C# / .NET

(chatbot: разработка слоя доступа к данным, модульное тестирование)

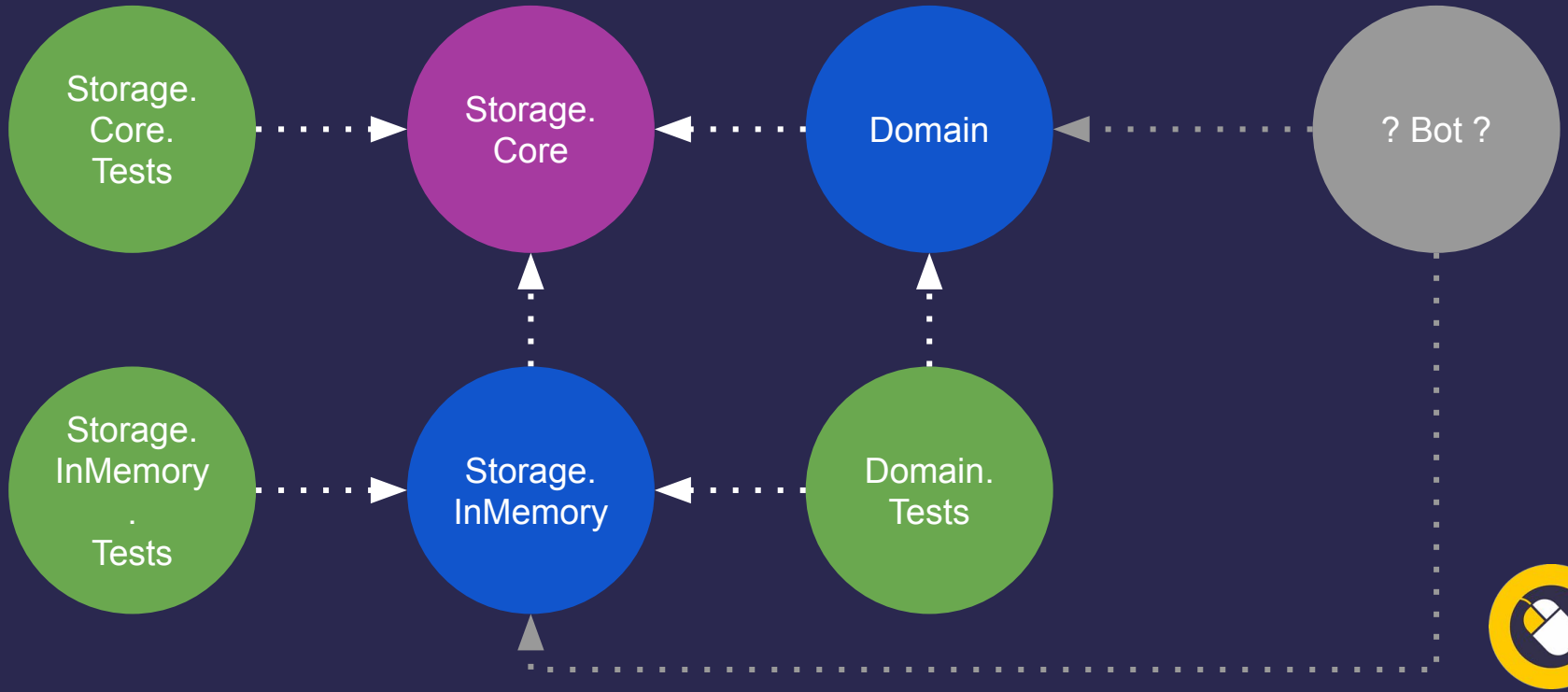
Андрей Голяков

Взаимосвязи между компонентами

Основные сборки, отвечающей за логику работы собственно ремайндера:

- Reminder.Storage.Core
  - Библиотека с описанием интерфейсов и классов, которые будут использоваться конкретными реализациями хранилища данных
  - Ей в пару будет создана сборка Reminder.Storage.Core.Tests для тестов логики классов, если она будет.
- Reminder.Storage.InMemory
  - Библиотека с реализацией хранилища данных в памяти. Реализует все интерфейсы Reminder.Storage.Core.
  - Ей в пару будет создана сборка Reminder.Storage.InMemory.Tests для тестов логики хранилища.
- Reminder.Domain
  - Основная библиотека логики.

Взаимосвязи между компонентами



Библиотека Reminder.Storage.Core

- `Reminder.Storage.Core.ReminderItem` класс
  - Определяет сущность единственной записи будильника которая будет лежать в хранилище данных.
  - Должен содержать, как минимум, необходимые по заданию поля с датой и текстом.
  - Должен также содержать уникальный идентификатор для быстрого поиска внутри хранилища.
- `Reminder.Storage.Core.ReminderItemStatus` перечисление
  - Определяет возможные статусы для единственной записи будильника которая будет лежать в хранилище данных.
  - Как минимум отправлено/не отправлено.
- `Reminder.Storage.Core.IReminderStorage` интерфейс
  - Описывает интерфейс хранилища данных.
  - Должен, как минимум, содержать методы по записи данных в хранилище и получению по данных из хранилища.

Библиотека Reminder.Storage.InMemory

- `Reminder.Storage.InMemory.InMemoryReminderStorage` класс
  - Реализует интерфейс хранилища данных на базе коллекции в оперативной памяти, например, словаря — Dictionary<Guid, ReminderItem>.
  - Внутренний словарь рекомендуется объявлять с модификатором internal, чтобы в библиотеке тестов он был доступен для проверки методов нашего класса через использование InternalsVisibleTo (см. слайд с заголовком “InternalsVisibleTo для доступа к internal-членам”).

Модульное тестрование (Unit tests)

Модульные тесты позволяют разработчикам и тест-инженерам быстро искать логические ошибки в реализации классов.

Для модульных тестов создаются отдельные проекты модульных тестов, как правило, по одному проекту на один тестируемый проект в солюшене.

Существует несколько форматов написания модульных тестов для платформы .NET Core:

- MSTest Test Project (.NET Core)
- NUnit Test Project (.NET Core)
- xUnit Test Project (.NET Core)

Они отличаются синтаксическими элементами оформления тестов и (немного) встроенными возможностями (см. таблицу сравнения).

Базовые атрибуты тестовых проектов

Обычно в одном классе теста покрывается одна единица тестирования. Это может быть класс, или даже метод.

Типичный файл проекта модульных тестов - это обычный класс, размеченный специальными атрибутами:

```
[TestClass] <---------------- Атрибут тестового класса
public class UnitTest1
{
    [TestMethod] <------------ Атрибут тестового метода
    public void TestMethod1()
    {
        ClassToBeTested obj = new ClassToBeTested();
        Assert.IsNull(obj); <-------- Тестовое утверждение
    }                                 Это и есть элементарный тест
}
```

Расширенные атрибуты тестовых проектов

Атрибут [DataTestMethod] представляет набор тестов, которые выполняют один и тот же код, но имеют разные входные аргументы. С помощью атрибута [DataRow] можно указать значения для этих входных аргументов.

```
[DataTestMethod]
[DataRow(0)]
[DataRow(1)]
[DataRow(int.MaxValue)]
public void ReturnFalseGivenValuesLessThan2(int value)
{
    ClassToBeTested cls = new ClassToBeTested();
    bool result = cls.IsPositive(value);

    Assert.IsTrue(result);
}
```

Расширенные атрибуты тестовых проектов

Также для сложных сценариев можно использовать методы настройки среды тестирования, которые будут подготавливать окружение для запуска.

- `[TestInitialize]` – Метод помеченный этим атрибутом будет запускаться перед каждым тестом в классе
- `[TestCleanup]` – Метод помеченный этим атрибутом будет запускаться после каждого теста в классе
- `[ClassInitialize]` – Метод помеченный этим атрибутом будет запускаться перед запуском первого теста в классе (и перед методом помеченным атрибутом `[TestInitialize]`, если такой есть)
- `[ClassCleanup]` – Метод помеченный этим атрибутом будет запускаться после запуска последнего теста в классе (и после метода помеченного атрибутом `[TestInitialize]`, если такой есть)

InternalsVisibleTo для доступа к internal-членам

Чтобы обеспечить доступ к своим internal-членам (например для тестирования их в проектах unit-тестов) необходимо добавить в начало любого файла *.cs такие строчки:

```
using System.Runtime.CompilerServices;
[assembly: InternalsVisibleTo("Reminder.Storage.InMemory.Tests")]
```

Я рекомендую для этого создавать отдельный файл ExportInternals.cs в котором и будут находиться только эти строки.

Полезные ссылки

- Документация Microsoft: Модульное тестирование кода C# с использованием MSTest и .NET Core: https://docs.microsoft.com/ru-ru/dotnet/core/testing/unit-testing-with-mstest
- Документация Microsoft: Модульное тестирование C# в .NET Core с использованием dotnet test и xUnit: https://docs.microsoft.com/ru-ru/dotnet/core/testing/unit-testing-with-dotnet-test
- Документация Microsoft: Модульное тестирование кода C# с использованием NUnit и .NET Core: https://docs.microsoft.com/ru-ru/dotnet/core/testing/unit-testing-with-nunit
- Habr: Unit тесты на практике: https://habr.com/ru/post/191986

Самостоятельная работа

Необходимо добавить к проектам Reminder.Storage.Core и Reminder.Storage.InMemory по одному проекту MSTest Test Project (.NET Core) с названиями:

- Reminder.Storage.Core.Tests
- Reminder.Storage.InMemory.Tests

Покрыть тестами

- Свойство TimeToAlarm класса ReminderItem библиотеки Reminder.Storage.Core,
- Метод Add(ReminderItem item) класса ReminderStorage библиотеки Reminder.Storage.InMemory.

Домашняя работа

Завершить реализацию интерфейса IReminderStorage в библиотеке Reminder.Storage.InMemory и покрыть тестами код реализации в библиотеке Reminder.Storage.InMemory.Tests.

Спасибо за внимание.