



Асинхронность в Entity Framework

Асинхронные операции

Последнее обновление: 31.10.2015



Начиная с версии 6.0 Entity Framework поддерживает асинхронные операции. Для сохранения результатов в базе данных в асинхронном режиме используется метод **SaveChangesAsync**.

Чтобы получить объект по id в асинхронном режиме, в классе DbSet определен метод **FindAsync**.

Некоторые методы Linq to Entities также имеют асинхронных двойников для осуществления запросов в асинхронном режиме:

- **ForEachAsync**: асинхронное извлечение данных и выполнение над ними определенных действий
- **AllAsync**: удовлетворяет ли все элементы в выборке определенному условию
- **AnyAsync**: удовлетворяет ли хотя бы один элемент выборки определенному условию
- **AverageAsync**: асинхронное получение среднего значения
- **ContainsAsync**: содержит ли выборка определенный элемент
- **CountAsync**: получение размера выборки
- **FirstAsync**: получение первого элемента
- **FirstOrDefaultAsync**: получение первого элемента или значения по умолчанию
- **LoadAsync**: асинхронная загрузка данных в кэш
- **MaxAsync**: получение максимального значения
- **MinAsync**: получение минимального значения
- **SingleAsync**: получение одного элемента
- **SingleOrDefaultAsync**: получение одного элемента или значения по умолчанию
- **SumAsync**: асинхронное получение суммы значений

Все методы возвращают объект задачи **Task** или **Task<T>**

Например, выполним асинхронное сохранение и асинхронную выборку из БД:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.Entity;
using System.Threading.Tasks;

class Program
```

```
{
    static void Main(string[] args)
    {
        Phone p = new Phone { Name = "Nokia Lumia 930", Price = 13000 };

        SaveObjectsAsync(p).Wait();

        Task t = GetObjectsAsync();
        t.Wait();

        Console.Read();
    }
    public static async Task GetObjectsAsync()
    {
        using (MobileContext db = new MobileContext())
        {
            await db.Phones.ForEachAsync(p =>
            {
                Console.WriteLine("{0} ({1})", p.Name, p.Price);
            });
        }
    }

    private static async Task SaveObjectsAsync(Phone p)
    {
        using (MobileContext db = new MobileContext())
        {
            db.Phones.Add(p);
            await db.SaveChangesAsync();
        }
    }
}
```

Кроме асинхронных операций Linq to Entities нам доступно асинхронное осуществление команд в БД с помощью метода **ExecuteSqlCommandAsync**:

```
private static async Task DbCommandAsync(Phone p)
{
    using (MobileContext db = new MobileContext())
    {
        System.Data.SqlClient.SqlParameter name = new System.Data.SqlClient.SqlParameter("name", p.Name);
        System.Data.SqlClient.SqlParameter price = new System.Data.SqlClient.SqlParameter("price", p.Price);
        await db.Database.ExecuteSqlCommandAsync("INSERT INTO Phones (Name, Price) VALUES (@name, @price)", name,
price);
    }
}
```

Применение:

```
Phone p2 = new Phone { Name = "iPhone 6", Price = 33000 };
DbCommandAsync(p2).Wait();
```

[Назад](#) [Содержание](#)



[Вконтакте](#) | [Twitter](#) | [Google+](#) | [Канал сайта на youtube](#) | [Помощь сайту](#)

Контакты для связи: metanit22@mail.ru

Copyright © metanit.com, 2012-2017. Все права защищены.