Анонимные типы

Последнее обновление: 15.10.2018

Анонимные типы позволяют создать объект с некоторым набором свойств без определения класса. Анонимный тип определяется с помощью ключевого слова **var** и инициализатора объектов:

```
var user = new { Name = "Tom", Age = 34 };
Console.WriteLine(user.Name);
```

В данном случае user - это объект анонимного типа, у которого определены два свойства Name и Age. И мы также можем использовать его свойства, как и у обычных объектов классов. Однако тут есть ограничение - свойства анонимных типов доступны только для чтения.

При этом во время компиляции компилятор сам будет создавать для него имя типа и использовать это имя при обращении к объекту. Нередко анонимные типы имеют имя наподобие "<>f__AnonymousType0'2".

Для исполняющей среды CLR анонимные типы будут также, как и классы, представлять ссылочный тип.

Если в программе используются несколько объектов анонимных типов с одинаковым набором свойств, то для них компилятор создаст одно определение анонимного типа:

```
var user = new { Name = "Tom", Age = 34 };
var student = new { Name = "Alice", Age = 21 };
var manager = new { Name = "Bob", Age = 26, Company = "Microsoft" };

Console.WriteLine(user.GetType().Name); // <>f__AnonymousType0'2
Console.WriteLine(student.GetType().Name); // <>f__AnonymousType0'2
Console.WriteLine(manager.GetType().Name); // <>f__AnonymousType1'3
```

Здесь user и student будут иметь одно и то же определение анонимного типа. Однако подобные объекты нельзя преобразовать к какому-нибудь другому типу, например, классу, даже если он имеет подобный набор свойств.

Следует учитывать, что свойства анонимного объекта доступны для установки только в инициализаторе. Вне инициализатора присвоить им значение мы не можем. Поэтому, например, в следующем случае мы столкнемся с ошибкой:

```
var student = new { Name = "Alice", Age = 21 };
student.Age = 32;   // ! Ошибка
```

Кроме исользованной выше формы инициализации, когда мы присваиваем свойствам некоторые значения, также можно использовать **инициализаторы с проекцией** (projection initializers), когда мы можем передать в инициализатор некоторые идиентификаторы, имена которых будут использоваться как названия свойств:

```
class User
{
    public string Name { get; set; }
}
class Program
{
    static void Main(string[] args)
    {
        User tom = new User { Name = "Tom" };
        int age = 34;
        var student = new { tom.Name, age}; // инициализатор с проекцией
        Console.WriteLine(student.Name);
        Console.WriteLine(student.age);
        Console.Read();
    }
}
```

В данном случае определение анонимного объекта фактически будет идентично следующему:

```
var student = new { Name = tom.Name, age = age};
```

Названия свойств и переменных (Name и age) будут использоваться в качесте названий свойств объекта.

Также можно определять массивы объектов анонимных типов:

```
var people = new[]
{
    new {Name="Tom"},
    new {Name="Bob"}
};
foreach(var p in people)
{
    Console.WriteLine(p.Name);
}
```

Зачем нужны анонимные типы? Иногда возникает задача использовать один тип в одном узком контексте или даже один раз. Создание класса для подобного типа может быть избыточным. Если нам захочется добавить свойство, то мы сразу же на месте анонимного объекта это можем сделать. В случае с классом придется изменять еще и класс, который может больше нигде не использоваться. Типичная ситуация - получение результата выборки из базы данных: объекты используются только для получения выборки, часто больше нигде не используются, и классы для них создавать было бы излишне. А вот анонимный объект прекрасно подходит для временного хранения выборки.

Дополнительные материалы

? Вопросы для самопроверки

Назад **Содержание** **Вперед**