM_RPM_TOO_HIGH, a);
30         }
31     }
32 }

```

Send the event

Sending events

Receiving events

```

118
119 @Inject @Optional
120 public void listenToAlarms(@UIEventTopic(EngineWatcher.ALARM_TOPIC) Alarm a)
121 {
122     // Receive event by injection
123     alarms.insertElementAt(a, 0);
124     if (viewer != null)
125     {
126         viewer.refresh();
127         viewer.setSelection(new StructuredSelection(a));
128     }
129 }

```

Receiving events

The event spy

- A specific spy receives all events and display their values.
- The Event spy can be opened with **Alt Shift F8**

Start or Stop capturing events

[Start capturing events](#) [Hide filters](#)

New filter:

Base topic of captured events: [Reset to default](#)

Capture event when: [Add filter](#)

Defined filters (relation between filters is AND):

Capture event when 'Topic' starts with 'Alarm'

Add your filters here [Update selected](#) [Remove selected](#) [Remove all](#)

Topic	Event publisher	Changed element
▼ Alarm/SpeedTooHigh		
org.eclipse.e4.data	Alarm : 2014.10.27 14:02:04 speed is too high (643) : 643	
event.topics	Alarm/SpeedTooHigh	
▶ Alarm/RpmTooHigh		
▶ Alarm/SpeedTooHigh		
▶ Alarm/RomTooHigh		

Open the capture and display event data

Event Spy

Event spy in action

Demo

Concept 4 : CSS

- It is possible to easily define a static CSS for an E4 application.

- Create a css file, and add it in the product parameters :

com.opcoach.ecf14.eap

Extensions

All Extensions

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

type filter text

- org.eclipse.core.runtime.products
 - com.opcoach.ecf14.eap (product)
 - applicationCSS (property)

Buttons: Add..., Remove

Extension Element Details

Set the properties of 'property' Required fields are denoted by '*'.

name*: applicationCSS

value*: platform:/plugin/com.opcoach.ecf14.eap/css/default.css

Overview | Dependencies | Runtime | **Extensions** | Extension Points | Build | MANIFEST

Using CSS

CSS Result

For the following CSS :

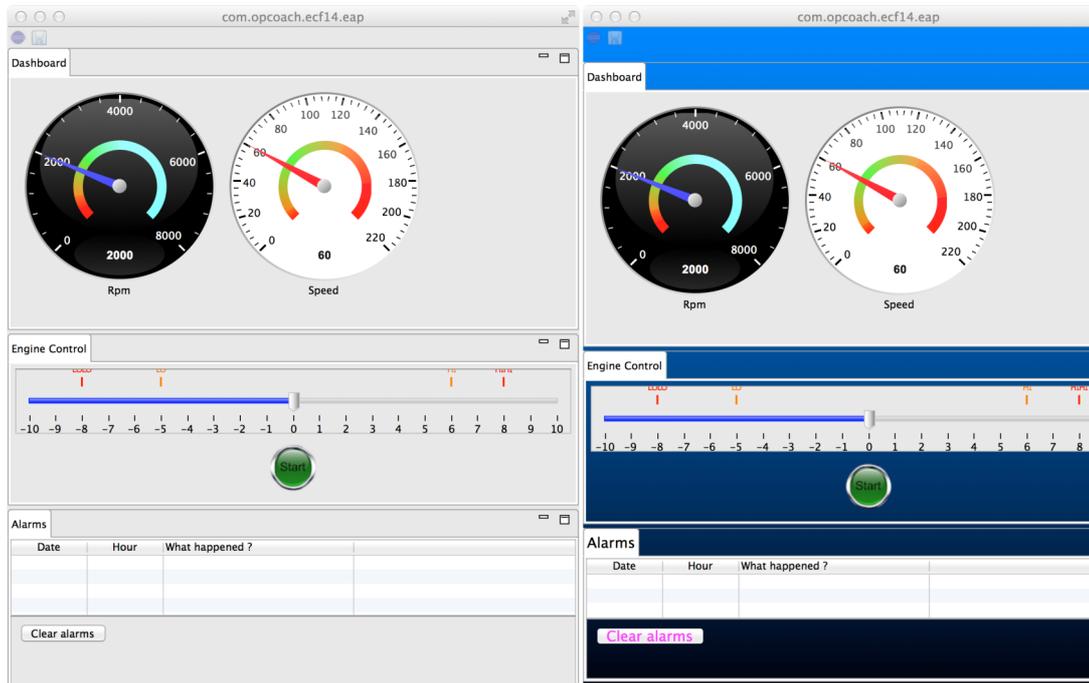
```

sample.css
1
2 Button {                               Use SWT class names as CSS classes
3   color: #FF00FF;
4   font-family: "Lucida Grande";
5   font-size: 15;
6   font-style: bold;
7 }
8
9 CTabFolder#com-opcoach-ecf14-eap-partstack-1 {
10   font-family: "Lucida Grande";        Can access to the specific widget
11   font-size: 15;                       of a rendered model element
12 }
13
14 .MTrimmedWindow.topLevel {
15   margin-top:15px;
16   margin-bottom:15px;
17   background-color: #08F #000 100%
18
19 }

```

CSS Sample

You will get :



Css effect

The CSS Spy

- How can you guess the content of your CSS ?
- Use the CSS Spy and the scratch pad to test.
- Open it with **Alt Shift F5** and the scratch pad with **Alt Shift F6**
- Click on a widget in your application and check the style

Select object in UI

Change a value

Get the CSS fragment

Widget

Widget	CSS Class	CSS Id
Shell (org.eclipse.swt.widgets)	MTrimmedWindow (+1 others)	
Composite (org.eclipse.swt.widgets)		
Composite (org.eclipse.swt.widgets)		
Composite (org.eclipse.swt.widgets)		
CTabFolder (org.eclipse.swt.widgets)		
ToolBar (org.eclipse.swt.widgets)		
Composite (org.eclipse.swt.widgets)		
ToolBar (org.eclipse.swt.widgets)		
ContributedPartRenderer-2		Dashboard
Canvas (org.eclipse.swt.widgets)		
CTabItem (org.eclipse.swt.widgets)		
CTabFolder (org.eclipse.swt.widgets)		com-opcoach-ecf14-eap-partstack-0

CSS

```
Canvas {
/* actual values */
background-color: #e8e8e8;
background-image: none;
color: #000000;
font-family: "Lucida Grande";
font-size: 11;
font-style: normal;
font-weight: normal;
swt-background-mode: none;
text-transform: none;
visibility: visible;
}
```

Escape to dismiss

CSS Properties

Property	Value
background	
background-color	#e8e8e8
background-image	none
color	#000000
cursor	
eclipse-perspective-keyline-...	
font	
font-adjust	

✓ Show unset properties

CSS Rules

SWT Style Bits:
SWT.LEFT_TO_RIGHT
SWT.DOUBLE_BUFFERED

CSS Class Element:
org.eclipse.e4.ui.css.swt.dom.CompositeElement

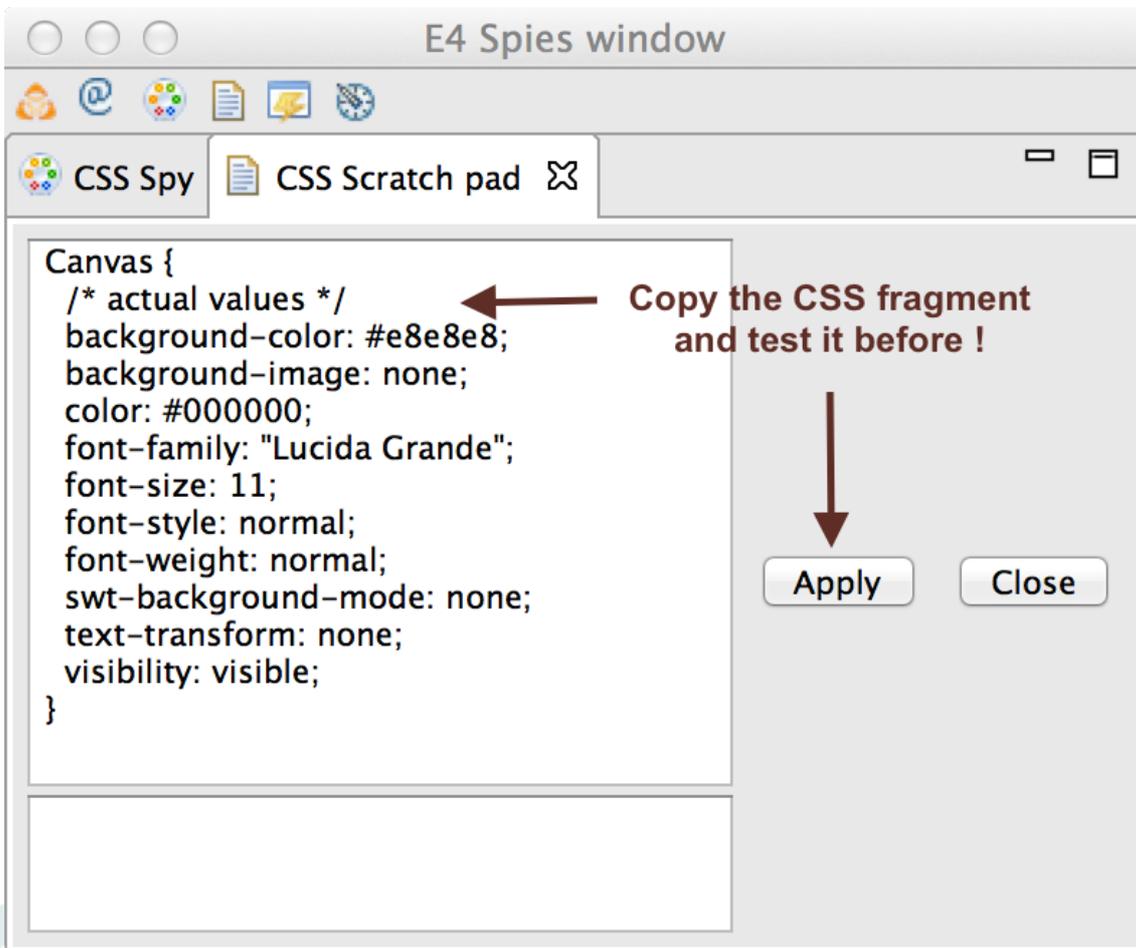
SWT Layout: null
Bounds: x=-320 y=98 h=234 w=594

Show CSS fragment

CSS Spy

The CSS Scratch pad

- Use it to test your CSS fragment on your application
- When it is ok, put it in your final CSS file
- CSS Scratchpad is fonctionnal if you add a theme engine
 - add an extension of [org.eclipse.e4.ui.css.swt.theme](#)



CSS scratch pad

CSS spy in action

Demo

Adding new spies

- It is very easy to add your own spy.
- Add a dependency to **org.eclipse.e4.tools.spy**
- Extend **org.eclipse.e4.tools.spy.spyPart**
- Your part is a pure E4 part (POJO, Injection...)
- The key binding is handled automatically

org.eclipse.e4.tools.bundle.spy

Extensions

All Extensions

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

type filter text

- org.eclipse.e4.tools.spy.spyPart
 - Bundles Spy (spyPart)

Add...
Remove

Extension Element Details

Set the properties of 'spyPart' Required fields are denoted by '*'.

description: Bundle Spy to display all bundles and their states

icon: icons/osgi.png

name: Bundles Spy

part: org.eclipse.e4.tools.bundles.spy.BundleSpyPart

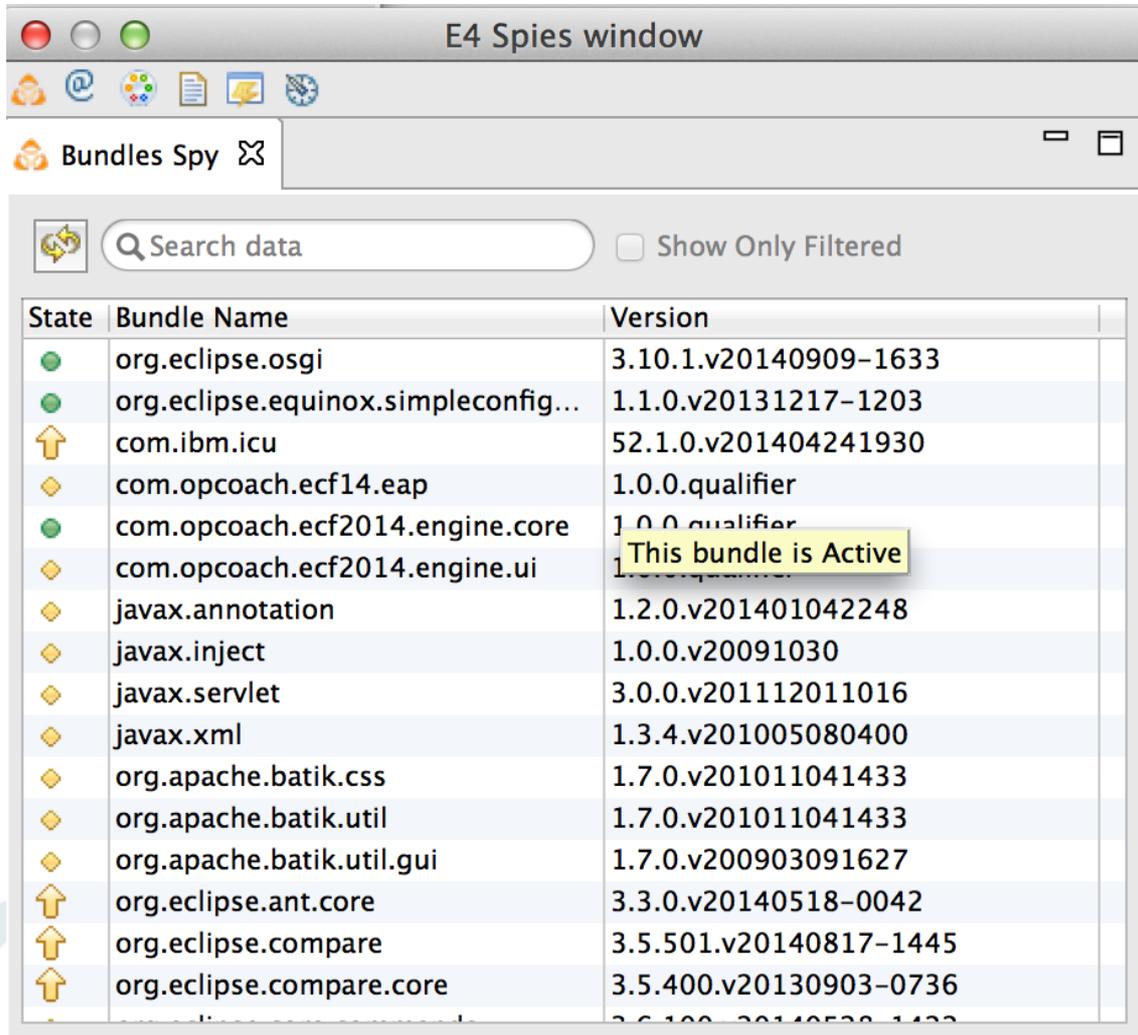
shortcut: M2+M3+F12

Overview | Dependencies | Runtime | Extensions | Extension Points | Build | MANIFEST.MF

Bundle spy definition

The bundle spy

- Is bound on **Alt Shift F12**
- It displays the bundle statuses like in the OSGi console
- Will add command start/stop later



Bundle Spy

Next steps for bundle spy

- It could be included in the E4 tools master branch
- For the moment Bundle spy is hosted on github :
 - <https://github.com/opcoach/Conferences/tree/master/ECE14>
- You can improve it !

Next steps for E4 spies

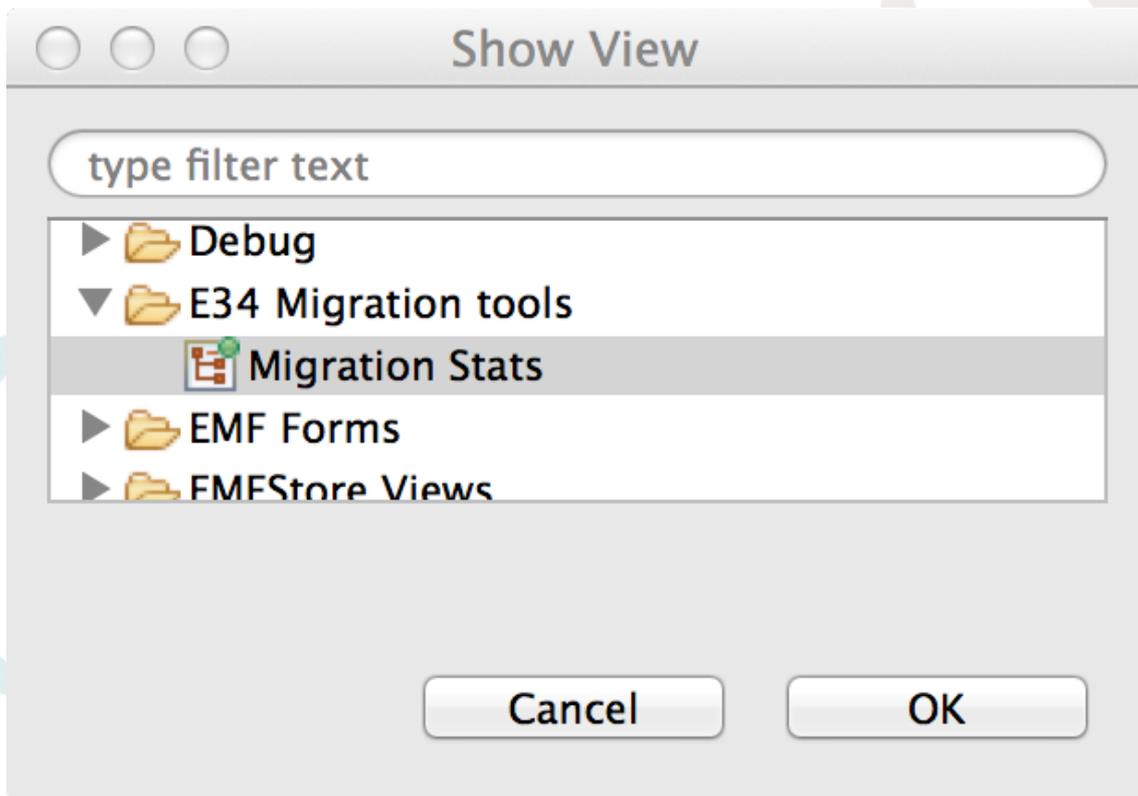
- Try it !
- Some ideas :
 - Memory spy
 - Objects suppliers (E4 concept)
 - OSGi service spy
 - Translation spy
 - System spy
 - Protocol spy (MQTT, HTTP, ...)
 - Encoding spy
 - Any spy specific to your business

E3E4 migration view

- If you want to migrate an E3 application it can be interesting to have an overview of the work to do.
- Use the E3E4 Statistics Migration view to get information
- It displays for each **org.eclipse.ui** extension point the number of extensions.
- It detects the deprecated attributes or extension points.
- This view could be improved but is already useful !

E3E4 migration view installation

- Get the project from github :
 - <https://github.com/opcoach/E34MigrationTooling>
- Launch an Eclipse instance with the project
- Import the '**org.eclipse.ui**' plug-in in your workspace (very important)
 - This plugin must contain the extension points schemas
- Open the Migration View :



Show view

- Import the projects you want to analyze
- Select a plugin, several plugins, or a feature
- Export the result as CSV for your migration stats.

Migration view content :

Select your project(s)

Get the stats

Use filter and export to CSV file

Extension Points	test.plugin	ui.editors	emf.ecp.core	emf.ecp.edit	emf.ecp.core.emfilter	Count
org.eclipse.ui.acceleratorScopes	2	0	0	0	0	2
acceleratorScope	2	0	0	0	0	2
org.eclipse.ui.actionSetPartAssociations	0	1	0	0	0	1
actionSetPartAssociation	0	1	0	0	0	1
org.eclipse.ui.actionSets	1	1	0	0	0	2
actionSet	1	3	0	0	0	4
org.eclipse.ui.bindings	0	1	0	0	0	1
key	0	7	0	0	0	7
org.eclipse.ui.commands	0	1	0	0	0	1
command	0	18	0	0	0	18
org.eclipse.ui.editorActions	0	1	0	0	0	1
editorContribution	0	1	0	0	0	1
org.eclipse.ui.editors	0	1	0	0	0	1
editor	0	1	0	0	0	1
org.eclipse.ui.handlers	0	1	0	0	0	1

Migration view

Migration Statistic view

Demo

Next step for migration view

- Import automatically the org.eclipse.ui with schemas or use it internally
- Convert count to time
- Compute some risk metrics

Questions

- Thank you for attending !
- Don't forget to evaluate this talk



- You can download the pdf of this presentation on the eclipse con web site
- Keep in touch :
 - olivier.prouvost@opcoach.com
 - <http://www.opcoach.com>