Remote Cache Application

This is an exercise of building a simple caching framework. Which comprises of the following components.

1. Cache Server
2. Cache Client
3. Test Application

Assignment is divided into three different tasks. All tasks need to be completed within the given time frame. Implement the tasks in sequence i.e. starting from task1 then task2 and so on.

Task 1

Cache Server

1. Cache Server is an application which acts as socket server (developed as a console based application) and hosts an object that implements the ICache interface.
2. This server should be able to receive requests from multiple clients at a time and respond to them.
3. The port to be used by your server should be read from app.config of the application.

Cache Client

1. Cache Client is a library (DLL) that has a public API (methods) which can be used by any application for adding and removing items from cache.
2. Cache client will connect to cache server via TCP channel.
3. The port to be used by your client should be read from app.config of the test application.

Test Application

1. Test Application will be a console application
2. It will give user a menu for all cache operations such as (adding, removing and fetching) items from cache
3. It will gave user option for register event on cache server.
4. It will invoke the methods on the client library accordingly.
5. It can be configured to run multiple threads requesting cache operations.

ICache interface is as follows:

```
interface ICache
{
    /// <summary>
    /// Initializes the cache instance
    /// </summary>
    public void Initialize();
    /// <summary>
    /// Adds the item to the cache.
    /// </summary>
    public void Add(string key, object value);
    /// <summary>
```

```
        /// Removes the item from the cache.
        /// </summary>
        public void Remove(string key);
        /// <summary>
        /// Retrieves the item from the cache.
        /// </summary>
        public object Get(string key);
        /// <summary>
        /// Clears the contents of the cache.
        /// </summary>
        public void Clear();
        /// <summary>
        /// Destroys the cache.
        /// </summary>
        public void Dispose();
    }
```

Task 2:

Client server application should have the following requirements:

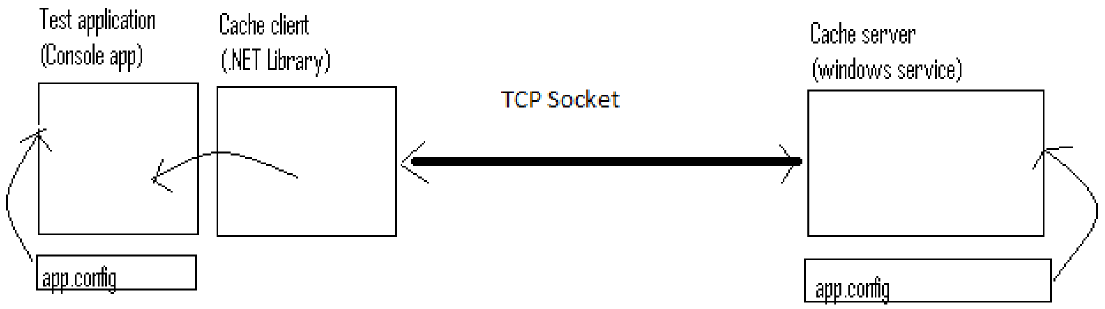
1. Client registers an event with server for any change in data. Means, client should receive an event whenever (add, remove and clear operations) are performed on cache server.
2. Only interested clients will receive events from server. Means that clients who register event with cache server will receive events.
3. Application should have proper exceptional handling.
4. Application should not break/crash under any circumstance.
5. All code needs to be thead-safe. As expecting multiple clients to be communicating with the server and performing different operations.

Task 3:

1. Deploy the chat server console application as a window service.
2. The following requirements are to be added at this stage:
    a. Add support to evict the items from cache when count reaches a max size specified in the service config file.
    b. Eviction should be based on LFU (Least Frequently Used) algorithm where items are evicted from the cache.
    c. Eviction should be performed in a background process which should run periodically after an interval specified in service config file.

Hint

The class that implements this interface can internally use a collection for storing the items of cache.



For help and guidance feel free to contact any NCache Team member.