.NET Framework Class Library
Библиотека классов

1. Общие положения.
2. Сборки и пространства имен.
3. Общая система типов CTS.
4. Пространство имен System.
5. Базовый объект Object.

Net.Framework. Библиотека классов

.Net Framework Class Library – библиотека базовых классов, на основе которых строятся все .Net-приложения.

Принципиальная новизна заключается в том, что.

- Унифицируются функциональные возможности разных языков.
- Реализуется модель управления оперативной памятью.
- Базовые классы не принадлежат пользовательским приложениям, а являются компонентом операционной системы.

Функциональность

1. Поддержка базовых типов и типов, определяемых пользователем.
2. Поддержка обработки исключительных ситуаций.
3. Операции ввода/вывода и работа с потоками.
4. Обращение к функциям операционной системы.
5. Поддержка доступа к данным.
6. Создание Windows-приложений.
7. Создание клиентских и серверных Web-приложений.
8. Создание Web-сервисов.

Реализация: сборки и динамические библиотеки

Сборка (assembly) – бинарные данные приложения, содержат код, графические изображения, ресурсы и другие.

Сборка – минимальная единица внедрения, контроля версий, повторного использования и системы безопасности.

Сборка также содержит метаданные – информацию о классах, типах и ссылках на другие сборки.

Динамические сборки создаются во время выполнения программы и на диске не сохраняются.

Библиотека классов .Net состоит из многих сборок, каждая из которых содержит много классов.

Динамические библиотеки

Функциональные возможности базовых классов распределены по библиотекам DLL. Управляемое приложение во время запуска загружает необходимые библиотеки DLL.

Реализация: пространства имен

Библиотека классов .NET Framework состоит из пространств имен **`namespace`**. Каждое пространство имен содержит типы, которые можно использовать в программах: классы, структуры, перечисления, делегаты и интерфейсы. Пространства имен группируют типы по функциональности.

Пространства имен обеспечивают ограничение области видимости: два класса с одним и тем же именем могут быть использованы, если они находятся в разных пространствах имен и их имена определены в рамках соответствующих пространств имен.

Пространства имен

Имя пространства имен является частью полного имени типа: **namespace.typename**

Использования полных имен **namespace.typename** в C++ позволяет избежать ключевое слово **using.**

```
using System;
. . .
void main()
    {
        Console.WriteLine("Hello");
        //  Без using, нужно использовать полное
  имя:
        // System.Console.WriteLine("Hello");
    }
```

Пространство имен System

Пространство имен **`System`** – корневое пространство имен в .NetClass Library, содержит фундаментальные типы данных, реализованные в .Net Framework.

- Класс **`Object`** – базовый для всех классов в библиотеке классов .Net.
- Примитивные и расширенные типы.
- Более 100 классов, используемых для обработки исключительных ситуаций, разработки интерфейса, сборки мусора и т.п.

Пространство имен System

Пространство имен **`System`** содержит фундаментальные и базовые классы, которые определяют распространенные типы значений и ссылочные типы данных, события и обработчики событий, интерфейсы, атрибуты и исключения обработки.

Также содержит классы, обеспечивающие поддержку преобразования типов данных, операций с параметрами методов, математических операций, удаленного и локального вызова программ, управления средой приложений и контроля управляемых и неуправляемых приложений.

Пространство имен System

AppDomain	Environment	Object
Array	Exception	Random
Console	GC	String
Convert	Math	Type

- System.Windows.Forms
- System.Drawing
- System.Collections
- System.IO

Пространство имен System

Базовые типы данных, используемые всеми приложениями, соответствуют простым типам данных, используемым в языках программирования.

При написании кода можно использовать типы .Net Framework или ключевое слово языка.

Таблица типов

`System.Windows.Forms` содержит классы для создания приложений Windows с использованием всех элементов пользовательского интерфейса, доступных в операционной системе Microsoft Windows.

Common Type System (CTS)

Общая система типов CTS предоставляет набор отношений между типами.

Множественное наследование не поддерживается.

`Object` – корневой объект иерархии. Из него наследуют все типы. Является базовым для всех управляемых типов, в том числе для примитивных типов, используемых в управляемых кодах.

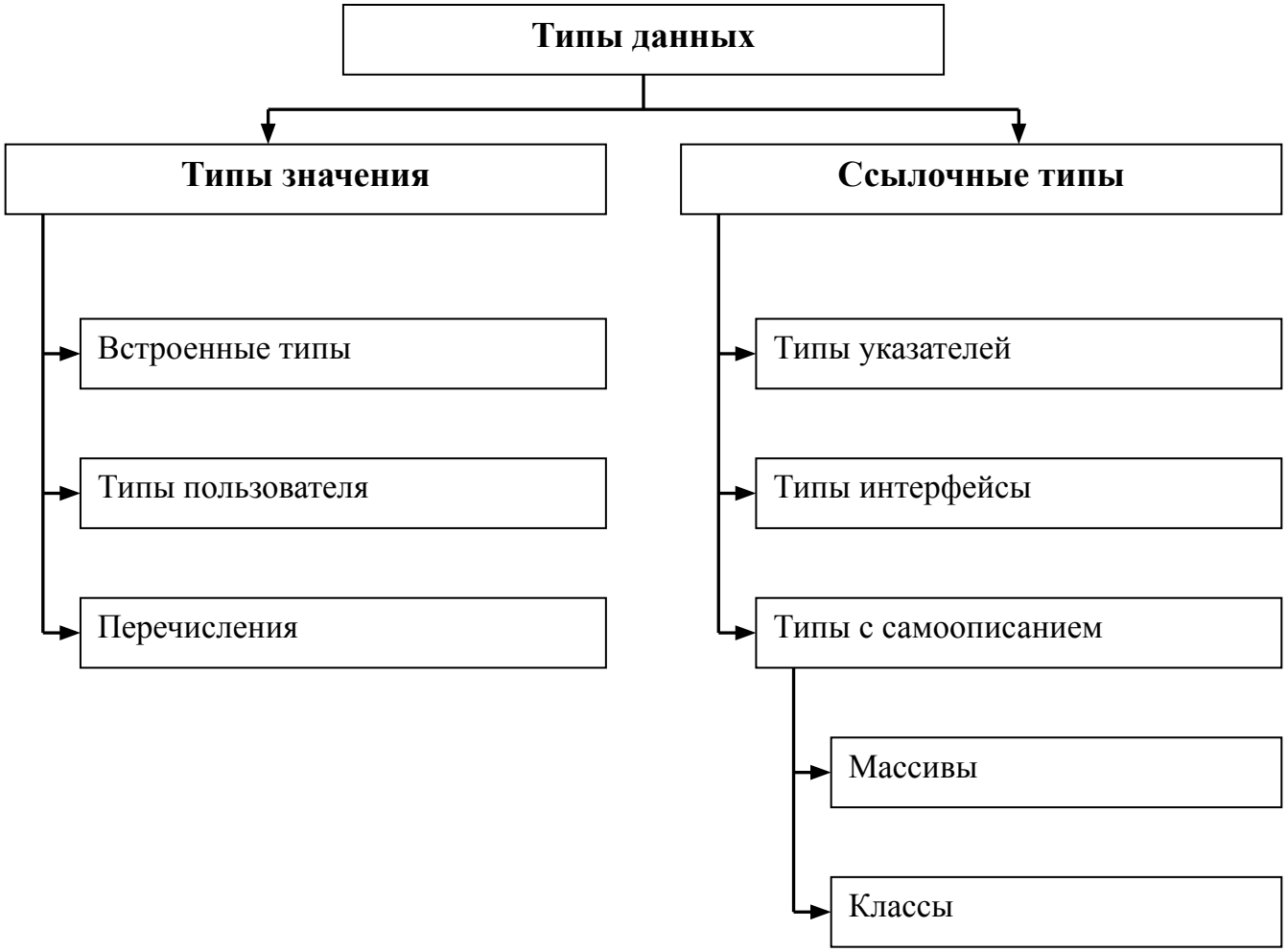
Общая система типов CTS

Примитивные типы CLI занимают место в системе типов в форме объектов, которые заключают в обертку каждый примитивный тип.

В управляемой куче можно распределить только ссылочные типы, и только они могут поддерживать наследование и виртуальные функции.

Базовые типы содержатся в пространстве имен **`System`**.

Общая система типов CTS



Ссылочные типы и типы значений

Управляемые типы делятся на две категории:

- тип значений,
- ссылочный тип.

Различие между ними в механизмах выделения памяти, хранения и выполнения операций.

Типы значения

Типы значения, это простые типы, которые содержат собственно данные.

- Встроенные типы реализованы в среде выполнения.
- Пользовательские типы-значения.
- Перечисления **`enum`** – типы, которые также может определить пользователь.

Значения типов-значений представлены как локальные и глобальные переменные, параметры методов, поля объектов и элементы массивов. Для каждого типа известен тип значений, которые она может содержать.

Типы значения не могут содержать объекты.

Ссылочные типы

Ссылочные типы описывают объектные ссылки (**`object references`**), которые представляют адреса объектов. Живут в куче, и к ним можно обращаться только через дескриптор.

Когда объект присваивается или передается как параметр, сам объект не копируется, но создается другая ссылка, которая ссылается на тот же объект.

Ссылочный тип используется везде, где моделируется объект. Могут унаследовать другой класс и быть унаследованы.

Ссылочные типы

Типы указателей – содержат описание значения, представленного адресом в памяти.

Типы интерфейсов – содержат только частичное описание значений.

Объектные типы (типы с самоописанием) – содержат самоописываемые значения.

Самоописывающие ссылочные типы

Ссылочные типы, представляющие объекты, называются самоописывающими (self-describing), потому что каждый объект в куче содержит информацию о своем типе.

Два таких типа являются встроенными:

`System.Object` и **`System.String`**.

`System.Object` является общим базовым классом, от которого непосредственно или косвенно наследует любой другой класс.

`System.String` используется для представления строковых данных в формате Unicode.

Классы

Классы – основа типов с описанием.

Классы могут:

- агрегировать значения других типов;
- наследоваться друг от друга (в .NET поддерживается только одиночное наследование).

Классы

Классы могут содержать следующие элементы:

Поля (`fields`) – для хранения значений других типов.

Методы (`methods`) – функции классов. Они бывают статическими (`static method`) и объектными (`instance method`). Вызываемый объектный метод всегда получает ссылку на объект, для которого он вызывается (`this`).

Свойства (`properties`) – те же методы, один из которых возвращает некоторое значение (`get`), а другой устанавливает это значение (`set`).

События (`events`) – используются для асинхронного внесения изменений в объект.

Типы-интерфейсы

Интерфейсы реализуются на основе классов, это такие типы данных, которые могут быть «лицом» для данных других типов.

Интерфейсы задают функциональность абстрактного объекта без ее реализации. Классы, использующие интерфейсы, наполняют конкретным содержанием методы, определенные в интерфейсах.

Примеры интерфейсов

Интерфейс **`IEnumerable`** описывает **итератор** , используемый для итерации по содержимому коллекции методом **`MoveNext()`** .

Метод **`GetEnumerator`** возвращает **`IEnumerator`** – энумератор, используемый для итерации.

Свойство **`Current`** типа **`Object`** возвращает текущий элемент коллекции.

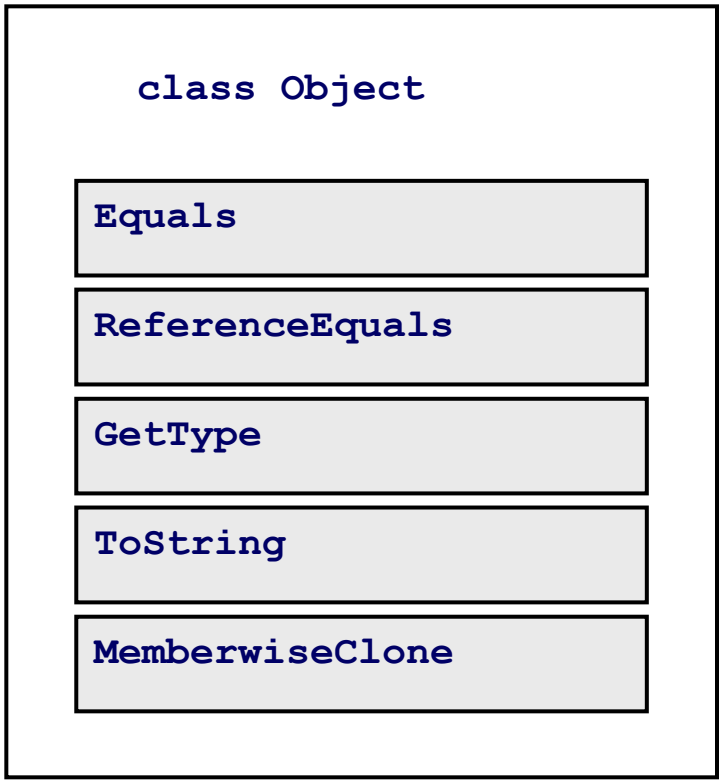
Интерфейс **`ICollection`** наследует интерфейс **`IEnumerable`** и служит базовым интерфейсом для реализации **коллекций**. Определяет методы, свойственные коллекциям – поддержку размера коллекции, энумераторов и синхронизации.

Встроенные типы

Класс **`Object`** пространства имен **`System`**.

Класс **`String`** пространства имен **`System`**.

Класс Object



Любой объект наследует классу **`Object`** явно или нет.

Конструктор **`Object`** инициализирует новый экземпляр класса.

Методы класса Object

`Equals` проверяет, являются ли два объекта одним и тем же экземпляром. Для типов значений метод переопределен и проверяет равенство значений, хранимых экземплярами объекта.

`ReferenceEquals` проверяет, являются ли два объекта одним и тем же экземпляром класса.

`GetType` возвращает объект типа **`Type`** для экземпляра класса.

`ToString` возвращает текстовое представление объекта. Может быть перегружен.

`MemberwiseClone` создает точную копию объекта.

Пример

Обычно не требуется объявлять класс наследником **`Object`**, так как наследование происходит неявно.

Все классы в платформе .NET Framework являются производными класса **`Object`**, и все методы, определенные в классе **`Object`**, доступны для всех объектов в системе.

В производных классах некоторые из этих методов могут переопределяться и переопределяются,

например, **`ToString()`**.

Класс Type и пространство имен Reflection

Класс **`Type`** – точка входа в пространство имен **`Reflection`**.

`Reflection` (отражение) – способность изучать возможности классов во время выполнения программы.

Используя отражения, можно извлекать классы, изучать их методы, свойства, конструкторы, поля, события, получая доступ к метаданным , ассоциированным с данным классом.

Класс **`Type`** содержит методы типа

`GetMethods()`,

`GetProperties()`,

которые возвращают информацию о полях класса.

Описание примера

1. Спецификация объекта – место в проекте. Доступность объекта.
2. Пространства имен по умолчанию.
3. Объявление объекта.
4. Инициализация объекта.
5. Интерфейс объекта: свойства и перегрузка `ToString()`.
6. Взаимодействие объекта с формой приложения и ее компонентами.
7. Управление поведением объекта из класса формы.
8. Инструменты для управления объектом.

Описание примера

Описание примера