EntitySpaces 2011 Release Notes

May 9th, 2011
EntitySpaces, LLC
The EntitySpaces Team

Persistence Layer and Business Objects for Microsoft .NET

CONTENTS

REQUIREMENTS

Microsoft .NET 3.5 Framework

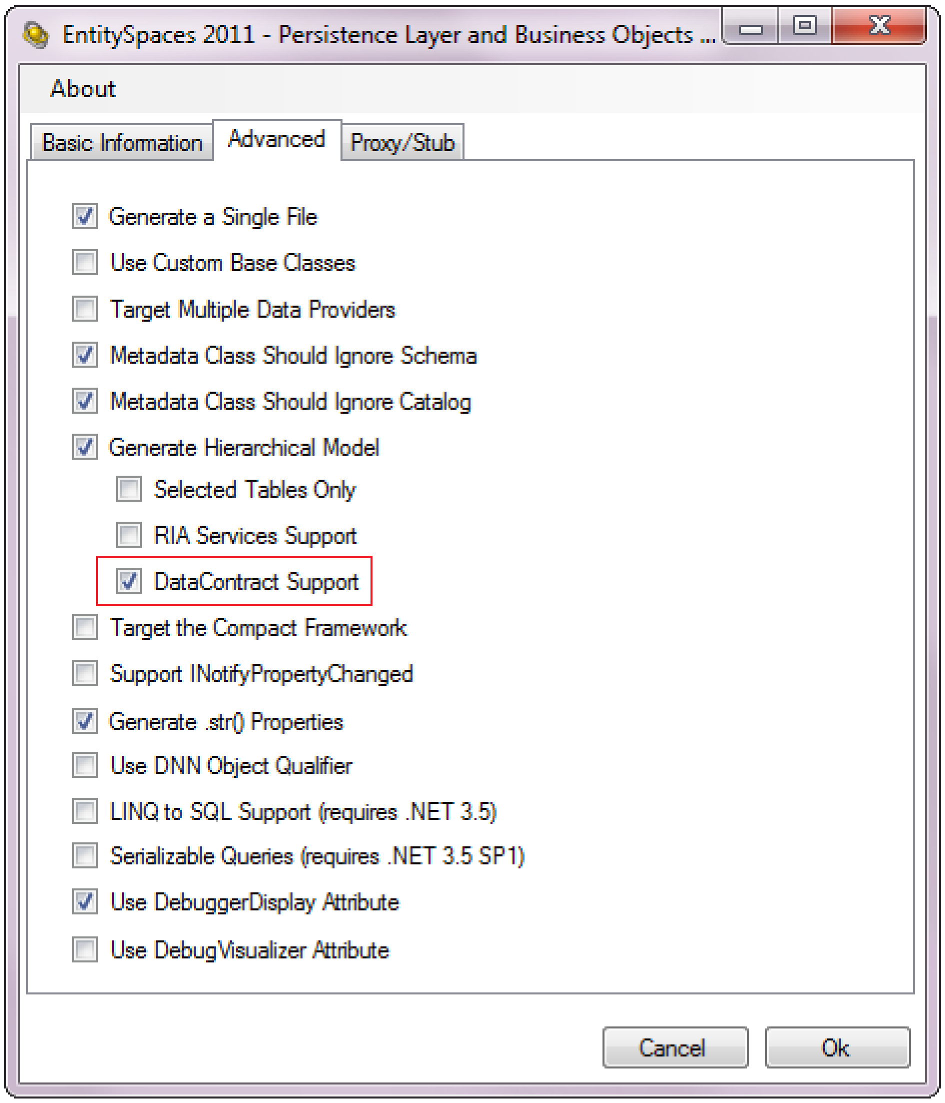
The EntitySpaces 2011 architecture requires the Microsoft .NET 3.5 Framework. It does not support running under .NET 2.0. Customers that wish to develop for the .NET 2.0 Framework will have to remain on EntitySpaces 2009.

INSTALLATION

EntitySpaces 2011 can be installed side-by-side with EntitySpaces 2010. You cannot install two copies of EntitySpaces 2011 side-by-side. We recommend accepting the defaults during installation. For Vista and Windows 7 installations, you should right-click the installer and "Run as Administrator". There is a "Manual Setup" PDF installed to your Start Menu if you have issues using the EntitySpaces 2011 AddIn in Visual Studio.

ENHANCED DATACONTRACT SUPPORT

The Generated classes now fully implement DataContract by default


EntitySpaces 2011 - Persistence Layer and Business Objects ...

About

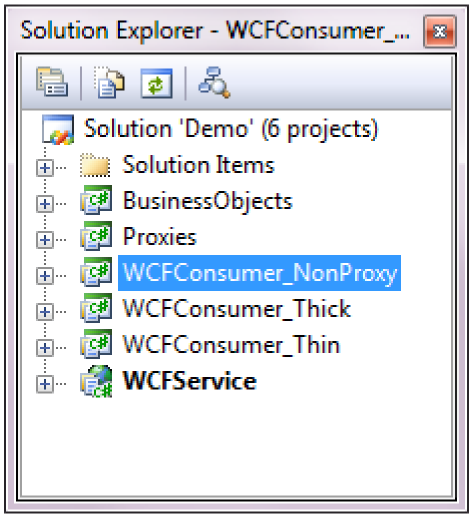
Basic Information | Advanced | Proxy/Stub

- [✓] Generate a Single File
- [ ] Use Custom Base Classes
- [ ] Target Multiple Data Providers
- [✓] Metadata Class Should Ignore Schema
- [✓] Metadata Class Should Ignore Catalog
- [✓] Generate Hierarchical Model
  - [ ] Selected Tables Only
  - [ ] RIA Services Support
  - [✓] DataContract Support
- [ ] Target the Compact Framework
- [ ] Support INotifyPropertyChanged
- [✓] Generate .str() Properties
- [ ] Use DNN Object Qualifier
- [ ] LINQ to SQL Support (requires .NET 3.5)
- [ ] Serializable Queries (requires .NET 3.5 SP1)
- [✓] Use DebuggerDisplay Attribute
- [ ] Use DebugVisualizer Attribute

Beginning with EntitySpaces 2011 the "Generated" classes themselves are serializable via DataContract. There is no checkbox for this, it is always enabled.

The checkbox that is checked (shown on the left) **takes this new serialization one step further** and enables serialization of the hierarchical model via DataContract as well.

An example of this new serialization is available in the "WCF Sample" which is available on your "EntitySpaces 2011" menu after installation. The solution is shown on the right.

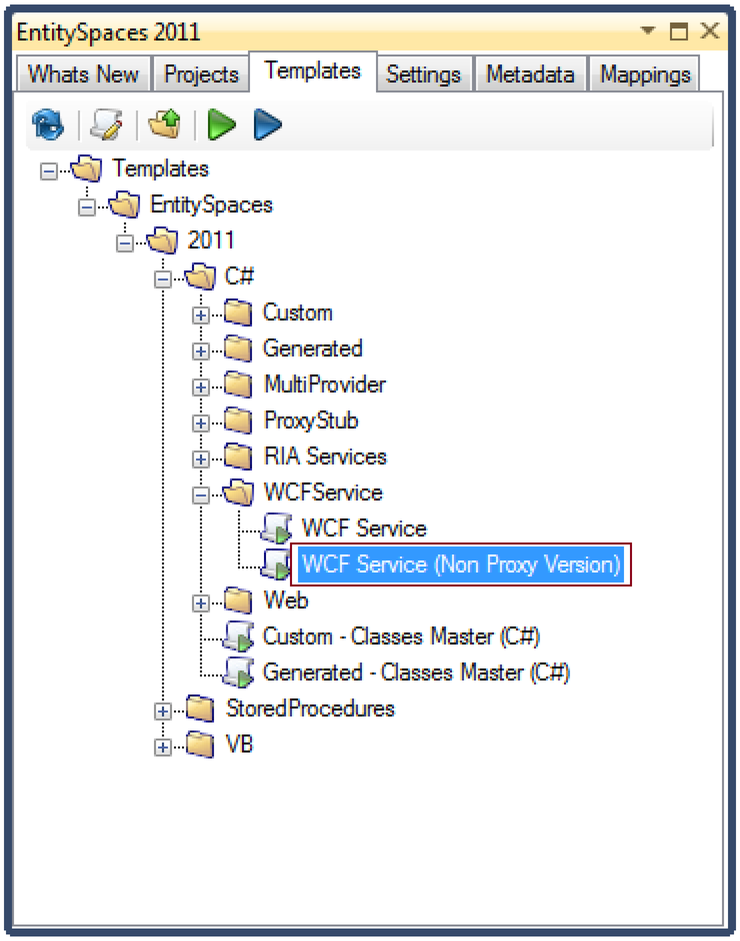

Solution Explorer - WCFConsumer_...

- Solution 'Demo' (6 projects)
  - Solution Items
  - BusinessObjects
  - Proxies
  - WCFConsumer_NonProxy
  - WCFConsumer_Thick
  - WCFConsumer_Thin
  - **WCFService**

The "Thick" and "Thin" proxies are still supported of course. But In many cases they are no longer needed. However, they still can be very useful, especially when it comes to JSON serialization. We plan to provide a set of JSON serialization samples soon.

NEW WCF SERVICE TEMPLATE

The Non-Proxy Based WCF Service Template



```
Templates
  EntitySpaces
    2011
      C#
        Custom
        Generated
        MultiProvider
        ProxyStub
        RIA Services
        WCFService
          WCF Service
          WCF Service (Non Proxy Version)
        Web
        Custom - Classes Master (C#)
        Generated - Classes Master (C#)
      StoredProcedures
      VB
```

This new template (shown highlighted in the image on the left) goes hand-in-hand with the new DataContract support. This template creates a WCF Service that use your EntitySpaces classes directly as opposed to either the thick or thin proxies.

If you check the "DataContract Support" checkbox on the Advanced tab of the "Generated Master" template then your WCF Service will serve up and receive entire object graphs.

If you desire to use your full EntitySpaces business objects on both sides of a WCF Service conversation this template is an excellent choice.

OBJECT GRAPH LEVEL API

The new Object Graph API operates hierarchically throughout the object model. It is available on both collections and single entities. The Object Graph API methods and or properties never cause hierarchical properties or collections to be lazy loaded. The Object Graph methods walk all hierarchical objects and collections, including UpTo's.

IsGraphDirty

This method will report true if there is any object in the object graph that is dirty.

```
Employees emp = new Employees();
if (emp.LoadByPrimaryKey(2))
{
    // Modify an object in the graph
    emp.EmployeesCollectionByReportsTo[0].FirstName = "Joe";
    if (emp.es.IsGraphDirty)
    {
        // Yes, it's dirty
    }

}
```

AcceptChangesGraph/RejectChangesGraph

You can now Accept or Reject changes across your entire object graph. Only AcceptChangesGraph() is shown below, however, you could simply replace the call to AcceptChangesGraph with RejectChangesGraph().

```
Employees emp = new Employees();
if (emp.LoadByPrimaryKey(2))
{
    // Modify an object in the graph
    emp.EmployeesCollectionByReportsTo[0].FirstName = "Joe";

    foreach (Orders order in emp.OrdersCollectionByEmployeeID)
    {
        order.Freight = 7.95M;
    }

    // Cause all changes to be accepted through the graph
    emp.AcceptChangesGraph();

}
```

PruneGraph

The PruneGraph method, if no parameters are passed in, removes all of the collections and entities in the object graph which have a RowState of esDataRowState.Unchanged. In other words, only collections and entities that have changes would remain in the graph. However, there is one exception, if an entity or collection is not dirty, but has a descendant that is dirty then that entity will remain in the graph in order to preserve the descendant. The important thing to note is that the object graph contains only those objects necessary to maintain all dirty objects.

This can be very useful. For instance, if you want to send changes from a client up to the server and you only want to send the objects in the graph that are dirty, then PruneGraph is for you. If you combine PruneGraph with the new hierarchical DataContract support you have a very solid communication method.

```
Employees emp = new Employees();
if (emp.LoadByPrimaryKey(2))
{
    // Modify a few object in the graph
    emp.EmployeesCollectionByReportsTo[0].FirstName = "Joe";
    emp.OrdersCollectionByEmployeeID[3].Freight = 7.95M;

    // Prune the graph so that it contains only objects with changes
    emp.PruneGraph();

}
```

The PruneGraph method does have an overload, it looks like this. Note that you can pass in multiple states.

*public void PruneGraph(***esDataRowState** *statesToPrune)*

```
EmployeesCollection coll = new EmployeesCollection();
if (coll.LoadAll())
{
    // Add a new entity
    Employees emp = coll.AddNew();

    // Modify a few object in the graph
    coll[0].EmployeesCollectionByReportsTo[0].MarkAsDeleted();
    coll[0].OrdersCollectionByEmployeeID[3].Freight = 7.95M;

    // Prune the graph so that it contains only objects with changes
    emp.PruneGraph(esDataRowState.Modified | esDataRowState.Deleted);
}
```

The example above will prune all of the objects which have a RowState of either Modified or Deleted. After the call, the EmployeesCollection would only contain one entity, the one we added.

THE ESVISITOR CLASS

The esVisitor Class, a new powerful API available to you.

The esVisitor class is the class we use internally to implement the Object Graph methods such as PruneGraph. We have made this available to you as a developer. Probably one of the questions running through your mind as you were reading through the list of new Object Graph methods is “Where is the MarkAsDeletedGraph() method?” Well, we didn’t provide such a method because of the potential for deleting far more data than you intend to. Remember, just inspecting an object in the debugger causes all its hierarchical properties to be lazy-loaded. That being said, one could write such a method using the esVisitor class.

For more insight into the esVisitor class see the BLOG post.

Also, for more information on the esVistor class see the .CHM Help files on your “EntitySpaces 2011” menu.

THE APPLYDEFAULTS METHOD

Providing Default Values Through the ApplyDefaults method

The ApplyDefaults() method is a new virtual method in the abstract esEntity class. You can override this method in your Custom class, in the single entity specifically, to supply defaults when a new entity is created.

There are two situations which will cause this method to be invoked.

1. Instantiating a new entity via the "New" operator.
2. Calling AddNew() on a collection.

However, ApplyDefaults is not invoked until you actually access a property on that newly created entity. The effect however is that it is instant since the moment you look at the entity to view a property the ApplyDefaults() method is called and any default values you apply will be present in the entity.

Here is an example of how you might use ApplyDefaults.

```
namespace BusinessObjects
{
    public partial class Employees : esEmployees
    {
        public Employees()
        {

        }

        protected override void ApplyDefaults()
        {
            this.DateAdded = DateTime.Now;
            this.ModifiedDate = DateTime.Now;
            this.AddedBy = "mgriffin";
            this.ModifiedBy = "mgriffin";
        }
    }

}
```

NEW AUDITING FIELD SUPPORT

The idea behind these new auditing fields is that EntitySpaces 2011 will handle these for you. EntitySpaces knows when they need to be updated and will do so automatically. EntitySpaces 2011 recognizes four new special fields, they are as follows:

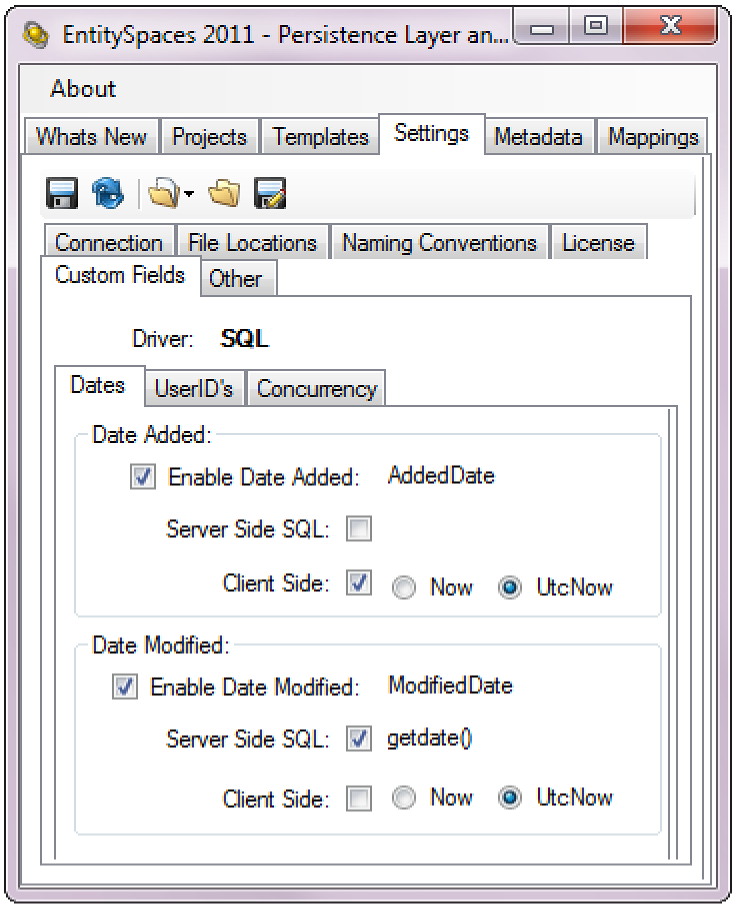
1. DateAdded
2. DateModified
3. AddedBy
4. ModifiedBy

The names are of the columns aren't set in stone. You can change them of course. You can adjust the names of these columns to fit your naming scheme by choosing the "Settings" tab and then the "Custom Fields" tab.

The new auditing field support is available in all providers. However, if you are using stored procedures only the Microsoft SQL Server stored procedure template supports the auditing fields in this release. The Oracle, PostgreSQL, and MySQL stored procedure templates will be updated in a future release.

DateAdded and DateModified

It is quite common to have these two fields in each of your tables so that you will know when the record was created and the last time the record was modified. Let's take a look at the "Custom Fields" tab for these two fields.



Driver: SQL

Date Added:
[✓] Enable Date Added: AddedDate
Server Side SQL: [ ]
Client Side: [✓] Now [✓ UtcNow]

Date Modified:
[✓] Enable Date Modified: ModifiedDate
Server Side SQL: [✓] getdate()
Client Side: [ ] Now [✓ UtcNow]

The first thing to notice is that we have enabled both auditing date types. We have also renamed the columns to match our schema, in this case, AddedDate and ModifiedDate are the names of the column in our tables.

The AddedDate field is set to be updated on the Client side using DateTime.UtcNow. We have setup the ModifiedDate to be filled in on the server side by the database using the Microsoft SQL "getdate()" function. You can use any valid SQL here for your database engine.

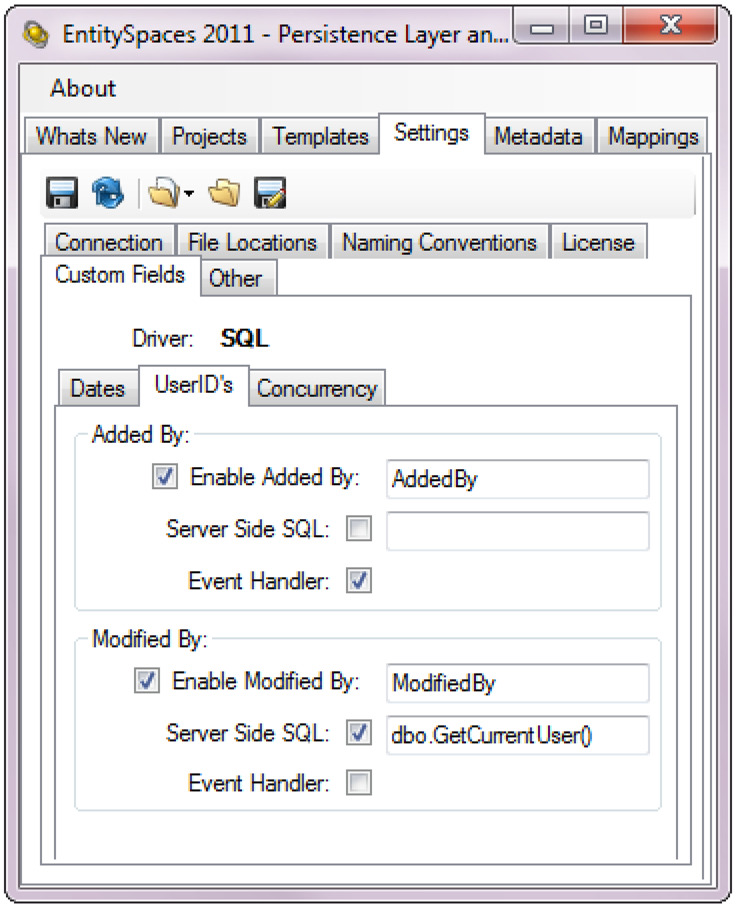
Both the AddedDate and ModifiedDate columns will be filled in automatically during an INSERT. However, only the ModifiedDate field is changed during an UPDATE.

It is during code generation that the logic is inserted into your EntitySpaces Business Objects to make all this happen. If a table has a column named "AddedDate" or "ModifiedDate" then the logic will be added for the appropriate field.

Finally, the Client Side values are applied in the Save() method itself, as are of course the Server Side changes. So, if you need to have these dates visible to your UI on your new entities before calling Save() you would need to use the new ApplyDefaults() method.

AddedBy and ModifiedBy

It is quite common to have these two fields in each of your tables so that you will know who the record was created by and who the last person was to modify the record. Let's take a look at the "Custom Fields" tab for these two fields.



Driver: SQL

Added By:
[✓] Enable Added By: AddedBy
Server Side SQL: [ ]
Event Handler: [✓]

Modified By:
[✓] Enable Modified By: ModifiedBy
Server Side SQL: [✓] dbo.GetCurrentUser()
Event Handler: [ ]

The first thing to notice is that we have enabled both User ID data types.

The AddedBy field is set to be updated on the Client side using an event handler that we will supply. We have setup the ModifiedBy field to be filled in on the server side by the database using our custom Microsoft SQL "dbo.GetCurrentUser()" function. You can use any valid SQL here for your database engine.

Both the AddedBy and ModifiedBy columns will be filled in automatically during an INSERT. However, only the ModifiedBy field is changed during an UPDATE.

It is during code generation that the logic is inserted into your EntitySpaces Business Objects to make all this happen. If a table has a column named "AddedBy" or "ModifiedBy" then the logic will be added for the appropriate field.

Finally, the Client Side values are applied in the Save() method itself, as are of course the Server Side changes. So, if you need to have these dates visible to your UI on your new entities before calling Save() you would need to use the new ApplyDefaults() method.

Below is how we setup the event handler for the AddedBy column. EntitySpaces will call it for you automatically.

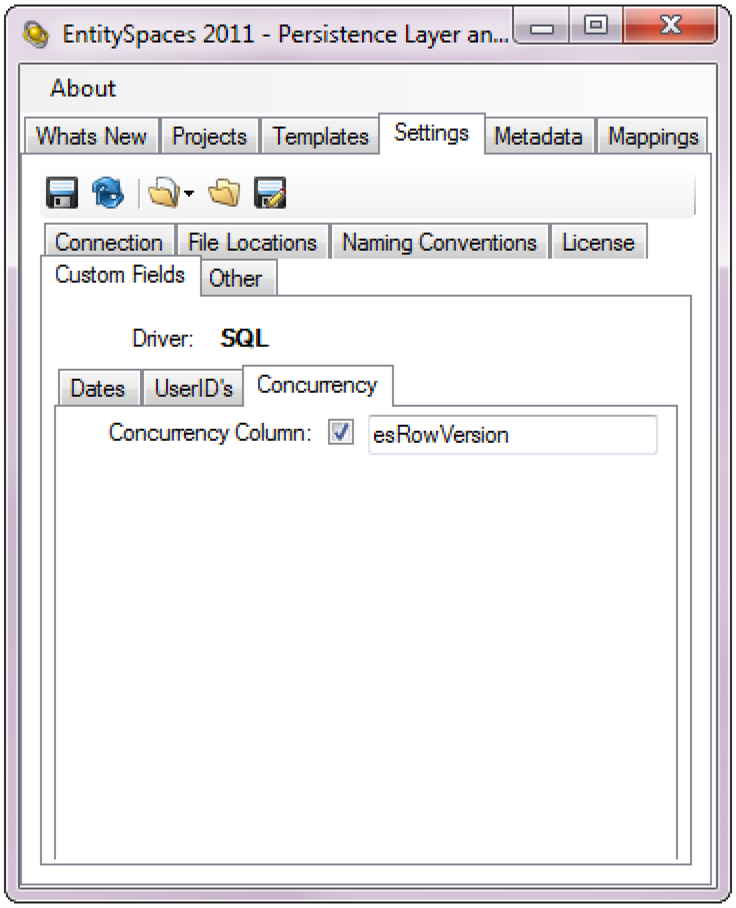
```
static void Main()
{
    // Hook up my AddedBy Event Handler
    esEntity.AddedByEventHandler += new ModifiedByEventHandler(AddedByEventHandler);

    // Register the loader
    esProviderFactory.Factory = new esDataProviderFactory();

    Application.Run(new Form1());
}

static object AddedByEventHandler()
{
    // Pull your value from the session, or call a
    // method, do whatever you need to here ...
    return "mgriffin";

}
```

Concurrency

EntitySpaces provides its own concurrency column support in addition to supporting native DBMS mechanisms such as Microsoft SQL's "timestamp" column. To use the EntitySpaces Concurrency feature you would add an "int" or "bigint" column to your table. In EntitySpaces 2010 you would set the "IsEntitySpacesConcurrency" property to true for that column in the Metadata tab. This can be quite cumbersome when you have 150 tables to perform this operation on.

Now, in EntitySpaces 2011, you can simply declare the column once, as shown below.

In this case our column is named "esRowVersion". The nice thing about the built-in EntitySpaces Concurrency feature is that you can write database independent applications and still have concurrency checking. This would normally be a problem as most native DBMS concurrency column types are quite different depending on what database you are working with. The EntitySpaces Concurrency feature solves this problem by making it generic across all DBMS systems.



Driver: SQL

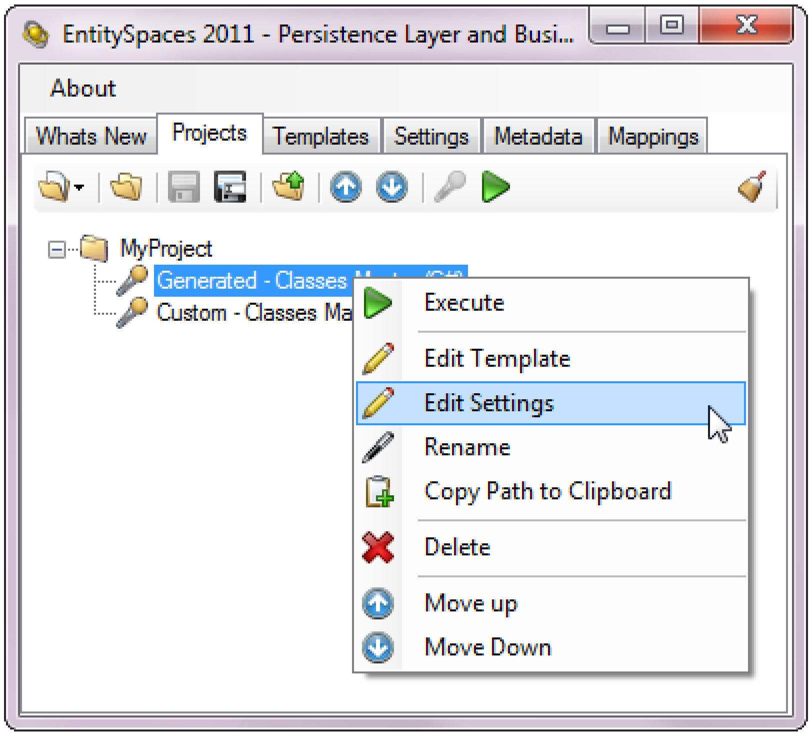
Concurrency Column: ✓ esRowVersion

PROJECT FILE ENHANCEMENTS

EntitySpaces project files allow you to record a template's execution so that you can play it back (re-execute) at a later time. This can save you a lot of time and alleviates you from having to try and remember what choices you made in the user interface weeks later when you need to regenerate your classes.

We put a lot of work into the EntitySpaces project files in ES2011. EntitySpaces users have always been able to re-edit a template recording (the UI choices). A template node contains all of the UI choices that were made when recording the template and the actual settings such as connection string and so on that were in place at the time of the recording. The settings of course were always in the project file, but there was no way (other than by hand) to edit them. EntitySpaces 2011 now allows you to edit the original settings on each individual template node. How? Simply right mouse on any template node and choose "Edit Settings" from the context menu.

Editing the Settings



Once you create a project file you can easily go back and edit the UI choices via the "Edit Template" menu choice.

However, new to EntitySpaces 2011 is the "Edit Settings" menu choice. This allows you to edit the settings (from the Settings Tab) as they were at the moment when the template was recorded. An EntitySpaces project file can, for instance, can contain many templates, some that access Oracle, SQL, MySQL and so on. If you ever need to go back and edit those settings you simply choose the "Edit Settings" context menu and make your changes there.

If you aren't using EntitySpaces Project Files you should consider using them as they can certainly make your life much easier.

EntitySpaces 2011 will convert your EntitySpaces 2010 project files over to our improved EntitySpaces 2011 format.

ESSETTING FILE ENHANCEMENTS

The esSettings.xml file which stores your settings has been simplified. In addition, we no longer encrypt the proxy settings which you are entered on the "Licensing" tab. They are stored in plain text. This has the added benefit of allowing your project files to be shared by all developers on your team.

WINDOWS PHONE 7 SUPPORT

The Runtimes folder now contains a new subfolder called "Windows Phone 7". In it you will find an assembly named "EntitySpaces.DynamicQuery.WP7.dll". This assembly allows you to use the full EntitySpaces DynamicQuery API under Windows Phone 7. This is similar to the Silverlight support we provide.

There is also a Windows 7 sample application on the "EntitySpaces 2011" menu after installation.

MISCELLANEOUS BUG FIXES AND ENHANCEMENTS

- The EntitySpaces.NpgsqlProvider.dll for Npgsql 1.x is no longer supported, the 1.x driver for PostgreSQL is long obsolete. The EntitySpaces.Npgsql2Provider.dll for Npgsql for 2.x is of course still supported.
- Both AttachEntity and DetachEntity now work properly when a Filter is ongoing.
- Fixed the bug in IBindingList.Find(PropertyDescriptor property, object key) reported on this thread.
- Fixed the bug in the hierarchical VB template reported on this thread.
- Modified the way our Many-To-Many's work in the hierarchical code. We now use the DynamicQuery API for these which allows them to work cross-schema.
- Updated the VistaDB and EffiProz drivers. We are now bound to the final version of VistaDB that did not require a license.
- Concurrency Checks are now done during Deletes for all providers.