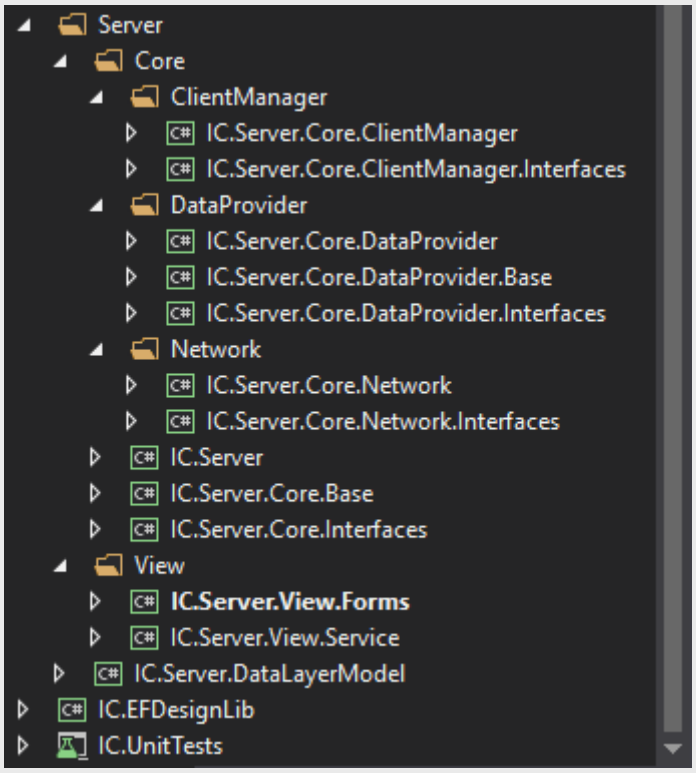
Руководство по разработке и оформлению кода в C# проектах

Содержание

Заметки по построению архитектуры проекта

При построении архитектуры проекта нужно соблюдать принципы проектирования **SOLID**. Все основные части проекта должны быть отделены по слоям, например: уровень данных, бизнес-логика, сетевое взаимодействие, вьюхи, и т.д. Действуя по принципу «Единственной обязанности» каждый класс должен выполнять только одну цель, ровно как и отдельный слой должен выполнять только ф-циональность отдельной части проекта. Все разделения по слоям должны разделяться по папкам, это позволяет более быстро ориентироваться в решении и придаёт единообразный вид решений. Рассмотрим пример организации проектов сервера в клиент-серверной архитектуре:



Серверная часть вынесена в отдельную папку Server. Дальше вынесены 3 основных слоя – Core, View и DataLayer. На уровне ядра выделены такие под слои как: ClientManager, отвечающий за всю работу с клиентами(обработка полученных запросов, сохранение подключенных клиентов, проверка доступности подключенных клиентов, и т.д.); DataProvider, обеспечивающий выполнение запросов с добавлением/получением данных из базы данных; Network, обеспечивающий всю сетевую работу. Остальные проекты касательно ядра сервера не вынесены в отдельную папку т.к. представляют исключительно сам слой сервера, а не отдельный под слой. На уровне View вынесены проекты касательно работы с executable приложениями и внешними видом.

Именование пространств имён

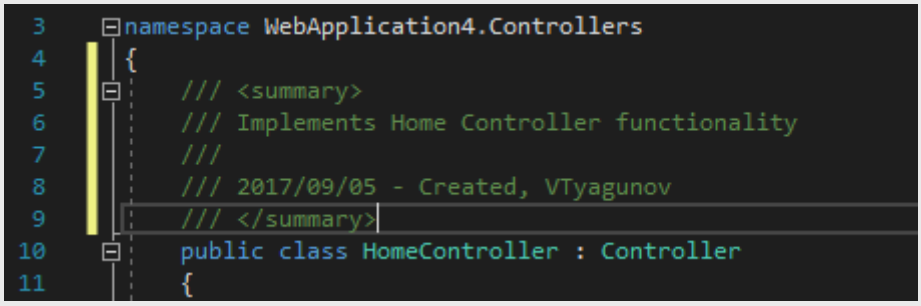
Пространства имён принято именовать по иерархии расположения проекта в солюшене. Рассмотрим пример, на скрине изображены проекты Серверной части солюшена, солюшен имеет абревиатуру IC, по этому первое имя в пространстве имён стоит IC, все имена в пространстве имён разделяются точками, следующее имя идёт по иерархии вниз, у нас это Server, далее, если к примеру мы создаём проекты для Дата Провайдера, то указываем имена Core и DataProvider, по расположению в солюшене, дальше идёт основное имя, указывающее на тип данного проекта – DataProvider, и после уже идёт разделение, если это реализация или основной проект, то больше имён не следует, если же нет, то следующее имя указывает принадлежность к типу данных, содержащихся в проекте. Проекты с любой ф-циональностью(кроме статичных хелперов), должны иметь интерфейсы, первым делом для нужной нам части мы создаём проект с интерфейсами, в данном случае – IC.Server.Core.DataProvider.Interfaces. Далее если в данном слое будут присутствовать перечисления, константы, события, делегаты, и прочие общие элементы, для них создаём проект с отношением Base – IC.Server.Core.DataProvider.Base, и в последнюю очередь уже идёт библиотека, содержащая реализацию нашей ф-циональности – IC.Server.Core.DataProvider. Подобное разделение при грамотном использовании позволяет избежать многих проблем при создании реализаций, в особенности предотвращает проблему обратной зависимости библиотек, где библиотеки вынуждены ссылаться друг на друга. В каждом проекте отдельные элементы должны обязательно получать пространства имён в зависимости от своего внешнего(среди проектов в солюшене) и внутреннего(внутри библиотеки) расположения, к примеру если у нас в проекте IC.Server.Core.DataProvider.Base есть папка Enums, а в ней располагаются различные перечисления, если они не определены по другим папкам, а располагаются в корне папки Enums, то у каждого такого перечисления будет пространство имён IC.Server.Core.DataProvider.Base.Enums. Касательно использования using для добавления пространств имён в файлах, следует выстраивать их строго по иерархии их важности, и в алфавинтом порядке, с разделением между пространствами имён, имеющими разное первое имя, например:

```
1   using System;
2   using System.Collections.Generic;
3
4   using Microsoft.AspNetCore.Builder;
5   using Microsoft.AspNetCore.Mvc;
6   using Microsoft.Extensions.DependencyInjection;
7
8   using WebApplication4.Util;
9
10  namespace WebApplication4
11  {
```

Не используемые пространства имён следует удалять

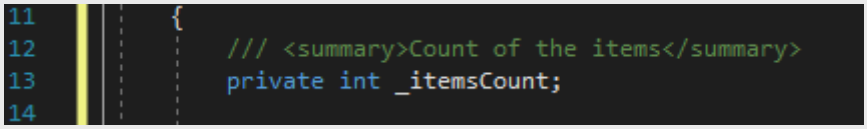
Комментирование кода

В проектах следует комментировать строго только на **английском языке** все перечисления, классы, структуры, интерфейсы, делегаты, переменные, свойства, конструкторы, методы через xml-doc комментарии, строго соблюдая вид комментария:



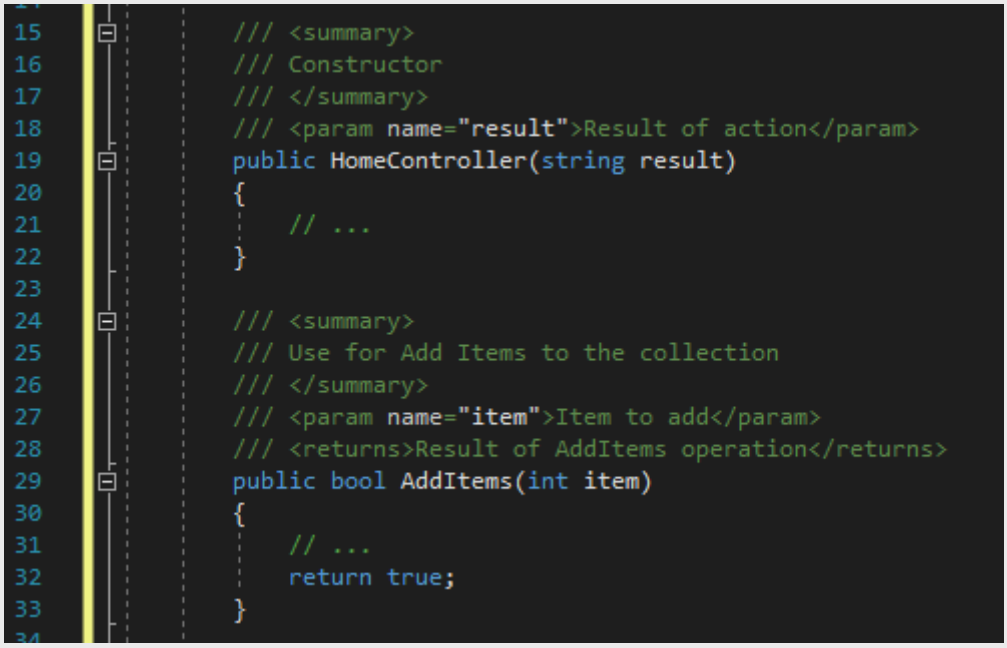
```
namespace WebApplication4.Controllers
{
    /// <summary>
    /// Implements Home Controller functionality
    ///
    /// 2017/09/05 - Created, VTyagunov
    /// </summary>
    public class HomeController : Controller
    {
```

Вначале идёт описание комментируемого элемента(в данном случае и далее класса), дальше через строку указывается дата в формате ГОД/МЕСЯЦ/ДЕНЬ, через тире указывается действие над данным классом – Created, и через запятую автор в формате – первая буква имени и следом фамилия – VTyagunov. Внутри класса переменные и свойства указывается через однострочный xml-doc комментарий:



```
    {
        /// <summary>Count of the items</summary>
        private int _itemsCount;
```

Конструкторы и методы комментируются через обычный вид xml-doc комментария:



```
        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="result">Result of action</param>
        public HomeController(string result)
        {
            // ...
        }

        /// <summary>
        /// Use for Add Items to the collection
        /// </summary>
        /// <param name="item">Item to add</param>
        /// <returns>Result of AddItems operation</returns>
        public bool AddItems(int item)
        {
            // ...
            return true;
        }
```

К комментированию ввиде обычных коментариев(//) следует прибегать только в исключительном случае(например когда слишком длинная логика в методе, нет возможности разделить её на методы, и есть необходимости отделить её комментариями для ясности), в остальном следует их избегать, комментирование через (/* */) так же не приветствуется

Именнование классов, методов, переменных и т.д.

Именовать элементы в проекте следует в строго установленной форме:

- Классы, структуры, перечисления и методы с любым типом доступа именуются с заглавной буквы
- **Публичные** переменные, свойства, делегаты именуются с заглавной буквы
- **Приватные** переменные, делегаты именуются со строчной буквы с нижним подчёрквaнием в постфиксе(например _count)
- Параметры в конструкторах и методах, и переменные, объявленные внутри конструкторов и методов именуются со строчной буквы

Именовать любые элементы стоит строго в понятной форме, сокращения и абревиатуры использовать только для общих элементов(например DBSet, ICComponent и т.д.), именовать интерфейсы следует строго начиная с буквы I, компоненты с буквы C, использования любых символов кроме букв и нижнего подчёркивания(_) не допускается.

Прочие положения

Внутри классов все отдельные части следует разделять на region’ы, переменные должны иметь свой регион с название Variables, для свойств нужно выделять регион Properties, для конструкторов Constructors, для методов Methods, если методов много и они разделяются по типам(например одни методы реализуют один интерфейс, вторые реализуют другой интерфейс, а третьи это приватные методы), их тоже следует выносить в отдельные регионы внутри региона Methods. Все регионы должны иметь названия только на **английском языке**.