

C# Essential

Универсальные шаблоны

C# Essential

После урока обязательно



Повторите этот урок в видео формате на [ITVDN.com](http://itvdn.com)



Проверьте как Вы усвоили данный материал
на [TestProvider.com](http://testprovider.com)

C# Essential

Тема

Универсальные шаблоны

Обобщение

Generic

Обобщение (Универсальные шаблоны) – элемент кода, способный адаптироваться для выполнения общих (сходных) действий над различными типами данных.



Универсальные шаблоны были добавлены в язык C# версии 2.0 и среду **CLR**. Эта возможность **CTS (Common Type System)** – общая система типов), названа обобщениями (**generics**).

Обобщение

Преимущества обобщений

Обобщения позволяют создавать открытые (**open-ended**) типы, которые преобразуются в закрытые во время выполнения.

Идентификатор **<T>** – это указатель места заполнения, вместо которого подставляется любой тип.

```
class Types<T>
{
    T[] mass = new T[5];
}
```

Создание открытого типа

```
static void Main()
{
    Types<int> type = new Types<int>();
}
```

Закрытый тип



Каждый закрытый тип получает свою собственную копию набора статических полей.

Обобщение

Boxing-Unboxing

Обобщения обеспечивают большую производительность, так как не происходит операции "упаковки-распаковки" (**Boxing-Unboxing**).

```
class MyClass<T>
{
    public T field;

    public void Method()
    {
        Console.WriteLine(field.GetType());
    }
}

static void Main()
{
    MyClass<int> instance1 = new MyClass<int>();
    MyClass<string> instance2 = new MyClass<string>();
    instance1.Method();
    instance2.Method();
}
```

Обобщения обеспечивают безопасность типов, так как могут содержать только типы, которые задаются при объявлении.

Ковариантность обобщений

UpCast

Ковариантность обобщений – **UpCast** параметров типов.

```
class Animal { }  
class Cat : Animal { }  
  
delegate T MyDelegate<out T>();  
  
static void Main()  
{  
    MyDelegate<Cat> delegateCat = () => new Cat();  
    MyDelegate<Animal> delegateAnimal = delegateCat;  
  
    Animal animal = delegateAnimal.Invoke();  
    Console.WriteLine(animal.GetType().Name);  
}
```



Ковариантность обобщений в C# 4.0 ограничена интерфейсами и делегатами.

Контрвариантность обобщений

DownCast

Контрвариантность обобщений – **DownCast** параметров типов.

```
class Animal { }  
class Cat : Animal { }  
delegate void MyDelegate<in T>(T a);  
  
static void Main()  
{  
    MyDelegate<Animal> delegateAnimal = (Animal animal) =>  
        Console.WriteLine(animal.GetType().Name);  
  
    MyDelegate<Cat> delegateCat = delegateAnimal;  
  
    delegateAnimal(new Animal());  
    delegateCat(new Cat());  
}
```



Контрвариантность обобщений в C# 4.0 ограничена интерфейсами и делегатами.

Универсальные шаблоны

Перегрузка обобщенных типов

Перегрузки обобщенных типов различаются количеством параметров типа, а не их именами.

```
class MyClass<T>
{
    T[] mass = new T[5];
}
```

```
class MyClass<T,R>
{
    T[] mass = new T[5];
    R[] mass = new R[5];
}
```

Универсальные шаблоны

Общие сведения

Общие сведения об универсальных шаблонах:

- Используйте универсальные типы для достижения максимального уровня повторного использования кода, безопасности типа и производительности.
- Наиболее частым случаем использования универсальных шаблонов является создание классов коллекции.
- Можно создавать собственные универсальные интерфейсы, классы, методы, события и делегаты.
- Доступ универсальных классов к методам можно ограничить определенными типами данных.

Nullable

Тип Nullable

Тип `Nullable<T>` представляет типы значений с пустыми (нулевыми) значениями.

```
int? a = null;  
int? b = a + 4;
```

b = null

При сравнении операндов один из которых `null` – результатом сравнения всегда будет – `false`.

```
int? a = null;  
int? b = -5;  
  
if (a >= b) // - false  
{  
    Console.WriteLine("a >= b");  
}  
else  
{  
    Console.WriteLine("a < b");  
}
```

Nullable

Операция поглощения

```
static void Main()
{
    int? a = null;
    int? b;

    b = a ?? 10; // b = 10

    a = 3;
    b = a ?? 10; // b = 3
}
```

Оператор `??` возвращает левый операнд, если он не `null` и правый операнд, если левый `null`.

C# Essential

Q&A

Информационный видеосервис для разработчиков программного обеспечения

