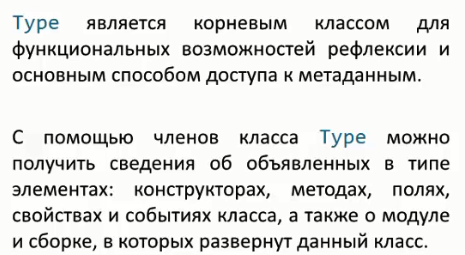
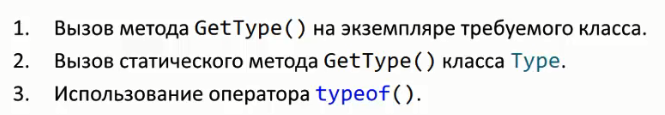
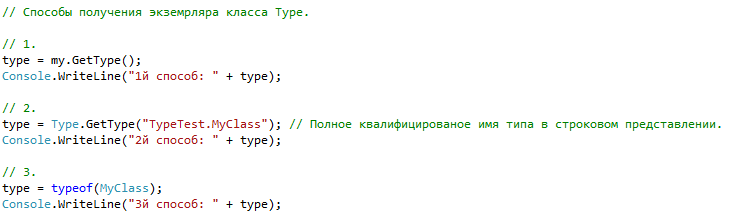


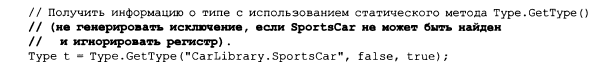
Type



Способы получения типа

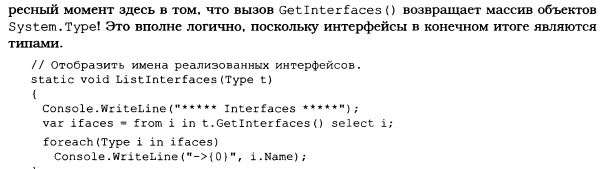












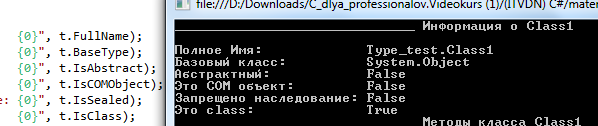
Чтобы вывести обобщенный тип то необходимо, где число количество параметров



Сравнивает два типа



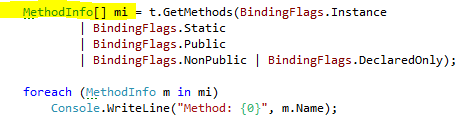
Получить описание класса



BindingFlags.DeclaredOnly – только те методы которые описаны в классе (наследники исключаются)

MethodInfo[]

Получить и использовать методы класса



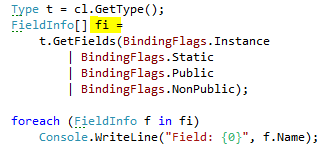
//Вызов private метода

MethodInfo methodC = type.GetMethod("MethodC", BindingFlags.Instance | BindingFlags.NonPublic);

methodC.Invoke(myClass, new object[] { "Hello", " world!" });

FieldInfo[]

Получить и использовать поля класса



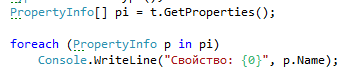
// Запись значения в private поле

FieldInfo mystring = type.GetField("mystring", BindingFlags.Instance | BindingFlags.NonPublic);

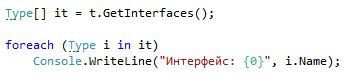
mystring.SetValue(myClass, "Привет Мир!");

PropertyInfo[]

Получить свойства класса

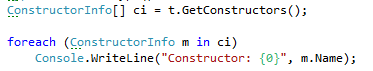


Получить интерфейсы класса





Получить конструкторы класса



В классе два конструктора, один явный по умолчанию и пользовательский с одним аргументом



TypeInfo – то же что и Type но присутствует только на платформе NEW in .NET 4.5

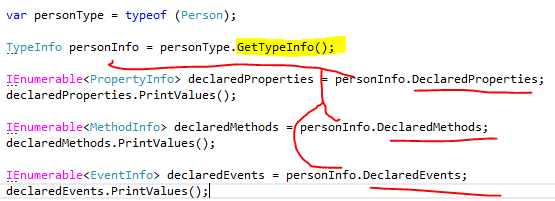
public abstract class TypeInfo : Type, IReflectableType

Представляет объявления типов для классов, интерфейсов, массивов, значений, перечислений, параметров, определений универсальных типов и открытых или закрытых сконструированных универсальных типов.

personInfo.DeclaredProperties – свойство возвращающее свойства типа

personInfo.DeclaredMethods - свойство возвращающее методы типа

personInfo.DeclaredEvents; - свойство возвращающее события типа

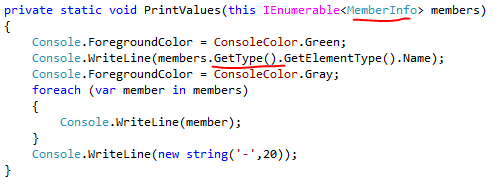


ОБРАТИТЬ ВНИМАНИЕ !!!!

Аргумент расширяющего метода должен быть наследован от – MemberInfo!

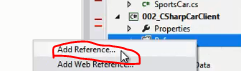
А после в теле расширяющего методы мы приводим (Даункаст) к типу и вызываем на НУЖНОМ ТИПЕ методы

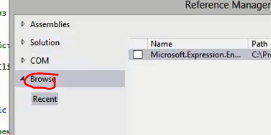
members.GetType().GetElementType().Name



Работа со сборкой

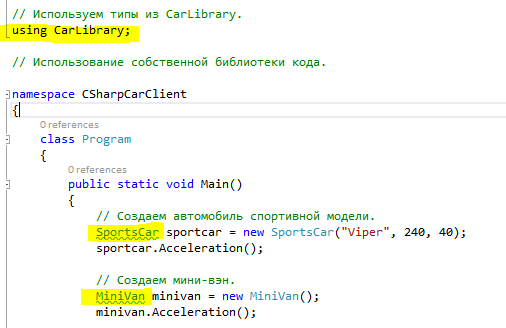
Добавить сборку





Два вида связывания сборок

Раннее – непосредственно при разработке программы



Позднее – ничего не знаем о сборке момент написания программы

Assembly

При помощи класса Assembly - можно динамически загружать сборки,

обращаться к членам класса в процессе выполнения (ПОЗДНЕЕ СВЯЗЫВАНИЕ),

а так же получать информацию о самой сборке.

Assembly.Load() - метод для загрузки сборки.

assembly = Assembly.Load("CarLibrary"); - при помощи метода .Load() происходит позднее связывание

Методы Assembly

GetTypes();

Type[] types = assembly.GetTypes(); - Метод для получения информации о всех типах в сборке, в том числе и те которые с модификатором iternal

foreach (Type t in types)

Console.WriteLine("Тип: {0}", t);

Выведет полные квалифицированные имена всех типов сборки



GetType(……….) GetMembers();

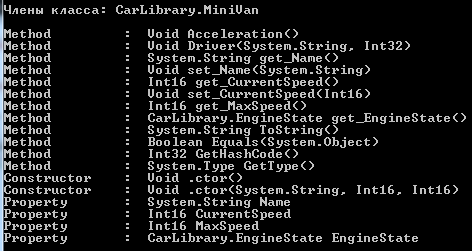
Type type = assembly.GetType("CarLibrary.MiniVan"); - Метод для получения информации о членах класса

Console.WriteLine("\nЧлены класса: {0} \n", type);

MemberInfo[] members = type.GetMembers(); - получаем все члены класса

foreach (MemberInfo element in members)

Console.WriteLine("{0,-15}: {1}", element.MemberType, element);



Получаем информацию о параметрах для метода

Type type = assembly.GetType("CarLibrary.MiniVan");

MethodInfo method = type.GetMethod("Driver"); // Equals , Acceleration, Driver

// Вывод информации о количестве параметров.

Console.WriteLine("\nИнформация о параметрах для метода {0}", method.Name);

ParameterInfo[] parameters = method.GetParameters();

Console.WriteLine("Метод имеет " + parameters.Length + " параметров");

// Выводим некоторые характеристики каждого из параметров.

foreach (ParameterInfo parameter in parameters)

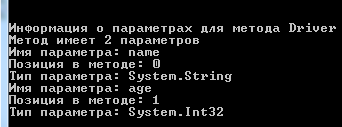
{

Console.WriteLine("Имя параметра: {0}", parameter.Name);

Console.WriteLine("Позиция в методе: {0}", parameter.Position);

Console.WriteLine("Тип параметра: {0}", parameter.ParameterType);

}



1-й способ

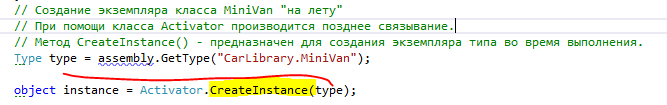
ПОЗДНЕЕ СВЯЗЫВАНИЕ

Activator

Создаем экземпляр нужного нам класса

Activator.CreateInstance(type); - При помощи класса Activator производится позднее связывание.

Метод CreateInstance() - предназначен для создания экземпляра типа во время выполнения.

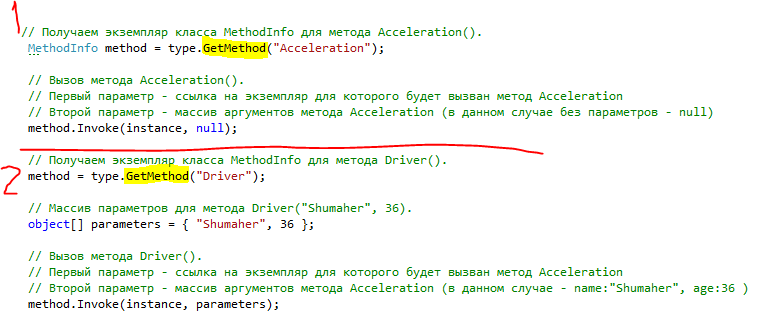


Выбираем нужные нам методы

Вызов метода Acceleration().

Первый параметр - ссылка на экземпляр для которого будет вызван метод Acceleration

Второй параметр - массив аргументов метода Acceleration (в данном случае без параметров - null)

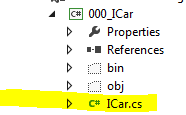


2-й способ

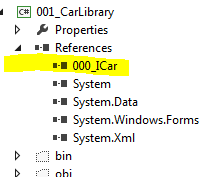
ПОЗДНЕЕ СВЯЗЫВАНИЕ

В отдельную сборку выносится интерфейс, который наследует класс сборки, к которой мы применим позднее связывание

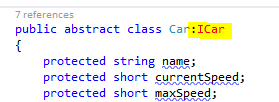
Сборка с интерфейсом



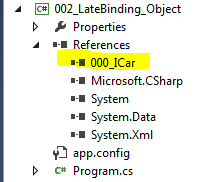
Сборка с подключенным интерфейсом



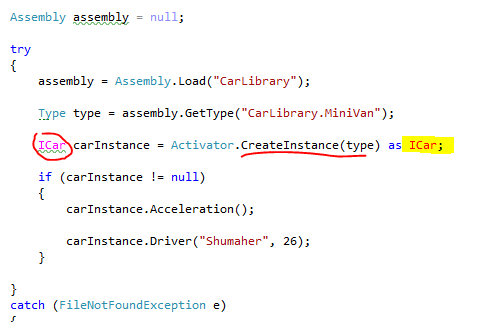
В этой сборке данный интерфейс наследует класс



Сборка где находится программа и где происходит позднее связывание (в ней так же подключается интерфейс)



Реализация создания экземпляра и приведение его к интерфейсу



3-й способ

ПОЗДНЕЕ СВЯЗЫВАНИЕ

Применение dynamic

