Создание первого сервиса.  
metanit.com/sharp/aspnet5/2.19.php  
[metanit.com/sharp/aspnet5/6.2.php](https://metanit.com/sharp/aspnet5/6.2.php)

При создании базового интерфейса появилась необходимость вывода на страницу последних добавленных документов. Естественно, обращаться к базе данный каждый раз при переходе пользователя на страницу с «последними статьями» - плохая идея. Что если на неё одновременной зайдет 100 человек? Будет создан сервис, который будет хранить в себе 5 последних статей и будет обновлять их раз в определенный интервал (по умолчанию 1 минута). Именно по такому принципу работают большие маркет-плейсы, они никак не могут обработать тысячи запросов путем прямого обращения к БД. Поэтому они заранее подготавливают ответы на определенное количество наиболее популярных запросов, обновляя их с определенным периодом.

Lifetime созданного сервиса – Singleton, или же «Одиночка». Экземпляр объекта создаётся один раз при первом обращении к нему и существует по мере существования самого приложения. В связи с этим невозможно внедрить зависимость непосредственно от DbContext, ведь его lifetime – Scope, т.е. он умирает вместе с окончанием исполнения метода. В данном случае необходимо внедрить зависимость от IServiceScopeFactory, и создать экземпляр DbContext уже непосредственно в самом сервисе.

Краткий код созданного сервиса:

RecentDocumentsBackgroundService : Microsoft.Extensions.Hosting.BackgroundService {

private IServiceScopeFactory DbContextScopeFactory;

public int Interval\_msec = 1000 \* 60 \* 1; // 1 min interval

public RecentDocumentsBackgroundService(IServiceScopeFactory DbContextScopeFactory)  
=> this.DbContextScopeFactory = DbContextScopeFactory;

public IQueryable<Website.Models.DocumentModel.DbDocument> RecentDocuments { get; private set; }

protected async override Task ExecuteAsync(CancellationToken ct)

{

using (WebsiteContext context = this.DbContextScopeFactory

.CreateScope().ServiceProvider.GetRequiredService<WebsiteContext>())

{

while (!ct.IsCancellationRequested)

{

var NewRecent = context.DbDocuments.OrderBy(d => d.CreatedDateTime);

NewRecent.Skip(NewRecent.Count() - 5);

this.RecentDocuments = NewRecent;

await Task.Delay(this.Interval\_msec);

}

}

}}