Article
Henrique Dias · Feb 5, 2023 4m read

Open Exchange

iris-tripleslash - let's rock together

Hey everyone,

Here we are again. New year, new contest, new project, old reasons.

Triple Slash is in the house!

1999, the year I learned how to code, my first "if," my first "Hello world."

I still remember my teacher explaining to everyone in that classroom the simple "while" and how we can figure out if a specific condition was achieved @Renato Banzai do you remember that? Teacher Barbosa, what a unique guy.

Since then, I have loved the idea of coding, transforming ideas into projects, into something useful. But we all know that to create something, we need to make sure it's working; we need not only create but test if it's working and if it doesn't break if we add something new.

And to be honest with all of you, creating tests it's boring. At least for me, nothing against you if you like to do it.

Using a poor analogy, I can say that creating testing methods is like cleaning your house or ironing your clothes. It's boring, but it's necessary if you want something better.

With that in mind, why not create a better and easiest way to test?

So, inspired in elixir style and in this idea from InterSystems Ideas (Thanks @Evgeny Shvarov)! We tried to improve the testing process and turn it into an enjoyable task again.

We simplify the %UnitTest and to show you how to use TripleSlash to create your unit tests, let use a simple example.

Let's say you have the following class and method which you'd like to write a unit test:

```
Class dc.sample.ObjectScript
{
ClassMethod TheAnswerForEverything() As %Integer
{

   Set a = 42
   Write "Hello World!",!

   Write "This is InterSystems IRIS with version ",$zv,!

   Write "Current time is: "_$zdt($h,2)
```

```
   Return a

}
}
```

As you can see, the **TheAnswerForEverything()** method just retruns the number **42**. So, let mark in the method documentation how TripleSlash should create a unit test for this method:

```
/// A simple method for testing purpose.
///
/// <example>
/// Write ##class(dc.sample.ObjectScript).Test()
/// 42
/// </example>
ClassMethod TheAnswerForEverything() As %Integer
{
    ...
}
```

Unit tests must be enclosed by tag ***<example></example>***. You can add any kind of documentation, but all tests must be within such a tag.

Now, start an IRIS terminal session, go to **IRISAPP** namespace, create an instance of the **Core** class passing the class name (or its package name for all its classes) and then run the **Execute()** method:

```
USER>ZN "IRISAPP"
IRISAPP>Do ##class(iris.tripleSlash.Core).%New("dc.sample.ObjectScript").Execute()
```

TripleSlash will interpret this like "Given the result of the **Test()** method, asserts that it is equals to **42**". So, a new class will be create within the unit test:

```
Class iris.tripleSlash.tst.ObjectScript Extends %UnitTest.TestCase
{

Method TestTheAnswerForEverything()
{
    Do $$$AssertEquals(##class(dc.sample.ObjectScript).TheAnswerForEverything(), 42)
}

}
```

Now let's add a new method for testing other ways to tell TripleSlash on how to write your unit tests.

```
Class dc.sample.ObjectScript
{

ClassMethod GuessTheNumber(pNumber As %Integer) As %Status
{
    Set st = $$$OK
    Set theAnswerForEveryThing = 42
    Try {
        Throw
:(pNumber '= theAnswerForEveryThing) ##class(
%Exception.StatusException).%New("Sorry, wrong number...")
```

```
    } Catch(e) {
        Set st = e.AsStatus()
    }

    Return st

}

}
```

As you can see, the **GuessTheNumber()** method expects a number, returns **$$$OK** just when the number **42** is passed or an error for any other value. So, let mark in the method documentation that how TripleSlash should create a unit test for this method:

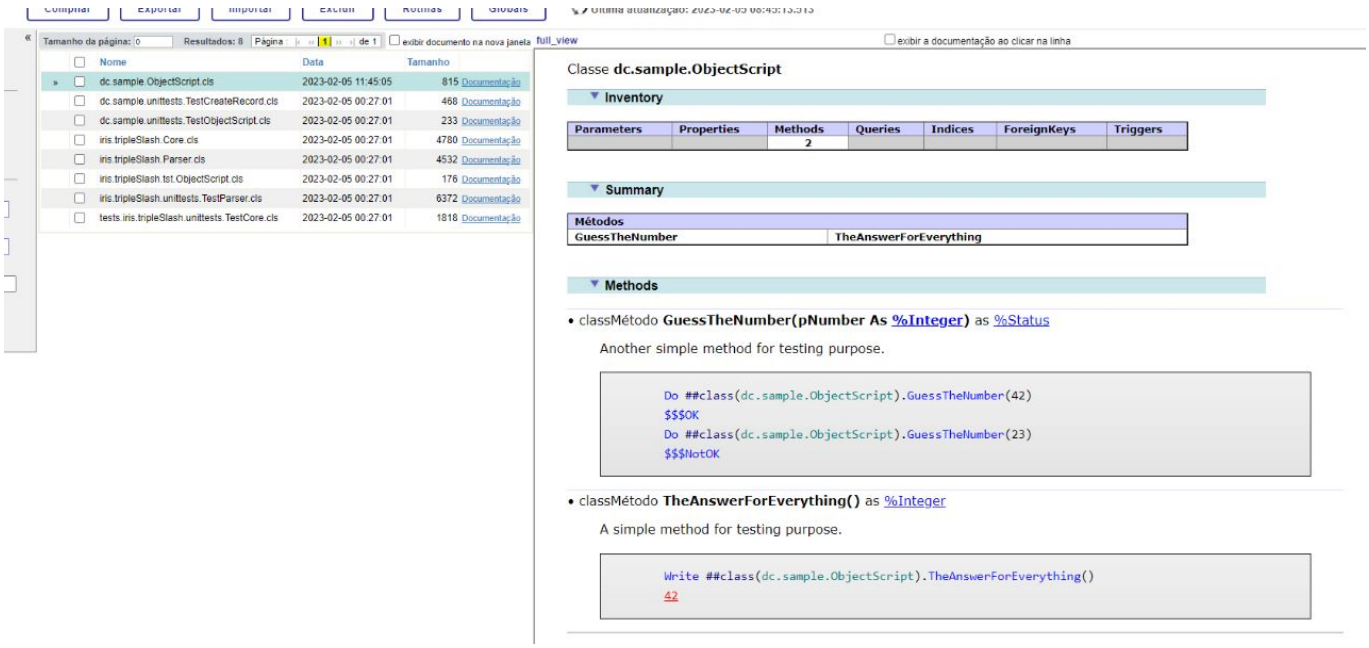
```
/// Another simple method for testing purpose.
///
/// <example>
/// Do ##class(dc.sample.ObjectScript).GuessTheNumber(42)
/// $$$OK
/// Do ##class(dc.sample.ObjectScript).GuessTheNumber(23)
/// $$$NotOK
/// </example>
ClassMethod GuessTheNumber(pNumber As %Integer) As %Status
{
    ...
}
```

Run again the **Execute()** method and you'll see a new test method in unit test class **iris.tripleSlash.tst.ObjectScript**:

```
Class iris.tripleSlash.tst.ObjectScript Extends %UnitTest.TestCase
{

Method TestGuessTheNumber()
{

    Do $$$AssertStatusOK(##class(dc.sample.ObjectScript).GuessTheNumber(42))
    Do $$$AssertStatusNotOK(##class(dc.sample.ObjectScript).GuessTheNumber(23))
}

}
```

Currently, the following assertions are available: **$$$AssertStatusOK**, **$$$AssertStatusNotOK** and **$$$AssertEquals**.

TripleSlash allows us to generate tests from code examples found in method descriptions. It helps you to *kill two birds with one stone*, improving your class documentation and creating test automation.



Acknowledgment

Once again, we would like to thank you for all the support from the community in each of the applications we create.

If you found our app interesting and contributed some insight, please vote for iris-tripleslash and help us on this journey!

#Contest #Framework #Testing #InterSystems Ideas Portal #InterSystems IRIS #InterSystems IRIS for Health #Open Exchange
Check the related application on InterSystems Open Exchange