

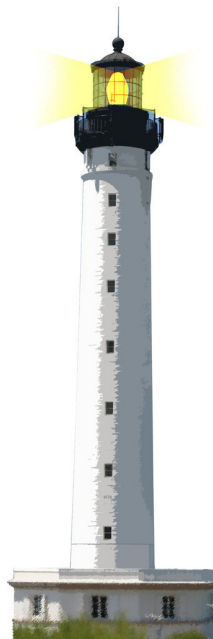
Xtreme Test Driven Development

Getting a productivity boost

S. Ducasse



<http://www.pharo.org>



Outline

- TDD on steroids
- Live programming
- Smart tools
- Absolutely gorgeous development flow



Principle

Do **not break** the flow

- Write a test
- When it breaks, define the method **on the fly in the debugger**
- Resume and continue until test is green



Studying an example

- A dead simple counter. Nothing simpler.
- Focus on essence of the process!
- You can do it.



An empty package

The screenshot shows the Counter IDE interface. The top bar has a title "Counter" and a dropdown arrow. Below it is a sidebar with a folder icon and the name "Counter". The main workspace is divided into three empty panels. At the bottom, there is a toolbar with buttons for "All Packages", "Scoped View", "Inst. side", and "Class side". Below the toolbar is a search bar with a question mark icon and the text "Comment". To the right of the search bar is a button with a plus icon and the text "New class". The bottom panel displays the following code:

```
Object subclass: #NameOfSubclass
  instanceVariableNames: ''
  classVariableNames: ''
  package: 'Counter'
```

An empty test case class

The screenshot shows the IntelliJ IDEA IDE interface. At the top, the title bar reads "CounterTest". Below it, a tab bar contains "Counter" and "CounterTest", with "CounterTest" selected. To the right of the tab bar, a button labeled "instance side" is visible. The main editor area is empty. At the bottom, a toolbar contains several icons and labels: "All Packages", "Scoped View", "Flat", "Hier.", "Inst. side", "Class side", "Methods", "Vars", "New class", "Comment", "CounterTest", "setUp", and "Inst. side meth". Below the toolbar, the code editor displays the following code:

```
TestCase subclass: #CounterTest
  instanceVariableNames: ''
  classVariableNames: ''
  package: 'Counter'
```

In the bottom-left corner, there is a small icon of a lighthouse and the text "6 / 28".

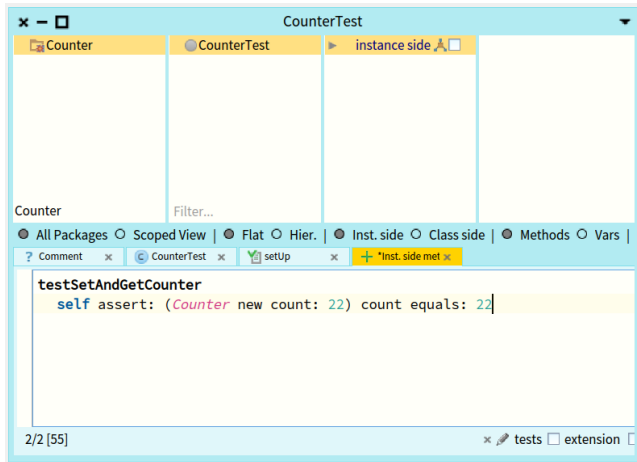
A first test

The screenshot shows the IntelliJ IDEA IDE interface. The top toolbar includes icons for file operations (x, -, □) and a dropdown arrow. Below the toolbar, the 'CounterTest' window is active, displaying a tree view with 'Counter' and 'CounterTest' nodes. The 'CounterTest' node is selected, and the 'instance side' view is active. The main editor area shows the 'testSetAndGetCounter' method with the following code:

```
testSetAndGetCounter
self assert: (Counter new count: 22) count equals: 22|
```

The bottom status bar shows the following tabs: '? Comment', 'CounterTest', 'setUp', and '*Inst. side met'. The 'setUp' tab is active, and the '*Inst. side met' tab is highlighted in yellow.

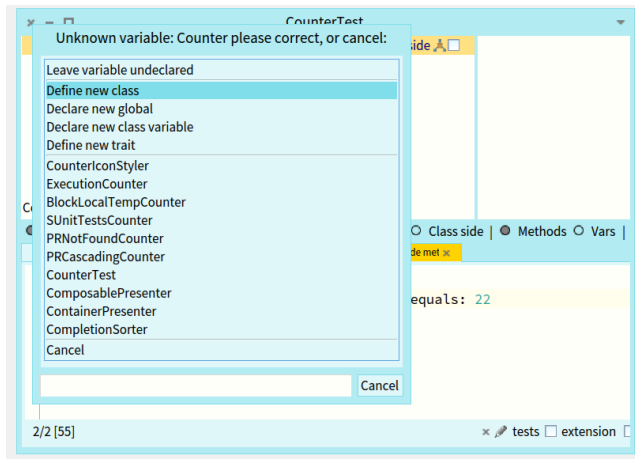
A first test



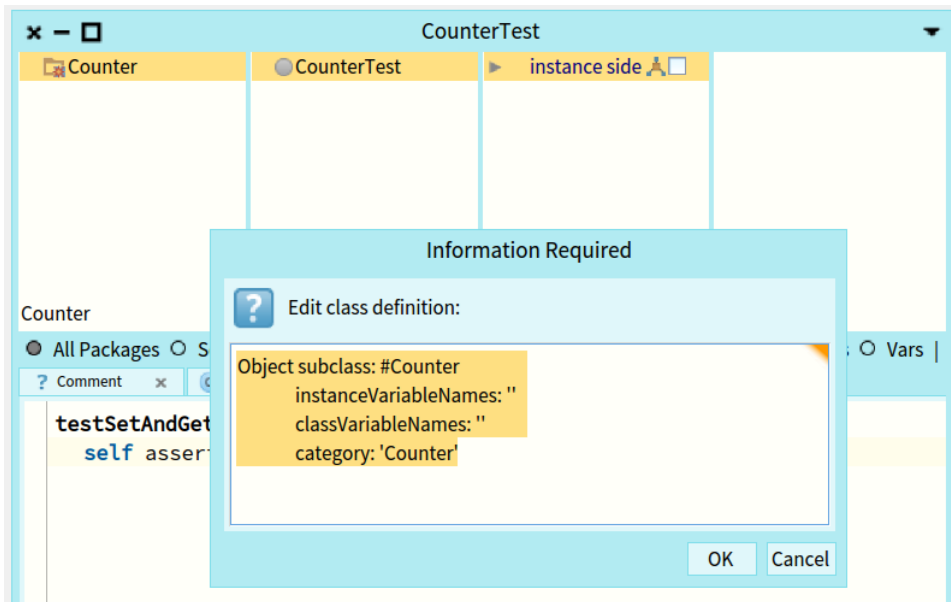
- Method is about to be compiled
- The system knows the class does not exist!

Define a class

- At compile time...



Define a class (II)



Test defined but not executed

The screenshot shows an IDE window titled "CounterTest>>testSetAndGetCounter". The interface is divided into several panes. On the left, a "Counter" package is shown. The middle pane displays a class hierarchy with "Counter" (marked with a red exclamation mark) and "CounterTest". The right pane shows the "instance side" with a "tests" section. Below the panes, a toolbar contains various view options: "All Packages", "Scoped View", "Flat", "Hier.", "Inst. side" (selected), "Class side", "Methods", "Vars", and "CL". Below the toolbar, a tab bar shows several open tabs: "Comment", "CounterTest", "setUp", "testSetAndGet" (highlighted in yellow), "Inst. side metr", and "Inst. side n". The main editor area displays the following code:

```
testSetAndGetCounter
  self assert: (Counter new count: 22) count equals: 22
```

Running the test

CounterTest>>testSetAndGetCounter

Run tests

Counter

Counter !

CounterTest

instance side

tests

testSetAndGetCounter

Counter

Filter...

All Packages | Scoped View | Flat | Hier. | Inst. side | Class side | Methods | Vars | CL

? Comment x CounterTest x setUp x testSetAndGet x + Inst. side met x + Inst. side n

```
testSetAndGetCounter
  self assert: (Counter new count: 22) count equals: 22
```

First Error

Instance of Counter did not understand #count:

Bytecode GT

Stack

+ Create ▶ Proceed 🔄 Restart ⚙ Step into ⚙ Step over ⚙ Step through ☰

Class	Method	Other	Package
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Core
CounterTest(TestCase)	runCase	[self setUp. self performTest	SUnit-Core
FullBlockClosure(BlockClosure)	ensure:		Kernel

Source

🔍 Where is? 📄 Browse

```
testSetAndGetCounter
  self assert: (Counter new count: 22) count equals: 22
```

Variables

Evaluator

Type	Variable	Value
implicit	self	CounterTest>>#testSetAndGetCounter
attribute	expectedFails	an Array [0 items] ()
attribute	testSelector	#testSetAndGetCounter
implicit	thisContext	CounterTest>>testSetAndGetCounter

Create a method on the fly

Create the missing class or method in the user prompted class, and restart the debugger at the location where it can be edited.

Instance of Counter d Bytecode GT

Stack + Create ▶ Proceed ⏮ Restart ⏴ Step into ⏵ Step over ⏶ Step through ⏷

Class	Method	Other	Package
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Core
CounterTest(TestCase)	runCase	[self setUp. self performTest	SUnit-Core
FullBlockClosure(BlockClosure)	ensure:		Kernel

Source 🔍 Where is? 📄 Browse

```
testSetAndGetCounter
  self assert: (Counter new count: 22) count equals: 22
```

Create a method on the fly (II)

Instance of Counter did not understand #count:

Bytecode GT

Stack

► Proceed ◀ Restart ⚙ Step into ⚙ Step over ⚙ Step through ⋮

Class	Method	Other	Package
Counter	count:		Counter
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Core
CounterTest(TestCase)	runCase	[self setUp. self performTest	SUnit-Core

Source

🔍 Where is? 📄 Browse

`count: anInteger
self shouldBeImplemented.`

Variables

Evaluator

Type	Variable	Value
implicit	self	a Counter

Edit the method in the debugger (III)

Instance of Counter did not understand #count: Bytecode GT

Stack

Proceed Restart Step into Step over Step through

Class	Method	Other	Package
Counter	count:		Counter
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Core
CounterTest(TestCase)	runCase	[self setUp. self performTest	SUnit-Core

Source

Where is? Browse

```
count: anInteger  
  count := anInteger
```

Variables Evaluator

Type	Variable	Value
implicit	self	a Counter
parameter	anInteger	22
implicit	thisContext	Counter>>count:
implicit	stack top	22

Add an instance variable on the fly

Instance of Counter did not understand #count:

Unknown variable: count please correct, or cancel:

- Declare new temporary variable
- Declare new instance variable
- Cancel

Restart Step into Step over Step through

Other Package

Counter

Counter

SUnit-Core

[self setUp. self performTest SUnit-Core]

Source Where is? Browse

```
count: anInteger  
count := anInteger
```

Variables Evaluator

Type	Variable	Value
implicit	self	a Counter
parameter	anInteger	22
implicit	thisContext	Counter>>count:
implicit	stack top	22

Compile....

✕ ▢ Instance of Counter did not understand #count: Bytecode GT ▼

Stack ▶ Proceed ↺ Restart 🔍 Step into 🔍 Step over 🔍 Step through ☰

Class	Method	Other	Package
Counter	count:		Counter
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Core
CounterTest(TestCase)	runCase	[self setUp. self performTest	SUnit-Core

Source 🔍 Where is? 📄 Browse

```
count: anInteger  
  count := anInteger|
```

Continue the execution...

Relinquish debugger control and proceed execution from the current point of debugger control.cmd+r

Instance of Counter did not un

Bytecode GT ▼

Stack

► Proceed ◀ Restart ↵ Step into ↗ Step over ↘ Step through ≡

Class	Method	Other	Package
Counter	count:		Counter
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Core
CounterTest(TestCase)	runCase	[self setUp. self performTest	SUnit-Core

Source

Where is? Browse

```
count: anInteger  
count := anInteger
```

Variables Evaluator

Type	Variable	Value
implicit	self	a Counter
parameter	anInteger	22

Supporting the programmer flow

- The system created a new method
- Removed the stack element with Error
- Replaced it with a call to the new method
- Relaunches execution
- We edited it and recompiled the method
- Continued execution



New method

The system created a new method

- Removed the stack element with Error
- Replace it with a **call** to the new method

```
count: anInteger  
self shouldBeImplemented
```

- `shouldBeImplemented` is just an exception so that the debugger stops again

Same story....

✕ - □

Instance of Counter did not understand #count

Bytecode GT ▾

Stack

+ Create ▶ Proceed ↺ Restart ↩ Step into ↗ Step over ↘ Step through ≡

Class	Method	Other	Package
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Cor
CounterTest(TestCase)	runCase	[self setUp. self performTest	SUnit-Cor
FullBlockClosure(BlockClosure)	ensure:		Kernel

Source

🔍 Where is? 📄 Browse

```
testSetAndGetCounter
  self assert: (Counter new count: 22) count equals: 22
```

Variables Evaluator

Debugger also precompiles methods

Instance of Counter did not understand #count Bytecode GT

Stack ▶ Proceed ⏮ Restart ⚙ Step into ⚙ Step over ⚙ Step through ≡

Class	Method	Other	Package
Counter	count		Counter
CounterTest	testSetAndGetCounter		Counter
CounterTest(TestCase)	performTest		SUnit-Cor
CounterTest(TestCase)	runCase	[self setUp. self performTest SUnit-Cor	

Source 🔍 Where is? 📄 Browse

count

^ count

Variables Evaluator

Type	Variable	Value
implicit	self	a Counter
attribute	count	22
implicit	thisContext	Counter>>count
implicit	stack top	nil

Test is green

The screenshot shows an IDE window titled "CounterTest>>testSetAndGetCounter". The interface is divided into several panes. On the left, a "Counter" package is shown. The middle pane displays the "Counter" class with a red exclamation mark icon and the "CounterTest" class with a green circle icon. The right pane shows the "instance side" view with a "tests" section containing a green circle icon and the text "testSetAndGetCounter". Below the panes, a toolbar includes options for "All Packages", "Scoped View", "Flat", "Hier.", "Inst. side", "Class side", "Methods", "Vars", and "CL". The bottom pane shows the test code:

```
testSetAndGetCounter
  self assert: (Counter new count: 22) count equals: 22
```

The test code is highlighted in yellow, indicating it has passed. The bottom status bar shows "24 / 28".

One Cycle

- Run all the tests
- Ready to commit
- New test



Why XTDD is powerful

- Avoid guessing context when coding
- Much much better context
 - inspect that **specific** instance state
 - talk to that **specific** object
- Inspectable / interactable context
- Tests are not a side effect artefact but the driving force

Protip from expert Pharo developers

- Get as fast as possible one object
- Cristalize your scenario with a test
- Xtreme TDD
- Loop

A course by

S. Ducasse, L. Fabresse, G. Polito, and Pablo Tesone



Except where otherwise noted, this work is licensed under CC BY-NC-ND 3.0 France
<https://creativecommons.org/licenses/by-nc-nd/3.0/fr/>