Cache کردن داده ها در هنگام Startup برنامه

مقدمه:

در دو مقاله ی قبلی چگونگی cache کردن داده ها در لایه ی Presentation و Caching Layer را دیدید. در Cache کردن داده ها با استفاده از ObjectDataSource , ما چگونگی استفاده از امکانات cache کنترل ObjectDataSource در لایه ی Presentation را بررسی کردیم. و درمبحث قبل , به cache کردن داده ها در لایه ای جدید و جدا به نام Caching Layer پرداختیم. در هر دو مقاله ی قبل, از بارگذاری القایی(reactive loading) در کار با data cache استفاده کردیم. با استفاده از Reactive loading , هر بار که داده ها درخواست می شوند, سیستم از وجود داده ها در Cache اطمینان حاصل می کند. اگر داده ها در Cache موجود نباشند, آنها را از منبع سازماندهی داده ها (برای مثال یک دیتابیس) می گیرد و سپس این داده ها را Cache می کند. مزیت اصلی Reactive loading پیاده سازی بسیار آسان این روش است. یکی از معایب این روش کارایی متغیر و ناجور آن در بین درخواست ها است. فرض کنید صفحه ای از Caching Layer مبحث قبلی جهت نشان دادن اطلاعات Product استفاده می کند. هنگامی که این صفحه برای اولین بار رویت می شود یا هنگامی که برای اولین بار بعد از خارج شدن داده های Cache شده بعد از زمان Expire رویت می شود, داده ها باید از دیتابیس بازیابی شود. بنابراین, این درخواست کاربر زمان بیشتری نسبت به درخواست ذخیره ی اطلاعات دارد.

بارگذاری پیش گستر(Proactive loading) یک استراتژی برای مدیریت تعاملی Cache را فراهم می کند. با این استراتژی, کارایی همه ی درخواستها توسط بارگذاری داده های cache شده قبل از این که به آنها نیازی باشد, یکسان می شود. به طور معمول , Proactive loading از یکسری فرایند ها که به طور دوره ای زمان به روزرسانی داده ها در لایه ی زیرین را چک می کنند یا خبر می دهند, استفاده می کند. سپس این فرایند, داده ها ی Cache را به منظور تازگی و جلوگیری از ماندگاری داده ها، بروز رسانی می کند. روش Proactive loading درصورتی که داده های لایه زیرین ازیک اتصال بسیار کند با دیتابیس, یا یک Web servise یا سایر data source های کند فراهم شوند, مفید و کارا است. اما این روش در پیاده سازی بسیار دشوار است زیرا نیازمند ایجاد, مدیریت و گسترش و آرایش یک فرایند به منظور چک کردن تغییرات و به روز رسانی cache است .

مد دیگر Proactive loadig , که ما قصد توضیح و بررسی آن را داریم, بارگذاری داده ها درCache هنگام Startup یک Application است. این برداشت برای cache کردن داده های استاتیک (مانند رکوردهای جداول مراجعه(lookup table) یک دیتابیس) بسیار مفید است.

مرحله ی 1: تعیین داده ها به منظور Cache در Application Startup

مثالهای caching ی که در دو مبحث قبلی به توضیح وبررسی آنها پرداختیم , از reactive loading استفاده می کردند و داده ها را به طور دوره ای تغییر می دادند و زمان گزافی صرف ایجاد، نمی شد. اما اگر داده های cache شده هرگز تغییر نمی کردند, زمان expire ی که توسط reactive loading استفاده شده بود, بیهوده است. بعلاوه, اگر زمان زیادی صرف تولید داده های cache شده شود(cache خالی باشد), کاربر درخواست کننده ی داده ها باید زمان زیادی را جهت بازیابی داده ها تحمل کند. بنابر این داده های استاتیک و داده هایی که زمان زیادی صرف ایجادشان می شود را در هنکام startup برنامه cache کنید.

هنگامی که دیتابیس مقدارهای پویا و تغییر پذیر زیادی دارد , ممکن است مقدار زیادی داده های استاتیک نیز داشته باشد.از این جمله می توان به همه ی مدلهای داده یک یا چند ستون که شامل

مقدار مشخصی از یک مجموعه ی انتخابات هستند, اشاره کرد. برای مثال در یک دیتابیس , جدول Patients ستونی به نام PrimaryLanguage که مجموعه ای از مقادیر English, Spanish, French, Russian, Japanese و... است را داراست. بعضی اوقات, این نوع ستون ها با استفاده از جداول مراجعه پیاده سازی می شوند. این به نسبت بهتر از ذخیره سازی رشته ای "English" یا "French" و... در جدول Patients است. جدول دوم معمولا با دو ستون (یک ستون به منظور نگهداری شناسه ی منحصر و یک ستون جهت تعریف رشته ای با یک رکورد برای مقدارهای ممکن ومجاز) ایجاد می شود. ستون PrimaryLanguage در جدولPatients , شناسه ی منحصر و مخصوصی متناظر با جدول مراجعه را نگهداری می کند. در شکل زیر , زبان اصلی John Doe's انگلیسی است حال آنکه زبان اصلی Ed Johnson روسی است.

Patients

Column	Type	Comments
PatientID	int	IDENTITY, Primary Key
Name	nvarchar(50)	
PrimaryLanguageID	int	Foreign Key

PatientID	Name	PrimaryLanguageID
1	John Doe	1
2	Jane Doe	1
3	Sam Smith	3
4	Sally Smith	1
5	Ed Johnson	4

Languages

Column	Type	Comments
LanguageID	int	IDENTITY, Primary Key
Description	nvarchar(50)	

LanguageID	Description
1	English
2	Spanish
3	French
4	Russian

شکل 1:جدول Languages ,جدول مراجعه ای است که توسط جدول Patients استفاده شده است.

رابط کاربری که به منظور تغییر یا ایجاد یک بیمار (patient) استفاده می شود, drop-down list ی از زبان های رکوردهای جدول Languages است. بدون عمل Cache , هر بار این interface رویت شود, سیستم باید درجدول Languags جستجویی انجام دهد.از آنجا که مقدارهای جدول مراجعه به ندرت تغییر می کنند, این کاری اضافی و غیر ضروری است.
اگرچه reactive loading از منقضی شدن بر مبنای زمان استفاده می کند و برای داده های استاتیک جدول مراجعه نیازی به منقضی شدن ندارد, اما می توانیم جهت cache کردن داده های Languages از reactive loading ی شبیه آنچه در مبحث قبل بکارگرفته شد, استفاده کنیم.
دراین مبحث ما به بررسی چگونگی cache کردن داده های جدول مراجعه و دیگر اطلاعات استاتیک می پردازیم.

مرحله ی 2: آزمودن روشهای مختلف cache کردن داده ها

در application های asp.net , می توانید اطلاعات را از طریق برنامه نویسی cache کنید. اشیا (object) را نیز می توانید با استفاده از static members یا application state , cache کنید.
وقتی با یک کلاس کار می کنید, برای این که اعضای کلاس در دسترس باشند باید یک نمونه از آن کلاس را ایجاد کنید. برای مثال, جهت درخواست یک تابع از کلاسی درBusiness Logic Layer ابتدا باید نمونه ای از آن کلاس را ایجاد کنیم:

```
ProductsBLL productsAPI = new ProductsBLL();
productsAPI.SomeMethod();
productsAPI.SomeProperty = "Hello, World!";
```

قبل از اینکه بتوانیم با متد یا خصوصیت خاصی از کلاسی کار کنیم , باید نمونه ای از آن کلاس را با استفاده از کلمه ی new , ایجادکنیم. متدها و خصوصیت ها با یک نمونه ی خاص پیوند خورده اند.

Lifetime این اعضا به lifetime شیی که به آن متصل هستند گره خورده است.از طرف دیگر, اعضای استاتیک, خصوصیات و متدهایی متغیری هستند که در میان همه ی نمونه های یک کلاس مشترک هستند, ودر نتیجه lifetime آنها به بلندی lifetime یک کلاس است. اعضای استاتیک توسط کلمه ی کلیدی static مشخص می شوند.

علاوه بر اعضای استاتیک, داده ها نیز می توانند با استفاده ازapplication state , cache شوند. هر application در ASP.NET مجموعه ای از نام/مقدار، که دربین همه ی کاربران و صفحات application مشترکند, را نگهداری می کنند. این مجموعه با استفاده از خصوصیت Application کلاس HttpContext قابل دسترسی است. استفاده از آن در code-behind به صورت زیر است:

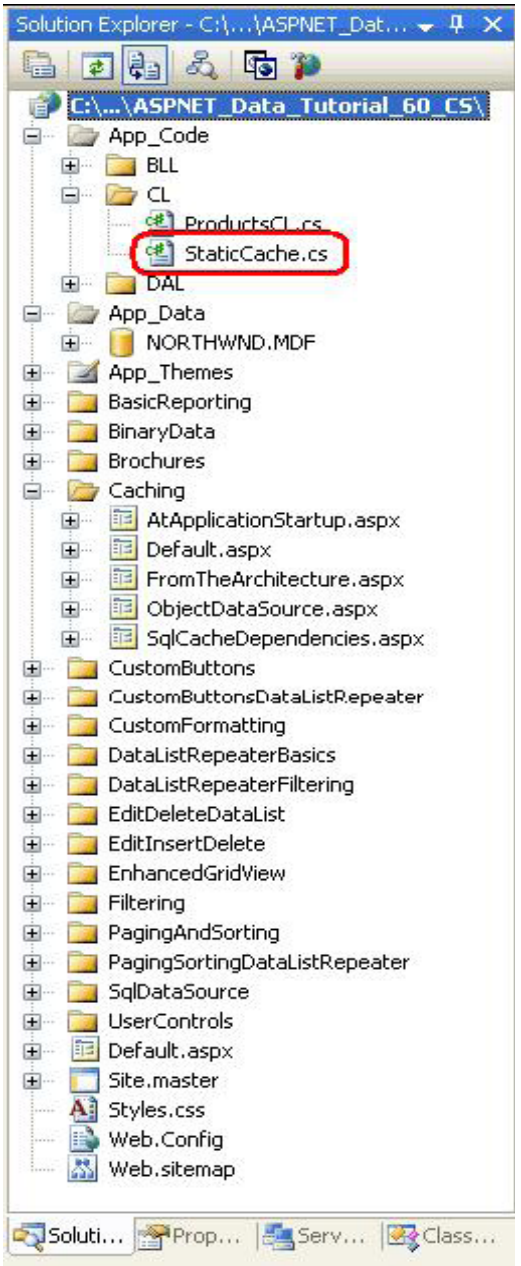
```
Application["key"] = value;
object value = Application["key"];
```

Data cache , API بسیار غنی برای cache کردن داده ها فراهم می کند. همچنین مکانیزمی برای منقضی شدن بر مبنای زمان یا بر مبنای وابستگی وتعلق, فراهم می کند. و اولویت آیتمهای cache را تعیین می کند. با وجود اعضای استاتیک و application state , بعضی ویژگی ها باید توسط برنامه نویس به طور دستی افزوده شود. در این مبحث ما کد مربوط به هر سه تکنیک Cache کردن داده های استاتیک را مورد بررسی قرار می دهیم.

مرحله ی 3: Cache کردن داده های جدول Suppliers

جداول دیتابیس Northwind از هیچگونه جداول مراجعه ای استفاده نکرده اند. 4 جدولی که در DAL ایجاد کردیم, فاقد داده با مقدار استاتیک بودند.به جای صرف وقت به منظور افزودن table ی جدید به DAL و سپس افزودن کلاسی جدید در BLL , اجازه دهید طوری وانمود کنیم که داده های جدول Suppliers استاتیک هستند. بنابراین, میتوانیم داده های این جدول را در Application startup , cache کنیم.

کار را با ایجاد کلاسی جدید به نام staticCache.cs در پوشه ی cl شروع کنید.



شکل2:ایجاد کلاس StaticCache.cs در پوشه ی cl

باید متدی جهت بارگذاری(load) داده ها در startup که مقدار برگشتی آن داده ها ی cache است, اضافه کنیم.

```
[System.ComponentModel.DataObject]
public class StaticCache
{
private static Northwind.SuppliersDataTable suppliers = null;
public static void LoadStaticCache()
{
// Get suppliers cache
using a static member variable
SuppliersBLL suppliersBLL = new SuppliersBLL();
suppliers = suppliersBLL.GetSuppliers();
}
[DataObjectMethodAttribute(DataObjectMethodType.Select, true)]
public static Northwind.SuppliersDataTable GetSuppliers()
{
return suppliers;
}
}
```

قطعه کد بالا از یک عضو متغیر استاتیک به نام supplier استفاده می کند. این عضو نتیجه ی برگشتی متد ()GetSuppliers کلاس SuppliersBLL که توسط متد ()LoadStaticCache فراخوانی می شود, را نگهداری می کند. متد ()LoadStaticCashe در زمان ابتدای application فراخوانی می شود. این داده ها در آغاز application یکبار بارگذاری(load) می شوند, و هر صفحه ای که نیازمند کار با داده های supplier است, می تواند متد ()GetSuppliers کلاس StaticCache را فراخواند. بنابراین, فراخوانی دیتابیس جهت گرفتن اطلاعات supplier ها تنها یکبار و آنهم در زمان آغاز application صورت می گیرد.
ما می توانیم متناوبا از application state یا data cache استفاده کنیم. قطعه کد زیر کلاسی با امکان استفاده از application state را نشان می دهد:

```
[System.ComponentModel.DataObject]
public class StaticCache
{
```

```
public static void LoadStaticCache()
{
// Get suppliers cache
using application state
SuppliersBLL suppliersBLL = new SuppliersBLL();
HttpContext.Current.Application["key"] = suppliersBLL.GetSuppliers();
}
[DataObjectMethodAttribute(DataObjectMethodType.Select, true)]
public static Northwind.SuppliersDataTable GetSuppliers()
{
return HttpContext.Current.Application["key"] as Northwind.SuppliersDataTable;
}
}
```

در متد ()loadStateCache , اطلاعات supplier در متغیری با سطح دسترسی application ذخیره می شود. مقدار برگشتی آن متناسب با نوع متد ()GEtSuppliers است. در صورتی application state در دسترس است که درکلاسهای Code-behind صفحات Asp.Net از [“key”]Application استفاده شود و باید در معماری از [“key”]HttpContext.Current.Application جهت گرفتن HttpContext جاری استفاده کرد.

بعلاوه data cache میتواند همچون یک انبار cache استفاده شود. اینچنین که در کد زیر می بینید:

```
[System.ComponentModel.DataObject]
public class StaticCache
{
public static void LoadStaticCache()
{
// Get suppliers cache
using the data cache
SuppliersBLL suppliersBLL = new SuppliersBLL();
HttpRuntime.Cache.Insert(
/* key */ "key",
/* value */ suppliers,
/* dependencies */ null,
/* absoluteExpiration */ Cache.NoAbsoluteExpiration,
/* slidingExpiration */ Cache.NoSlidingExpiration,
/* priority */ CacheItemPriority.NotRemovable,
/* onRemoveCallback */ null);
```

```
}
[DataObjectMethodAttribute(DataObjectMethodType.Select, true)]
public static Northwind.SuppliersDataTable GetSuppliers()
{
return HttpRuntime.Cache["key"] as Northwind.SuppliersDataTable;
}
}
```

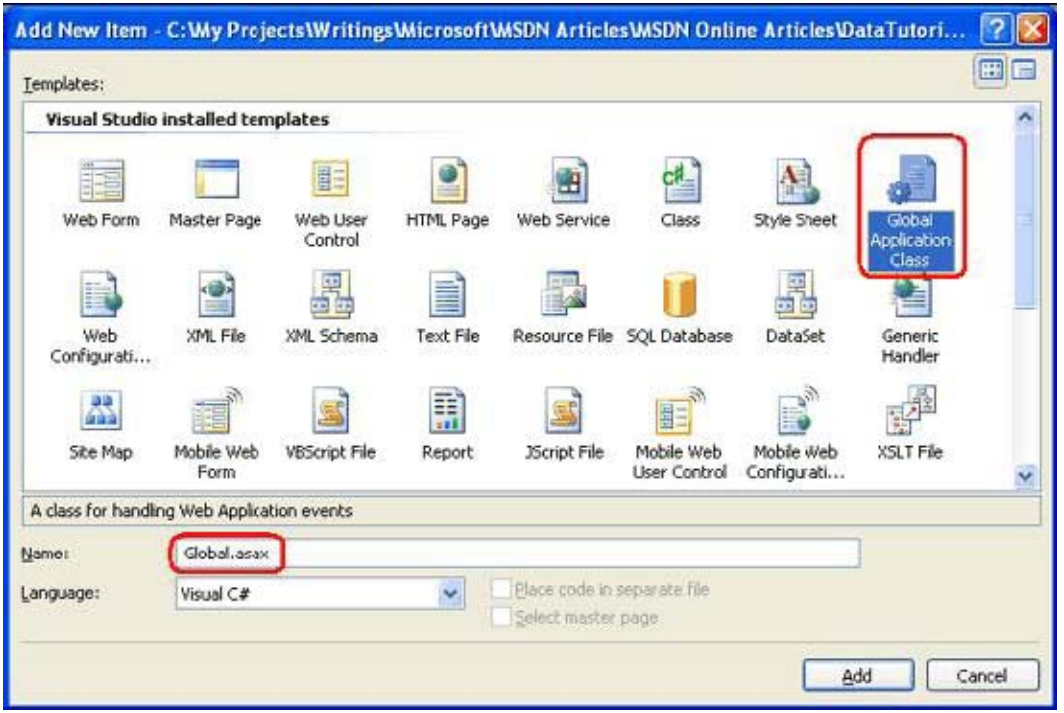
جهت افزودن یک آیتم به data cache بدون expire مبتنی بر زمان از مقدارهای System.Web.Cache.NoAbsoluteExpiration و System.Web.Caching.Cache.NoSlidingExpiration به عنوان پارامترهای ورودی استفاده کنید. به علت این که بتوانیم اولویت آیتم های cache را تعیین کنیم, این overload خاص متد Insert متعلق به data cache را انتخاب کردیم. این اولویت بندی به منظور تعیین آیتمی است که در زمان کمبود حافظه در دسترس باید از cache خارج شود. در اینجا ما از اولویت NotRemovable استفاده کرده ایم. این اولویت ما را از این که داده ها ازcache خارج نمی شوند, مطمئن می کند.

مرحله ی 4: اجرای کد در زمان startup یک Application

جهت اجرای کد در زمانی که Application برای اولین بار شروع می شود, لازم است فایل مخصوصی با نام Global.asax ایجاد شود.این فایل می تواند شامل event handler هایی برای application , session و رویدادهای سطح درخواست(request-level) باشد. در این فایل می توانیم کدهایی را بیفزاییم که هر زمان application شروع می شود, اجرا شوند.

یک فایل Global.asax را به دایرکتوری ریشه ی Application تان اضافه کنید. جهت انجام این کار, روی اسم وب سایتتان در Solution Explorer راست کلیک کنید و گزینه ی Add New Item را انتخاب کنید.در diolog box ظاهر شده, آیتم Global Application Class را انتخاب کنید و دکمه ی Add را کلیک کنید.

توجه: اگر فایلGlobal.asax را از قبل در پروژه ای که ایجاد کرده اید, داشته باشید, آیتم Global Application Class در لیست این Add New Item)dialog box)نخواهد بود.



شکل 3: افزودن فایلGlobal.asax به دایرکتوری اصلی Web Application

الگوی فایل Global.asax به طور پیش فرض شامل 5 متد با یک تگ script سمت سرور(server-side) است :

- Application_Start –هنگامی کهweb application برای اولین بار آغاز می شود, اجرا می شود.
- Application-end –زمانی که application در حال بسته شدن است, اجرا می شود.
- Application_Error –هر زمان که یک exception کنترل نشده(unhandled) از Application برسد, اجرا می شود.
- Start_Session –هنگامی که یک session جدید ایجاد شود, اجرا می شود.

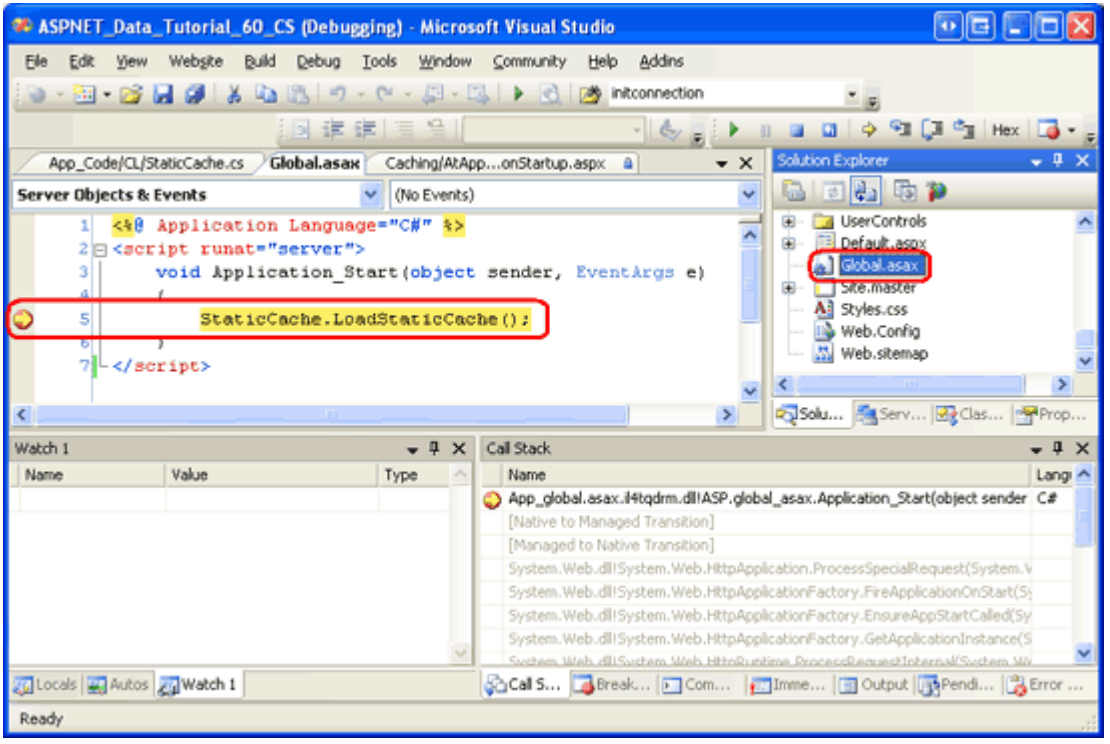
- Session_End –زمانی که یک session منقضی یا فاسد شود , اجرا می شود.

Application_Start تنها یکبار در طول حیات application فراخوانی می شود.و اجرای آن تا زمانی که Application دوباره شروع نشده است(restart), ادامه دارد. این متد می تواند به دلیل ویرایش محتویات پوشه ی bin/ , ویرایشGlobal.asax , ویرایش محتویات پوشه یApp_Code , یا ویرایش فایل Web.Config , یا دلایل دیگر اتفاق افتد.

در این مبحث , ما تنها به متد Application_Start , کدها و دستوراتی را می افزاییم, بنابراین می توانید بقیه ی قسمتها را حذف کنید. در Application_Start , متد ()LoadStaticCache کلاس StaticCache را که اطلاعات تولید کننده(Supplier) را لود کرده و آنها را cache می کند , را فرا می خوانیم.

```
<%@ Application Language="C#" %> <script runat="server"> void
Application_Start(object sender, EventArgs e) { StaticCache.LoadStaticCache(); }
</script>
```

در startup این application , متد ()LoadStaticCache اطلاعات تولید کننده را از BLL می گیرد, و آن را در یک متغیر استاتیک ذخیره می کند. برای بررسی این عملکرد , یک breakpoint درمتد Application_Start قرار دهید و آن را اجرا کنید. توجه کنید که breakpoint به محض اینکه application آغاز می شود, فعال می شود(روشن می شود). درخواستهای بعدی سبب اجرای متد Application_Start نمی شوند.



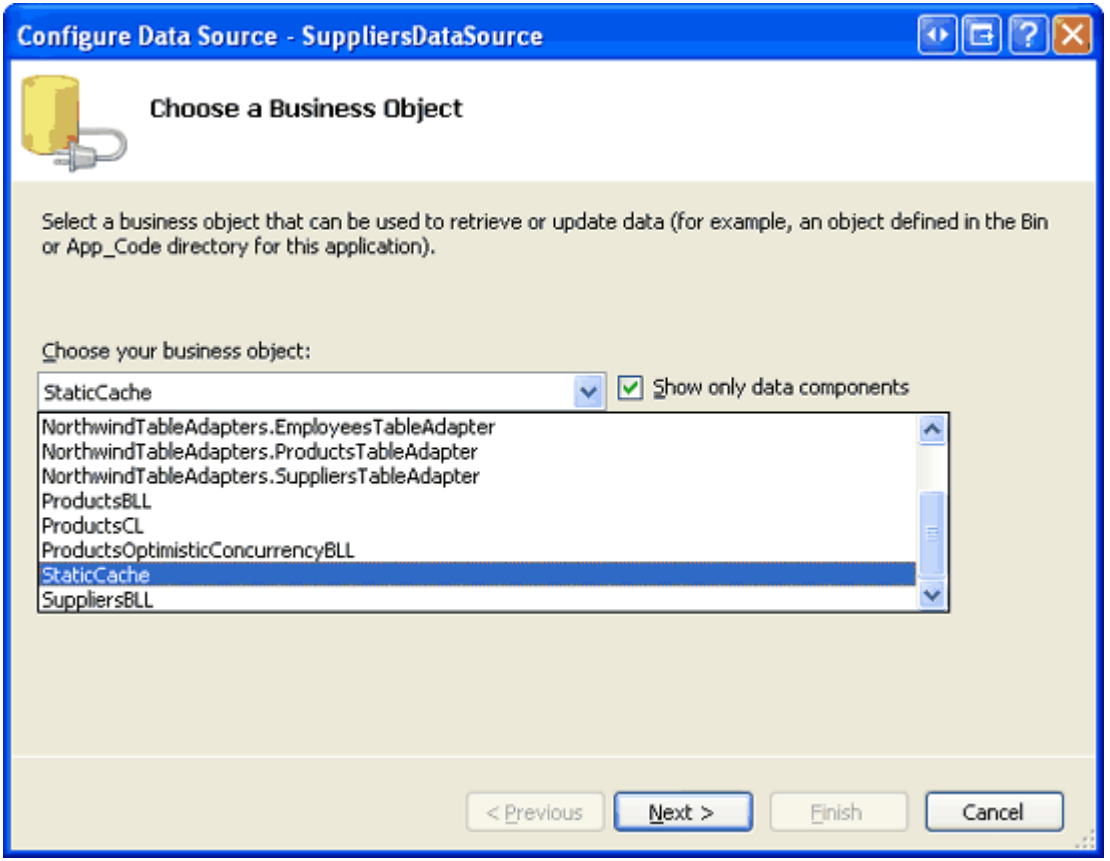
شکل4:استفاده از breakpoint جهت بررسی عملکردevent handler (کنترلگر رویداد) Applicaqtion_Startی در حال اجرا

توجه: اگر breakpoint مربوط به Application_Start در زمان شروع اولین debug موفق نبود, به این دلیل است که application تان قبلا شروع شده است. Application تان را با ویرایش Global.asax یا فایلهایWeb.CXonfig مجبور به restart کنید و دوباره سعی کنید. می توان با افزودن(یا حذف) یک خط خالی در آخر هر کدام ازاین فایلها , applicationتان را خیلی ساده و سریع restart کنید.

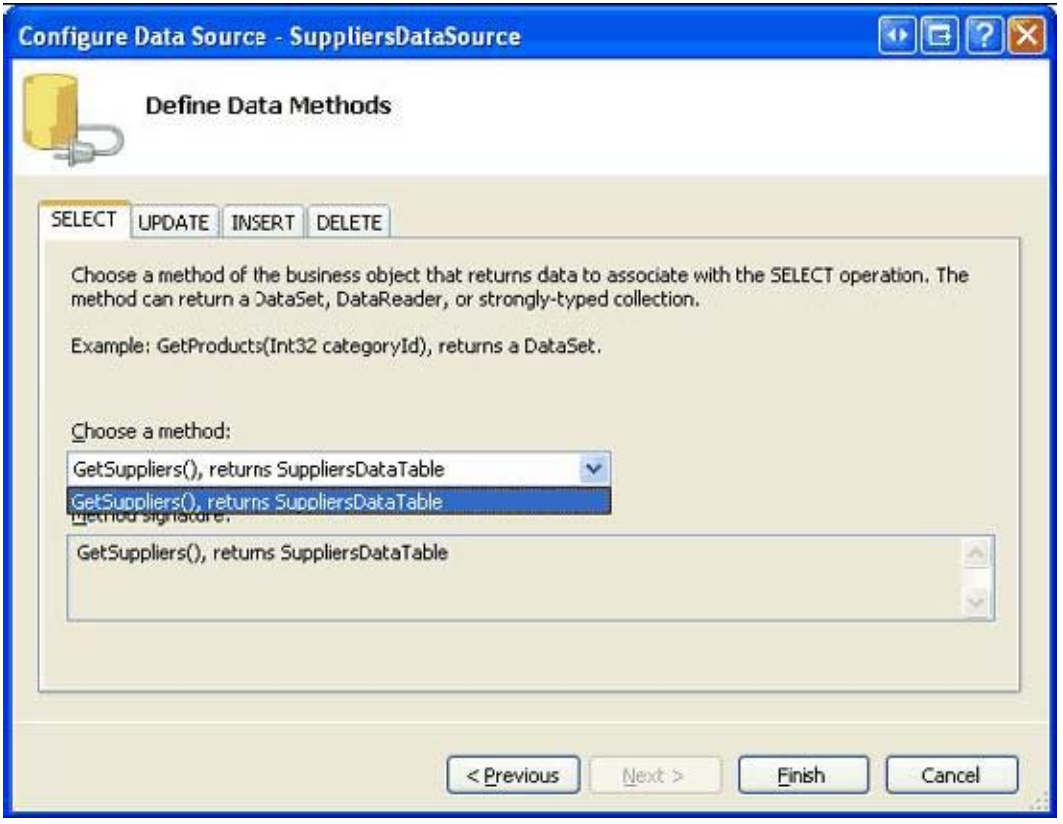
مرحله ی 5:نمایش داده های cache شده

در آغاز application , کلاس StaticCache محتوی یک نسخه cache شده از داده های supplier است که میتواند در متد ()GetSuppliers در دسترس باشد. جهت کار با این داده ها در لایه ی presentstion , می توانیم از یک ObjectDataSource استفاده کنیم یا از طریق برنامه نویسی متد ()GetSupplier کلاس StaticCache را از code-behind صفحه درخواست کنیم. بیاید چگونگی بکارگیری کنترلهای ObjectDataSource و GridView جهت نمایش دادن اطلاعات supplier (تولیدکننده ی) cache شده را با هم بررسی کنیم.

کار را با بازکردن صفحه ی AtApplicationStartup.aspx در پوشه یCaching , شروع کنید. یک GridView از ToolBox به designer اضافه کنید, خصوصیت ID آن را Suppliers کنید. سپس در تگ smart متعلق به GridView , یک ObjectDataSource جدید با نام SuppliersCacheDataSource ایجاد کنید. ObjectDataSource را به گونه ای پیکربندی کنید که از متد ()GetSuppliers کلاس StaticCache استفاده کند.



شکل5:پیکربندی ObjectDataSource جهت کار با کلاس StaticCache



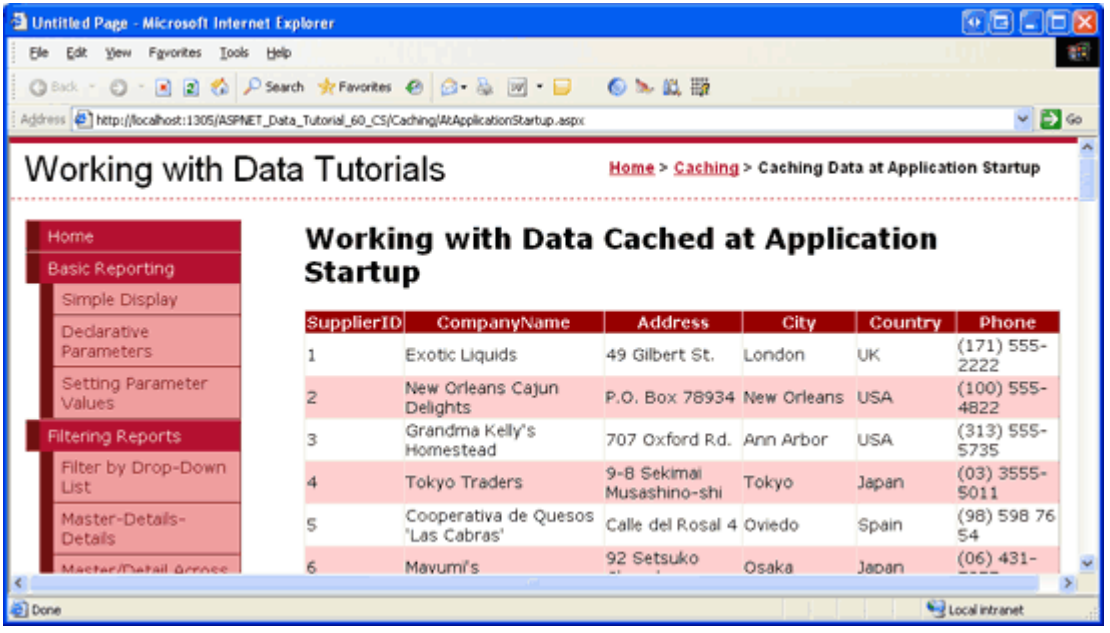
شکل6:استفاده از متد ()GetSuppliers جهت بازیابی دادهای cache شده ی supplier

پس از اتمام این ویزارد, VS به طور خودکار برای هر فیلد داده در SuppliersDataTable یک Boundfield اضافه می کند .markup اعلانی Gridview و ObjectDataSource باید به صورت زیر باشد:

```
<asp:GridView ID="Suppliers" runat="server" AutoGenerateColumns="False"
DataKeyNames="SupplierID" DataSourceID="SuppliersCachedDataSource"
EnableViewState="False"> <Columns> <asp:BoundField DataField="SupplierID"
HeaderText="SupplierID" InsertVisible="False" ReadOnly="True"
SortExpression="SupplierID" /> <asp:BoundField DataField="CompanyName"
HeaderText="CompanyName" SortExpression="CompanyName" /> <asp:BoundField
DataField="Address" HeaderText="Address" SortExpression="Address" />
```

```
<asp:BoundField DataField="City" HeaderText="City" SortExpression="City" />
<asp:BoundField DataField="Country" HeaderText="Country" SortExpression="Country"
/> <asp:BoundField DataField="Phone" HeaderText="Phone" SortExpression="Phone" />
</Columns> </asp:GridView> <asp:ObjectDataSource ID="SuppliersCachedDataSource"
runat="server" OldValuesParameterFormatString="original_{0}"
SelectMethod="GetSuppliers" TypeName="StaticCache" />
```

شکل 7 صفحه را هنگامی که در browser دیده می شود, نشان می دهد. خروجی باید مشابه داده های گرفته شده از کلاس SuppliersBLL لایه ی BLL باشد, اما استفاده از کلاس StaticCache باعث شده است تا داده های supplierرا در آغاز application , cache شده بر گرداند. می توانید breakpoint ی در متد ()GetSuppliers کلاس StaticCache جهت بررسی این عملکرد قرار دهید.



شکل7: داده های cache شده ی supplier در GridView نشان داده شده است.

خلاصه

بیشتر مدلهای داده شامل مجموعه ای از داده های استاتیک هستند, که معمولا به فرم جداول مراجعه (lookup tables) پیاده سازی می شوند. از آنجا که این اطلاعات استاتیک هستند, هیچ دلیلی برای ادامه ی دسترسی به دیتابیس در هر بار که این اطلاعات لازم است نمایش داده شوند, وجود ندارد.
بعلاوه, به دلیل ماهیت استاتیک آنها, وقتی داده ها cache می شوند هیچ نیازی به انقضا و خاتمه نیست. در این مبحث دیدیم که چگونه داده ها را بگیریم و آنها را در data cache , cache کنیم. اطلاعات در آغاز یک application , cache شدند و تا زمان حیات application در cache باقی می مانند.
در این مبحث و دو مبحث قبل ,جهت cache کردن داده ها برای مدت حیات application , از منقضی شدن مبتنی بر زمان استفاده کردیم. جهت کار با دیتابیس ها cache کردن مبتنی بر زمان ایده آل و خیلی کارا نیست. Cache کردن در صورتی بهینه وایده آل است که زمانی که داده ها در لایه ی زیرین ویرایش شدند از cache خاج شوند .جهت بکارگیری این تکنیک باید از SQL cache dependencies استفاده کنیم که موضوع مبحث بعدی است .

شاد و موفق باشید
پاییز 86