Artículo
Mathew Lambert · 16 abr, 2020 Lectura de 10 min

Open Exchange

Breve introducción al Desarrollo Basado en Pruebas con Caché y CosFaker

¡Hola Comunidad!

Descubrí el Desarrollo Basado en Pruebas (TDD) hace casi 9 años y me enamoré del concepto inmediatamente.

Hoy se ha vuelto muy popular pero, desafortunadamente, muchas empresas no lo usan. Es más, muchos desarrolladores, sobre todo principiantes, ni siquiera saben exactamente qué es ni como usarlo.

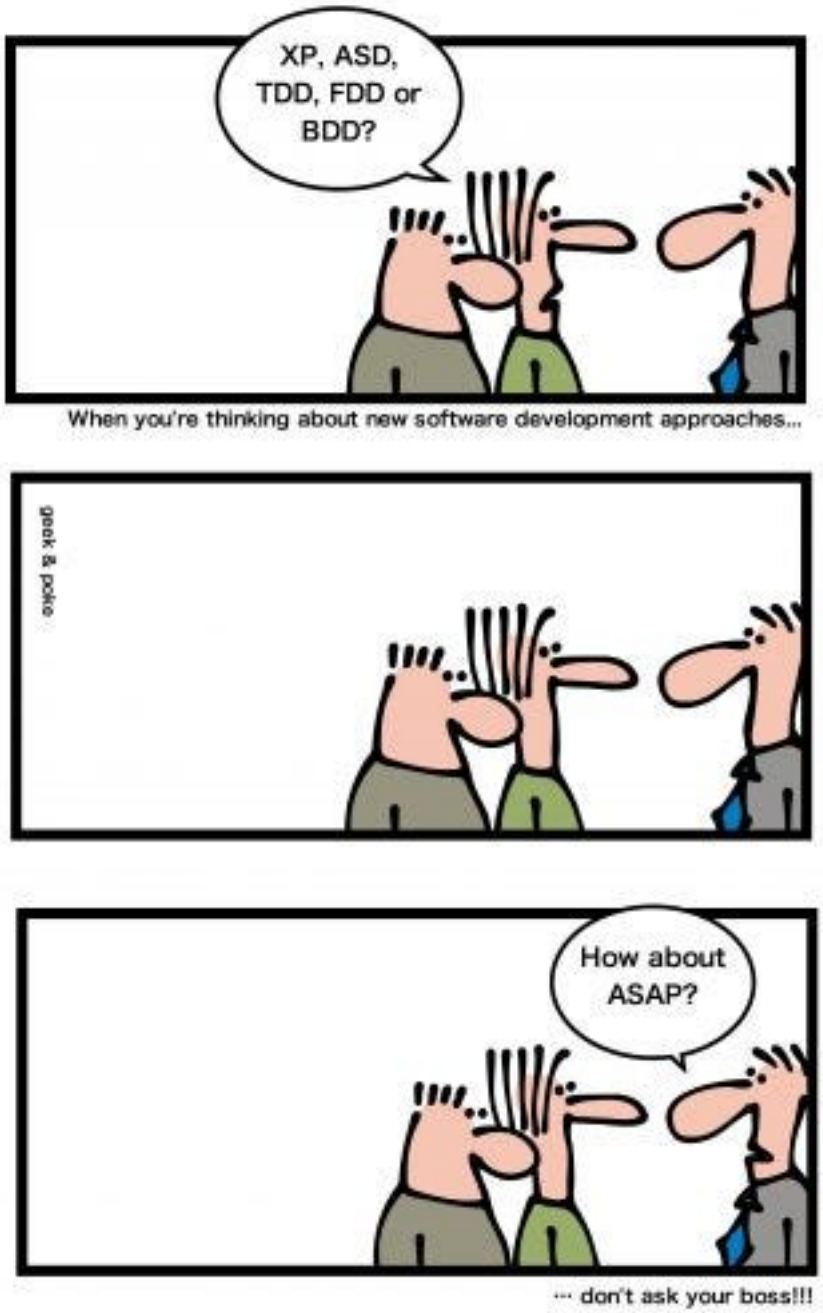
Resumen

Mi objetivo con este artículo es mostrar cómo usar TDD con %UnitTest. Mostraré mi flujo de trabajo y explicaré cómo usar cosFaker, un proyecto de @Henry Pereira Pereira, usando Caché y que hace poco lo subió a OpenExchange.

Así que... ¡preparaos que allá vamos!

¿Qué es TDD?

El Desarrollo basado en pruebas (TDD) puede definirse como una práctica de programación que enseña a los desarrolladores a escribir código solo cuando haya fallado una prueba automática.
Hay montones de artículos, ponencias, charlas, etc., sobre sus ventajas, y todas tienen razón.
El código nace ya probado, te aseguras de que tu sistema realmente cumple con los requerimientos que se le definieron, evitas la sobreingeniería y obtienes feedback continuo.



Entonces... ¿por qué no se usa TDD? ¿Cuál es el problema con TDD? La respuesta es sencilla: ¡el coste! Cuesta mucho.

Al tener que escribir más líneas de código con TDD es un proceso lento. Pero con TDD tienes el coste final de un producto AHORA, sin tener que sumarle en un futuro.
Si ejecutas las pruebas todo el tiempo, encontrarás los errores pronto, lo que reduce el coste de su corrección.
Así que mi consejo es: ¡Hazlo Ya!

Configuración

InterSystems tiene documentación y un tutorial sobre cómo usar %UnitTest, puedes leerlo aquí.

Yo uso vscode para desarrollar. De esta forma, creo una carpeta separada para las pruebas. Agrego la ruta del código de mi proyecto a UnitTestRoot y, cuando ejecuto pruebas, paso el nombre de la subcarpeta de prueba. Y siempre paso el calificador loadudl

```
Set ^UnitTestRoot = "~/code"
Do ##class(%UnitTest.Manager).RunTest("myPack","/loadudl")
```

Pasos

Probablemente hayas oído hablar del famoso ciclo TDD: rojo ➡ verde ➡ refactor. Primero escribes una prueba que falla, luego escribes un código de producción simple para hacer que pase la prueba y luego refactorizas el código de producción.

Así que vamos a ponernos manos a la obra y crear una clase para hacer unas operaciones matemáticas y otra clase para probarla. La segunda de estas clases será una extensión de %UnitTest.TestCase.

Ahora crearemos un Método de clase que devuelva el cuadrado de un número entero:

```
Class Production.Math
{

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

}

}
```

Y probaremos qué sucede si le pasamos 2. Debería devolver 4.

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

Method TestSquare()
{
```

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

}
```

Si ejecutas:

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

La prueba Fallará.

¡Rojo! El siguiente paso es convertirlo en verde.
Para hacer que funcione, devolvamos 4 como resultado de la ejecución de nuestro método Square.

```
Class Production.Math
{

ClassMethod Square(pValue As %Integer) As %Integer
{
  Quit 4
}

}
```

y volvamos a ejecutar nuestra prueba.

```
Load finished successfully.

    TDD.Math begins ...
      TestSquare() begins ...
        AssertEquals:##class(Production.Math).Square(2)== 4 (passed)
        LogMessage:Duration of execution: .000034 sec.
      TestSquare passed
    TDD.Math passed
  test passed

Use the following URL to view the result:
http://10.0.0.1:52773/csp/sys/%25UnitTest.Portal.Indices.cls?Index=11&$NAMESPACE=DEV
All PASSED
DEV>
```

Probablemente no estés muy contento con esta solución, porque solo funciona para un escenario. ¡Bien! Vamos al siguiente paso. Creemos otro escenario de prueba, ahora pasándole un 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)
}

}
```

Cuando ejecutamos la prueba:

```
Load finished successfully.

    TDD.Math begins ...
      TestSquare() begins ...
        AssertEquals:##class(Production.Math).Square(2)== 4 (passed)
        LogMessage:Duration of execution: .000024 sec.
      TestSquare passed
      TestSquareNegativeNumber() begins ...
AssertEquals:##class(Production.Math).Square(-3)== 9 was '4' (failed)  <<==== **FAILED**
test:TDD.Math:TestSquareNegativeNumber:
        LogMessage:Duration of execution: .000019 sec.
      TestSquareNegativeNumber failed
    TDD.Math failed
  test failed
```

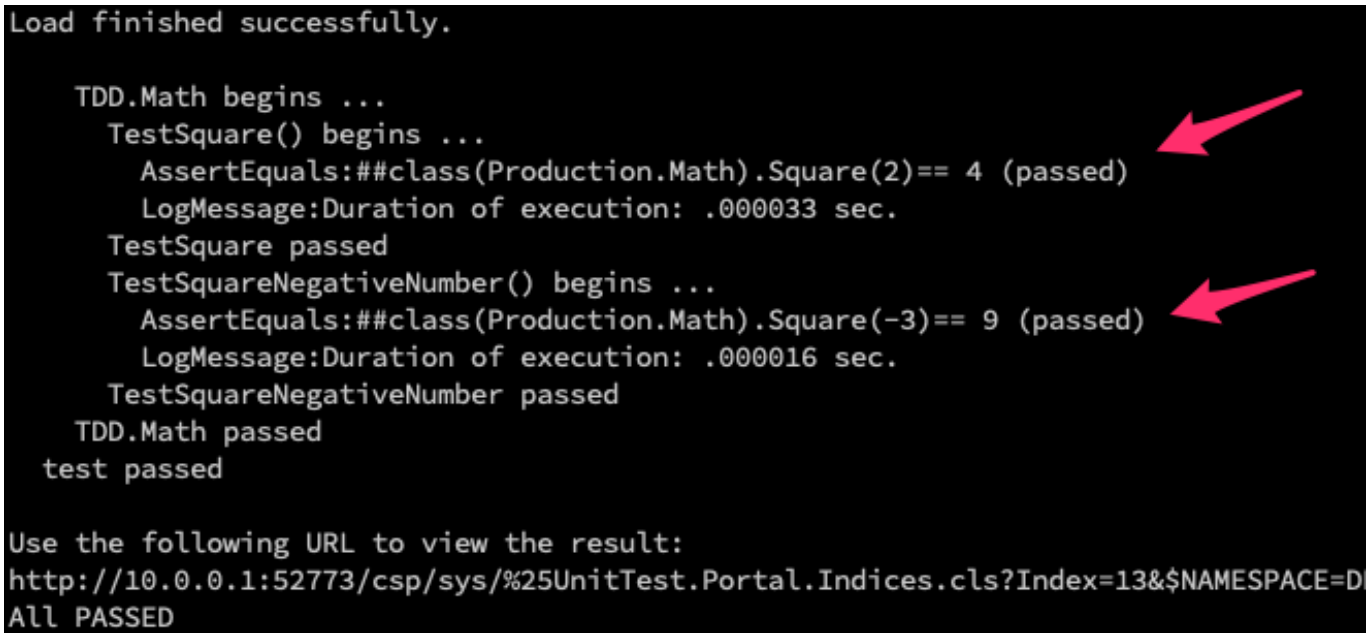
Fallará nuevamente. Refactoricemos entonces el código de producción:

```
Class Production.Math
{
```

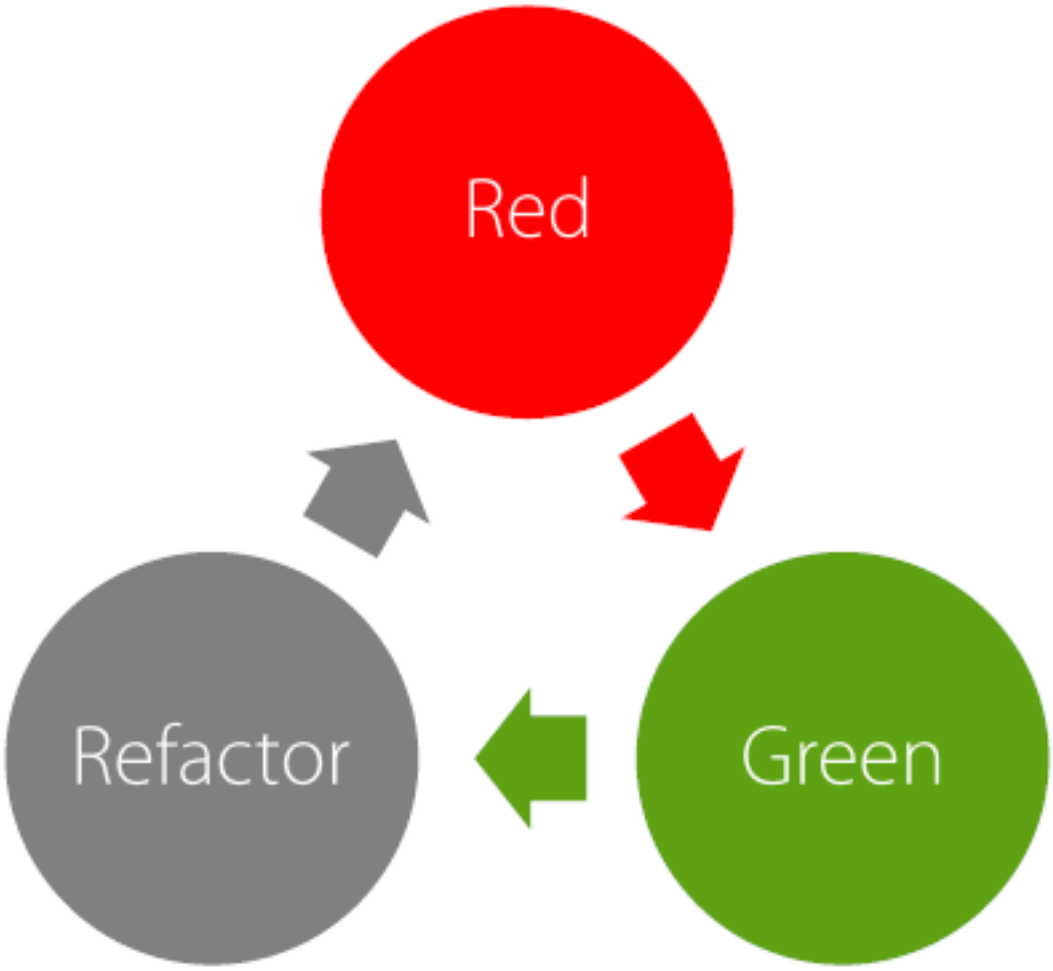
```
ClassMethod Square(pValue As %Integer) As %Integer
{
  Quit pValue * pValue
}

}
```

y volvamos a ejecutar nuestras pruebas:



Ahora todo va bien... Este es el ciclo de TDD, en pequeños pasos.



En este punto te estarás preguntando: ¿por qué debo seguir estos pasos? ¿Por qué tengo que ver fallar la prueba?
He trabajado en equipos que escribían el código de producción y después escribían las pruebas. Pero prefiero seguir estos pequeños pasos por los siguientes motivos:
Uncle Bob (Robert C. Martin) dijo que escribir pruebas después de escribir el código no es TDD, sino que en realidad se le podría llamar "una pérdida de tiempo".
Otro detalle es que, al ver fallar la prueba y luego ver que se pasa, estoy probando la propia prueba.
Tu prueba no deja de ser un código, y también puede tener errores. Y la forma de ponerlo a prueba es garantizar que falla y pasa cuando debe fallar y pasar. Esta es la forma de "probar la prueba" y asegurarte de que no tienes falsos positivos.

cosFaker

Para escribir buenas pruebas, puede que primero tengas que generar datos de prueba. Una forma de hacer esto es generar un volcado de datos y usarlo en tus pruebas.
Otra forma es usar cosFaker para generar fácilmente datos falsos cuando los necesites. https://openexchange.intersystems.com/package/CosFaker

Solo necesitas descargar el archivo xml y luego ir a Management Portal -> System Explorer -> Classes -> Import. Elige el archivo xml file a importar, o arrastra el archivo en Studio.
También puedes importarlo usando Terminal.

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

Traducciones (Localization)

cosFaker añadirá archivos de localización en la carpeta de aplicación CSP predeterminada. Por ahora solo hay dos idiomas: inglés y portugués de Brasil.
El idioma de los datos se elige de acuerdo con la configuración de tu Caché.
La localización de cosFaker es un proceso en desarrollo. Si quieres ayudar, no dudes en crear un proveedor localizado para tu propio idioma y envíar una pull request.
Con cosFaker puedes generar textos aleatorios como palabras, párrafos, números telefónicos, nombres, direcciones, direcciones de correo electrónico, precios, nombres de productos, fechas, códigos de color hexadecimales, etc.

Todos los métodos se agrupan por asuntos en clases, es decir, para generar una Latitud, debes llamar al método Latitude en la clase Address

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

-37.6806
```

También puedes generar un Json para tus pruebas

```
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"
}
```

Esta es la lista completa de las clases y métodos de **cosFaker**:

- **cosFaker.Address**
  - StreetSuffix
  - StreetPrefix
  - PostCode
  - StreetName
  - Latitude
    - *Output:* -54.7274
  - Longitude
    - *Output:* -43.9504
  - Capital(Location = “ ”)
  - State(FullName = 0)
  - City(State = “ ”)
  - Country(Abrev = 0)
  - SecondaryAddress
  - BuildingNumber
- **cosFaker.App**
  - FunctionName(Group= “ ” , Separator = “ ”)
  - 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**
    - 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
    - CreditCard(Type = “ ”)
        - *Output:* 3476-581511-6349
    - BitcoinAddress(Min = 24, Max = 34)
        - *Output:* 1WoR6fYvsE8gNXkBkeXvNqGECPUZ
- **cosFaker.Game**
    - MortalKombat
        - *Output:* Raiden
    - StreetFighter
        - *Output:* Akuma
    - Card(Abrev = 0)
        - *Output:* 5 of Diamonds
- **cosFaker.Internet**
    - UserName(FirstName = “ ” , LastName = “ ”)
    - Email(FirstName = “ ” , LastName = “ ” , Provider = “ ”)
    - Protocol
        - *Output:* http
    - DomainWord

    - DomainName
    - Url
    - Avatar(Size = “ ”)
        - *Output:* http://www.avatarpro.biz/avatar?s=150
    - 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 = “ ” // JSON template string to create data)
        - *Parameter Example*: "{dates:'5 date'}"
        - *Output*: {"dates":["2019-02-19","2019-12-21","2018-07-02","2017-05-25","2016-08-14"]}
- **cosFaker.Job**
    - Title
    - Field
    - Skills
- **cosFaker.Lorem**
    - Word
    - Words(Num = “ ”)
    - Sentence(WordCount = “ ” , Min = 3, Max = 10)
        - *Output:* Sapiente et accusamus reiciendis iure qui est.
    - Sentences(SentenceCount = “ ” , Separator = “ ”)
    - Paragraph(SentenceCount = “ ”)
    - Paragraphs(ParagraphCount = “ ” , Separator = “ ”)
    - Lines(LineCount = “ ”)
    - Text(Times = 1)
    - Hipster(ParagraphCount = “ ” , Separator = “ ”)
- **cosFaker.Name**
    - FirstName(Gender = “ ”)
    - LastName
    - FullName(Gender = “ ”)
    - Suffix
- **cosFaker.Person**
    - cpf(Pretty = 1)
        - CPF is the Brazilian Social Security Number
        - *Output:* 469.655.208-09
- **cosFaker.Phone**
    - PhoneNumber(Area = 1)
        - *Output:* (36) 9560-9757
    - CellPhone(Area = 1)
        - *Output:* (77) 94497-9538
    - AreaCode
        - *Output:* 17
- **cosFaker.Pokemon**
    - 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
    - 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
    - Featherweight(Country = “ ”)
        - *Output:* Yair Rodriguez
    - Middleweight(Country = “ ”)
        - *Output:* Elias Theodorou
    - Welterweight(Country = “ ”)
        - *Output:* Charlie Ward
    - Lightweight(Country = “ ”)
        - *Output:* Tae Hyun Bang
    - Bantamweight(Country = “ ”)
        - *Output:* Alejandro Pérez
    - Flyweight(Country = “ ”)
        - *Output:* Ben Nguyen
    - Heavyweight(Country = “ ”)
        - *Output:* Francis Ngannou
    - LightHeavyweight(Country = “ ”)
        - *Output:* Paul Craig
    - Nickname(Fighter = “ ”)
        - *Output:* Abacus

Vamos a crear una clase para el usuario con un método que devuelva su nombre de usuario, es decir, FirstName concatenado con 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)
}

}
```

```
    TDD.Math passed
    TDD.User begins ...
      TestUsername() begins ...
LogStateStatus:0:TestUsername: ERROR #5002: ObjectScript error: <COMMAND>zTestUsername+6^TDD.User.1 *Functi
on must return a value at zUsername+1^Production.User.1  <<==== **FAILED**
test:TDD.User:TestUsername:
      TestUsername failed
    TDD.User failed
  test failed

Use the following URL to view the result:
http://10.0.0.1:52773/csp/sys/%25UnitTest.Portal.Indices.cls?Index=19&$NAMESPACE=DEV
```

```
Class Production.User Extends %RegisteredObject
{

Property FirstName As %String;
Property LastName As %String;

Method Username() As %String
{
  Quit ..FirstName _ "." _ ..LastName
}


}
```

```
    TDD.Math passed
    TDD.User begins ...
      TestUsername() begins ...
        AssertEquals:user.Username()== firstName _ "." _ lastName (passed)
        LogMessage:Duration of execution: .001561 sec.
      TestUsername passed
    TDD.User passed
  test passed

Use the following URL to view the result:
http://10.0.0.1:52773/csp/sys/%25UnitTest.Portal.Indices.cls?Index=20&$NAMESPACE=
All PASSED
```

Ahora añadimos una fecha de vencimiento de la cuenta y la validamos.

```
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())
}

}
```

```
    TDD.User begins ...
      TestUsername() begins ...
        AssertEquals:user.Username()== firstName _ "." _ lastName (passed)
        LogMessage:Duration of execution: .004456 sec.
      TestUsername passed
      TestWhenIsNotExpired() begins ...
LogStateStatus:0:TestWhenIsNotExpired: ERROR #5002: ObjectScript error: <COMMAND>zTestWhenIsNotExpired+3^TD
D.User.1 *Function must return a value at zExpired+1^Production.User.1  <<==== **FAILED**
test:TDD.User:TestWhenIsNotExpired:
      TestWhenIsNotExpired failed
    TDD.User failed
  test failed
```

Hacemos relación.

```
Method Expired() As %Boolean
{
  Quit ($system.SQL.DATEDIFF("dd", ..AccountExpires, +$Horolog) > 0)
}
```

```
    TDD.User begins ...
      TestUsername() begins ...
        AssertEquals:user.Username()== firstName _ "." _ lastName (passed)
        LogMessage:Duration of execution: .003934 sec.
      TestUsername passed
      TestWhenIsNotExpired() begins ...
        AssertNotTrue:user.Expired() (passed)
        LogMessage:Duration of execution: .000106 sec.
      TestWhenIsNotExpired passed
    TDD.User passed
  test passed
```

Ahora vamos a probar cuando expira la cuenta.

```
Method TestWhenIsExpired() As %Status
{
  Set user = ##class(Production.User).%New(),
    user.AccountExpires = ##class(cosFaker.Dates).Backward(40)
  Do $$$AssertTrue(user.Expired())
}
```

```
    TDD.User begins ...
      TestUsername() begins ...
        AssertEquals:user.Username()== firstName _ "." _ lastName (passed)
        LogMessage:Duration of execution: .00448 sec.
      TestUsername passed
      TestWhenIsExpired() begins ...
        AssertTrue:user.Expired() (passed)
        LogMessage:Duration of execution: .000105 sec.
      TestWhenIsExpired passed
      TestWhenIsNotExpired() begins ...
        AssertNotTrue:user.Expired() (passed)
        LogMessage:Duration of execution: .000072 sec.
      TestWhenIsNotExpired passed
    TDD.User passed
  test passed

Use the following URL to view the result:
http://10.0.0.1:52773/csp/sys/%25UnitTest.Portal.Indices.cls?Index=40&$NAMESPACE=DEV
All PASSED
```

Y todo está verde...

Sé que estos ejemplos son un poco tontos, pero de esta forma conseguirás que no solo el código sea sencillo, sino también el diseño de la clase.

Conclusión

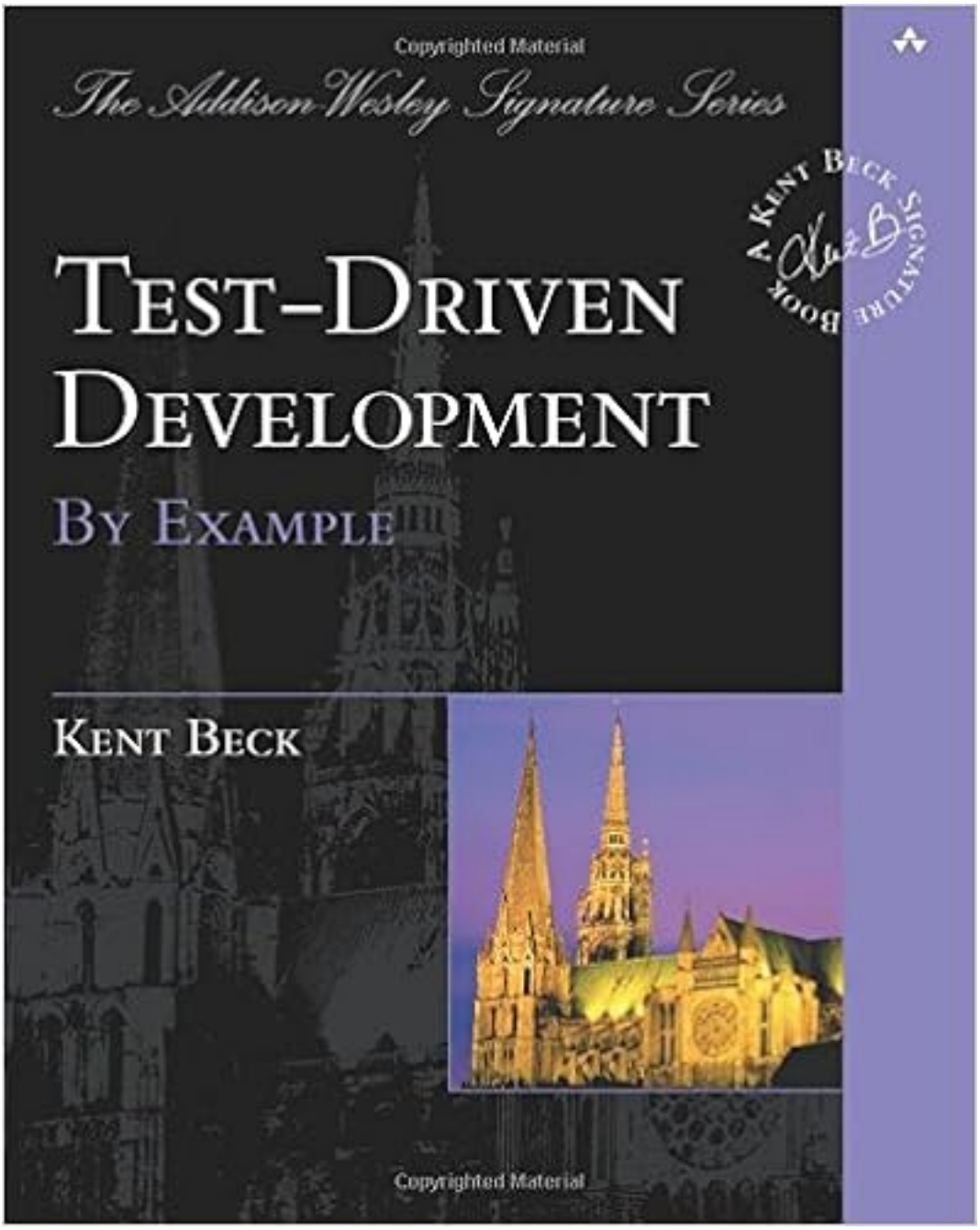
En este artículo hemos aprendido un poco más sobre el Desarrollo Basado en Pruebas (Test Driven Development / TDD) y cómo usar la clase %UnitTest. También tratamos cosFaker y cómo generar datos falsos para pruebas.

Hay mucho más para aprender sobre pruebas y TDD, cómo usar estas prácticas con código legacy, pruebas de integración, pruebas de aceptación (ATDD), bdd, etc. Si quieres saber más sobre estos temas, te recomiendo muy especialmente estos dos libros:

Test Driven Development Teste e design no mundo real com Ruby - Mauricio Aniche. No sé si este libro tiene una versión en inglés o español. Hay ediciones para Java, C#, Ruby y PHP. Este libró me explotó la cabeza de lo

bueno que es.

Y, por supuesto, el libro Test Driven Development by Example, de Kent Beck.



No dudes en escribir tus comentarios o preguntas.
Es todo, amigos.

#Prueba #Caché #InterSystems IRIS
Ir a la aplicación en InterSystems Open Exchange

URL de fuente:https://es.community.intersystems.com/post/breve-introducci%C3%B3n-al-desarrollo-basado-en-pruebas-con-cach%C3%A9-y-cosfaker