Telefónica

NTT DATA

Capacitación Pruebas Unitarias (xUnit)

30 Noviembre 2022 – Sesión 9

FUTURE AT HEART

Contenido

Modulo I – Conceptos Básicos

- ¿Qué es una prueba unitaria?
- Otros niveles de prueba
- Conociendo las pruebas unitarias
- Tecnicas de diseño de casos de prueba
- Principios FIRST

Modulo II – Test Driven Development

- Definición
- Desarrollo Ágil – Características
- Ciclo de desarrollo TDD
- Como escribir código que se pueda probar
- Cobertura de código
- Ventajas/Desventajas

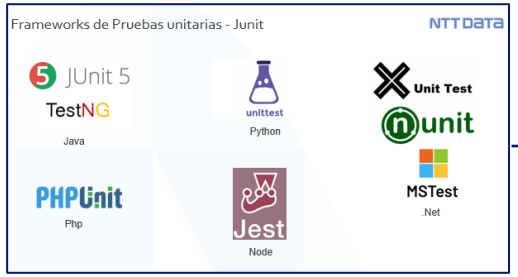
Modulo III – xUnit

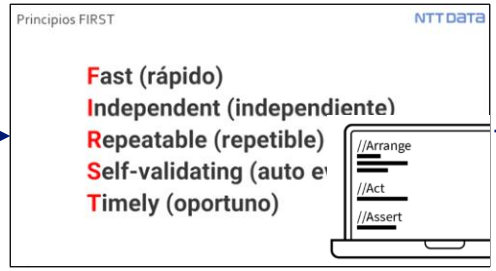
- ¿Qué es xUnit?
- ¿Por qué xUnit?
- ¿Cómo funciona xUnit?
- Conceptos básicos
- Escenarios con xUnit
- Patrones de nombramiento de los escenarios
- Simulaciones
- Aserciones

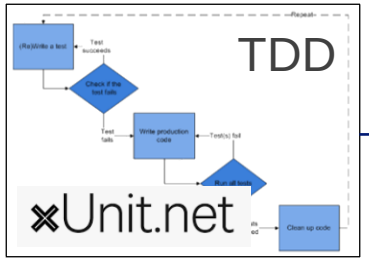
Modulo IV – Taller

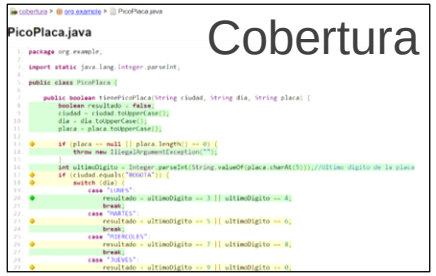
- Preparación del Entorno
- Ejemplo práctico
- Creando casos de prueba
- Ejecución de casos
- Mutation Testing
- Conclusiones/Recomendaciones

En capítulos anteriores...







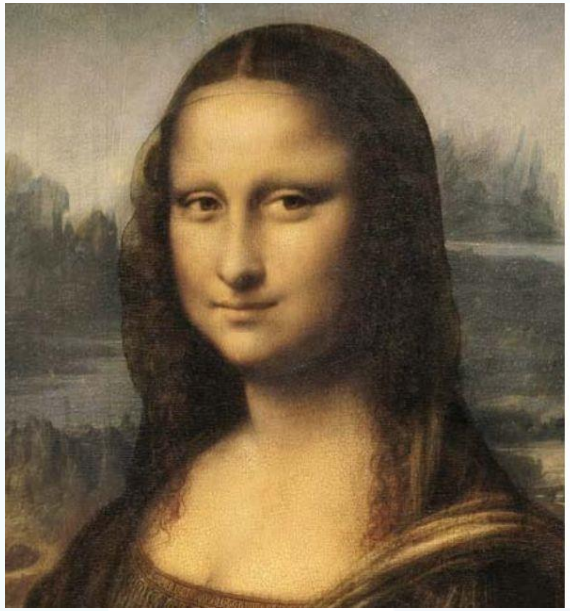


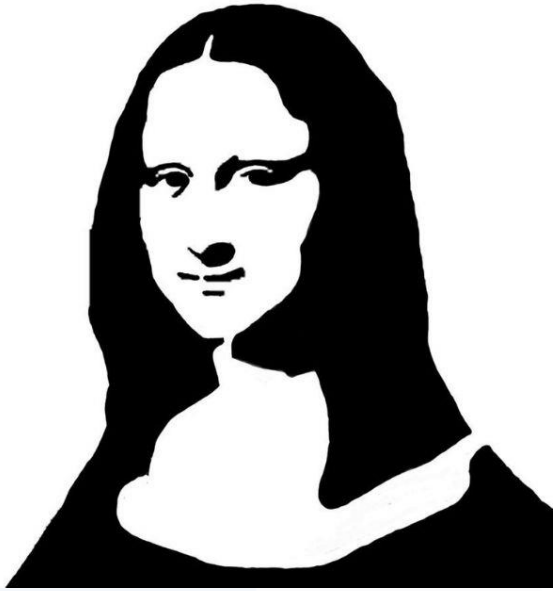
https://github.com/oacarrillo/curso-xUnit

Mocks

El término mock, técnicamente en español significa burlarse pero uno de sus sinónimos es “mimic” el cual significa mímica, que es lo mismo que imitar.

Las conexiones con bases de datos, operaciones contra el sistema de ficheros o interacciones con APIs externas en general pueden hacer más difícil que nuestros test unitarios sean realmente unitarios, ya que añaden una dependencia sobre la que no siempre vamos a tener control. Por otro lado, dicha dependencia causa a menudo problemas de velocidad de ejecución, lo que hace pesado ejecutar nuestra batería de test. Una manera de solucionar estos problemas es utilizando “**mocks**”, que no son más que objetos simulados que imitan el comportamiento de objetos reales.





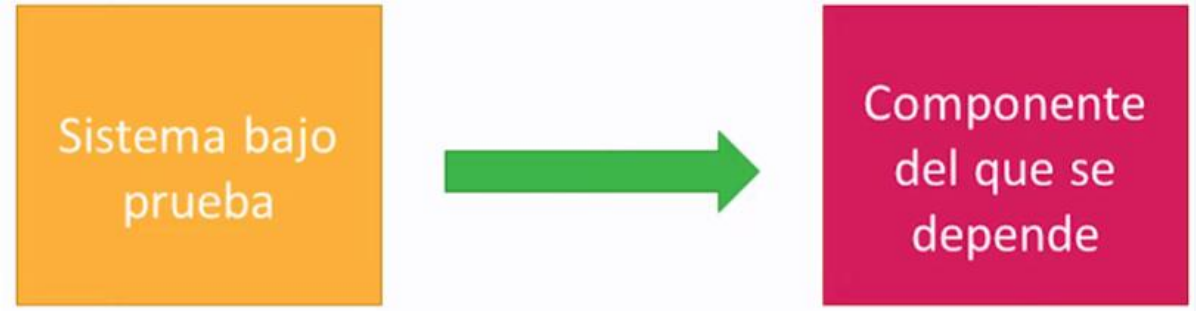
https://https://www.netmentor.es/Entrada/mock-unit-test-csharp/#.Y39513bMKiM
https://vuejsdevelopers.com/images/posts/jest_mock.jpg

Mocks en los niveles de pruebas

Los mocks se usan tanto para las pruebas unitarias , como en pruebas de integración. Una de las principales razones de su uso:

- Disminuir tiempos de respuesta de los componentes dependientes,
- Los ambientes de pruebas de estos sistemas externos, al no tener control alguno sobre ellos, no podemos controlar el resultado de las llamados. Así es muy poco probable que las pruebas pasen.
- También es posible que no exista disponibilidad del sistema dependiente en ambiente de desarrollo.
- Puede ser que sea costoso acceder a estos sistemas (Tarifas Cloud)

(Base de datos, Sistema de archivos, API´s publicas)



https://https://www.netmentor.es/Entrada/mock-unit-test-csharp/#.Y39513bMKiM

Principios FIRST - I

NTT DATA

Independent (independiente)

En las pruebas unitarias, el objeto de prueba no debe depender de otra unidad de código. Esto no significa que no requiera de datos para poder ejecutar ese código, pero esto se realizará a través de **mocks o stubs**, que vienen a ser «objetos falsos» creados específicamente para realizar las pruebas. Pero, en ningún caso, utilizaremos otras partes del código que no forme parte del unit test.

Dummy : Objetos ficticios que se pasan al método pero no se usan. Por lo general, solo se usan para rellenar listas de parámetros.
Fake : Objetos falsos que toman un atajo para cumplir suplir un valor de entrada o componente. Ej: Base de datos en memoria.
Stubs : Respuestas enlatadas para un solo contexto de prueba. Ej: Un parámetro constante (Cedula, Tipo)
Spies : Proporcionan información especifica reaccionando de acuerdo a como fueron llamados. Ej: Ingreso una cedula y me retorna # de líneas
Mocks : Objetos pre programados complejos (Subrutinas).

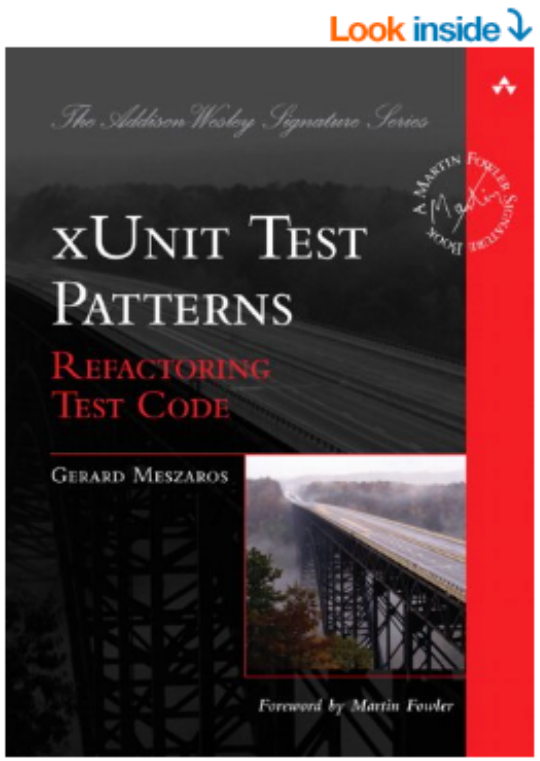
Cada prueba unitaria debe ser independiente de otra

- Se pueden ejecutar las pruebas en diferente orden
- El resultado no se ve alterado por el orden de ejecución de la prueba
- Su nombre lo dice prueba unitaria (No de integración)

https://martinfowler.com/articles/mocksArentStubs.html#TheDifferenceBetweenMocksAndStubs

Mocks - Libro

Tienda Kindle › eBooks Kindle › Computadoras y Tecnología



xUnit Test Patterns: Refactoring Test Code (Addison-Wesley Signature Series (Fowler)) 1st Edición, Edición Kindle

de Gerard Meszaros (Author) | Formato: Edición Kindle

★★★★½ 63 calificaciones

Ver todos los formatos y ediciones

Kindle	Pasta dura
US$47.99	US$97.01 - US$127.99
Leer con nuestra Aplicación gratuita	9 Usado de US$93.00 5 Nuevo de US$127.99

Automated testing is a cornerstone of agile development. An effective testing strategy will deliver new functionality more aggressively, accelerate user feedback, and improve quality. However, for many developers, creating effective automated tests is a unique and unfamiliar challenge.

Leer más

Seguir al autor

Gerard Meszaros — Seguir

https://https://www.netmentor.es/Entrada/mock-unit-test-csharp/#.Y39513bMKiM

Mocks en Java

Mockito es un framework que permite escribir hermosas pruebas con una API limpia y simple. Mockito no te da resaca porque las pruebas son muy legibles y producen errores de verificación limpios. Lea más sobre características y motivaciones .

- La comunidad masiva de StackOverflow votó a Mockito como el mejor marco de simulación para Java. A pesar de que StackOverflow evita las preguntas que probablemente generen debates emocionales, el hecho es que Mockito tiene la mayor cantidad de votos .
- Las 10 principales bibliotecas de Java en todas las bibliotecas, no solo las herramientas de prueba. A finales de 2013 se hizo un análisis de 30.000 proyectos de GitHub. Aunque Mockito alcanzó el número 9 en el informe principal, mockito-core y mockito-all son la misma herramienta y, por lo tanto, la posición real de Mockito es el número 4, superando a herramientas famosas como Guava o Spring. Trate este estudio como un indicador del gran impacto que Mockito tiene todos los días en las pruebas unitarias escritas en Java.
- Dan North , el creador de Behavior-Driven Development escribió esto en 2008:

> “Decidimos durante la conferencia principal que deberíamos usar JUnit 4 y Mockito porque creemos que son el futuro de TDD y burlas en Java”

https://site.mockito.org/

Mocks en C#

Moq (pronunciado "Mock-you" o simplemente "Mock") es la única biblioteca de simulación para .NET desarrollada desde cero para aprovechar al máximo los árboles de expresión y las expresiones lambda de .NET Linq, lo que la convierte en la biblioteca más productiva, con seguridad de tipos y Biblioteca de simulación compatible con la refactorización disponible. Y admite interfaces simuladas, así como clases. Su API es extremadamente simple y directa, y no requiere ningún conocimiento previo o experiencia con conceptos de simulación.

La biblioteca se creó principalmente para desarrolladores que actualmente no usan ninguna biblioteca de simulación (o están disgustados con las complejidades de alguna otra implementación) y que normalmente escriben manualmente sus propias simulaciones (con más o menos "elegancia"). La mayoría de los desarrolladores en esta situación también son bastante pragmáticos y se adhieren al TDD estatal (o clásico). Es el resultado de sentir que la barrera de entrada de otras bibliotecas de simulación es un poco alta, y es posible un enfoque más simple, ligero y elegante. Moq logra todo esto aprovechando al máximo las características elegantes y compactas del lenguaje C# y VB conocidas colectivamente como LINQ (no son solo para consultas, como lo indica el acrónimo).


NTT DATA
GRACIAS
FUTURE
AT HEART