



## eCFSmarter Caching Keeping Data Fresh

### Overview

A great example of a double-edged sword in the eCF is its caching mechanism. ecF provides a mechanism in the web.config file to control how quickly the cache will update. A value can be set anywhere from 0 to 5 which will determine how quickly the cache will be updated.

The eCF's modular design requires that caching be implemented. Without the implementation of caching, smaller modules would needlessly perform multiple look-ups to the database when, *with caching*, it would only need one query then thereafter look into the cache. Using caching keeps the pages snappy but there is a time frame period of the cache that will not allow a recently updated page to become immediately available.

The problem is that items in cache will not be refreshed for x number of minutes depending on your eCF cache setting in the web.config. Another issue is that some data records are only cached for a few seconds.

With smart caching, items stay in cache for several hours or until the record is updated in the database. This gives you the best of both worlds, because not every commerce site has a web user on every page every few seconds, or even every few hours.

The eCF uses a MetaData engine to provide extended field values to category, product and sku objects. This is a shining example of the engineering of the eCF. However, if you have a highly developed custom Category Page using these MetaValues with 36 Products represented by 36 MetaClass records this can add up fast. Then, if you have just 3 Skus for each product, it would amount to 108 Sku records and 108 Sku MetaClass records.

Altogether this would cause 216+ queries to be sent to the database server for each page load. Also, depending on the number of MetaFields in the class it could be dramatically more. Once the page is cached there will be no queries sent to the database server but this may only last a few minutes. With active retail sales, Products and Skus change constantly.

Asp.Net supports dependency caching however it's designed with a File-watching mechanism looking for new DataSet xml files to be deployed. Then you can add a Delegate function to re-cache results. I wasn't interested in trying to persist the Mediachase API objects to disk. I just wanted the cache in this



# PROJECTTHUNDER.COM, INC.

HOW E-COMMERCE IS DONE

---

process to be dropped when I changed or deleted a record in the Commerce Manager.

The work-around we found incorporates an apparatus around SQL 2005 and Trigger running in the .Net CLR. The trigger running on our Product, Category, Sku and MetaData tables is creating new files with a 0-file size in a location where the ASP.Net process has linked dependences to. Causing ASP.Net to drop the cache and the eCF API will refresh the cache on the next page refresh.

There are two assumptions here. One is that you are using .Net 1.1 or 2.0 version of the eCF platform, and also that you are using SQL 2005 with Visual Studio 2005. What if you're not using SQL 2005? There are a few ways to skin a cat, someone told me once.

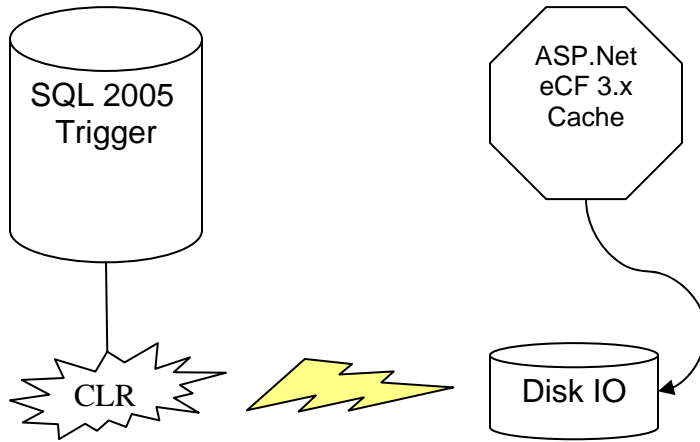
You can write a Com object and expose it to SQL to create these files. You can write an extended SP in C++. You can use the shell extended store procedure in SQL 2000. I like the 2005 approach because it runs the process in a sandbox, which gives us some protection if the process becomes exploited. It's also so easy to deploy a trigger in Visual Studio 2005 written in VB.Net or C#. I was almost teary-eyed.



# PROJECTTHUNDER.COM, INC.

HOW E-COMMERCE IS DONE

---





# PROJECTTHUNDER.COM, INC.

HOW E-COMMERCE IS DONE

We started with a modified Insert method in the StoreCache.cs file. It needed to look at what kind of object was being stored in the Cache. Depending on the object being stored, we can create the dependency on the naming convention that our SQL Trigger will also be following. You need to code a case statement for every object you want to cache.

```
public static void Insert(string key, object obj, CacheDependency dep, int seconds, CacheItemPriority
priority)
{
    if(!IsEnabled)
        return;

    if(dep==null)
    {
        string[] files;

        ArrayList list = new ArrayList();
        string path;
        //What kind of object are you do you come here often?
        switch(obj.GetType().ToString())
        {
            case "Mediachase.Store.Business.Product":

                Product pro = (Product)obj;

                path = SettingsManager.GetParamValue("SitePhysicalPath");

                if(path.EndsWith("\\")==false) path+="\\";
                //If there are more than one product in the cache then let's
                create a watch for each record

                foreach(Product lilpro in pro)
                {
                    list.Add("Product." + lilpro.ProductId);
                }

                files = (System.String[])list.ToArray(typeof(System.String));

                dep = new CacheDependency(files);

                break;

            case "Mediachase.Store.Business.ObjectMetaFields":

                ObjectMetaFields fields = (ObjectMetaFields)obj;
                ObjectMetaField field;
                field = fields.Fields[0];

                string stringfile = field.Owner.MetaClass.TableName + "." + field.Owner.Id.ToString();

                path = SettingsManager.GetParamValue("SiteCachePath");

                if(path.EndsWith("\\")==false) path+="\\";

                list.Add(path + stringfile);

                files = (System.String[])list.ToArray(typeof(System.String));
```



# PROJECTTHUNDER.COM, INC.

HOW E-COMMERCE IS DONE

---

```
        dep = new CacheDependency(files);

        break;

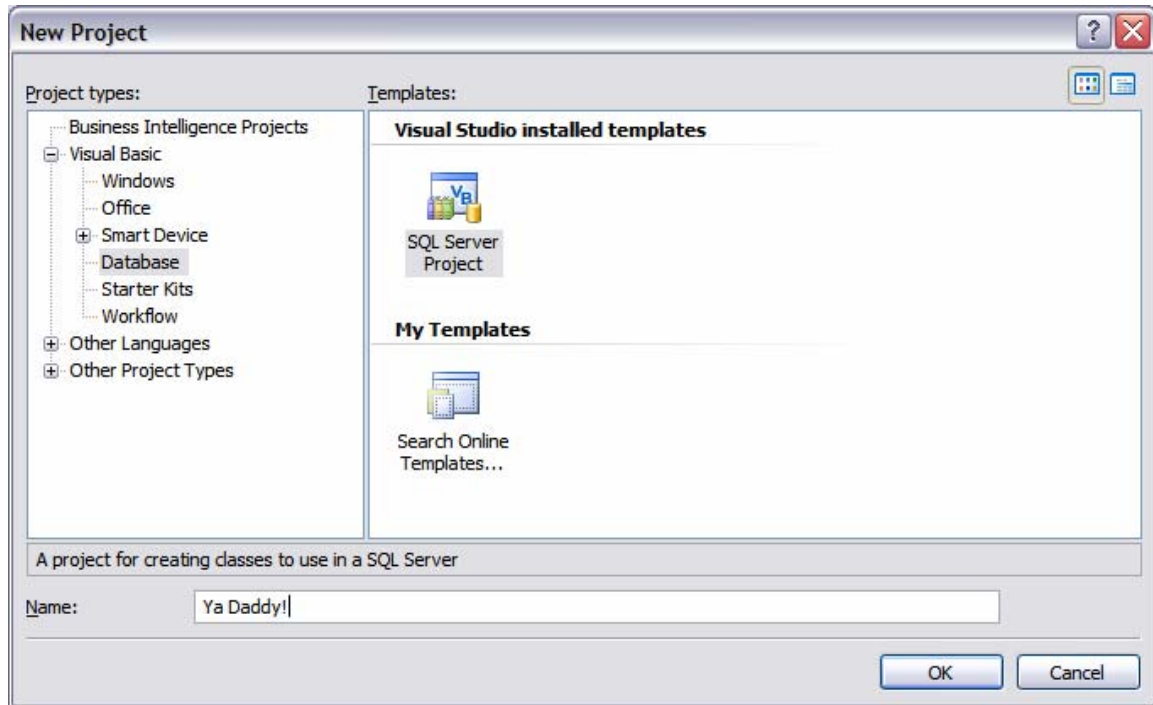
    }

}

if(obj != null)
{
_cache.Insert(key,obj,dep,DateTime.Now.AddSeconds(Factor * seconds), TimeSpan.Zero,priority,null);
}
}
```



Now onward and upward to the SQL Trigger. In Visual Studio 2005, create a new SQL Server Database Project.



Note: before you try this at home, make sure you have enabled CLR in your SQL 2005 database. You need to set the database compatibility to SQL 2005 if you restored an older version of your database from SQL 2000. You also have to allow the database to make external calls since our database now creates files on the disk. If you receive any errors while deploying, just search for the context of the error on Google Groups.

### Enables CLR

```
sp_configure 'show advanced options', 1;  
GO  
RECONFIGURE;  
GO  
sp_configure 'clr enabled', 1;  
GO  
RECONFIGURE;  
GO
```

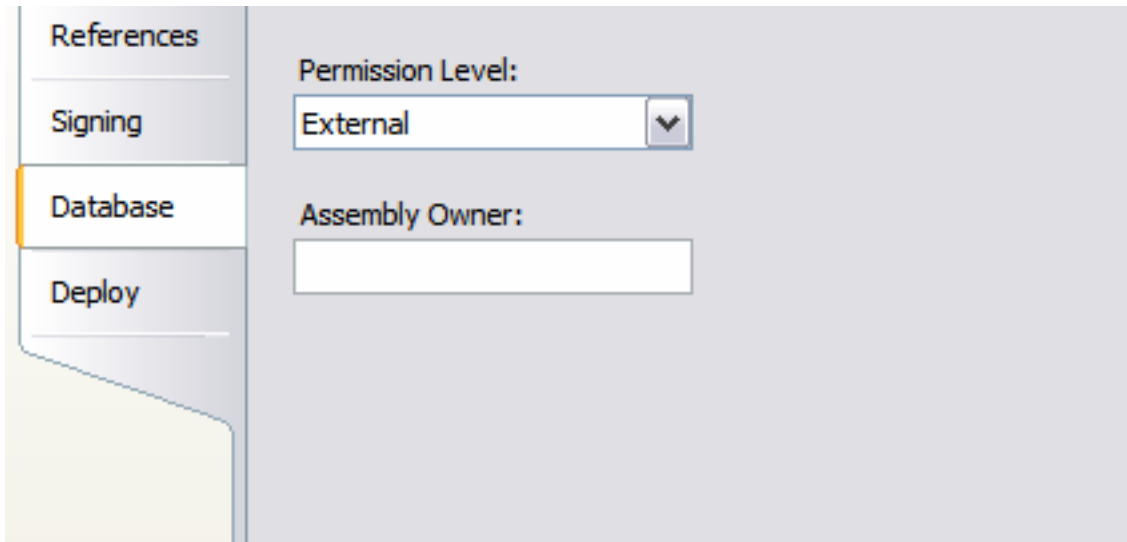


# PROJECTTHUNDER.COM, INC.

HOW E-COMMERCE IS DONE

---

The wizard will walk you through setting up the database connection. When that is completed, right -click on the project and select Properties. On the Database tab, bump up the permission level to External so the process can get to the Disk.





Now right-click on the Project, select Add, and then click on Trigger. Visual Studio will add a new class-file with a sample trigger. We have to set the Target for each table in the Beta of SQL 2005. We could not specify a Target, and it would fire for all tables. But alas, not any more, which means we need one of these bad-boy methods for every table we want to monitor changes on. The tables for MetaData are created dynamically and will have names like ProductEx\_Everything\_Apparel.

```
'This is our SQL Trigger
<Microsoft.SqlServer.Server.SqlTrigger(Name: ="Product", Target: ="Product", Event: ="FOR UPDATE,
INSERT, DELETE")> _
Public Shared Sub Product()
```

Try

```
Dim Context As SqlTriggerContext = SqlContext.TriggerContext()
Dim Connection As New SqlConnection("context connection=true")
Connection.Open()

Dim Settings As New DataTable

'Get Global Setting from eCF
Dim TableSettings As New SqlDataAdapter("select * from globalsetting where paramname
='SiteCachePath'", Connection)
TableSettings.Fill(Settings)

Dim cachepath As String = Settings.Rows(0).Item(2).ToString
If Not cachepath.EndsWith("\") Then cachepath += "\"

Dim DeletedCommand As SqlCommand = Connection.CreateCommand()
Dim InsertedCommand As SqlCommand = Connection.CreateCommand()

'If you try to select a ntext, image or text from deleted it will crash.
DeletedCommand.CommandText = "select ProductId from deleted"

Dim DeletedReader As SqlDataReader

Dim TableName As String = "Product"

DeletedReader = DeletedCommand.ExecuteReader

Select Case Context.TriggerAction

Case TriggerAction.Update

Dim cachekey As String = ""

While DeletedReader.Read
cachekey = TableName.ToUpper() + "." + DeletedReader(0).ToString()
CreateFile(cachepath + cachekey)

End While

DeletedReader.Close()
```



```
Case TriggerAction.Delete
  Dim cachekey As String = ""

  While DeletedReader.Read
    cachekey = TableName.ToUpper() + "." + DeletedReader(0).ToString()
    File.Delete(cachepath + cachekey)

  End While

End Select

Connection.Close()
Catch e As Exception

  'Whoops
  Throw

End Try

End Sub
```

## Conclusion

SQL and Visual 2005 open a lot of possibilities for developers to get in to SQL. One thing I noticed is that the connection available in the context of the trigger can only be used to do one thing at a time and the bug with text and image data from the deleted table still exists. This is still a useful tool for performing operations like this for caching, but we have also used it for creating a Stored Procedure to strip HTML from descriptions, Geo Code addresses to calculate distance.

Solving this issue for caching and how cache is stored allows us to bump up the cache setting in the web.config for the eCF and let our customers and content editors change content and products, allowing them to view their changes in real time instead of taking 5-minute breaks waiting for cache to drop. We get greater performance gain with no little or no pain waiting for the cache to clear.