Provides efficient storage for cached items.

```
//Microsoft Public License (Ms-PL)
//http://visualizer.codeplex.com/license
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Redwerb.BizArk.Core.Util
{
///

/// Provides efficient storage for cached items.
///
public class Cache
{

#region Initialization and Destruction

///

/// Creates a new instance of Cache.
///
public Cache()
{
// The default expiration time is 5 minutes.
DefaultExpiration = new TimeSpan(0, 5, 0);
}

#endregion

#region Fields and Properties

private Dictionary mItems = new Dictionary();

///
```

```
/// Gets or sets a value in the cache. Can be set even if the item hasn't been set before.
///
///  /// The object that was cached or null if it hasn't been cached yet.
public object this[string key]
{
get { return GetValue(key); }
set { SetValue(key, value, DefaultExpiration); }
}

///

/// Gets or sets the default expiration date.
///
public TimeSpan DefaultExpiration { get; set; }

#endregion

#region Methods

///

/// Puts a value into the cache.
///
///  ///  public void SetValue(string key, object value)
{
SetValue(key, value, DefaultExpiration);
}

///

/// Puts a value into the cache.
///
///  ///  ///  public void SetValue(string key, object value, TimeSpan expiration)
{
if (mItems.ContainsKey(key))
mItems.Remove(key);

// We don't store nulls.
```

```
if (value == null) return;

mItems.Add(key, new CacheItem(key, value,
DateTime.Now.Add(expiration)));
}

///

/// Removes an item from the cache.
///
///  public void ClearValue(string key)
{
SetValue(key, null);
}

///

/// Gets a value from the cache.
///
///  /// The value corresponding to the key. Null if the key
is not defined.
public object GetValue(string key)
{
if (mItems.ContainsKey(key))
{
var item = mItems[key];
if (item.HasExpired)
return null;
else
return item.Value;
}
else
return null;
}

///

/// Gets a value from the cache.
///
///
```

```
///  ///  /// The value corresponding to the key. defaultVal
if the key is not defined.
public T GetValue(string key, T defaultVal)
{
var value = GetValue(key);
if (value == null)
return defaultVal;
else
return (T)value;
}

///

/// Removes expired items from the cache.
///
public void PurgeCache()
{
var items = mItems.Values;
foreach (var item in items)
{
if (item.HasExpired)
mItems.Remove(item.Key);
}
}

///

/// Completely clears the cache.
///
public void ClearCache()
{
mItems.Clear();
}

#endregion

#region CacheItem

private class CacheItem
```

```
{
public CacheItem(string key, object value, DateTime
expirationDate)
{
Key = key;
ExpirationDate = expirationDate;
mValRef = new WeakReference(value);
}

private WeakReference mValRef;

public string Key { get; private set; }

///

/// Must call HasExpired before getting the value.
///
public object Value
{
get
{
if (mValRef.IsAlive)
return mValRef.Target;
else
return null;
}
}
public DateTime ExpirationDate { get; private set; }
public bool HasExpired
{
get
{
if (!mValRef.IsAlive)
return false;
else if (DateTime.Now < ExpirationDate) return false; else
return true; } } } #endregion } } [/csharp]
```