

第79期-使用变更令牌检测目标变更

2019年12月18日 13:23

变更令牌用于跟踪状态变化的目标对象（文件变更通知、CLR对象变更通知，缓存过期通知）。

IChangeToken 接口

IChangeToken 用于传播已发生更改的通知。

Install-Package Microsoft.Extensions.Primitives -Version 3.1.0

ASP.NET CORE 共享框架集已包含此包，其它 .NET Core 其它项目模板需要单独安装。

IChangeToken 接口

该接口2个属性和1个方法

ChangeToken 类

ChangeToken 是静态类，用于传播已发生更改的通知，ChangeToken 类没有实现 IChangeToken 接口，跟 IChangeToken 接口没有任何关系，ChangeToken 类提供了两个 OnChange 静态方法来注册令牌与回调。

ChangeToken 类主要是将实现了令牌变更接口的类与回调通知关联起来，提供一个注册机制。

OnChange 返回 IDisposable。调用 Dispose 将使令牌停止侦听更多更改并释放令牌的资源。

ASP.NET Core 中更改令牌的使用示例

更改令牌主要用于在 ASP.NET Core 中监视对象更改：

- IFileProvider 的 Watch 方法将为要监视的指定文件
- 可以将 IChangeToken 令牌添加到缓存条目，以在更改时触发缓存逐出。
- IOptionsChangeTokenSource<TOptions> 检测配置项变化

监视配置文件更改

默认情况下, ASP.NET Core 配置文件监听配置文件, 此设置出现在 Host 便捷方法 CreateDefaultBuilder 中。

<https://github.com/aspnet/Extensions/blob/master/src/Hosting/Hosting/src/Host.cs#L74>

```
config.AddJsonFile("appsettings.json", optional: true, reloadOnChange: true)
    .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true,
        reloadOnChange: true);
```

ConfigurationBuilder->FileConfigurationSource->PhysicalFileProvider->IFileMonitor->
FileSystemWatcher

监控缓存文件更改

可以使用 缓存技术将内容缓存在内存中, 如果源数据更改, 更新缓存或删除缓存。

```
// Obtain a change token from the file provider whose
// callback is triggered when the file is modified.
var changeToken = _fileProvider.Watch(fileName);

// Configure the cache entry options for a five minute
// sliding expiration and use the change token to
// expire the file in the cache if the file is
// modified.
var cacheEntryOptions = new MemoryCacheEntryOptions()
    .SetSlidingExpiration(TimeSpan.FromMinutes(5))
    .AddExpirationToken(changeToken);

// Put the file contents into the cache.
_cache.Set(filePath, fileContent, cacheEntryOptions);
```

CompositeChangeToken 类

要在单个对象中表示一个或多个 IChangeToken 实例, 请使用 CompositeChangeToken 类。

```
var firstCancellationTokenSource = new CancellationTokenSource();
var secondCancellationTokenSource = new CancellationTokenSource();

var firstCancellationToken = firstCancellationTokenSource.Token;
var secondCancellationToken = secondCancellationTokenSource.Token;

var firstCancellationChangeToken = new
CancellationChangeToken(firstCancellationToken);
var secondCancellationChangeToken = new
```

```
CancellationChangeToken(secondCancellationToken);
```

```
var compositeChangeToken =  
    new CompositeChangeToken(  
        new List<IChangeToken>  
        {  
            firstCancellationChangeToken,  
            secondCancellationChangeToken  
        }  
    );
```

如果任何表示的令牌 HasChanged 为 true，则复合令牌上的 HasChanged 报告 true。如果任何表示的令牌 ActiveChangeCallbacks 为 true，则复合令牌上的 ActiveChangeCallbacks 报告 true。如果发生多个并发更改事件，则调用一次复合更改回调。

实现自己的 IChangeToken 类

<https://github.com/aspnet/Extensions/blob/master/src/Primitives/src/CancellationChangeToken.cs>

<https://github.com/aspnet/Extensions/blob/master/src/Configuration/Config/src/ConfigurationReloadToken.cs>

<https://github.com/aspnet/Extensions/blob/master/src/FileProviders/Abstractions/src/NullChangeToken.cs>

附录阅读

<https://www.cnblogs.com/artech/p/inside-asp-net-core-04-02.html>

<https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/change-tokens>