Implement distributed caching in an ASP.NET Core Step by step (With Redis)

Defined distributed cache

A distributed cache is a cache shared by multiple app servers, typically maintained as an external service to the app servers that access it. A distributed cache can improve the performance and scalability of an ASP.NET Core app

Example Source : https://github.com/abolfazlSadeqi/CurdRedisDistributedCaching

Steps:

1.Install Microsoft.Extensions.Caching.StackExchangeRedis nuget package.

2. configure the Redis service with `AddStackExchangeRedisCache(in Program file or Startup)`

Example:

```
services.AddStackExchangeRedisCache(options =>
            {
                options.Configuration =
Configuration.GetConnectionString("RedisConnection");

  });
```

3.Add Connection into appsettings.json

Example:

```
"RedisConnection": " localhost:6379 "
```

4. Implement all of Methods `IDistributedCache` For projects(`Set, Get, Remove,...`)

5. Create a list of `CacheKey per Cache`

Example:

```
public static readonly  string PersonCacheKey = "Personall";
```

6.Use Cache in mvc(in Controller,Action)

Example:

a.Controller

```
private readonly IDistributedCache _DistributedCache;

public PersonController(IDistributedCache DistributedCache )
        {
            _DistributedCache = DistributedCache;
        }
```

b.Add Items(Check Exists Cache Based on Key and use SetAsync)

```
if (!_DistributedCache.TryGetValue(ListCache.PersonCacheKey, out
IEnumerable<PersonDto>? PersonDtos))
            {
                var Persons = unitOfWork.Person.GetAll();
                var newperson = _mapper.Map<List<PersonDto>>(Persons);

                await _DistributedCache.SetAsync(ListCache.PersonCacheKey,
newperson);
                return View(newperson);
            }

            return View(PersonDtos);
```