C# Essential

Классы и объекты

После урока обязательно

Повторите этот урок в видео формате на ITVDN.com

Проверьте как Вы усвоили данный материал на TestProvider.com

Тема

Классы и объекты

OOP

Object-Oriented Programming

ООП (Объектно-ориентированное программирование) – парадигма программирования, в которой основными концепциями являются понятия объектов и классов.

Class

Класс – это конструкция языка, состоящая из ключевого слова `class`, идентификатора (имени) и тела.

Класс может содержать в своем теле: поля, методы, свойства и события.

Поля определяют состояние, а ***методы*** поведение будущего объекта.

```
class MyClass
{
   public int field;  // Поле

   public void Method()  // Метод
   {
      Console.WriteLine(field);
   }
}
```

Объект и экземпляры

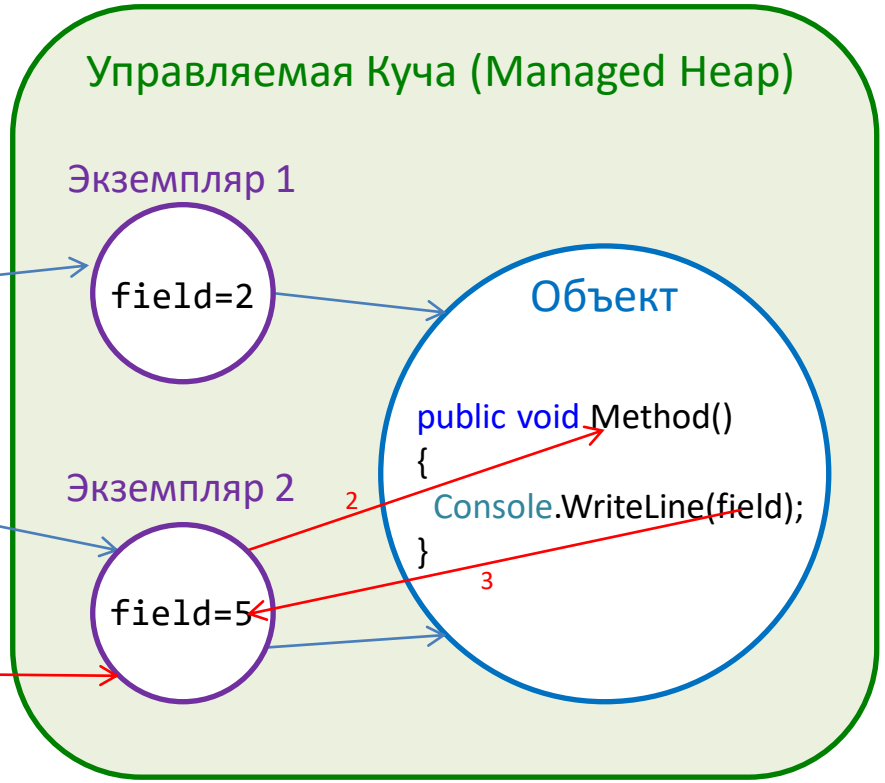
Object and instances

Объекты содержат в себе статические поля и все методы.
Экземпляры содержат нестатические поля.

```
MyClass instance1 = new MyClass();
MyClass instance2 = new MyClass();

instance1.field=2;
instance2.field=5;

instance1.Method();
instance2.Method();
```



Сокрытие реализации членов класса

Использование модификаторов доступа

Модификаторы доступа – `private` и `public` определяют видимость членов класса.

Никогда не следует делать поля открытыми, это плохой стиль.
Для обращения к полю рекомендуется использовать методы доступа.

Свойства

Property

Свойство – это конструкция языка C#, которая заменяет собой использование обычных методов доступа.

```
int field;

public int Property
{
   get
   {
      return field;
   }

   set
   {
      field = value;
   }
}
```

Работа со свойством экземпляра напоминает работу с полями экземпляра.

Свойство состоит из имени, типа и тела. В теле задаются методы доступа, через использование ключевых слов `set` и `get`.

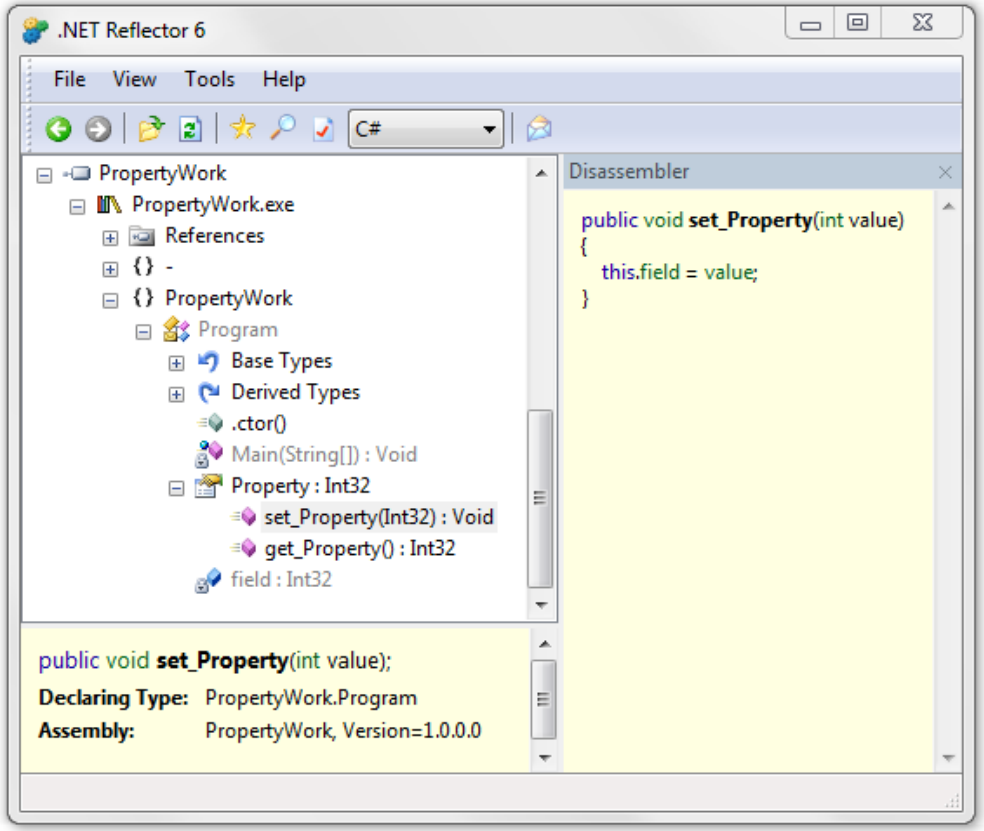
Метод `set` автоматически срабатывает тогда, когда свойству пытаются присвоить значение. Это значение представлено ключевым словом `value`.

Метод `get` автоматически срабатывает тогда, когда мы пытаемся получить значение.

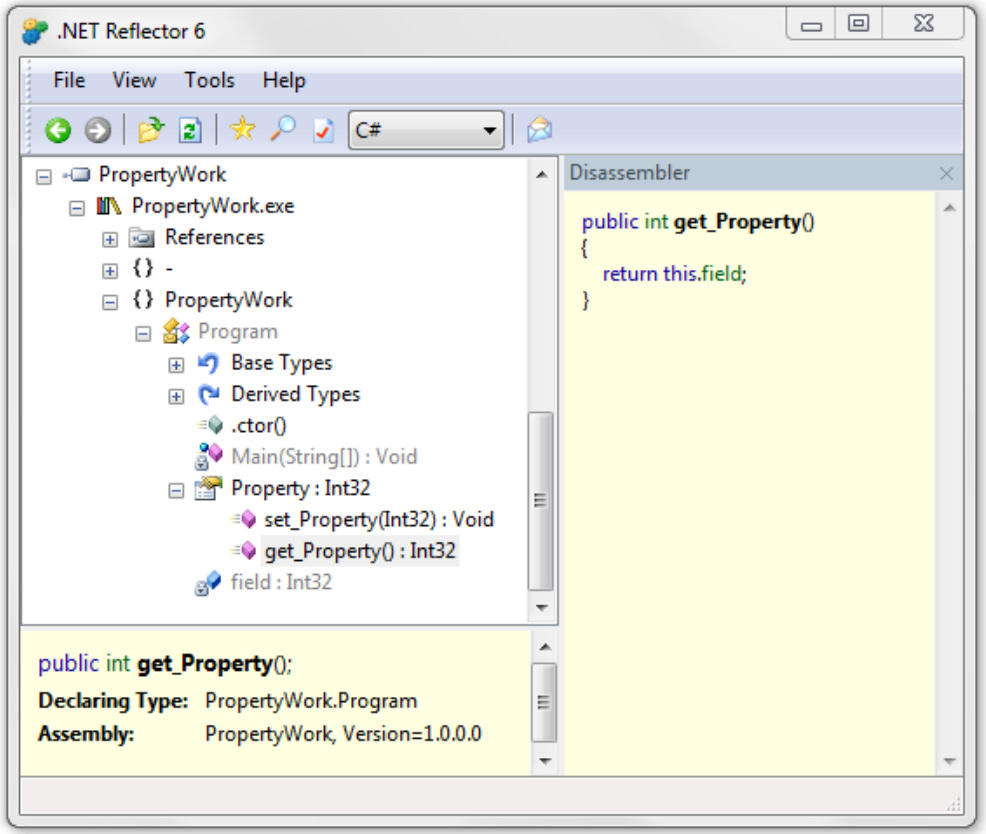
Свойства

Анализ

Анализ кода реализации свойств с использованием программы .NET Reflector.



Метод доступа `set`



Метод доступа `get`

Свойства

ReadOnly и WriteOnly

Метод доступа `get` – используется для получения значения из переменной.

Метод доступа `set` – используется для записи значения в переменную.

```
int field;

public int Property
{
   get
   {
      return field;
   }
}
```

Свойство только для чтения

```
int field;

public int Property
{
   set
   {
      field = value;
   }
}
```

Свойство только для записи

Конструктор

Constructor

Конструктор класса – специальный метод, который вызывается во время построения класса.

Конструкторы бывают двух видов:

Конструкторы по умолчанию

```
public MyClass()
{
}
```

Пользовательские конструкторы

```
public MyClass (int arg)
{
}
```

Если в теле класса не определен явно ни один конструктор, то всегда используется «невидимый» конструктор по умолчанию.
Имя конструктора всегда совпадает с именем класса. Конструкторы не имеют возвращаемых значений.

Конструктор

Constructor

Задача конструктора по умолчанию – инициализация полей значениями по умолчанию.

Задача пользовательского конструктора – инициализация полей предопределенными пользователем значениями.

Если в классе имеется пользовательский конструктор, и при этом требуется создавать экземпляры класса с использованием конструктора по умолчанию, то конструктор по умолчанию должен быть определен в теле класса явно, иначе возникнет ошибка на уровне компиляции.

Конструкторы, вызывающие другие конструкторы

Один конструктор может вызывать другой конструктор того же класса, если после сигнатуры вызывающего конструктора поставить ключевое слово `this` и указать набор параметров, который должен совпадать по количеству и типу с набором параметров вызываемого конструктора .

Вызывающий конструктор

```
public Point(string name)
             : this(300, 400)
{
    this.name = name;
}
```

Вызываемый конструктор

```
public Point(int x, int y)
{
    this.x = x;
    this.y = y;
}
```

Попытка вызова конструктора с не существующим набором параметров приведет к ошибке уровня компиляции.

Автоматически реализуемые свойства

Auto-Implemented Properties

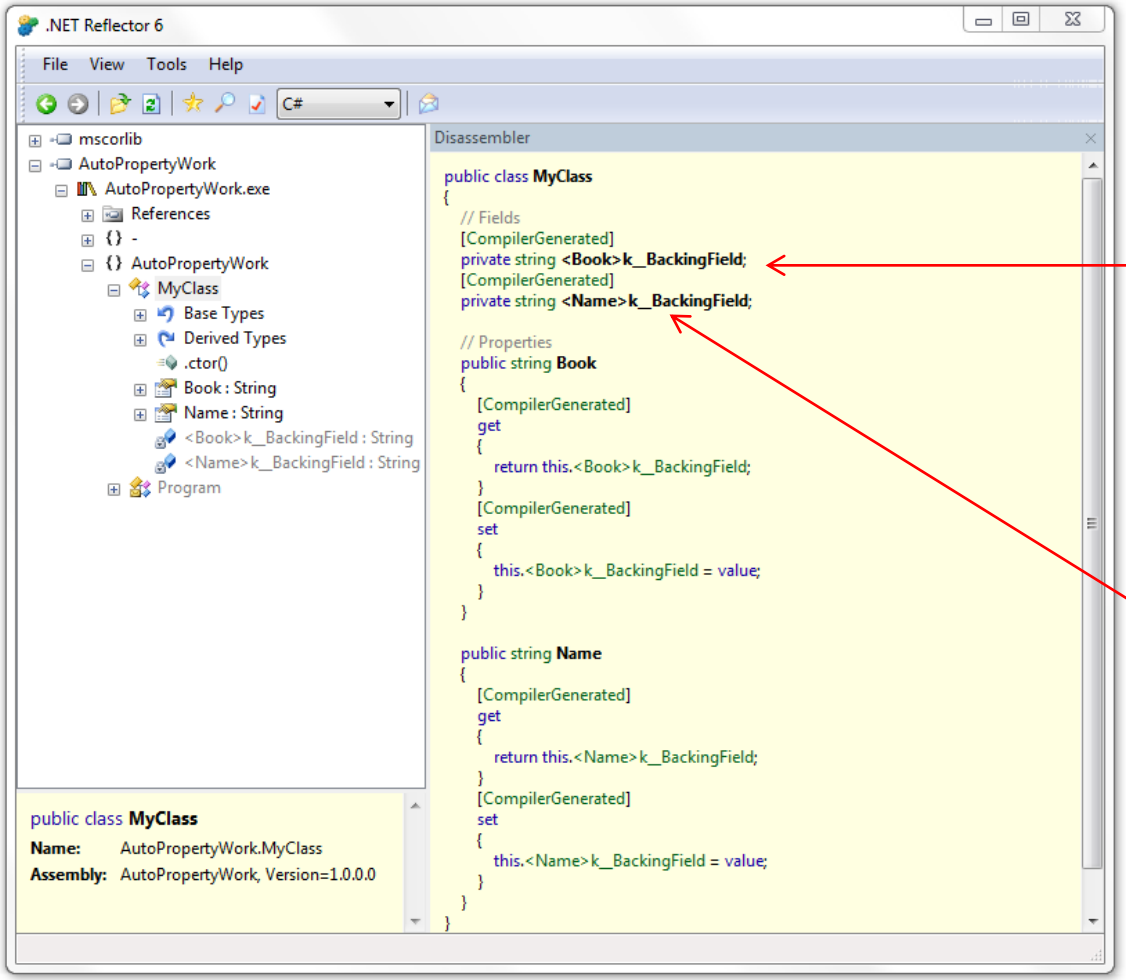
Автоматически реализуемые свойства это более лаконичная форма свойств, их есть смысл использовать, когда в методах доступа `get` и `set` не требуется дополнительная логика.

При создании автоматически реализуемых свойств, компилятор создаст закрытое, анонимное резервное поле, которое будет доступно с помощью методов `get` и `set` свойства.

```
public class MyClass
{
    public string Name { get; set; }
    public string Book { get; set; }
}
```

Автоматически реализуемые свойства

Auto-Implemented Properties



При создании автоматически реализуемых свойств, компилятор создаст закрытое, анонимное резервное поле, которое будет доступно с помощью методов доступа `get` и `set`.

```
public class MyClass
{
    public string Book { get; set;
}
    public string Name { get; set;
}
}
```

Сильные и слабые

Создание экземпляра класса по сильной ссылке

```
MyClass instance = new MyClass();
instance.Method();
```

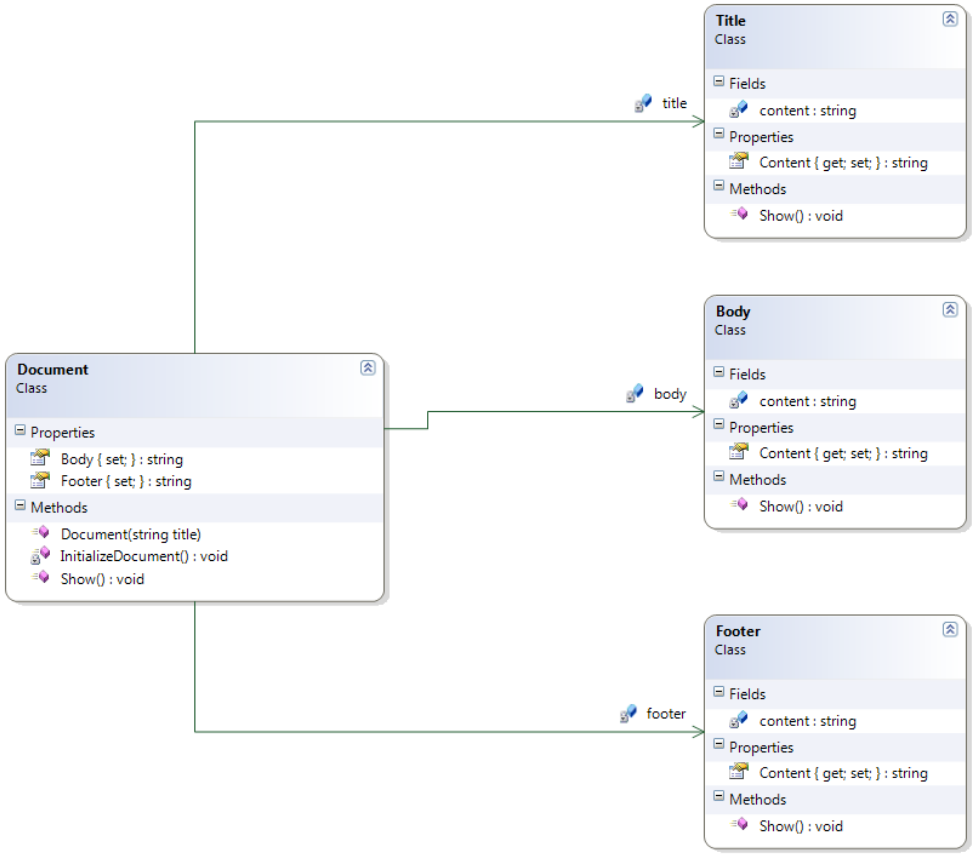
Создание экземпляра класса по слабой ссылке

```
new MyClass().Method();
```

Инкапсуляция

Первая парадигма ООП

Инкапсуляция *(инкапсуляция вариаций)* – техника сокрытия частей объектно-ориентированных программных систем.



Q&A

Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association