

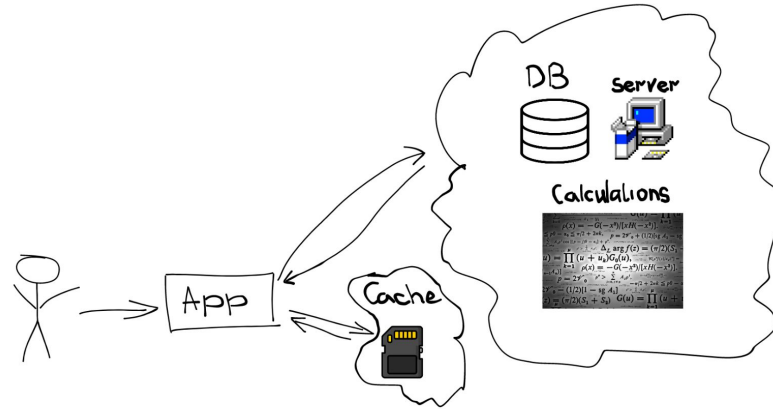


Кеширование в .NET 7

Кузьмин Сергей

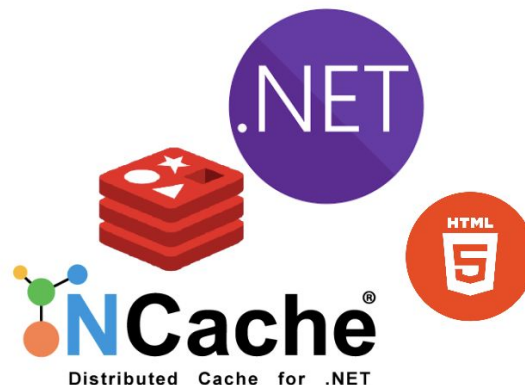
Кеширование

Кеш - ограниченный по размеру буфер данных, доступ к которому осуществляется быстрее, чем к исходному источнику



Кеширование в .NET

- In-memory caching
- Distributed Cache
- Cache Tag Helper
- Distributed Cache Tag Helper
- Response caching
- **Output Caching**











Response Caching

- Основан на HTTP-заголовках
- Кеш хранится на сервере
- Ключом кеша является url запроса с набором заголовков Vary (параметры запроса не учитываются)
 - Параметры запроса можно настроить с помощью VaryByQuery
- Ограничен In-Memory кешем
- Использует default policy, которую невозможно перенастроить

Response Caching: Default Policy

- Кешируются только GET / HEAD запросы
- Кешируются только ответы с HTTP-кодом 200
- Не кешируются ответы, если у запроса есть Authorization заголовок
- Не кешируются ответы с Set-Cookie заголовком

 GET	 POST	 PUT	 DELETE	 PATCH	 HEAD
retrieve data from server	add data to an existing file or resource	update(replace) an existing file or resource in server	delete data from server	update a resource partially (modify)	retrieve the resource's headers



Output Caching

- Конфигурируется **вами**
- Поддерживает Minimal APIs
- Можно инвалидировать кодом
- Поддерживает блокировку ресурсов
- Позволяет выбрать хранилище для кеша

```
61  
62 app.MapGet(pattern: "/cached", handler: () => $"Hello World! Now is: {DateTime.UtcNow}").CacheOutput();  
63
```



Output Caching: Настройка Policy

```
25
26 builder.Services.AddOutputCache(options =>
27 {
28     options.AddBasePolicy( build: policyBuilder =>
29     {
30         policyBuilder.SetVaryByHost(false);
31         policyBuilder.SetVaryByHeader( headerName: "User-Agent");
32         policyBuilder.SetVaryByQuery( queryKey: string.Empty);
33     }, excludeDefaultPolicy: true);
34
35     options.AddPolicy( name: "WithAllParams", build: policyBuilder => policyBuilder.SetVaryByQuery( queryKey: "*"));
36 });
--
```



Output Caching: Настройка Policy

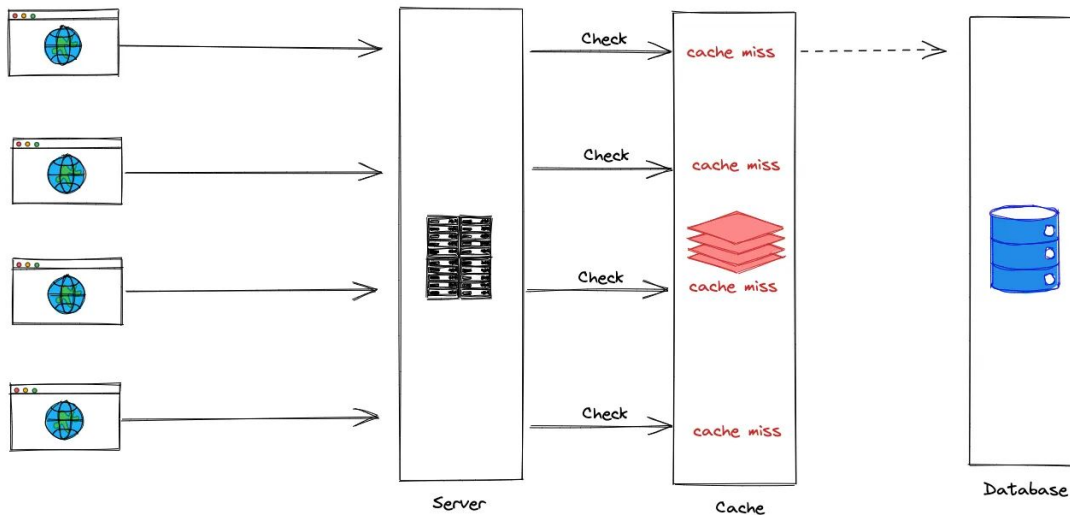
```
71
72 var cachedMapGroup = app.MapGroup(prefix: "/cached").CacheOutput(policyBuilder =>
73 {
74     policyBuilder.With(x:OutputCacheContext => x.HttpContext.Request.Query.ContainsKey("test-param"));
75     policyBuilder.AddPolicy<MyCustomPolicy>();
76     policyBuilder.Expire(TimeSpan.FromSeconds(30));
77 });
78
```


Output Caching: Инвалидация

```
69 var cachedMapGroup = app.MapGroup(prefix: "/cached").CacheOutput("Expire30");
70
71 cachedMapGroup // RouteGroupBuilder
72     .MapGet(pattern: "/tags", handler: () => $"Hello World! Tag1 and tag2 are here! Now is: {DateTime.UtcNow}")
73     .CacheOutput(policyBuilder =>
74     {
75         policyBuilder.Tag("tag1", "tag2");
76     });
77
78 cachedMapGroup.MapGet(pattern: "/evict/{tag}", handler: async (IOutputCacheStore cacheStore, string tag) =>
79 {
80     await cacheStore.EvictByTagAsync(tag, cancellationToken: default);
81 });
```

Output Caching: Блокировка Ресурсов

- Thundering herd problem





Output Caching: Cache Storage

- MemoryCache
- DistributedCache

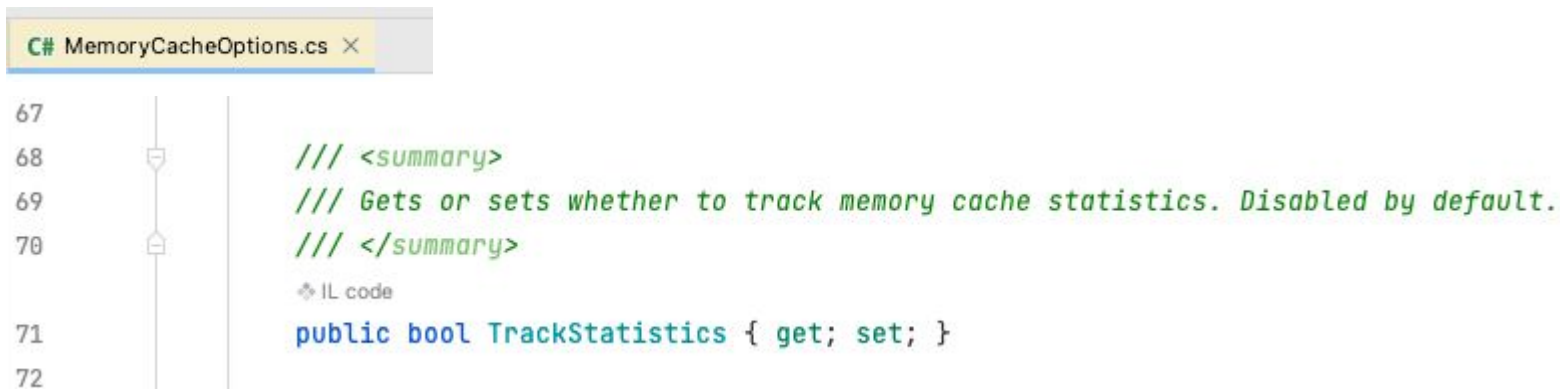
```
20     builder.Services.AddSingleton<IOutputCacheStore>(sp:IServiceProvider =>
21     {
22         var memoryCache = sp.GetRequiredService<IMemoryCache>();
23         return new MyOutputCacheStore(memoryCache);
24     });
```



Output Caching: Выводы

- Управление политиками
- Управление хранилищем
- Блокировка ресурсов
- Minimal API

MemoryCache: Метрики



The image shows a screenshot of a Visual Studio code editor window. The title bar at the top reads "C# MemoryCacheOptions.cs" with a close button on the right. The editor contains C# code for the `TrackStatistics` property. Line numbers 67 through 72 are visible on the left margin. The code includes XML documentation comments in green and a property definition in blue. A vertical line is positioned at line 68. To the left of the code, there are two small icons: a square with a right-pointing arrow at line 68, and a square with a left-pointing arrow at line 70. Below the code, there is a small icon of a diamond with a cross inside, followed by the text "IL code".

```
67
68 /// <summary>
69 /// Gets or sets whether to track memory cache statistics. Disabled by default.
70 /// </summary>
    ⬢ IL code
71 public bool TrackStatistics { get; set; }
72
```



MemoryCache: Метрики

- `public long CurrentEntryCount { get; init; }`
- `public long? CurrentEstimatedSize { get; init; }`
- `public long TotalMisses { get; init; }`
- `public long TotalHits { get; init; }`



MemoryCache: Метрики

```
/// <summary>  
/// Gets a snapshot of the cache statistics if available.  
/// </summary>  
/// <returns>  
/// An instance of <see cref="MemoryCacheStatistics"/> containing a snapshot of the cache statistics.  
/// </returns>
```

⇨ IL code

```
MemoryCacheStatistics? GetCurrentStatistics() => null;
```

MemoryCache: Метрики

- Response Cache ❌
- Output Cache ✅

```
13     builder.Services.AddMemoryCache(options =>
14     {
15         options.TrackStatistics = true;
16     });
17
18     builder.Services.AddSingleton<IOutputCacheStore>(sp :IServiceProvider =>
19     {
20         var memoryCache = sp.GetRequiredService<IMemoryCache>();
21         return new MyOutputCacheStore(memoryCache);
22     });
```




Ссылки

- <https://learn.microsoft.com/en-us/aspnet/core/performance/caching/overview?view=aspnetcore-7.0>
- <https://timdeschryver.dev/blog/prevent-a-net-api-from-adding-cache-headers-to-unsuccessful-requests#bad-cache-attributes>
- <https://devblogs.microsoft.com/dotnet/asp-net-core-updates-in-dotnet-7-rc-2/#output-caching-improvements>
- <https://timdeschryver.dev/blog/exploring-the-new-output-caching-middleware>
- <https://devblogs.microsoft.com/dotnet/announcing-dotnet-7-preview-4/#added-metrics-for-microsof-extensions-caching>
- <https://www.rfc-editor.org/rfc/rfc9111>



Спасибо за внимание!