記事
Toshihiko Minamoto · 2022年4月21日 6m read

Open Exchange

ObjectScript Package Manager のユニットテストとテストカバレッジ

この記事では、ObjectScript Package Manager（https://openexchange.intersystems.com/package/ObjectScript-Package-Manager を参照）を使用して、ユニットテストを実行するためのプロセスを説明します。テストカバレッジ測定（https://openexchange.intersystems.com/package/Test-Coverage-Tool を使用）も含まれます。

ObjectScript でのユニットテスト

ObjectScript でユニットテストを記述する方法については、素晴らしいドキュメントがすでに存在するため、ここでは繰り返しません。 ユニットテストのチュートリアルは、こちらをご覧ください: https://docs.intersystems.com/irislatest/csp/docbookj/Doc.View.cls?KEY=TUNTpreface

ユニットテストは、「/tests」など、ソースツリーの別の場所に含めるのがベストプラクティスです。InterSystems 内では、デファクトスタンダードとして、/internal/testing/unittests/ を使用しています。テストは社内用/非配布用であり、ユニットテスト以外の種類のテストもあるため、これが意に適うためではありますが、単純なオープンソースプロジェクトでは、多少複雑になるかもしれません。 InterSystems の一部の GitHub リポジトリでは、この構造が使用されています。

ワークフローの観点では、VSCode では非常に簡単に行えます。ディレクトリを作成して、そこにクラスを配置するだけです。より旧式のサーバー中心型のソース管理アプローチ（Studio で使用されているアプローチ）の場合、このパッケージを適切にマッピングする必要があります。このアプローチは、ソース管理拡張機能ごとに異なります。

ユニットテストクラスの命名規則の観点では、個人的には以下を気に入っています（私の所属するグループにおいてはベストプラクティスです）。

UnitTest.<テストされるパッケージ/クラス>[.<テストされるメソッド/機能>]

たとえば、MyApplication.SomeClass クラスのメソッド Foo のユニットテストであれば、ユニットテストクラスは UnitTest.MyApplication.SomeClass.Foo と名付けられます。一方で、クラス全体のテストであれば、単に UnitTest.MyApplication.SomeClass となります。

ObjectScript Package Manager でのユニットテスト

ObjectScript Package Manager でユニットテストを認識させるのは簡単です！ 以下のように、module.xml に行を 1 つ追加するだけです（https://github.com/timleavitt/ObjectScript-Math/blob/master/module.xml から抜粋。Open Exchange の @Peter Steiwer による優れた数学パッケージのフォークです。これは単純なサンプルとして使用しています）。

```
<Module>
...
<UnitTest Name="tests" Package="UnitTest.Math" Phase="test"/>
</Module>
```

上記は次のように解釈します。

- ユニットテストは、モジュールのルート配下にある「tests」ディレクトリにあります。
- ユニットテストは「UnitTest.Math」パッケージにあります。
  テストされるクラスは「Math」パッケージに含まれるため、合理的です。
- ユニットテストは、パッケージライフサイクルの「test」フェーズで実行します。
  （テストが実行できる「verify」フェーズもありますが、これについては後日お話しします。）

ユニットテストの実行

上記で説明されたとおりにユニットテストを定義したら、Package Manager
でそれを実行するための非常に便利なツールを使用することができます。 %UnitTest.Manager と同じように ^UnitTestRoot などを設定することも可能ですが、特に同じ環境で複数のプロジェクトの作業を行っている場合には、次のオプションを使用する方がはるかに簡単だと思います。

このすべては、上記にリストされている objectscript-math リポジトリのクローンを作成してから zpm "load /path/to/cloned/repo/" で読み込むか、「objectscript-math」を自分のパッケージ名（とテスト名）に入れ替えて独自のパッケージで実行することができます。

モジュールを再読み込みしてからすべてのユニットテストを実行する:

```
zpm "objectscript-math test"
```

ユニットテストのみを実行する（再読み込みなし）:

```
zpm "objectscript-math test -only"
```

ユニットテストのみを実行し（再読み込みなし）、詳細出力を取得する:

```
zpm "objectscript-math test -only -verbose"
```

再読み込みを行わずに特定のテストスイート（テストのディレクトリ、この場合は UnitTest/Math/Utils のすべてのテスト）のみを実行し、詳細出力を取得する:

```
zpm "objectscript-math test -only -verbose -DUnitTest.Suite=UnitTest.Math.Utils"
```

再読み込みを行わずに特定のテストケース（この場合は UnitTest.Math.Utils.TestValidateRange）のみを実行し、詳細出力を取得する:

```
zpm "objectscript-math test -only -verbose -DUnitTest.Case=UnitTest.Math.Utils.TestValidateRange"
```

または、1 つのテストメソッドに存在する問題を解決するだけである場合は、以下のようにします。

```
zpm "objectscript-math test -only -verbose -DUnitTest.Case=UnitTest.Math.Utils.TestValidateRange
-DUnitTest.Method=TestpValueNull"
```

ObjectScript Package Manager を使ったテストカバレッジ測定

ユニットテストがあっても、それがうまく機能するのかを知るにはどうすればよいのでしょうか。 テストカバレッジを測定するだけでは、この疑問に完全に答えることはできませんが、少なくとも役には立ちます。
このことについては、2018 年の Global Summit でお話ししました。https://youtu.be/nUSeGHwN5pc
をご覧ください。

最初に行うべきことは、テストカバレッジパッケージをインストールすることです。

```
zpm "install testcoverage"
```

これには ObjectScript Package Manager
によるインストール/実行は不要であることに注意してください。詳しくは、Open Exchange をご覧ください:
https://openexchange.intersystems.com/package/Test-Coverage-Tool

とは言っても、ObjectScript Package Manager
も使用するのであれば、テストカバレッジツールを最大限に活用することができます。

テストを実行する前に、テストでカバーしたいクラス/ルーチンを指定する必要があります。
非常に大型のコードベース（HealthShare など）においては、プロジェクト内のすべてのファイルに使用するテストカバレッジを測定して収集してしまうと、システムに装備されている以上のメモリが必要となる場合があるため、このステップは重要です。 （具体的には、行単位モニターの gmheap です。）

ファイルのリストは、ユニットテストのルート内の coverage.list というファイルに出力されます。つまり、ユニットテストの別のサブディレクトリ（スイート）にそれぞれのリストが格納されるため、テストスイートが実行する間、追跡されるクラス/ルーチンがオーバーライドされます。

objectscript-math
を使った単純な例については、https://github.com/timleavitt/ObjectScript-Math/blob/master/tests/UnitTest/coverage.list
をご覧ください。テストツールのユーザーガイドではさらに詳しく説明されています。

テストカバレッジ測定を有効にしてユニットテストを実行する場合、コマンドにはもう 1
つ引数を追加します。テストを実行するために、%UnitTest.Manager の代わりに TestCoverage.Manager
が使用することを指定するのです。

```
zpm "objectscript-math test -only -DUnitTest.ManagerClass=TestCoverage.Manager"
```

出力には、詳細モードでなくても、ユニットテストでカバーされたクラス/ルーチンの行と一部の集計された統計情報を確認できる URL が含まれます。

今後の内容

このすべてのプロセスを CI で自動化してはどうでしょうか。
ユニットテストの結果とカバレッジのスコア/差をレポートするにはどうすればよいでしょうか。
こういったことも、実現可能です！ Docker、Travis CI、および codecov.io
を使用した簡単な例について、https://github.com/timleavitt/ObjectScript-Math
をご覧ください。将来、いくつかの異なるアプローチを説明する記事を書く予定です。

#InterSystems Package Manager (IPM) #継続的インテグレーション #InterSystems IRIS #InterSystems IRIS for Health
InterSystems Open Exchangeで関連アプリケーションを確認してください

ソースURL:https://jp.community.intersystems.com/post/objectscript-package-manager-%E3%81%AE%E3%83%A6%E3%83%8B%E3%83%83%E3%83%88%E3%83%86%E3%82%B9%E3%83%88%E3%81%A8%E3%83%86%E3%82%B9%E3%83%88%E3%82%AB%E3%83%90%E3%83%AC%E3%83%83%E3%82%B8