# Setup

Visual Studio 2010

AppFabric Caching installed and configured

ZoomIT

PowerShell as administrator

# Demo 1 - .NET 4 Caching

New console application

Save solution

Make sure this is NOT in the client profile

Add following to Program.cs

1. class Users
2. {
3. public IList<string> GetUsers()
4. {
5. Console.WriteLine("Start");
6. IList<string> users = new List<string>();
7. users.Add("Robert");
8. users.Add("Rein");
9. users.Add("Fred");
10. users.Add("Wilma");
11. Console.WriteLine("Users added");
12. Thread.Sleep(2500);
13. Console.WriteLine("Woken up");
14. return users;
15. }
16. }

Add following to main method

1. Users users = new Users();
2. Console.WriteLine(users.GetUsers()[0]);
3. Console.WriteLine(users.GetUsers()[1]);
4. Console.WriteLine(users.GetUsers()[3]);

Demo and show that each call is slow because it has to happen 3 times!

Add reference to System.Runtime.Caching

Change GetUsers to:

1. Console.WriteLine("Start");
2. ObjectCache cache = MemoryCache.Default;
3. IList<string> users;
4. if (cache.Contains("users"))
5. {
6. users = (IList<string>)cache.Get("users");
7. }
8. else
9. {
10. users = new List<string>();
11. users.Add("Robert");
12. users.Add("Rein");
13. users.Add("Fred");
14. users.Add("Wilma");
15. Console.WriteLine("Users added");
16. Thread.Sleep(2500);
17. Console.WriteLine("Woken up");
18. cache.Add("users", users, new CacheItemPolicy());
19. }
20. return users;

Demo and show it is fast and that users added and woken up only occurs once!

# Demo 2 – AppFabric Hello World

Fire up PowerShell as Administrator and type:

1. Import-Module DistributedCacheAdministration
2. Use-CacheCluster
3. Grant-CacheAllowedClientAccount < Do not execute, just talk about it!
4. Get-CacheHost
5. Start-CacheCluster
6. Get-CacheClusterHealth
7. Get-Cache
8. Get-CacheStatistics default

Switch back to the VS project from earlier and add references found in %windir%\SystNative\AppFabric (32bit it is %windir%\System32\AppFabric)

* Microsoft.ApplicationServer.Caching.Client.dll
* Microsoft.ApplicationServer.Caching.Core.dll

Add following class

1. class CacheManager
2. {
3. private static DataCacheFactory factory = null;
4. private static DataCache cache = null;
5. public static DataCache GetCache()
6. {
7. if (cache != null)
8. {
9. return cache;
10. }
11. IList<DataCacheServerEndpoint> servers = new List<DataCacheServerEndpoint>();

 servers.Add(new DataCacheServerEndpoint("work",22233));

1. DataCacheFactoryConfiguration config = new DataCacheFactoryConfiguration();
2. config.Servers = servers;
3. config.LocalCacheProperties = new DataCacheLocalCacheProperties();
4. DataCacheClientLogManager.ChangeLogLevel(System.Diagnostics.TraceLevel.Off);
5. factory = new DataCacheFactory(config);
6. cache = factory.GetCache("default");
7. return cache;
8. }
9. }

Change GetUsers to be:

1. public IList<string> GetUsers()
2. {
3. Console.WriteLine("Start");
4. //ObjectCache cache = MemoryCache.Default;
5. DataCache cache = CacheManager.GetCache();
6. IList<string> users = (IList<string>)cache.Get("users");
7. if (users ==null)
8. {
9. users = new List<string>();
10. users.Add("Robert");
11. users.Add("Rein");
12. users.Add("Fred");
13. users.Add("Wilma");
14. Console.WriteLine("Users added");
15. Thread.Sleep(2500);
16. Console.WriteLine("Woken up");
17. cache.Add("users", users);
18. }
19. return users;
20. }

Run it once and show it works like before, now switch back to powershell and run

1. Get-CacheStatistics default

Show the cache has some stuff in it.

Run app again and show it gets it from memory – no adding this run :D

Jump back to code and remove the like where we check if users are null – run and show it fails on add.

Change add to put and run again:

1. cache.Put("users", users);

Add a constructor to Users:

1. public Users()
2. {
3. DataCache cache = CacheManager.GetCache();
4. cache.CreateRegion("Users");
5. cache.Put("Robert", "Robert", new List<DataCacheTag>() { new DataCacheTag("Male"), new DataCacheTag("ZA") }, "Users");
6. cache.Put("Rein", "Rein", new List<DataCacheTag>() { new DataCacheTag("Male"), new DataCacheTag("USA") }, "Users");
7. cache.Put("Fred", "Fred", new List<DataCacheTag>() { new DataCacheTag("Male"), new DataCacheTag("ZA") }, "Users");
8. cache.Put("Wilma", "Wilma", new List<DataCacheTag>() { new DataCacheTag("Female"), new DataCacheTag("ZA") }, "Users");
9. }

Now change GetUsers to

1. public IEnumerable<string> GetUsers()
2. {
3. DataCache cache = CacheManager.GetCache();
4. foreach (var user in cache.GetObjectsInRegion("Users"))
5. {
6. yield return (string)user.Value;
7. }
8. }

Finally change Main to

1. foreach (string user in users.GetUsers())
2. {
3. Console.WriteLine(user);
4. };

Demo – show we can partition our cache

Now change the foreach to be:

1. foreach (var user in cache.GetObjectsByAllTags(new List<DataCacheTag>() { new DataCacheTag("Male"), new DataCacheTag("ZA") }, "Users"))