



6. Структури от данни в .NET

Част втора Generics



Съдържание (1/2)

- Какво е Generic?
- Разлики между Generics и шаблоните в C++
- Деклариране
 - Generic класове
 - Generic методи
- Повече от един параметър
- Ограничения



Съдържание (2/2)

- Как да използваме Generic класове и методи?
- Generic интерфейси – `IEnumerable<T>`, `ICollection<T>`, `IList<T>`, `IDictionary<T,V>`
- Важни Generic класове
- Generic колекции и списъци vs. нетипизирани колекции и списъци?



Generic?

- В превод:

GENERIC

1. *родов*
2. *общ*
3. *обширен, с широко приложение*

- В програмирането



Generics vs. Templates (1/2)

Generics в C# не са толкова гъвкави като шаблоните в C++.

- Само типове могат да са параметри
- Не се поддържат специализации
- Не се поддържат частични имплементации



Generics vs. Templates (2/2)

- Не се поддържат стойности по подразбиране
- Кодът трябва да е валиден за всеки тип, с който може да бъде заместен параметъра
- Получаването на крайния (CLR) тип се получава по време на изпълнение



Generic класове и методи (1/4)

- С помощта на Generics можем да имплементираме функционалност без да конкретизираме тип
- Типът който не конкретизираме е параметър
- Могат да се параметризират както класове така и методи



Generic класове и методи (2/4)

- Един клас е generic, като след името му следва <T>, където T е параметър
- В рамките на класа това име означава тип

```
class GenericClass<T>
{
    public static T GetValue()
    {
        return default(T);
    }
}
```




Generic класове и методи (3/4)

- Поддържат се и generic методи в рамките на клас който или е, или не е generic
- Това става като след името на метода следва <T>, където T е име на параметър

```
class NonGenericClass
{
    public static T GetGenericValue<T>()
    {
        return default(T);
    }
}
```



Generic класове и методи (4/4)

- Обръщение - <T> след името, където T е име на тип
- Обръщения към дефинираните на предходния слайд класове/методи

```
GenericClass<int>.GetValue();
```

```
NonGenericClass.GetGenericValue<int>();
```



(де

ве и



По-сложни сценарии

- Повече параметри - <T, R, E, ...>
- Ограничения на типове, които могат да заместят параметъра – оказват се посредством ключовата дума **where**



Ограничения на параметрите

- where T: struct
- where T: class
- where T: new()
- where T: <base class name>
- where T: <interface name>
- where T: U



(пое



тър;



Generic интерфейси?

- Generic могат да бъдат не само методите и класове, но и интерфейсите
- `IEnumerable<T>` vs. `IEnumerable`
- `ICollection<T>` vs. `ICollection`
- Всъщност `System.Collections.Generic` е новата версия на `System.Collections...`



Пир

лове





Generic колекции за всички!

Generic колекциите са силно типизирани. От това следват следните неща:

- **Сигурни са** – не могат да съдържат елемент, който не е от посочения тип
- Не предизвикват излишен **boxing** и **unboxing** на елементите си
- Не ни е нужно да правим преобразования
- Съдържат повече функционалност (понеже са създадени по-късно)



Въпроси?

