



My own preferred testing techniques





OXiane

 valtech

NOV^EDIA
VISION

 zenika
ARCHITECTURE INFORMATIQUE







 Xebia





www.parisjug.org

My own preferred testing techniques



Plugins

More expressive tests

More robust

Faster

Frameworks



JUnit 4+

Personal taste only

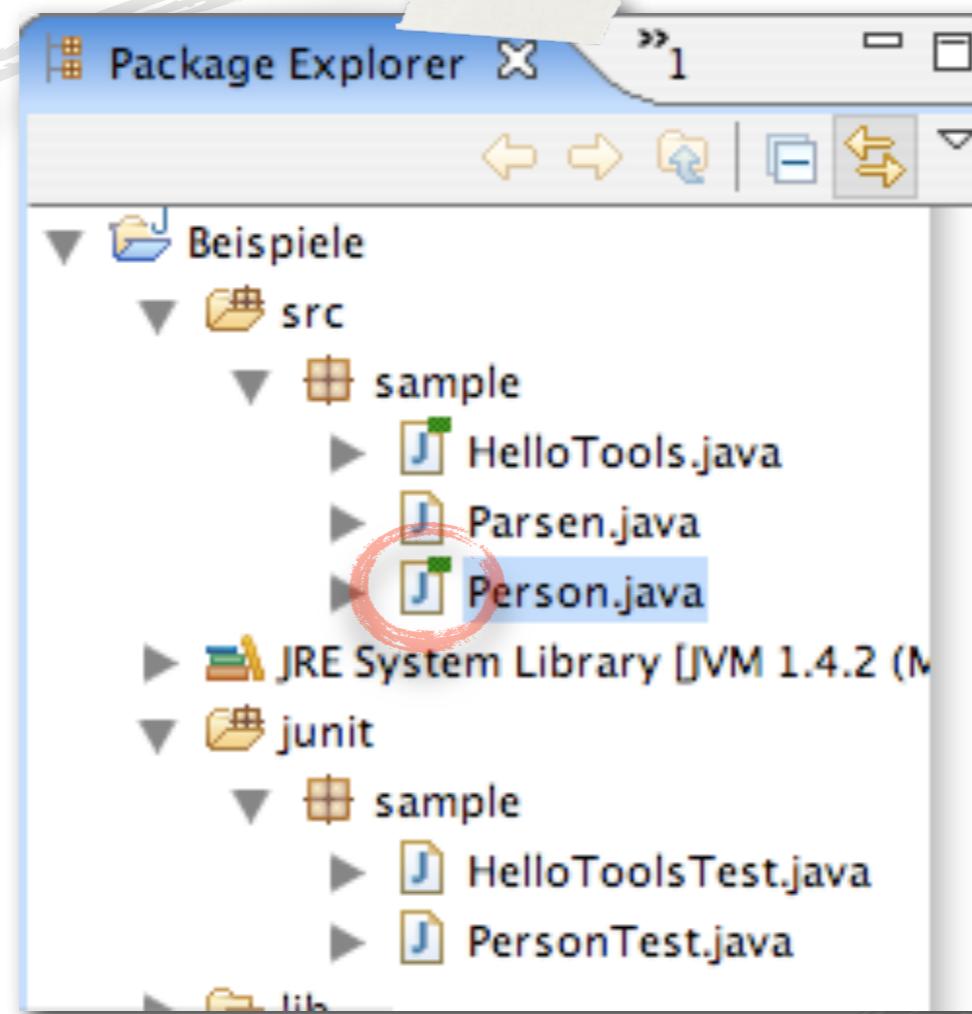


Plugins



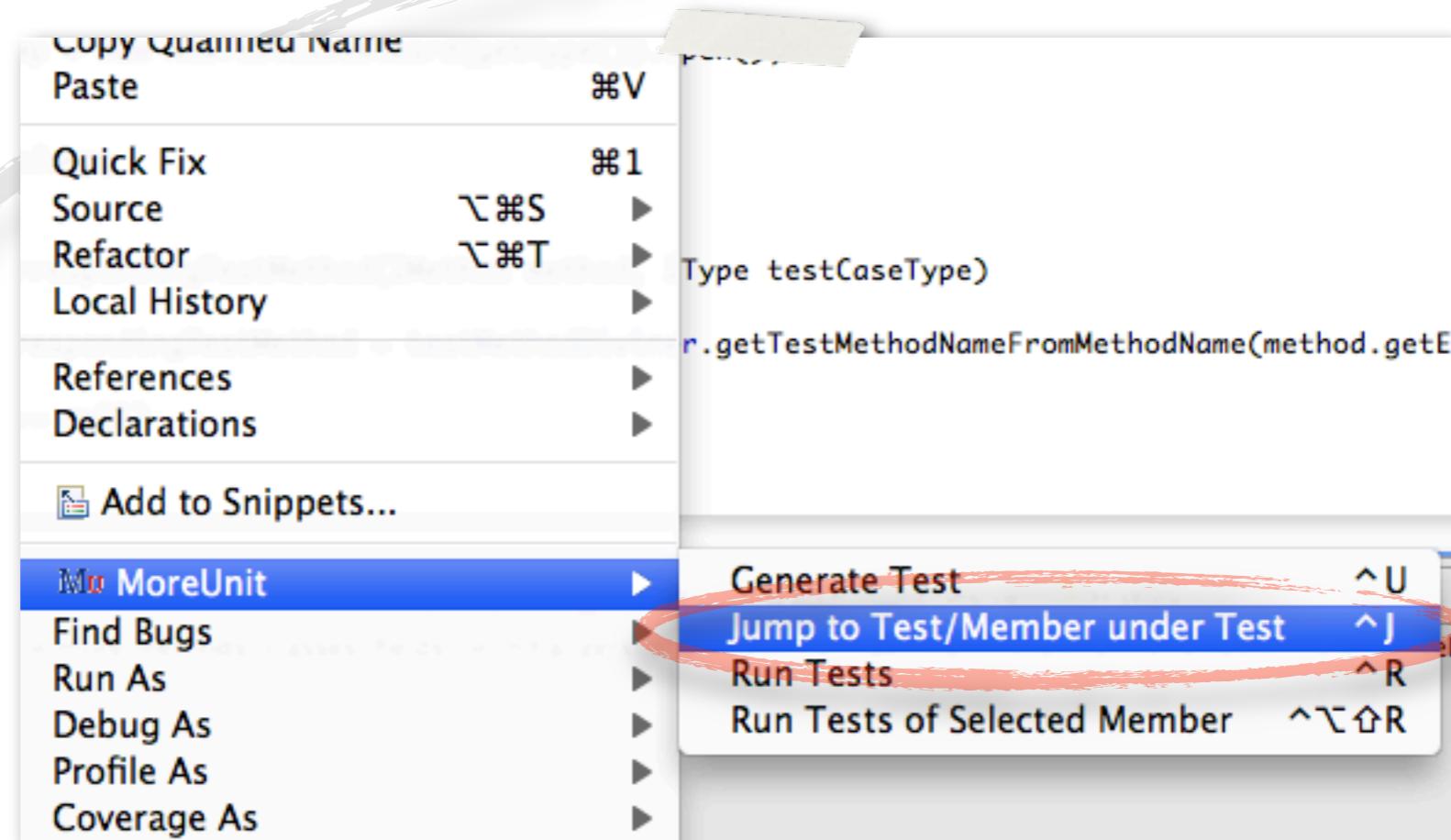
MoreUnit, a unit test friendly Eclipse

See what's tested



MoreUnit

Jump to Test



MoreUnit

Run current test

Refactoring friendly
(move, rename, delete)

<http://moreunit.sourceforge.net/>

Demo



Continuous testing

Runs all impacted tests after each code change

Eclipse and IntelliJ

OpenSource, Free

<http://infinitest.org>



A screenshot of an IDE interface. At the top, there is a navigation bar with tabs: Problems, JUnit (which is selected, indicated by a red arrow), Breakpoints, Variables, Call Hierarchy, and History. Below the navigation bar, a message states "1 error, 0 warnings, 0 others". The main area is divided into two columns: "Description" and "Resource". Under "Description", there is a section titled "Errors (1 item)" with one item: "AssertionError (actual value:<35> should be greater than or equal to:<65>) in PersonTest.canMakeInfinitestFail". The "Resource" column shows the file "PersonTest". In the bottom left corner of the screenshot, there is a small icon of a lightbulb with a red 'X' inside it, which is highlighted by a large red arrow pointing towards it. To the right of the screenshot, there is a yellow starburst shape containing the word "Demo".

```
@Test  
public void canMakeInfinitestFail() {  
    Person person = new Person("David", 35);  
  
    assertThat(person.getName()).isSameAs("David");  
    assertThat(person.getAge()).isGreaterThanOrEqualTo(65);  
}
```

Demo

JUnit Max

Continuous testing

Runs all ~~impacted~~ tests after each code change

Try to run tests failing most often, first

~~Eclipse and IntelliJ~~

~~OpenSource, Free~~

Developed by Kent Beck

<http://www.junitmax.com/>

Frameworks



Hamcrest

More readable assertions

```
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.*;  
  
public void test() {  
    Biscuit theBiscuit = new Biscuit("Ginger");  
    Biscuit myBiscuit = new Biscuit("Ginger");  
  
    assertThat(theBiscuit, equalTo(myBiscuit));  
    assertThat(theBiscuit, is(equalTo(myBiscuit)));  
    assertThat(theBiscuit, is(myBiscuit));  
}
```

Hamcrest

More samples

```
assertThat("Hello World", is(equalTo(greeter.greeting())));
assertThat("Hello", is(allOf(notNullValue(), instanceOf(String.class))));
assertThat("Hello", is(any(String.class)));
assertThat("Hello", is(anything()));
assertThat(object, is(sameInstance(sameObject)));
```

+ easy to write custom matchers

So



FEST-Assert

More readable assertions

```
int removed = employees.removeFired();
assertThat(removed).isZero();
```

```
List<Employee> newEmployees = employees.hired(TODAY);
assertThat(newEmployees).hasSize(6)
    .contains(frodo, sam);
```

```
assertThat(yoda).isInstanceOf(Jedi.class)
    .isEqualTo(foundJedi)
    .isNotEqualTo(foundSith);
```



Demo

FEST-Assert

More readable assertions

OpenSource

<http://fest.easytesting.org/>

JUnit 4.8.2



JUnit 4.8.2

Do you know/use all the features?

@Test

@Before, @After, @BeforeClass, @AfterClass

@RunWith

@Theory, @DataPoint(s)

@Rule

@SuiteClasses

@Category (beta)



Demo

JUnit 4.8.2

@Categories and @SuiteClasses

```
@Category(FastTests.class)
public class WebTest {
    @Test
    @Category(SlowTests.class)
    public void longTest() {
        ...
    }

    @Test
    public void shortTest1() {
        ...
    }

    @Test
    public void shortTest2() {
        ...
    }
}
```

JUnit 4.8.2

@Categories and @SuiteClasses

```
@RunWith(Categories.class)
@IncludeCategory(SlowTests.class)
@SuiteClasses( { WebTest.class, SomeTest.class, MoreTests.class } )
public class LongRunningTestSuite {}
```

```
@RunWith(Categories.class)
@ExcludeCategory(SlowTests.class)
@SuiteClasses( { WebTest.class, SomeTest.class, MoreTests.class } )
public class UnitTestSuite {}
```

JUnit 4.8.2

@Categories and @SuiteClasses

Cumbersome

Need not forget to add each
test manually in all suites

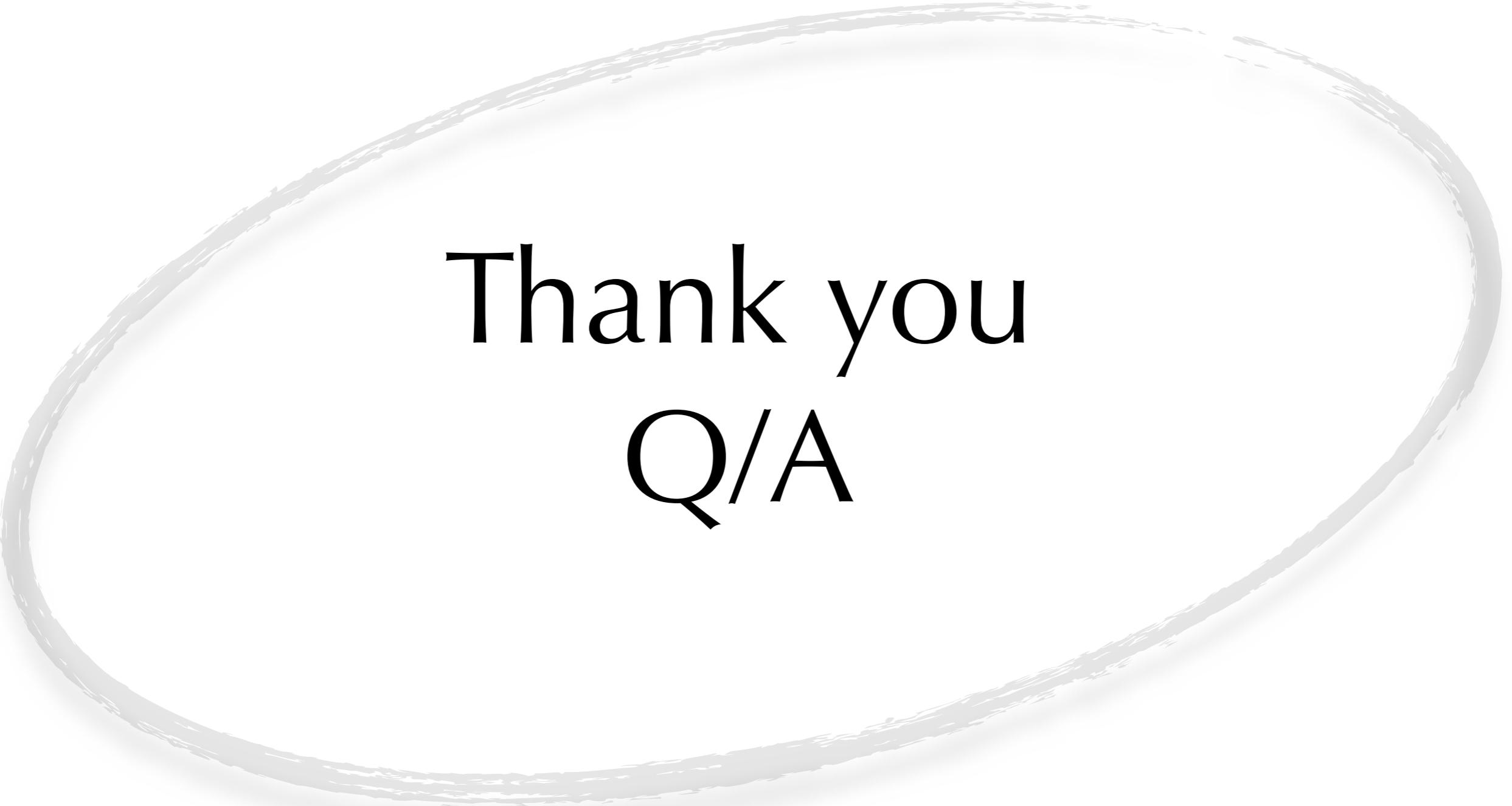
Who doesn't run all the tests
every build? :-)

JUnit 4.8.2

@Rule

Take a look at the code of MethodRule,
ExternalResource, TestWatchman, Verifier...

Write your own rules if they make the tests
easier to read.



Thank you
Q/A



OXiane

 valtech

NOV^EDIA
VISION

 zenika
ARCHITECTURE INFORMATIQUE








Xebia





www.parisjug.org