Artigo

Henry Pereira · jan 7, 2021 13min de leitura

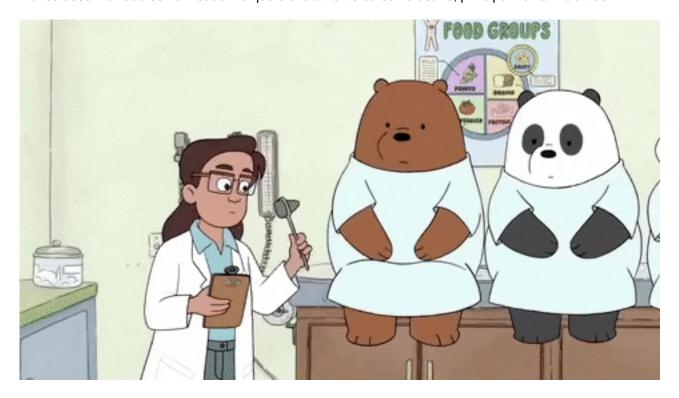
Open Exchange

# Breve introdução ao Desenvolvimento Guiado por Testes (TDD) com Caché e CosFaker

Tempo estimado de leitura: 6 minutos

Olá a todos.

Fui apresentado ao TDD há quase 9 anos e imediatamente me apaixonei por ele. Hoje em dia se tornou muito popular, mas, infelizmente, vejo que muitas empresas não o utilizam. Além disso, muitos desenvolvedores nem sabem o que é exatamente ou como usá-lo, principalmente iniciantes.



# Visão Geral

Meu objetivo com este artigo é mostrar como usar TDD com %UnitTest. Vou mostrar meu fluxo de trabalho e explicar como usar o <u>cosFaker</u>, um dos meus primeiros projetos, que criei usando o Caché e recentemente carreguei no <u>OpenExchange</u>.

Então, aperte o cinto e vamos lá.

# O que é TDD?

O Desenvolvimento Guiado por Testes (TDD) pode ser definido como uma prática de programação que instrui os desenvolvedores a escrever um novo código apenas se um teste automatizado falhar. Existem toneladas de artigos, palestras, apresentações, seja o que for, sobre suas vantagens e todas estão

Breve introdução ao Desenvolvimento Guiado por Testes (TDD) com Caché e CosFaker Published on InterSystems Developer Community (https://community.intersystems.com)

#### corretas.

Seu código já nasce testado, você garante que seu sistema realmente atenda aos requisitos definidos para ele, evitando o excesso de engenharia, e você tem um feedback constante.

Então, por que não usar o TDD? Qual é o problema com o TDD? A resposta é simples: o Custo! Isso custa muito! Como você precisa escrever mais linhas de código com TDD, é um processo lento. Mas com o TDD você tem um custo final para criar um produto AGORA, sem ter que adicioná-lo no futuro.

Se você executar os testes o tempo todo, encontrará os erros antecipadamente, reduzindo assim o custo de sua correção.

Portanto, meu conselho: Simplesmente Faça!

# Configuração

A InterSystems tem uma documentação e tutorial sobre como usar o %UnitTest, que você pode ler aqui.

Eu uso o vscode para desenvolver. Desta forma, crio uma pasta separada para testes. Eu adiciono o caminho para código do meu projeto ao UnitTestRoot e quando executo testes, passo o nome da subpasta de teste. E eu sempre passo no qualificador loadudl

```
Set ^UnitTestRoot = "~/code"

Do ##class(%UnitTest.Manager).RunTest("myPack","/loadudl")
```

## **Passos**

Provavelmente você já ouviu falar sobre o famoso ciclo TDD: vermelho verde refatorar. Você escreve um teste que falha, você escreve um código de produção simples para fazê-lo passar e refatora o código de produção. Então, vamos sujar as mãos e criar uma classe para fazer cálculos matemáticos e outra para testá-la. A última classe deve estender de %UnitTest.TestCase.

Agora vamos criar um ClassMethod para retornar um quadrado de um número inteiro:

```
Class Production.Math
{

ClassMethod Square(pValue As %Integer) As %Integer
{
}

E teste o que acontecerá se passarmos 2. Deve retornar 4.

Class TDD.Math Extends %UnitTest.TestCase
{
```

```
Method TestSquare()
{
    Do $$$AssertEquals(##class(Production.Math).Square(2), 4)
}
}
Se você executar:
Do ##class(%UnitTest.Manager).RunTest("TDD","/loadudl")
o teste irá Falhar
Vermelho! O próximo passo é torná-lo Verde.
Para fazer funcionar, vamos retornar 4 como resultado da execução do nosso método Square.
Class Production.Math
{
ClassMethod Square(pValue As %Integer) As %Integer
{
  Quit 4
}
}
e executar novamente nosso teste.
Provavelmente você não está muito satisfeito com esta solução, porque ela funciona para apenas um cenário.
Ótimo! Vamos dar o próximo passo. Vamos criar outro cenário de teste, agora enviando um número negativo.
Class TDD.Math Extends %UnitTest.TestCase
{
Method TestSquare()
{
```

Do \$\$\$AssertEquals(##class(Production.Math).Square(2), 4)

```
Method TestSquareNegativeNumber()
{
        Do $$$AssertEquals(##class(Production.Math).Square(-3), 9)
}

Quando executamos o teste:
ele Falhará novamente, então vamos refatorar o código de produção:
Class Production.Math
{
ClassMethod Square(pValue As %Integer) As %Integer
```

e executar novamente nossos testes:

Quit pValue \* pValue

Agora tudo funciona bem... Esse é o ciclo do TDD, em pequenos passos.

Você deve estar se perguntando: por que devo seguir esses passos? Por que eu tenho que ver o teste falhar? Trabalhei em equipes que escreveram o código de produção e só depois escrevi os testes. Mas eu prefiro seguir estes passos de bebê pelos seguintes motivos:

Tio Bob (Robert C. Martin) disse que escrever testes depois de escrever o código não é TDD e, em vez disso, é chamado de " perda de tempo ".

Outro detalhe, quando vejo o teste falhar, e depois vejo passar, estou testando o teste.

Seu teste também é um código; e pode conter erros também. E a maneira de testá-lo é garantir que ele falhe e seja aprovado quando for necessário. Desta forma, você "testou o teste".

# cosFaker

{

}

}

Para escrever bons testes, você pode precisar gerar dados de teste primeiro. Uma maneira de fazer isso é gerar um despejo (dump) de dados e usá-lo em seus testes.

Outra maneira é usar o <u>cosFaker</u> para gerar facilmente dados falsos quando você precisar deles. <u>https://openexchange.intersystems.com/package/CosFaker</u>

Basta fazer o download do arquivo xml, em seguida vá para o Portal de Gerenciamento -> System Explorer -> Classes -> Import. Selecione o arquivo xml a ser importado ou arraste o arquivo no Studio. Você também pode importá-lo usando o Terminal

```
Do $system.OBJ.Load("yourpath/cosFaker.vX.X.xml","ck")
```

### Localização

O cosFaker adicionará arquivos de localidades na pasta da aplicação CSP padrão. Por enquanto, existem apenas dois idiomas: Inglês e Português do Brasil (minha língua nativa).

O idioma dos dados é escolhido de acordo com a configuração do seu Caché.

A localização do cosFaker é um processo contínuo, se você quiser ajudar, não hesite em criar um provedor localizado para sua própria localidade e enviar um Pull Request.

Com o cosFaker você pode gerar palavras aleatórias, parágrafos, números de telefone, nomes, endereços, emails, preços, nomes de produtos, datas, códigos de cores hexadecimais... etc.

Todos os métodos são agrupados por assunto nas classes, ou seja, para gerar uma Latitude você chama o método Latitude na classe Address

```
Write ##class(cosFaker.Address).Latitude()
-37.6806
```

Você também pode gerar Json para seus testes

```
Write ##class(cosFaker.JSON).GetDataJSONFromJSON("{ip:'ipv4',created_at:'date.backwar
d 40',login:'username', text: 'words 3'}")

{
    "created_at":"2019-03-08",
    "ip":"95.226.124.187",
    "login":"john46",
    "text":"temporibus fugit deserunt"
}
```

Aqui está uma lista completa das classes e métodos do cosFaker:

- cosFaker.Address
  - StreetSuffix
  - StreetPrefix
  - o PostCode
  - StreetName
  - Latitude
    - Output: -54.7274
  - Longitude
    - Output: -43.9504
  - o Capital( Location = " " )
  - State(FullName = 0)
  - o City( State = " " )
  - o Country( Abrev = 0 )

```
    SecondaryAddress

       o BuildingNumber
cosFaker.App
       o FunctionName( Group= " ", Separator = " ")
       o AppAction( Group= " " )
       AppType
· cosFaker.Coffee
       • BlendName
              Output: Cascara Cake

    Variety

              Output: Mundo Novo
       Notes
              Output: crisp, slick, nutella, potato defect!, red apple
       • Origin
              Output: Rulindo, Rwanda

    cosFaker.Color

    Hexadecimal

              Output: #A50BD7
       RGB
              Output: 189,180,195
       Name

    cosFaker.Commerce

    ProductName

       Product
       • PromotionCode
       Color

    Department

       • Price(Min = 0, Max = 1000, Dec = 2, Symbol = "")
              Output: 556.88
       CNPJ( Pretty = 1 )

    CNPJ is the Brazilian National Registry of Legal Entities

              Output: 44.383.315/0001-30
cosFaker.Company
       Name

    Profession

      Industry

    cosFaker.Dates

       o Forward( Days = 365, Format = 3 )
       Backward( Days = 365, Format = 3 )

    cosFaker.DragonBall

       Character
              Output: Gogeta
· cosFaker.File
       Extension
              Output: txt
       MimeType
              Output: application/font-woff
       • Filename( Dir = " ", Name = " ", Ext = " ", DirectorySeparator = " / " )
              Output: repellat.architecto.aut/aliquid.gif

    cosFaker.Finance

    Amount( Min = 0, Max = 10000, Dec = 2, Separator= ", ", Symbol = "")

              Output: 3949,18
       o CreditCard( Type = " " )
              Output: 3476-581511-6349
       BitcoinAddress( Min = 24, Max = 34 )
              Output: 1WoR6fYvsE8gNXkBkeXvNqGECPUZ
· cosFaker.Game

    MortalKombat

              Output: Raiden
```

```
StreetFighter
              Output: Akuma
       o Card( Abrev = 0 )
              Output: 5 of Diamonds

    cosFaker.Internet

       o UserName( FirstName = " ", LastName = " ")
       o Email( FirstName = " ", LastName = " ", Provider = " ")

    Protocol

              Output: http

    DomainWord

    DomainName

       Url
       o Avatar( Size = " " )
              Output: <a href="http://www.avatarpro.biz/avatar?s=150">http://www.avatarpro.biz/avatar?s=150</a>
       o Slug( Words = " ", Glue = " ")
       ° IPV4
              Output: 226.7.213.228
       IPV6
              Output: 0532:0b70:35f6:00fd:041f:5655:74c8:83fe
       • MAC
              Output: 73:B0:82:D0:BC:70

    cosFaker.JSON

    GetDataOBJFromJSON( Json = " " // String de modelo JSON para criar dados )

              Parameter Example: "{dates:'5 date'}"
              Output: {"dates":["2019-02-19","2019-12-21","2018-07-02","2017-05-25","2016-08-14"]}

    cosFaker.Job

       o Title
       Field

    Skills

    cosFaker.Lorem

       Word
       o Words( Num = " " )
       Sentence( WordCount = " ", Min = 3, Max = 10)
              • Output: Sapiente et accusamus reiciendis iure qui est.
       Sentences (SentenceCount = " ", Separator = " ")
       o Paragraph( SentenceCount = " ")
       Paragraphs( ParagraphCount = " ", Separator = " ")
       o Lines( LineCount = " " )
       o Text( Times = 1 )
       • Hipster( ParagraphCount = " ", Separator = " ")

    cosFaker.Name

       o FirstName( Gender = " " )
       LastName
       o FullName( Gender = " " )
       Suffix

    cosFaker.Person

       o cpf( Pretty = 1)

    CPF is the Brazilian Social Security Number

              Output: 469.655.208-09

    cosFaker.Phone

       PhoneNumber( Area = 1 )
              Output: (36) 9560-9757
       o CellPhone( Area = 1 )
              Output: (77) 94497-9538
       • AreaCode

    Output: 17

    cosFaker.Pokemon

       o Pokemon( EvolvesFrom = " " )
              Output: Kingdra
```

```
    cosFaker.StarWars

    Characters

             Output: Darth Vader
      Droids
             Output: C-3PO
      Planets
             Output: Takodana
      Quotes
             Output: Only at the end do you realize the power of the Dark Side.
      Species
             Output: Hutt
      Vehicles
             Output: ATT Battle Tank
      WookieWords
             Output: nng
      o WookieSentence( SentenceCount = " " )
             Output: ruh ga ru hnn-rowr mumwa ru ru mumwa.

    cosFaker.UFC

    Category

             Output: Middleweight
      • Fighter( Category = " ", Country = " ", WithISOCountry = 0 )
             Output: Dmitry Poberezhets
      o Featherweight( Country = " " )
             Output: Yair Rodriguez
      o Middleweight( Country = " " )
             Output: Elias Theodorou
      o Welterweight( Country = " " )
             Output: Charlie Ward
      o Lightweight( Country = " " )
             Output: Tae Hyun Bang
      o Bantamweight( Country = " ")
             Output: Alejandro Pérez
      o Flyweight( Country = " " )
             Output: Ben Nguyen
      • Heavyweight( Country = " " )
             Output: Francis Ngannou
      o LightHeavyweight( Country = " ")
             Output: Paul Craig
      o Nickname( Fighter = " " )
             Output: Abacus
```

Vamos criar uma classe para o usuário com um método que retorna seu nome de usuário, que será FirstName concatenado com LastName.

```
Class Production.User Extends %RegisteredObject
{
Property FirstName As %String;
Property LastName As %String;
Method Username() As %String
{
```

```
}
}
Class TDD.User Extends %UnitTest.TestCase
{
Method TestUsername()
  Set firstName = ##class(cosFaker.Name).FirstName(),
    lastName = ##class(cosFaker.Name).LastName(),
    user = ##class(Production.User).%New(),
    user.FirstName = firstName,
    user.LastName = lastName
  Do $$$AssertEquals(user.Username(), firstName _ "." _ lastName)
}
}
Refatorando:
Class Production.User Extends %RegisteredObject
{
Property FirstName As %String;
Property LastName As %String;
Method Username() As %String
{
  Quit ..FirstName _ "." _ ..LastName
}
```

```
}
Agora vamos adicionar uma data de expiração da conta e validá-la.
Class Production.User Extends %RegisteredObject
Property FirstName As %String;
Property LastName As %String;
Property AccountExpires As %Date;
Method Username() As %String
{
  Quit ..FirstName _ "." _ ..LastName
}
Method Expired() As %Boolean
{
}
}
Class TDD.User Extends %UnitTest.TestCase
{
Method TestUsername()
{
  Set firstName = ##class(cosFaker.Name).FirstName(),
    lastName = ##class(cosFaker.Name).LastName(),
    user = ##class(Production.User).%New(),
    user.FirstName = firstName,
```

```
user.LastName = lastName
    Do $$$AssertEquals(user.Username(), firstName _ "." _ lastName)
}
Method TestWhenIsNotExpired() As %Status
{
  Set user = ##class(Production.User).%New(),
    user.AccountExpires = ##class(cosFaker.Dates).Forward(40)
  Do $$$AssertNotTrue(user.Expired())
}
}
Refatorando:
Method Expired() As %Boolean
{
  Quit ($system.SQL.DATEDIFF("dd", ..AccountExpires, +$Horolog) > 0)
}
Agora vamos testar quando a conta expirou:
Method TestWhenIsExpired() As %Status
{
  Set user = ##class(Production.User).%New(),
    user.AccountExpires = ##class(cosFaker.Dates).Backward(40)
  Do $$$AssertTrue(user.Expired())
}
```

E tudo está verde...

Eu sei que esses são exemplos bobos, mas dessa forma você verá a simplicidade não apenas no código, mas também no design da classe.

# Conclusão

Breve introdução ao Desenvolvimento Guiado por Testes (TDD) com Caché e CosFaker Published on InterSystems Developer Community (https://community.intersystems.com)

Neste artigo, você aprendeu um pouco sobre Desenvolvimento Guiado por Testes e como usar a classe %UnitTest.

Também cobrimos o cosFaker e como gerar dados falsos para seus testes.

Há muito mais para aprender sobre testes e TDD, como usar essas práticas com código legado, testes de integração, testes de aceitação (ATDD), bdd, etc...

Se você quiser saber mais sobre isso, recomendo fortemente 2 livros:

<u>Test Driven Development Teste e design no mundo real com Ruby - Mauricio Aniche</u>, realmente não sei se este livro tem versão em inglês. Existem edições para Java, C #, Ruby e PHP. Este livro me surpreendeu com sua grandiosidade.

E, claro, o livro de Kent Beck Test Driven Development by Example

Sinta-se à vontade para deixar comentários ou perguntas. Isso é tudo, pessoal

#Testes #Caché #InterSystems IRIS Confira o aplicativo relacionado no InterSystems Open Exchange

URL de

origem: <a href="https://pt.community.intersystems.com/post/breve-introdu%C3%A7%C3%A3o-ao-desenvolvimento-guiado-por-testes-tdd-com-cach%C3%A9-e-cosfaker">https://pt.community.intersystems.com/post/breve-introdu%C3%A7%C3%A3o-ao-desenvolvimento-guiado-por-testes-tdd-com-cach%C3%A9-e-cosfaker</a>