Working With Caching in C#

## **Caches**

A cache is the most valuable feature that Microsoft provides. It is a type of memory that is relatively small but can be accessed very quickly. It essentially stores information that is likely to be used again. For example, web browsers typically use a cache to make web pages load faster by storing a copy of the webpage files locally, such as on your local computer.

### **Why do we need caching?**

Our applications often calls same method, again and again and fetch data from database, but sometimes, data doesn't get's changed or updated in database, in that case, we can use caching to reduce database calls and get's same data directly from memory-cache.

## **Caching**

Caching is the process of storing data into cache. Caching with the C# language is very easy. System.Runtime.Caching.dll provides the feature for working with caching in C#.

1. ObjectCache
2. MemoryCache
3. CacheItemPolicy

**ObjectCache:**The CacheItem class provides a logical representation of a cache entry, that can include regions using the [RegionName](http://msdn.microsoft.com/en-us/library/system.runtime.caching.cacheitem.regionname(v=vs.110).aspx) property. It exists in the System.Runtime.Caching.  
  
**MemoryCache:** This class also comes under System.Runtime.Caching and it represents the type that implements an in-cache memory.  
  
**CacheItemPolicy:** Represents a set of eviction and expiration details for a specific cache entry.

Lets take an example of storing stock items in cache memory

I have kept some stock items in the cache memory for further use that is to understand the availability of a stock item in the future. When the application does not find any cached data in cache memory, it will redirect to the Get default stock function.

1. **private** **const** **string** CacheKey = "availableStocks";
2. CacheItemPolicy cacheItemPolicy = **new** CacheItemPolicy();
3. cacheItemPolicy.AbsoluteExpiration = DateTime.Now.AddHours(1.0);
4. cache.Add(CacheKey, availableStocks, cacheItemPolicy);

Code above the line number 02 tells the system to create a policy for storing information in the cache, Line 03 specifies which policy to use. There are the following two types of policies available in the cashitempolicy class.

1. **AbsoluteExpiration:** Indicating whether sliding expiration is enabled.

**In this the cache will be expired after a particular time , irrespective of whether the cache has been used or not in that time span.**

1. **SlidingExpiration:** Policy specifies when a cache entry should be evicted after a specified duration.

**In this case , the cache will be expired after a particular time only if it has not been used during that time span.**

I have given one hour duration to expire cache memory and line number 04 stores information with a unique key and cache policy. The Add() method stores information into memory and the Add() method is three types based its parameter. They are as in the following:

1. Add(CacheItem Item,CacheItemPolicy policy)
2. Add(string key,object value,CacheItemPolicy policy, string regionname)
3. Add(string key,object value,DateTimeOffset absoluteExpiration, string regionname)

**CacheKey** is a unique key by which we can query in the future. The **CacheItem** class represents an individual cache entry in the cache, in other words this class has a structure to store information such as Region name, Key and value to store in memory. absoluteExpiration is like a time duration for expiring or releasing cached information.

## **Retrieve Information from Cache**

Getting or querying information from cache memory is as easy as storing into memory. The Get function of the ObjectCache class helps to retrieve information from cache memory. This Get() method takes two parameters for querying data from cache memory. Key and region name. See the following code that helps to retrieve information from memory.

1. ObjectCache cache = MemoryCache.Default;
2. Enumerable information=(IEnumerable)cache.Get(CacheKey);

Example 1

**using** System.Runtime.Caching;

**static** **void** **Main**(**string**[] args)

{

ObjectCache cache = MemoryCache.Default;

//add cache

cache.Add("CacheName", "Value1", null);

cache.Add("CacheName2", 0, null);

// create cache item policy

**var** cacheItemPolicy = **new** CacheItemPolicy

{ AbsoluteExpiration = DateTimeOffset.Now.AddSeconds(60.0),

};

//add cache with cache item policy

cache.Add("CacheName3", "Expires In A Minute", cacheItemPolicy);

//add cache with CacheItem object

**var** cacheItem = **new** CacheItem("fullName", "Nandini Shamdasani");

cache.Add(cacheItem, cacheItemPolicy);

//get cache value and print

Console.WriteLine("Full Name "+ cache.Get("fullName"));

//print all cache

Console.WriteLine("All key-values");

PrintAllCache(cache);

//remove cache

cache.Remove("CacheName");

//update cache value, from 0 to 1

cache.Set("CacheName2", 1, null);

//print all cache key value again to check updates

Console.WriteLine("All key-values after updates");

PrintAllCache(cache);

}

**public** **static** **void** **PrintAllCache**(ObjectCache cache)

{

//loop through all key-value pairs and print them

**foreach** (**var** item **in** cache)

{

Console.WriteLine("cache object key-value: "+ item.Key + "-" + item.Value);

}

Example 2

**namespace** CachingWithConsoleApplication

{

**class** Program

    {

**static** **void** Main(**string**[] args)

        {

            StockItems PS = **new** StockItems();

            List<**string**> items= (List<**string**>)  PS.GetAvailableStocks();

            items = (List<**string**>)PS.GetAvailableStocks();

        }

    }

**public** **class** StockItems

    {

**private** **const** **string** CacheKey = "availableStocks";

**public** IEnumerable GetAvailableStocks()

        {

            ObjectCache cache = MemoryCache.Default;

**if** (cache.Contains(CacheKey))

**return** (IEnumerable)cache.Get(CacheKey);

**else**

            {

                IEnumerable availableStocks = **this**.GetDefaultStocks();

                // Store data in the cache

                CacheItemPolicy cacheItemPolicy = **new** CacheItemPolicy();

                cacheItemPolicy.AbsoluteExpiration = DateTime.Now.AddHours(1.0);

                cache.Add(CacheKey, availableStocks, cacheItemPolicy);

**return** availableStocks;

            }

        }

**public** IEnumerable GetDefaultStocks()

        {

**return** **new** List<**string**>() { "Pen", "Pencil", "Eraser" };

        }

    }

}

Caching is the mechanism of storing data , making subsequent call to the data for faster retrieval. Improves performance of the application.

It is best when the change is frequent in the data store.

In-Memory caching :- the consumer of the cache has a control over Absolute and SlidingExpriation

CacheEntry :- Priority :- Gets or sets the priority for the cache entry during cleanup process.