



## **E4 Tester**





# E4 Tester



## *E4 Tester*

---



## *Who ?*

---

### **Olivier Prouvost :**

- OPCoach ([www.opcoach.com](http://www.opcoach.com))
- Eclipse expert since 2004
- Provide training and consulting on Eclipse technologies
- France / Europe ... or even further
- Committer on e4 tools and platform UI
- [olivier@opcoach.com](mailto:olivier@opcoach.com)
- Twitter : [@OPCoach\\_Eclipse](https://twitter.com/OPCoach_Eclipse)

## *Goals of this talk*

---

It will explain how to test E4 POJOs using E4 Tester

## *Use case for core plugins*

---

Testing core plugins is simple:

- create a test fragment (depending on junit)
- write junit tests
  - create samples
  - mock some objects
- in general no specific runtime for running the test
- create a basic launch configuration (JUnit Test Plugin)
- call the tests in the tycho build (eclipse-test-plugin)

## *Use case for UI Plugins*

---

Testing UI plugins is more complicated:

- need to simulate the UI framework
- difficult to test only a part of UI, usually whole application is tested
- in E3 : must have a full RCP stack

- in E4 : must have a minimal E4 stack (with context and services)
- difficult to find the good launch configuration (UI Thread ? headless ? ...).
- difficult to call these tests using Tycho (must have a specific configuration)

### E4 Tester

The goal of E4 tester is to provide a simple E4TestCase to:

- test POJO in a simple way
- test interaction between POJOS
- send events in **EventBroker** to test the Pojo interaction
- define a test model application to create the UI Elements

Setup is provided:

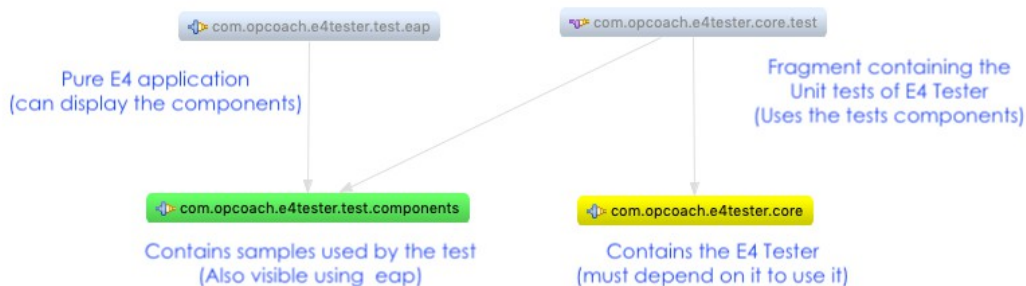
- to run the tests using a dedicated application in Eclipse
- to run the tests in the tycho build with a basic pom

### E4TestCase features

The E4TestCase provides convenient methods:

- create a part in the provided test model application
- open a perspective
- select an object in a tree
- get the value of any widget (label, combo, button...)
- assert that values in UI are as expected
- sendEvent to POJOS
- simulate selection
- ...

### Architecture overview



### Tests of E4 Tester

E4Tester is tested using the good practices:

- **com.opcoach.e4tester.core.test** contains tests in a plugin fragment
- launch configuration for the tests is also available
- included in the build process

All methods defined in the E4TestCase are tested using the sample defined in **com.opcoach.e4tester.test.components**

### **Build overview**

---

E4 Tester is built using the good practices:

- `com.opcoach.e4tester.tp`
- `com.opcoach.e4tester.feature`
- `com.opcoach.e4tester.repository`
- `com.opcoach.e4tester.parent` (defined in the `.project` of the root directory)

The build can be launched manually in the root directory:

- `mvn clean install`
- get the p2 repository generated in `com.opcoach.e4tester.repository/target/repository`

### **Delivery concerns**

---

To use E4 Tester you can use your local built repository

Or you can get the latest stable version on the p2 repository :

- <https://www.opcoach.com/repository/photom>
- use the : `com.opcoach.e4tester.feature.feature.group`

### **E4 Tester development tooling**

---

E4 Tester is hosted on github

- <http://www.github.com/opcoach/E4Tester>
- Follows the github workflow:
  - issues
  - pull request
- **please 'star' the project to see who is interested in !**

E4 Tester is built on travis :

- <https://travis-ci.org/opcoach/E4Tester>
- badge available on the project

### **How to use E4 Tester ?**

---

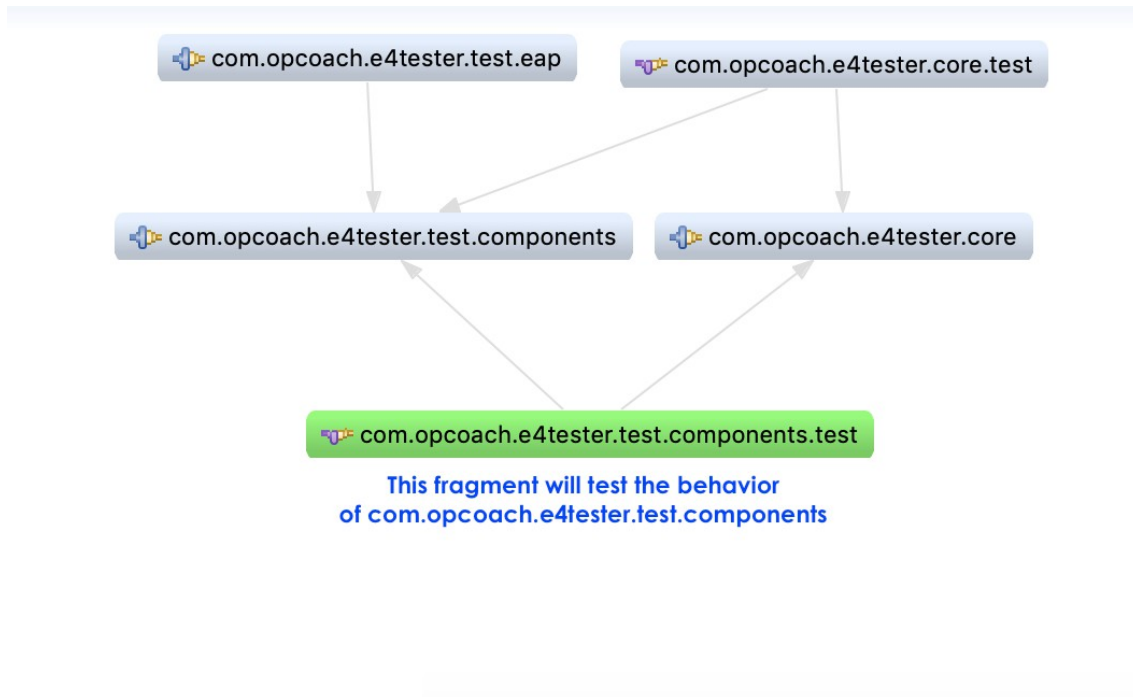
- Update your target platform to get the latest E4 Tester version
- Create a plugin fragment of your UI plugin to be tested
- Write your test cases by extending the `E4TestCase`
- Create a launch configuration and run the tests
- Update your pom files to get the tests in your build

### **Demo concerns**

---

For this demo, we will test the test.components !

- it provides views and perspectives that can be tested
- the goal is to test if that application works as expected
  - we can check the behaviors using the E4 application that uses it
- Of course we will not test the `E4TestCase` content



---

### ***The E4 Application***

#### **Demo :**

- open the eap project and check the model
- open the test.components project and check the model fragment
- launch the E4 Application that displays the test components

---

### ***Setting the target platform***

Use the tpd editor to edit this file

- <https://github.com/eclipse-cbi/targetplatform-dsl>

```

com.opcoach.training.e4.targetplatform.tpd
1 target "Training RCP 4" with source requirements
2
3 location "http://download.eclipse.org/releases/photon" {
4
5     org.eclipse.platform.feature.group
6     org.eclipse.e4.rcp.feature.group
7     org.eclipse.emf.databinding.feature.group
8     org.eclipse.equinox.executable.feature.group
9
10    // Add plugins for test running.
11    org.eclipse.jdt.junit.runtime
12    org.eclipse.pde.junit.runtime
13 }
14
15 // Add delta pack
16 location "http://download.eclipse.org/eclipse/updates/4.8" {
17     org.eclipse.equinox.sdk.feature.group
18 }
19
20 // Add the E4 test platform from opcoach website
21 location "https://www.opcoach.com/repository/photon" {
22     com.opcoach.e4tester.feature.feature.group
23 }
24
25 // Add E4 spies
26 location "http://download.eclipse.org/e4/snapshots/org.eclipse.e4.tools/latest/" {
27     org.eclipse.e4.tools.spies.feature.feature.group
28 }
29

```

This file is also available on gist :

- browse : <https://gist.github.com/opcoach>
- <https://gist.github.com/opcoach/7c96488b14b276df4b749a8431861bac>

For the demo, as all the needed plugins are in the workspace we do not need to create a new target platform.

### Create your plugin fragment

Tests must be defined in a plugin fragment:

- it is modular and gives access to all internal classes of the host plug-in
- it will be built with tycho
- it will be not included in the final delivery

### Writing a TestCase

- The testCase must extends E4TestCase
- The global setup is inherited and is done once
- Create the part in the setup
- Set and check widget values in the test

### Demo 1

Demo 1 : create the Part and check the content inside.

### Demo 2

Demo 2 : test a perspective and its content

### **Demo 3**

---

Demo 3 : Synchronize the 2 views and check selection received

### **Demo 4**

---

Demo 4 : Test EventBroker and send event.

### **Next steps**

---

Fix bugs for selection

- parts must be created in a good order to test selection (focus problem ? )

Add Handler test support

Distinguish SWT tests from JavaFx tests

- create an SWT TestCase and JavaFxTestCase

Manage more types of widgets

- canvas -> should compare images
- sliders
- nebula widgets
- ...

### **How to contribute to this project**

---

- Open an issue to fix on [github.com/opcoach/E4Tester](https://github.com/opcoach/E4Tester) project
- Fork the project into your github account.
- Clone the repository
  - in a shell : `git clone https://github.com/YourAccount/E4Tester.git`
  - in Eclipse : copy the URL of **YOUR GIT repository** in GIT Repository view
- Select the E4 Tester target platform (provided in the project)
  - Preferences -> Target Platform
- Make your changes
- Launch the test (use the E4Tester\_Core\_Tests launch configuration)
- Commit and push your changes to your repository
- Create a pull request.

### **Conclusion**

---

- E4 Tester is easy to use
- It will be completed if it interests people
- This is a simple solution to make simple UI tests

### **Evaluate**

---

Don't forget to evaluate this talk !





## Evaluate the Sessions

Sign in and vote at [eclipsecon.org](http://eclipsecon.org)

-1      0      +1

### *Any Questions ?*

---

- [olivier@opcoach.com](mailto:olivier@opcoach.com)
- The PDF is available on <http://www.opcoach.com/en/opcoach-eclipse-intervention/>