
Сборка_
GoodLine
Оператор связи Кузбасса
Атвинта

Тестирование бизнес-логики в .NET

Швец Сергей,
директор по ИТ,
ManageBand

О себе

- 9 лет в профессиональной разработке:
  - PHP, Python, Ruby
  - Любимый и основной язык C#
  - Немного занимался машинным обучением

- Текущие интересы: архитектура ПО, маркетинг, продажи, бережливый стартап

Зачем нужны тесты?

1. Выявление регрессий
2. Новый функционал
3. Рефакторинг
4. Документация
5. TDD (Test Driven Development)

Препятствия

1. Большие инвестиции времени и сил
2. Нужно подстраивать код под тесты
3. Тесты нужно поддерживать
4. Сроки
5. ~~Лень~~

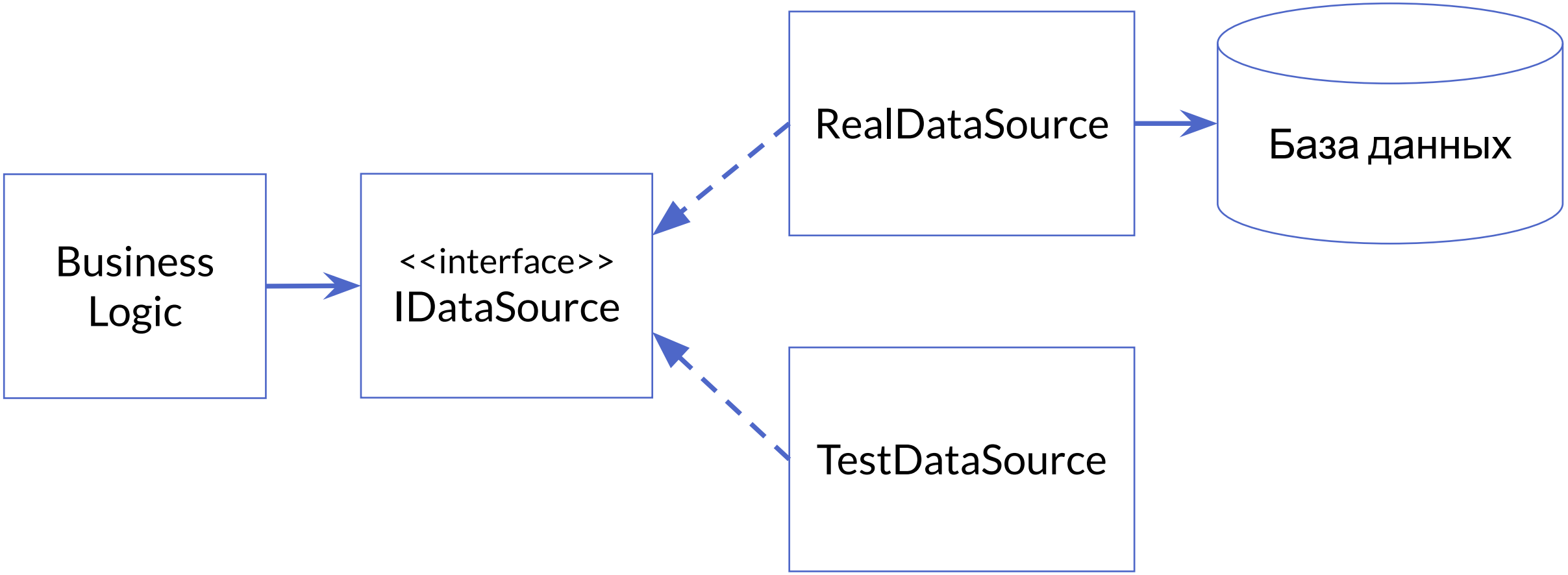
Case1.

Задача:	Обработка телеметрии от оборудования
Особенности:	Windows-сервис (автономность)
	Трудно понять, что произошла ошибка
	Высокая ответственность
Инструменты:	SqlServer, Postgresql
	C#, Linq2Db

```
class MessageProcessor
{
    private EFContext _db;
    private Configuration _configuration;
    public MessageProcessor(EFContext db)
    {
        _db = db;
    }

    public void Initialize()
    {
        _configuration = new Configuration();
        _configuration.Devices = _db.Devices.Include(x => x.Ports);
    }

    public void Process(RawData rawData)
    {
        var message = new Message();

        // Do some work to convert rawData to message...

        _db.Messages.Add(message);
        _db.SaveChanges();
    }
}
```

- Работа с БД через ORM
- Невозможно протестировать логику

Абстракция вместо реальной базы



```
interface IDataSource
{
    Configuration GetConfiguration();
    void SaveMessage(Message alarm);
}

class MessageProcessor
{
    private IDataSource _dataSource;
    private Configuration _configuration;
    public MessageProcessor(IDataSource dataSource)
    {
        _dataSource = dataSource;
    }

    private void Initialize()
    {
        _configuration = _dataSource.GetConfiguration();
    }

    public void Process(RawData rawData)
    {
        var message = new Message();

        // Do some work to convert rawData to message...

        _dataSource.SaveMessage(message);
    }
}
```

- Источник данных стал абстрактным (его можно подменить в тестах)
- Полностью протестирован MessageProcessor (15-20 зеленых тестов)
- Можно выкладывать в production

```
interface IDataSource
{
    Configuration GetConfiguration();
    void SaveMessage(Message alarm);
}

class MessageProcessor
{
    private IDataSource _dataSource;
    private Configuration _configuration;
    public MessageProcessor(IDataSource dataSource)
    {
        _dataSource = dataSource;
    }

    private void Initialize()
    {
        _configuration = _dataSource.GetConfiguration();
    }

    public void Process(RawData rawData)
    {
        var message = new Message();

        // Do some work to convert rawData to message...

        _dataSource.SaveMessage(message);
    }
}
```

Ошибка:

1. Изменилась схема данных

```
interface IDataSource
{
    Configuration GetConfiguration();
    void SaveMessage(Message alarm);
}

class MessageProcessor
{
    private IDataSource _dataSource;
    private Configuration _configuration;
    public MessageProcessor(IDataSource dataSource)
    {
        _dataSource = dataSource;
    }

    private void Initialize()
    {
        _configuration = _dataSource.GetConfiguration();
    }

    public void Process(RawData rawData)
    {
        var message = new Message();

        // Do some work to convert rawData to message...

        _dataSource.SaveMessage(message);
    }
}
```

Ошибка:

1. Изменилась схема данных

2. Ошибка при построении конфигурации (NRE)

```
interface IDataSource
{
    Configuration GetConfiguration();
    void SaveMessage(Message alarm);
}

class MessageProcessor
{
    private IDataSource _dataSource;
    private Configuration _configuration;
    public MessageProcessor(IDataSource dataSource)
    {
        _dataSource = dataSource;
    }

    private void Initialize()
    {
        _configuration = _dataSource.GetConfiguration();
    }

    public void Process(RawData rawData)
    {
        var message = new Message();

        // Do some work to convert rawData to message...

        _dataSource.SaveMessage(message);
    }
}
```

Ошибка:

1. Изменилась схема данных

2. Ошибка при построении конфигурации (NRE)

3. Ошибка при вставке данных (внешний ключ)

```
interface IDataSource
{
    Configuration GetConfiguration();
    void SaveMessage(Message alarm);
}

class MessageProcessor
{
    private IDataSource _dataSource;
    private Configuration _configuration;
    public MessageProcessor(IDataSource dataSource)
    {
        _dataSource = dataSource;
    }

    private void Initialize()
    {
        _configuration = _dataSource.GetConfiguration();
    }

    public void Process(RawData rawData)
    {
        var message = new Message();

        // Do some work to convert rawData to message...

        _dataSource.SaveMessage(message);
    }
}
```

Ошибка:

1. Изменилась схема данных
2. Ошибка при построении конфигурации (NRE)
3. Ошибка при вставке данных (внешний ключ)
4. Ошибка при вставке данных (ограничение столбца)

Плюсы

+ Чисто и красиво
+ Быстро работают

Минусы

- Не протестирован DataSource
- Нужно подстраивать код под тесты
- Не протестировать SQL запросы
- Не проверить особенности БД (ключи, ограничения, триггеры)
- Трудоемко реализовывать TestDataSource

Опрос. Есть ли тесты в бизнес-логике?

C#	PHP	Python
Нет	Да	Да
Нет	Да	
Да (но нет БД)		

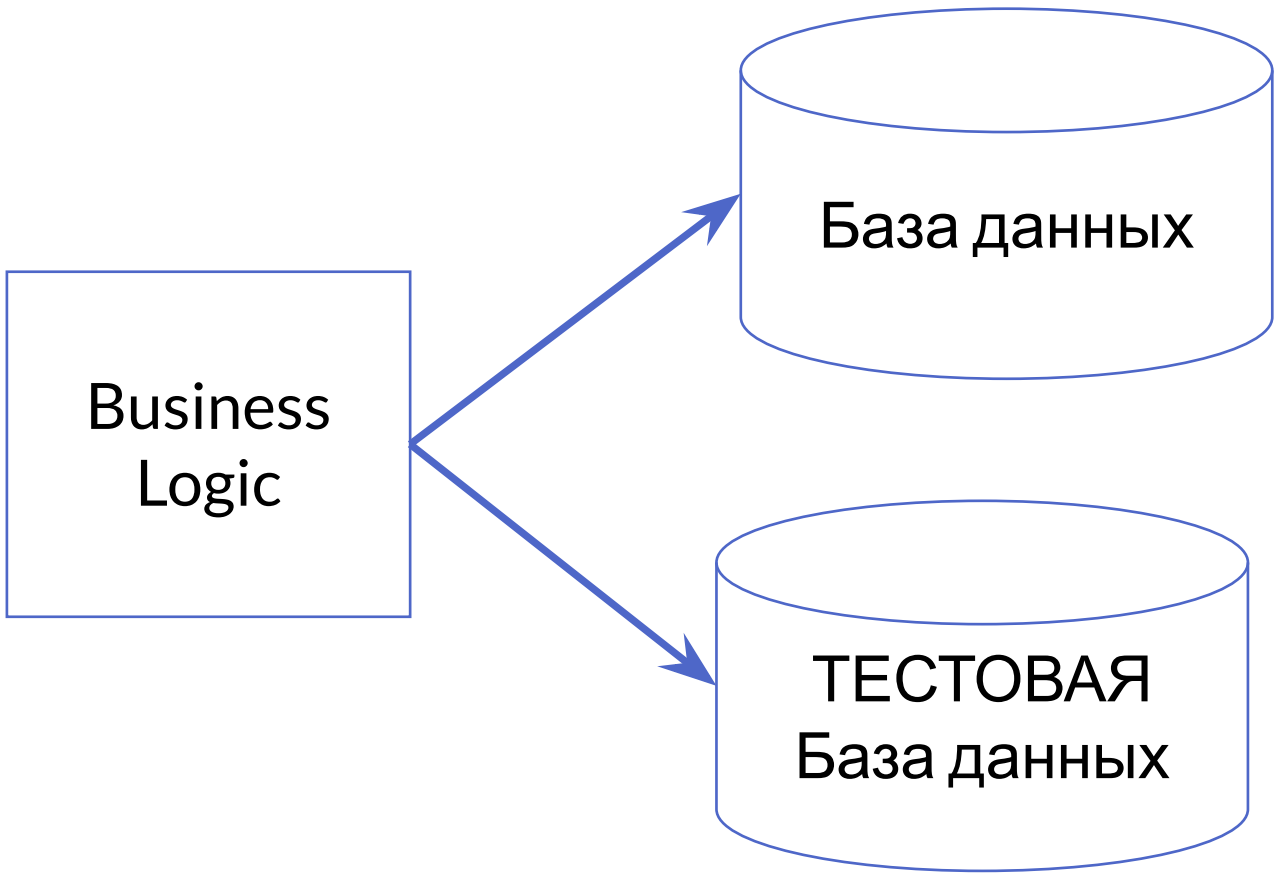
* выборка не репрезентативна

Опрос. Есть ли тесты в бизнес-логике?

C#	PHP	Python
Нет	Да	Да
Нет	Да	
Да (но нет БД)		

* выборка не репрезентативна

Тестовая база



1. Настоящие таблицы, триггеры, ограничения
2. Очищается перед каждым тестом
3. Загружаются общие данные (фикстуры)
4. В базе формируется сценарий
5. В базе проверяется результат

Case2.

Задача: Офлайн расчет остатков товара на планшете

Особенности: Очень много не очевидной логики

Рефакторинг (уже запущено и работает)

Работает в офлайн (сложно поймать ошибку и доставить обновление)

Инструменты: SQLite

C#, Xamarin.IOS, очень простая ORM

Решение

nuget

0. DbTest

Актуализирует структуру БД

Очищает перед каждым тестом

Заливает фикстуры

1. Фикстуры

Данные, которые не влияют на тесты, но без них не собрать тестовый сценарий

2. TestBuilder

Помогает собрать тестовый сценарий в базе данных

Фикстуры

```
public class Countries : IModelFixture<Country>
{
    public string TableName => "Countries";

    public static Country Scotland => new Country
    {
        Id = 1,
        Name = "Scotland",
        IsDeleted = false
    };

    public static Country Ireland => new Country
    {
        Id = 2,
        Name = "Ireland",
        IsDeleted = false
    };

    public static Country USA => new Country
    {
        Id = 3,
        Name = "USA",
        IsDeleted = false
    };
}
```

Фикстуры

```
public class Countries : IModelFixture<Country>
{
    public string TableName => "Countries";

    public static Country Scotland => new Country
    {
        Id = 1,
        Name = "Scotland",
        IsDeleted = false
    };

    public static Country Ireland => new Country
    {
        Id = 2,
        Name = "Ireland",
        IsDeleted = false
    };

    public static Country USA => new Country
    {
        Id = 3,
        Name = "USA",
        IsDeleted = false
    };
}
```

```
class Manufacturers : IModelFixture<Manufacturer>
{
    public string TableName => "Manufacturers";

    public static Manufacturer BrownForman => new Manufacturer
    {
        Id = 1,
        Name = "Brown-Forman",
        CountryId = Countries.USA.Id,
        IsDeleted = false
    };

    public static Manufacturer TheEdringtonGroup => new Manufacturer
    {
        Id = 2,
        Name = "The Edrington Group",
        CountryId = Countries.Scotland.Id,
        IsDeleted = false
    };

    public static Manufacturer Diageo => new Manufacturer
    {
        Id = 3,
        Name = "Diageo",
        CountryId = Countries.Scotland.Id,
        IsDeleted = false
    };

}
```

Тесты

```
public class RemainsTest
{
    [SetUp]
    public void SetUp()
    {
        World.InitDatabase();
    }

    [Test]
    public void CalculateRemainsForMoveDocuments()
    {
        /// ARRANGE
        var builder = new TestBuilder();

        // Income to remote storage
        var doc1 = builder.CreateDocument("15.01.2016 10:00:00", Storages.MainStorage, Storages.RemoteStorage);
        builder.AddGood(doc1, Goods.JackDaniels, 10);
        builder.AddGood(doc1, Goods.JohnnieWalker, 15);

        // Outcome from remote storage
        var doc2 = builder.CreateDocument("16.01.2016 20:00:00", Storages.RemoteStorage, Storages.MainStorage);
        builder.AddGood(doc2, Goods.JohnnieWalker, 7);

        /// ACT
        var remains = new RemainsService(World.GetContext()).GetRemainFor(Storages.RemoteStorage, new DateTime(2016, 02, 01));

        /// ASSERT
        Assert.AreEqual(2, remains.Count);
        Assert.AreEqual(10, remains.Single(x => x.GoodId == Goods.JackDaniels.Id).Count);
        Assert.AreEqual(8, remains.Single(x => x.GoodId == Goods.JohnnieWalker.Id).Count);
    }
```

Тесты

```
public class RemainsTest
{
    [SetUp]
    public void SetUp()
    {
        World.InitDatabase();
    }

    [Test]
    public void CalculateRemainsForMoveDocuments()
    {
        /// ARRANGE
        var builder = new TestBuilder();

        // Income to remote storage
        var doc1 = builder.CreateDocument("15.01.2016 10:00:00", Storages.MainStorage, Storages.RemoteStorage);
        builder.AddGood(doc1, Goods.JackDaniels, 10);
        builder.AddGood(doc1, Goods.JohnnieWalker, 15);

        // Outcome from remote storage
        var doc2 = builder.CreateDocument("16.01.2016 20:00:00", Storages.RemoteStorage, Storages.MainStorage);
        builder.AddGood(doc2, Goods.JohnnieWalker, 7);

        /// ACT
        var remains = new RemainsService(World.GetContext()).GetRemainFor(Storages.RemoteStorage, new DateTime(2016, 02, 01));

        /// ASSERT
        Assert.AreEqual(2, remains.Count);
        Assert.AreEqual(10, remains.Single(x => x.GoodId == Goods.JackDaniels.Id).Count);
        Assert.AreEqual(8, remains.Single(x => x.GoodId == Goods.JohnnieWalker.Id).Count);
    }
```

ARRANGE

Поступление

Отгрузка

Тесты

```
public class RemainsTest
{
    [SetUp]
    public void SetUp()
    {
        World.InitDatabase();
    }

    [Test]
    public void CalculateRemainsForMoveDocuments()
    {
        /// ARRANGE
        var builder = new TestBuilder();

        // Income to remote storage
        var doc1 = builder.CreateDocument("15.01.2016 10:00:00", Storages.MainStorage, Storages.RemoteStorage);
        builder.AddGood(doc1, Goods.JackDaniels, 10);
        builder.AddGood(doc1, Goods.JohnnieWalker, 15);

        // Outcome from remote storage
        var doc2 = builder.CreateDocument("16.01.2016 20:00:00", Storages.RemoteStorage, Storages.MainStorage);
        builder.AddGood(doc2, Goods.JohnnieWalker, 7);

        /// ACT
        var remains = new RemainsService(World.GetContext()).GetRemainFor(Storages.RemoteStorage, new DateTime(2016, 02, 01));

        /// ASSERT
        Assert.AreEqual(2, remains.Count);
        Assert.AreEqual(10, remains.Single(x => x.GoodId == Goods.JackDaniels.Id).Count);
        Assert.AreEqual(8, remains.Single(x => x.GoodId == Goods.JohnnieWalker.Id).Count);
    }
```

ACT

Тестируемый метод

Тесты

```
public class RemainsTest
{
    [SetUp]
    public void SetUp()
    {
        World.InitDatabase();
    }

    [Test]
    public void CalculateRemainsForMoveDocuments()
    {
        /// ARRANGE
        var builder = new TestBuilder();

        // Income to remote storage
        var doc1 = builder.CreateDocument("15.01.2016 10:00:00", Storages.MainStorage, Storages.RemoteStorage);
        builder.AddGood(doc1, Goods.JackDaniels, 10);
        builder.AddGood(doc1, Goods.JohnnieWalker, 15);

        // Outcome from remote storage
        var doc2 = builder.CreateDocument("16.01.2016 20:00:00", Storages.RemoteStorage, Storages.MainStorage);
        builder.AddGood(doc2, Goods.JohnnieWalker, 7);

        /// ACT
        var remains = new RemainsService(World.GetContext()).GetRemainFor(Storages.RemoteStorage, new DateTime(2016, 02, 01));

        /// ASSERT
        Assert.AreEqual(2, remains.Count);
        Assert.AreEqual(10, remains.Single(x => x.GoodId == Goods.JackDaniels.Id).Count);
        Assert.AreEqual(8, remains.Single(x => x.GoodId == Goods.JohnnieWalker.Id).Count);
    }
```

ASSERT

Проверка результатов

Плюсы

+ Малые трудозатраты на тесты
+ Тестирование настоящей базы (SQL запросы, тригеры, ограничения)
+ Можно тестировать legacy-код без его модификации
+ Универсальность подхода – любая база, любая ORM

Минусы

- Работает медленнее (300ms vs 3ms на один тест)

Ссылки

@justserega

https://github.com/justserega/DbTest

Install-Package DbTest
Install-Package DbTest.EF6
Install-Package DbTest.EFCore

Помогите сделать Сборку лучше

Сборка_

goo.gl/LRwjPo

Сборка_

СЛЕДИТЕ ЗА НАМИ

t.me/sborkacamp

vk.com/sborkacamp

GoodLine
Оператор связи Кузбасса

Атвинта