Раздел 1

Задача

Сервис который предоставляет значение текущей цены, некой крипто валюты.

К примеру BTC/USD 8000$

Так же эту цену каким-то образом можно обновлять извне

## 01.Primitive.Types.cs

Крипто пара: как примитив, используем службу для обновления цены

служба внутри использует словарь string/float

public static class PriceService

{

private static readonly Dictionary<string, float> PairPrice = new Dictionary<string, float>();

public static float GetPriceFor(string pair)

{

return PairPrice[pair];

}

public static void ApplyPairPrice(string pair, float price)

{

PairPrice[pair] = price;

}

}

//Коротко объяснение

И тест

[TestMethod]

public void PriceShouldBeApplied()

{

var pair = "BTC/USD";

var newPrice = 10.0f;

PriceService.ApplyPairPrice(pair, newPrice);

PriceService.GetPriceFor(pair).Should().Be(newPrice);

}

Раздел 1.1

Программа расширяется, добавляем Exchange к ценам

public class ExchangePair

{

public string Exchange;

public string Pair;

}

//

public static class PriceService

{

private static readonly Dictionary<ExchangePair, float> PairPrice = new Dictionary<ExchangePair, float>();

public static float GetPriceFor(ExchangePair exchangePair)

{

return PairPrice[exchangePair];

}

public static void ApplyPrice(ExchangePair exchangePair, float price)

{

PairPrice[exchangePair] = price;

}

}

Тестовый набор

public class TestsInput

{

public static float TestPrice => 10.0f;

public const string **Binance** = "Binance";

public const string **BtcToUsd** = "BTC/USD";

public static ExchangePair BinanceBtcUsd => new ExchangePair()

{

Exchange = Binance,

Pair = BtcToUsd

};

}

И сами тесты

[TestMethod]

public void ExchangePairShouldBeSame()

{

//Todo: fix this duplicates. Code smell's

var pair1 = new ExchangePair() { Exchange = TestsInput.Binance, Pair = TestsInput.BtcToUsd };

var pair2 = new ExchangePair() { Exchange = TestsInput.Binance, Pair = TestsInput.BtcToUsd };

pair1.Should().Be(pair2);

}

[TestMethod]

public void PriceShouldBeApplied()

{

PriceService.ApplyExchangePairPrice(TestsInput.BinanceBtcUsd, TestsInput.TestPrice);

PriceService.GetPriceFor(TestsInput.BinanceBtcUsd).Should().Be(TestsInput.TestPrice);

}

Раздел 1.1.1

03.ValueTypes.Adding.CodeGuards

public struct ExchangePair

{

public readonly string Exchange;

public readonly string Pair;

public ExchangePair(string exchange, string pair)

{

Exchange = exchange.ToLower();

if (!pair.Contains("/"))

throw new FormatException(pair);

Pair = pair;

}

}

И тесты.

public const string **BtcToUsdInvalid** = "BTC-USD";

[TestMethod]

[ExpectedException(typeof(FormatException))]

public void OnlyValidPairsCanBeCreated()

{

var shouldNeverHappenPair = new ExchangePair(TestsInput.Binance, TestsInput.BtcToUsdInvalid);

}

Раздел 2

Добавление функциональности к службе

public static void RemoveExchange (string exchange)

{

//Одержимость примитивными типами антипаттерн

// https://refactoring.guru/ru/smells/primitive-obsession

//STOP Is the Type of exchange a string???

Он ведь стал значимым для домена, потому что начал фигурировать в логике (обращение заказчика и требование было удалять по Эксченджу)

//Maybe it is class?

//Maybe it is struct?

//1.1) How to recognize class or struct?

//1.1.1) Should Binance match BINANCE ? (Equality) (Structural Equality)

//1.1.2) Does exchange type has some methods?, Lets remind yourself a definition of class

//1.1.3) Maybe it's enum? and what enum is(structs)

}

Структуры в дотнете, почему прикладные програмисты их так редко используют?

Потому что лепят чушь про то что они быстрые, про то что они попадают в стек

Но почему то самое важное, что они это более высокоуровневое представление ваших ДАННЫХ забывают

Перформанс? Стек? Может быть. а может и нет

Как лечить?

Вычленяем отдельный тип значений

public struct Exchange

{

public readonly string Name;

public Exchange(string name)

{

if (string.IsNullOrEmpty(name))

throw new ArgumentException(name);

Name = name;

}

}

Исправление ошибок компиляции

public static Exchange Binance => new Exchange("Binance");

Добавление теста на Equality

[TestMethod]

public void TwoSameExchangesShouldBeSame()

{

TestsInput.Binance.Should().Be(new Exchange("BINANCE"));

}

Тест падает. Плохо, как исправить?

1. Приводим всё в один регистр
2. При сравнении не учитываем регистр
3. Капитализация букв

Выбираем способ 3

Исправляем конструктор

var cultureInfo = Thread.CurrentThread.CurrentCulture;

var textInfo = cultureInfo.TextInfo;

Name = textInfo.ToTitleCase(name.ToLower());

Выводы:

Когда Эксчендж стал значимым для логики он стал ЗНАЧИМЫМ типом =)

попробуем создать кривой эксчендж, почему бы нет?

[TestMethod]

[ExpectedException(typeof(Exception), AllowDerivedTypes = true)]

public void OnlyValidExchangeCanBeCreated()

{

var shouldNeverHappenPair = new Exchange("");

}

Продолжим нашу логику сервиса, очистка по эксченджу

public static class DictionaryExtensions

{

public static void RemoveAll<TKey, TValue>(this IDictionary<TKey, TValue> dic,

Func<TKey, TValue, bool> predicate)

{

var keys = dic.Keys.Where(k => predicate(k, dic[k])).ToList();

foreach (var key in keys)

{

dic.Remove(key);

}

}

}

И добавим метод

public static void RemoveExchange(Exchange exchange)

{

PairPrice.RemoveAll((pair, f) => pair.Exchange.Equals(exchange));

}

Попытаемся попробовать наш код в действии

[TestMethod]

[ExpectedException(typeof(KeyNotFoundException))]

public void ExchangeShouldBeRemoved()

{

PriceService.ApplyPrice(TestsInput.BinanceBtcUsd, TestsInput.TestPrice);

PriceService.RemoveExchange(TestsInput.Binance);

PriceService.GetPriceFor(TestsInput.BinanceBtcUsd);

}

Просмотр сделаной работы

Разносим код по разным файлам

Ищем что можно улучшить

Пара к ... ЦЕНЕ? сначала флоат (демонстрация что флоат структура ЛОЛ) инт структура,

Цена это **может** быть структура если она имеет смысл в самом домене

Пара это **может** быть структура

(демонстрация что класс ExchangePair отобрал по сути на себя проверку входных данных пары)

Рассмотрение типов дотнет под другим углом.

Просмотр Инта, флоута, гуида, дайттайма на выбор

Это ДАННЫЕ

Таймспан\ДейТайм\Поинт - данные без методов

**А теперь немного о СТРОКАХ в .НЕТ**

Какова Эквивалентность строк? (ссылочная или значимая?)

Содержит ли он только данные без методов? (методы возвращают новые данные)

Есть ли у строк поведение?

**Тогда класс это или структура?**

СТРИНГ класс ! (почему?) (обьяснение)

ИТОГ:

Если бы я сказал закодируйте Карандаш и у него есть цвет

Вы взяли бы КОЛОР, вряд ли вы бы брали стринг.

Для дат обновления вы используете ДейтТайм, тогда почему

Для АЙДИ клиента вы используете ИНТ? Только потому что в дотнете

Нет такого типа данных?

Пример с АЙДИ пользователя и посылки

Айдишки разных сущностей, это разные айдишки даже если одинаково их значение тогда зачем мы используем ИНТ теряя при этом контекст?

Внутреннее поле ИД посылки может быть и номер и строка, может иметь массу сложных проверок на входе в саму логику (как по примеру что мы делали)

Выводы, когда у вас в вашем языке продукта (ubiquitous language) появляются слова вроде

НомерЗаказа, Обменник, Пара, Посылка, Человек возможно это валю – тип?

Отдельного разговора заслуживают Entities из EntityFramework

Подумайте на досуге сами, могли ли бы они быть обьектами значений?

**Раздел 2.1**

Жить без удобных штучек порой очень больно

Демонстрация боли

new ExchangePair(new Exchange(), ), Весьма неудобно. Что можно сделать с этим?

Фиксим боль

public static class Exchanges

{

public static Exchange Binance = new Exchange("Binance");

public static Exchange Bittrex = new Exchange("Bittrex");

public static ExchangePair WithPair(this Exchange exchange, string pair)

{

return new ExchangePair(exchange, pair);

}

}

Я не могу смотреть без слёз на пару как стринг.

public struct CurrencyPair

{

public readonly string Value;

public CurrencyPair(string value)

{

if (!value.Contains("/"))

throw new FormatException(value);

Value = value;

}

}

Исправляем компиляцию

TestInput

public static CurrencyPair BtcToUsd => new CurrencyPair("BTC/USD");

public static CurrencyPair BtcToUsdInvalid => new CurrencyPair("BTC-USD");

Добавим еще пару методов расширений для удобства в класс Exchanges

public static CurrencyPair ToPair(this string plainTextPair)

{

return new CurrencyPair(plainTextPair);

}

**И заодно**

**ToString() методы у стурктур**

**И демонстрация NCrunch output (не работает?!)**

**Пробуем создать ExchangePair**

**Попутно создаём класс**

public static class Pairs

{

public static CurrencyPair BtcToUsd => new CurrencyPair("BTC/USD");

}

Стало удобнее не так ли?

**Раздел 3**

Связять ФИЛЬТР ДАТУ и прайс сервис?!

GetHashCode Иммутабельность, IEQutable<T>, ISerializable, Взаимодействие с ТайпСкрипт

public enum Mode

{

**Include**,

**Exclude**,

}

struct Filter

{

public Mode Mode;

public HashSet<string> Data;

}

private static Filter CreateFilter()

{

return new Filter() { Data = new HashSet<string>() { "a", "b" }, Mode = Mode.**Include** };

}

Добавим тест

[TestMethod]

public void TwoSameValueTypesShouldExistOnceInHashset()

{

var e = new HashSet<Filter>();

var filterOne = CreateFilter();

var filterTwo = CreateFilter();

filterOne.Should().BeEquivalentTo(filterTwo);

e.Add(filterOne);

e.Add(filterTwo);

e.Count.Should().Be(1);

}

Встроеный Equals Структур сравнивает всё по значению,

Если есть ссылочный тип мы сравним его по значению (да еще и используя рефлексию)

Попробуем починить эквивалентность

public override bool Equals(object obj)

{

if (obj is Filter other)

return other.Mode == this.Mode && Data.SetEquals(other.Data);

return false;

}

Поменяем Data/ Mode местами.

И …. Магия, Вы всё еще не используете тесты?

Хорошо как вам такой финт?

public override int GetHashCode()

{

return 10;

}

Теперь давайте разбираться что тут происходит

Обьяснение Что такое ГетХешКод и как это работает:

По сути Это сделано для того чтоб в словаре не запускать прожорливый Икуалс

На ВСЕХ элементах

Отдельный разговор почему это сделано для класса обжект и это может быть темой дискусии на афтерпати

Оказывается базовый ГетХешКод который есть у класса обжект

Использует первое не нулевое поле обьекта, а хешсет не переопределяет свой гет хеш код для содержащихся элементов

Поэтому два одинаковых хеш сета дадут разные хеш коды)

В этом легко убедится

filterTwo.Data.GetHashCode().Should().Be(filterOne.Data.GetHashCode());

Выходит что два обьекта имеют разный хеш и хотя данные СТРУКТУРНО эквивалентны,

Но будут лежать в нашем словаре два раза

Фикс:

public override int GetHashCode()

{

return Mode.GetHashCode() ^ Data.Select(element => element.GetHashCode()).Sum();

}

Какова теперь будет разница между struct Filter и class Filter ?

Подводим короткий итог

Тесты, на такие штуки, замечательно, ГетХеш код не такой уж бесполезный

Интерфейс IEquatable<T>

public bool Equals(Filter f)

{

return true;

}

Демонстрация как колекция будет автоматически использовать этот интерфейс,

с обьяснением почему так (компарер дефолтный)

# ИММУТАБЕЛЬНОСТЬ

Обьяснение иммутабельности

Из иммутабельности вытекает сериализация , показать почему (закрытые поля, конструктор итп)

Сериализация

[TestMethod]

public void FilterValuesShouldBeSerializable()

{

var filter = CreateFilter();

filter.Should().BeBinarySerializable();

}

Коротко о аттрибуте [Serializable]

Попробуем джсон

private static void CheckJsonSerialization<T>(T instance)

{

var serialized = JsonConvert.SerializeObject(instance);

Debug.WriteLine("Serialized: " + serialized + Environment.NewLine);

var deserialized = JsonConvert.DeserializeObject<T>(serialized);

Debug.WriteLine("Deserialized: " + JsonConvert.SerializeObject(deserialized) + Environment.NewLine);

deserialized.Should().Be(instance);

}

Играем с названиями полей конструктора

Реализуем ISerializable

public void GetObjectData(SerializationInfo info, StreamingContext context)

{

info.AddValue(nameof(Mode), Mode);

info.AddValue(nameof(Data), Data, typeof(HashSet<string>));

}

И восстановление:

private Filter(SerializationInfo info, StreamingContext para)

{

Mode = (Mode)info.GetInt32(nameof(Mode));

Data = (HashSet<string>)info.GetValue(nameof(Data), typeof(HashSet<string>));

}

Отметим что их используют оба сериализатора (Бинари, джсон)

Отметим конструктор (нельзя одно поле «забыть»)

*Добавление*

*filter.Should().BeXmlSerializable();*

*Грязный фикс,*

*Делаем паблик, делаем класс, снимаем ридонли, делаем второй пустой конструктор. Вуаля*

*Почему это ОЧЕНЬ ПЛОХО?!*

Раздел 4

Оптимизация в конце не так ли ?

Интернирование – когда и зачем? РЕАЛЬНЫЙ кейс

Недостатки : выгрузка домена – освобождение памяти интернированых строк

Структуры которые передаются по ссылке. ВТФ? (пример разработать с рекурсией)

struct BigData

{

public decimal a1;

public decimal a2;

public decimal a3;

public decimal a4;

public decimal a5;

public decimal a6;

public decimal a7;

public decimal a8;

public decimal a9;

public decimal a0;

public decimal a11;

public decimal a12;

public decimal a13;

public decimal a14;

public decimal a15;

public decimal a16;

public decimal a17;

public decimal a18;

public decimal a19;

public decimal a20;

public decimal a21;

public decimal a22;

public decimal a23;

public decimal a24;

}

int RecursiveMethod(BigData b, int limit)

{

try

{

limit++;

return RecursiveMethod(b, limit);

}

catch (Exception)

{

return limit;

}

}

[TestMethod]

public void ComputeRecursionLimit()

{

var counter = RecursiveMethod(new BigData(), 1);

Debug.WriteLine(counter);

}

Коротко о

RuntimeHelpers.EnsureSufficientExecutionStack();

Адские тормоза при выборке из Дикшинари (разработать пример)

**Раздел 5 «не успели» или «не подготовил»**

Аттрибут [Pure]

Аттрибут [Layout]

Enums + Invalid values, (default struct)

БОЛЬ - структуры не наследуются?!!

можно пообщаться о этом на афтерпати

Конечно же Компараторы и почему мы их так мало используем?

Почему .НЕТ разработчики добавили ГетХешКод в Обжект (в само сердце)

Как распознать структуру, и как распознать когда она лишняя

IEqutable,IComparable, IComparer ,IEquityComparer,IStructureEquality и как в этом не запутаться

Попытка заставить класс быть структурой

Ну и в лучших традициях финального раскрытия задумки в конце фильмов голивуда,

у нас ТДД ребятки вышло в конце. Бонусом.

Интересно кто обратил внимание?

Ну и следующий Митап мог бы быть посвящен .NET TPL Tasks, Schedulers, TCS, и async await

https://github.com/BruceBayne/CSharp-Study

Всем спасибо

20.10.2018