TestView Plugin

A Nautilus Plugin to Facilitate Unit Testing
Overview

1. How to facilitate unit testing
2. Nautilus & TV Plugin
3. Comparison
4. Conclusion
1. How to facilitate unit testing

• View methods and tests side by side
• Test search
• Allow quick creation of new tests
2. Nautilus & TV Plugin

- Default system browser in the Pharo IDE
- TV Plugin is a Nautilus plugin to facilitate unit testing
next

| token |
buffer reset.
tokenStart := stream position.
token := characterType = #eof
ifTrue:
    [RBTToken start: tokenStart + 1 "The EOF token should occur after the end of input"]
ifFalse: [self scanToken].
self stripSeparators.
The TestView Plugin
| token |
buffer reset.
tokenStart := stream position.
token := characterType = #eof
  ifTrue: [RBToken start: tokenStart + 1 "The EOF token should occur after the end of input"]
  ifFalse: [self scanToken].
3. Comparison
Viewing methods and tests side by side
asCamelCase

"Convert to CamelCase. Can be convenient in conjunction with asLegalSelector.
'A man, a plan, a canal, panama' asCamelCase.
'Here 123 should % be 6 the name 6 of the method' asCamelCase.
'Here 123 should % be 6 the name 6 of the method' asCamelCase.

testAsCamelCase

    self assert: 'A man, a plan, a canal, panama' asCamelCase = 'AMan,APlan,ACanal,Panama'.
    self assert: 'Here 123 should % be 6 the name 6 of the method' asCamelCase = 
        'Here123should%Be6TheName6OfTheMethod'.
String >> #asCamelCase

"Convert to CamelCase. Can be convenient in conjunction
'A man, a plan, a canal, panama' asCamelCase.
'A man, a plan, a canal, panama' asCamelCase asLegal
'Here 123should % be 6 the name6 of the method' asCamelCase
'Here 123should % be 6 the name6 of the method' asCamelCase

^ self class streamContents: [stream |
    self substrings do: [:sub |}
**asClassIfPresent:**

```
"returns a global class with my name"
^ presentBlock cull: (self asClassIfAbsent: [ ^ nil ])
```

**asCamelCase**

"Convert to CamelCase. Can be convenient in conjunction with #asLegalSelector
'A man, a plan, a canal, Panama' asCamelCase.
'A man, a plan, a canal, Panama' asCamelCase asLegalSelector.
Give me 6 the name6 of the method' asCamelCase
With the TV Plugin

• In the same Nautilus window
• At the same time
testNextWithAnIdentifierTokenTestToken

| source scanner token |

source := 'identifierToken'.
scanner := self buildScannerForText: source.
token := scanner next.
sel assert: token isIdentifier.
Nautilus' test search

• Nautilus' test search criteria
  • Inherits from *TestCase*
  • *Stack* => *StackTest*
  • *pop* => *testPop*

• *Only one test per method*
String>>#asCamelCase

"Convert to CamelCase. Can be convenient in conjunction with #asLegalSelector.
'A man, a plan, a canal, panama' asCamelCase.
'A man, a plan, a canal, panama' asCamelCase asLegalSelector.
'tHere 123should % be 6 the name6 of the method' asCamelCase.
'tHere 123should % be 6 the name6 of the method' asCamelCase asLegalSelector."

^self class streamContents: [:stream |
  self substrings do: [:sub | ]
}
Test search of the TV Plugin

• Class level
  • Inherits from TestCase
  • Stack => StackTest, TestStack, Teststack2, etc.
  • Customizable

• Method level
  • pop => testPop, popTest, poptest2, etc.
  • Test should call pop
RBS\text{Scanner} \gg \#next

---

all

accessing

error handling

initialize-release

private

private-scanning

testing

next

def new Test

RBS\text{ScannerTest} \gg \#testNextWithAnIdentifierTokenGetThe

RBS\text{ScannerTest} \gg \#testNextWithAWrongSymbolGet\text{Error}

RBS\text{ScannerTest} \gg \#testNextWithAnOpeningStringsGet\text{Error}

RBS\text{ScannerTest} \gg \#testNextWithAnUnknownCharacterGet

RBS\text{ScannerTest} \gg \#testNextWithAnOpeningCommentGet

178

token \text{ := } \text{scanner next.}

token should occur after the end of input"

token := characterType = \#eof

if True:

[RB\text{Token} \text{ start: } tokenStart + 1 "The EOF

token shall occur after the end of input"]

self assert: token is\text{identifier}.  

RBS\text{ScannerTest} \gg \#testNextWithAnIdentifierTokenGetThe

RBS\text{ScannerTest} \gg \#testNextWithAWrongSymbolGet\text{Error}

RBS\text{ScannerTest} \gg \#testNextWithAnOpeningStringsGet\text{Error}

RBS\text{ScannerTest} \gg \#testNextWithAnUnknownCharacterGet

RBS\text{ScannerTest} \gg \#testNextWithAnOpeningCommentGet


Creating new tests

• In an existing test class
• In a new test class
asClassIfPresent: presentBlock

"returns a global class with my name"

^presentBlock cull: (self asClassIfAbsent: [ ^nil ])

asCamelCase

"Convert to CamelCase. Can be convenient in conjunction with"

'A man, a plan, a canal, panama' asCamelCase.

'A man, a plan, a canal, panama' asCamelCase asLegalSelector.

'^here % should be 6, the name6 of the method' asCamelCase.
4. Conclusions

• Proper support for unit testing is often lacking
• Developers are thus discouraged from writing tests
• Does the TV Plugin effectively encourage unit testing?
Where to get the TVPlugin?

• Search "TestView" on SmallTalkHub
Questions?