Unit-тестирование бизнес-логики

№ урока: 2 **Курс:** Тестирование ASP.NET Core MVC приложений

Средства обучения: Visual Studio 2019 Community Edition

Обзор, цель и назначение урока

Вспомнить, что такое unit-тест. Разобраться со структурой unit-тестов, рассмотреть признаки хорошего unit-теста и типичные ошибки. Рассмотреть функциональную архитектуру в применении к unit-тестам бизнес-логики. Рассмотрение демо проекта.

Изучив материал данного занятия, учащийся сможет

- Разбираться в том, как правильно писать unit-тесты для бизнес-логики.
- Разбираться в том, какие признаки хорошего unit-теста.
- Разобраться в том, как избежать типичных ошибок при реализации unit-теста.
- Рассмотреть на практике, как и с чего начинать писать unit-тесты для бизнес-логики.

Содержание урока

1. Что такое unit-тест
2. Признаки хорошего unit-теста
3. Структура unit-теста
4. Что может проверять unit-тест
5. Типичные ошибки в unit-тестах
6. Обзор логики тестового проекта
7. Демо

Резюме

- **Unit-тест** – быстрая изолированная проверка наименьшего возможного логически смыслового блока программного кода.
- **Быстрота Unit-теста** напрямую проистекает из его изолированности и относительно малого размера, также благодаря изолированности unit-тесты отлично распараллеливаются.
- **Изолированность unit-тестов –** это то, что делает их unit-тестами по их природе, никаких реальных манипуляций с внешним неконтролируемым миром не производится, используются абстракции, которые легко мокаются.
- **Unit-тесты можно делить** структурно, когда мы тестируем отдельные методы, а также классы, и по поведению, когда проверяется некий атомарный аспект поведения, тесно связанный с бизнес-логикой, при этом он может затрагивать сразу несколько методов или классов.
- **Хороший unit-тест** – предсказуемый, защищен от регрессии, устойчив к рефакторингу и понятный.
- **Ваш unit-тест уже на половину хороший**, если он соблюдает принципы Single Responsibility & Dependency Inversion.

CyberBionic systematics

CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

- **Невероятно важное свойство unit-теста** — это его детерминированность, любые источники случайностей (случайные числа, строки, зависимость от времени) должны быть повторяемыми, предсказуемыми, контролируемыми.
- **Структура unit-теста** - triple A, Arrange, Act, Assert.
- **Секция Arrange** – предусловия, необходимые для выполнения секции Act
- **Секция Act** – само действие, которое мы проверяем в тесте.
- **Секция Assert** – конкретные проверки, их может быть несколько, в случае их независимости, когда одновременно проверяются, допустим, выходной результат и состояние, можно использовать специальный метод под названием Assert.Multiple.
- **Unit-тест** может проверять выходной результат, состояние, коммуникацию.
- **Выходной результат –** то, что возвращает тестируемый метод.
- **Состояние –** больше характерно для проверки последствий выполнения метода, для проверки того, что метод изменил какое-либо состояние (в классе, глобальное). Не рекомендуется делать глобальные состояние, так как их достаточно тяжело проверять.
- **Коммуникация –** метод, который проверяется, может вызывать остальные методы, используя зависимости, которые были переданы в сам метод, или в конструктор класса. Для проверки таких взаимодействий идеально подходят моки, они предоставляют все необходимые для этого инструменты, например, такие, как задание необходимого поведения на лету и подсчёт количества вызовов нужного мокированого метода. Проверка коммуникации является важной частью проверки внешних сервисов. Как раз таки проверка коммуникации используется при тестирования взаимодействии с абстракциями внешних сервисов.
- **Типичные ошибки в unit-тестах** – непредсказуемость и зависимость.
- **Функциональная архитектура** – внешний мир зависит от чистого ядра приложения.
- **С чего начать описывать бизнес-логику проекта?** Продумайте базовые составляющие вашей логики Entity & ValueObject. Продумайте взаимодействие ваших сущностей с помощь доменных сервисов. Любое взаимодействие внешней логики с внешним миром — это чистая абстракция, которая реализуется на другом уровне. Сохранение бизнес-логики чистой и не привязанной к конкретной реализации приложения, позволит радикально упростить unit-тестирование.
- **С чего начать unit-тестирование бизнес-логики?** Выберите логику для тестирования. Напишите предусловия, необходимые для тестирования выбранной логики. Обеспечьте детерминированность выполнения тестов. Начните с написания тестов для успешных случаев. Не забывайте про классы эквивалентности, для адекватной проверки всевозможных вариантов выполнения логики. Не обязательно проверять абсолютно все комбинации, да и в большинстве случаев не получится, достаточно проверить самые вероятные.

Закрепление материала

- Что такое unit-тест?
- Опишите каждый термин из определения.
- Назовите признаки хорошего unit-теста.
- Что может проверять unit-тест?
- Какая типичная структура unit-теста?
- Опишите дилемму WhiteBox vs BlackBox.
- Какие типичные ошибки в unit-тестах?

CyberBionic systematics
CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

- Подумайте о том, как бы вы писали unit-тесты в случае анемичной модели данных.

Дополнительное задание

Задание
Выберите какой-нибудь метод из стандартной библиотеки языка программирования C# и напишите для него unit-тесты. Это отличный способ изучения на практике BlackBox тестирования и правильного учёта пограничных случаев в тестировании.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные понятия, рассмотренные на уроке.

Задание 2
Дополнительно найдите сопутствующую информацию по теме. Составьте сводную таблицу по основным понятиям, найдите схожие и отличающиеся моменты в описаниях, постарайтесь сформулировать своё понимание данной темы.

Задание 3
Выберите или продумайте с нуля некоторую бизнес-логику и реализуйте эту бизнес-логику на языке программирования C#. Эта реализованная бизнес-логика будет ядром вашего приложения. Обязательным требованием является то, что эта логика НЕ должна зависеть от фреймворков вывода данных (WinForms, WPF, ASP. NET Core, UWP, Xamarin Forms как пример) и в общем случае бизнес-логика не должна зависеть от внешнего мира.
Допускается использование любых библиотек, упрощающих разработку бизнес-логики.

Задание 4
Напишите unit-тесты на бизнес-логику из предыдущего задания. Используйте любой тестовый фреймворк на языке программирования C# и любые библиотеки, помогающие написанию unit-тестов.

Рекомендуемые ресурсы

Test ASP.NET Core MVC apps
https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/test-asp-net-core-mvc-apps

Unit testing in .NET Core and .NET Standard
https://docs.microsoft.com/en-us/dotnet/core/testing

Test naming
https://enterprisecraftsmanship.com/posts/you-naming-tests-wrong/

Functional architecture: a definition
https://blog.ploeh.dk/2018/11/19/functional-architecture-a-definition/

Entity vs Value Object
https://enterprisecraftsmanship.com/posts/entity-vs-value-object-the-ultimate-list-of-differences/

CyberBionic systematics
CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com