EntitySpaces Getting Started

From Download to Your First Application

March 19th, 2012
EntitySpaces, LLC
The EntitySpaces Team

Persistence Layer and Business Objects for Microsoft .NET

CONTENTS

..... 1
Purpose 4
Requirements 5
  Microsoft .NET 3.5 Framework or Higher 5
  EntitySpaces 2012 5
Additional Resources 5
Installing EntitySpaces 6
Launching EntitySpaces 6
  The EntitySpaces Standalone Version 6
  The EntitySpaces Visual Studio Add-In 7
Generating Your EntitySpaces Architecture 8
  Connect to your Database 8
    The Connection Settings Tab 8
  The File Locations Settings Tab 10
  Saving Default Settings 10
  The Templates Tab 13
    Executing the EntitySpaces Templates 14
    Running the Generated Classes Master Template 15
    Running the Custom Classes Master Template 20
The EntitySpaces Configuration File 21
  The Config File 21
  Configless Execution 22
  Standard .NET Connection Strings 22
  DotNetNuke Configuration Settings 23
Registering the Loader 24
High Level Architectural Overview 26
High Level Object Usage 27
  Collections 27
    Loading a Collection 27
    Querying a Collection 27
    Saving a Collection 27
    Adding a New Record Through a Collection 27
    Deleting a Record Through a Collection 28
  Entities 29
    Loading a Single Entity 29
    Querying an Entity 29

Saving an Entity 29

Deleting an Entity 29

The EntitySpaces Assemblies 30

The EntitySpaces Assemblies 31

PURPOSE

The purpose of this document is to provide you with very simple but accurate instructions to help you get your first EntitySpaces application up and running quickly. There is also a high-level architectural overview at the end of the document showing some common coding techniques using EntitySpaces.

Here is a very quick checklist list of what you need to do:

- ✓ Install EntitySpaces.
- ✓ Launch EntitySpaces
- ✓ Connect to your database.
- ✓ Generate your EntitySpaces classes into your Visual Studio Solution folder(s).
- ✓ Add the proper connection information to your app.config /web.config file.
- ✓ Add references to
  - EntitySpaces.Core
  - EntitySpaces.Interfaces
  - EntitySpaces.DynamicQuery
- ✓ Add a reference to **one** of the EntitySpaces Loaders
  - EntitySpaces.Loader
  - EntitySpaces.LoaderMT (medium trust mode).
- ✓ Register the Loader
- ✓ Add reference(s) to one or more of the EntitySpaces database provider(s). Just the ones you need.
- ✓ Add a using statement (Imports in VB.NET) for the namespace of your generated business objects (the default is BusinessObjects).
- ✓ Write your application in record time.

It is very simple to do all of the above, and most of it is one time setup. If you run into an issue, just register and post in the EntitySpaces forums.

REQUIREMENTS

MICROSOFT .NET 3.5 FRAMEWORK OR HIGHER

The EntitySpaces 2012 architecture uses many of the features that are present in the 3.5 .NET Framework. You will need to have this installed before you install EntitySpaces. If you need to target the 2.0 .NET Framework, then use EntitySpaces 2009. They can be installed side-by-side.

ENTITYSPACES 2012

The latest production version as of this writing is EntitySpaces 2012 (2012.1.0319.0). The download link for the EntitySpaces 2012 Trial can be found on our home page at http://www.entityspaces.net.

ADDITIONAL RESOURCES

This document will take you through the essentials necessary for a working solution using EntitySpaces. However, there is a lot more power waiting for you under the hood when you are ready. Here is a list of additional resources that are available to you once you are familiar with "Getting Started".

- ✓ The EntitySpaces API Reference, installed to your start menu, is a fully indexed, compiled help file documenting the EntitySpaces classes, methods, and properties.
- ✓ The Documentation menu on the EntitySpaces site contains links to videos and our Developer Documentation site, which includes a tremendous number of coding examples.
- ✓ Our Team Blog will not only keep you informed of releases and upcoming events, but contains detailed examinations of new features as they are released. Browsing back through some of the older posts will reveal some gems on using EntitySpaces with the Compact Framework, LINQ, Oracle Sequences, and more.
- ✓ VB and C# EntitySpaces Demos, with source code are installed to your program folder. The demos are designed to run against SQL Server Northwind database. Running the demo is as simple as changing the connection string in App.config, compiling, and executing.
- ✓ VB and C# esDataSource examples are installed, as well.
- ✓ The EntitySpaces forums are your primary means of support. Registration is free, open to all, and we have a very strict privacy policy. We take great pride in our response times, and have many generous, knowledgeable users that will help answer your questions. You should find the FAQ section particularly interesting.

INSTALLING ENTITYSPACES

The Trial version is available from our Trial Download page. If you are a customer, once you are logged in to our Home Page, you can find the production release, and source if you purchased that, under the Products menu (the Download Page).

EntitySpaces 2012 can be installed side-by-side with EntitySpaces 2011. You cannot install two versions of EntitySpaces 2012 side-by-side.

Be sure that Visual Studio is closed during installation. For Vista or Windows 7, you should download the installer, right-click, and "Run as Administrator".

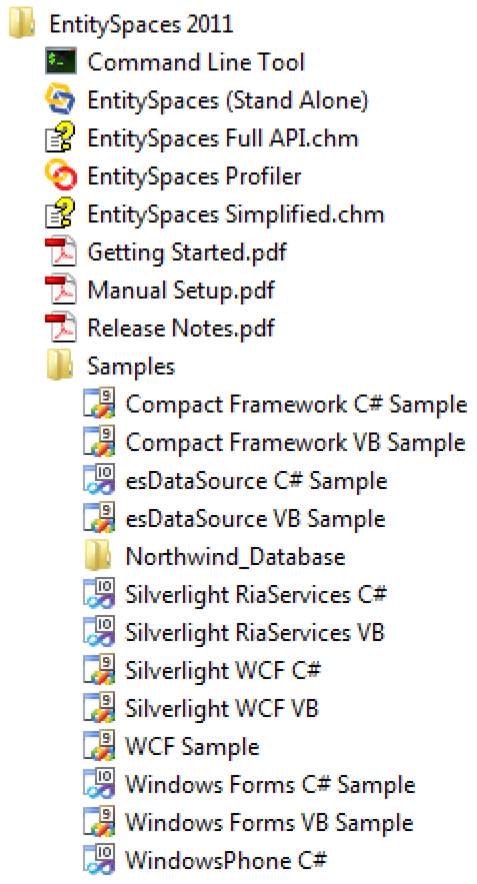
You must first read and accept the license agreement by pressing the “I Agree” button to install EntitySpaces. We recommend that you accept the default installation path.

LAUNCHING ENTITYSPACES

At this point, you are ready to use EntitySpaces. There are two ways to launch EntitySpaces.

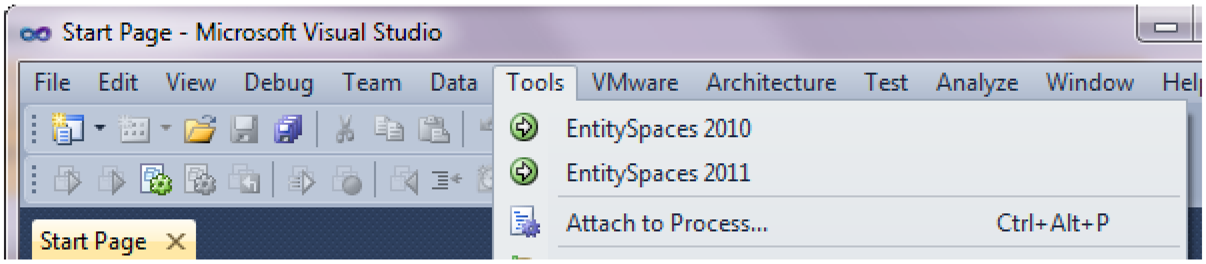
THE ENTITYSPACES STANDALONE VERSION

After installation, the EntitySpaces menu will be available on your Windows Start Menu. You can launch the StandAlone version via the menu as shown below. The StandAlone version has the same UI as the Visual Studio Add-In, but does not require Visual Studio to run.



THE ENTITYSPACES VISUAL STUDIO ADD-IN

After you launch Visual Studio (2005/2008/2010), you will find the EntitySpaces 2012 Add-In under the “Tools” menu as shown below.


Start Page - Microsoft Visual Studio
File Edit View Debug Team Data Tools VMware Architecture Test Analyze Window Help
EntitySpaces 2010
EntitySpaces 2011
Attach to Process... Ctrl+Alt+P
Start Page

If you are using an international version of windows, and you do not see “EntitySpaces 2012” under your “Tools” menu, follow the instructions in the “Manual Setup" PDF. The PDF is located under your EntitySpaces Start Menu folder. It is really quite simple to register manually; just perform a few minor steps.

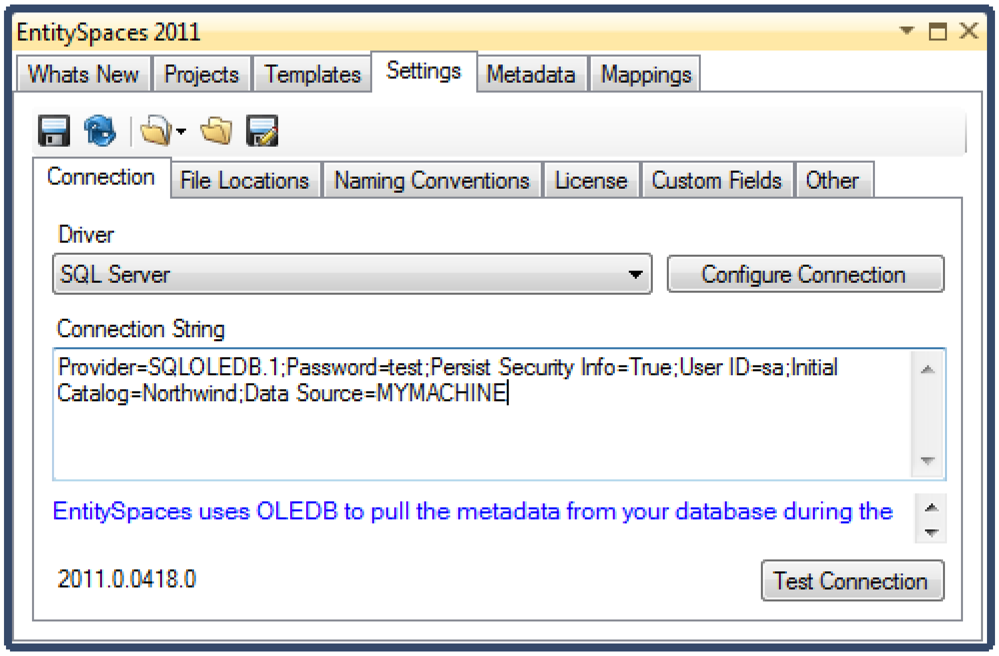
GENERATING YOUR ENTITYSPACES ARCHITECTURE

The EntitySpaces Architecture (C# or VB) can be generated quickly using the EntitySpaces Templates. The metadata from your database is used to generate all of your code.

CONNECT TO YOUR DATABASE

The first step is to connect to your database.

THE CONNECTION SETTINGS TAB



```
Provider=SQLOLEDB.1;Password=test;Persist Security Info=True;User ID=sa;Initial Catalog=Northwind;Data Source=MYMACHINE
```

EntitySpaces uses OLEDB to pull the metadata from your database during the

2011.0.0418.0

Choose your "Driver. The drivers of interest for EntitySpaces are as follows:

Driver Name	Supports
Microsoft SQL Server	SQL 2000/2005/2008 (including MSDE and Express) *
Microsoft SQL CE	SQL CE 3.x
Microsoft Access	Microsoft Access 2000 or later *
MySQL	MySQL 4.1, 5.0
Oracle	Oracle 9i, 10g *
PostgreSQL	PostgreSQL 8.x
Sybase	Sybase SQL Anywhere 11
SQLite	SQLite 3.x
VistaDB	VistaDB 3.x
VistaDB4	VistaDB 4.x

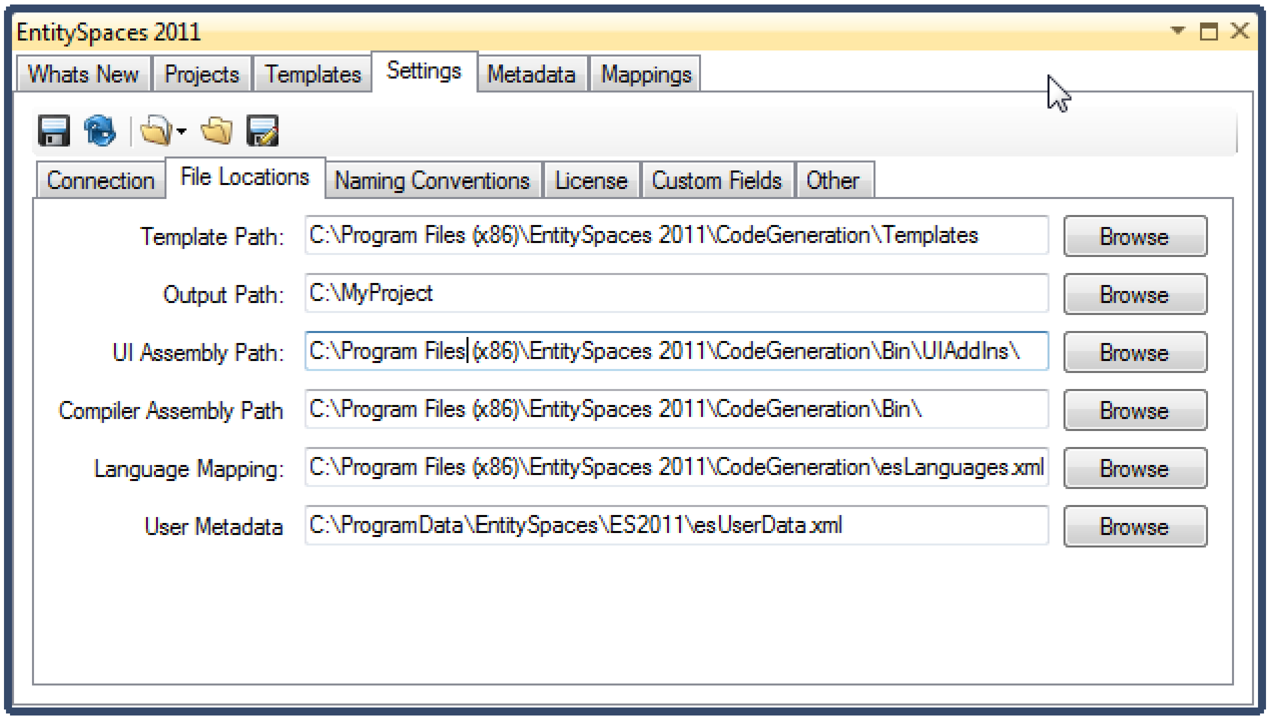
* The EntitySpaces Code Generator uses OLEDB drivers to connect to these. You will need to use an OLEDB connection string for them in this Settings tab. (NOTE: The connections in the EntitySpaces applications you create use the databases' ADO.NET provider, not OLEDB. For example, you would not have "Provider=SQLOLEDB.1;" in an SQL Server ADO.NET connection string in your config file.)

All you really need to do on this tab is make sure you can connect to your database. Use the “Configure Connection” button to create an OLEDB connection string, or manually enter one. Use the "Test Connection" button to confirm it.

Some databases have some additional requirements for code generation. You will need to follow the instructions below for the database you are using.

- **SQL CE**
  - You must remove any password from the database before generating the templates. EntitySpaces cannot read the necessary schema information from an encrypted database. Once you have generated your classes, you can add the password protection back, and the EntitySpaces API will work against the encrypted database based on the credentials in the connection string.
  - EntitySpaces requires a special "Version" parameter to determine which version of SQL CE was used to create your .sdf database file. The version number and PublicKeyToken can be determined by looking at the assembly in the GAC. Some examples are:
    - **3.0 -** Data Source=C:\SomeDatabase.sdf;Version="9.0.242.0, Culture=neutral, PublicKeyToken=89845dcd8080cc91";
    - **3.5 -** Data Source=C:\SomeDatabase.sdf;Version=" 3.5.1.0, Culture=neutral, PublicKeyToken=89845dcd8080cc91";
- **VistaDB**
  - If you need to compile the EntitySpaces VistaDB MetadataEngine Plugin against your version of the VistaDB provider installed by VistaDB here are the instructions. The default location of the solution is "C:\Program Files\EntitySpaces 2012\CodeGeneration\EntitySpaces.MetadataEngine.VistaDB".
    - Open EntitySpaces.MetadataEngine.VistaDB.sln.
    - Delete EntitySpaces.snk.
    - Under References, remove the reference to EntitySpaces.MetadataEngine, and add a reference to it from "C:\Program Files\EntitySpaces 2012\CodeGeneration\Bin".
    - Copy EntitySpaces.MetadataEngine.VistaDB.dll from the solution bin folder to "C:\Program Files\EntitySpaces 2012\CodeGeneration\Bin".
- **Microsoft Access**
  - If you encounter ADODB errors with the EntitySpaces Visual Studio Add-In, use the EntitySpaces StandAlone code generator instead.

THE FILE LOCATIONS SETTINGS TAB



Template Path: C:\Program Files (x86)\EntitySpaces 2011\CodeGeneration\Templates
Output Path: C:\MyProject
UI Assembly Path: C:\Program Files (x86)\EntitySpaces 2011\CodeGeneration\Bin\UIAddIns\
Compiler Assembly Path C:\Program Files (x86)\EntitySpaces 2011\CodeGeneration\Bin\
Language Mapping: C:\Program Files (x86)\EntitySpaces 2011\CodeGeneration\esLanguages.xml
User Metadata C:\ProgramData\EntitySpaces\ES2011\esUserData.xml

The important thing here is your “Output Path”. If you point this to your Visual Studio solution project folder for the EntitySpaces application you are creating, this will create the files right in your folder (in subfolders). Then, all you need to do in Visual Studio is click “Show All Files” and include them. In the future, you can just regenerate, and you are off and running. If you have not created your Visual Studio solution, yet, just set this later once you have. At generation time, the “Output Path” on this screen will be the default path in the Template UI. You can change the output path in the template UI, and override this default.

On the screen, above, we are setting the default path to "C:\MyProject". It could just as easily be a folder in a Windows.Forms application or the App_Code folder of our ASP.NET web site.

When the EntitySpaces Templates are executed, based on the path above, the following folders will automatically be created:

1. C:\MyProject\Custom
2. C:\MyProject\Generated
3. C:\MyProject\Proxies (these classes are optional)

NOTE: Do not tack “Custom” or “Generated” onto your default path, the templates will do that for you.

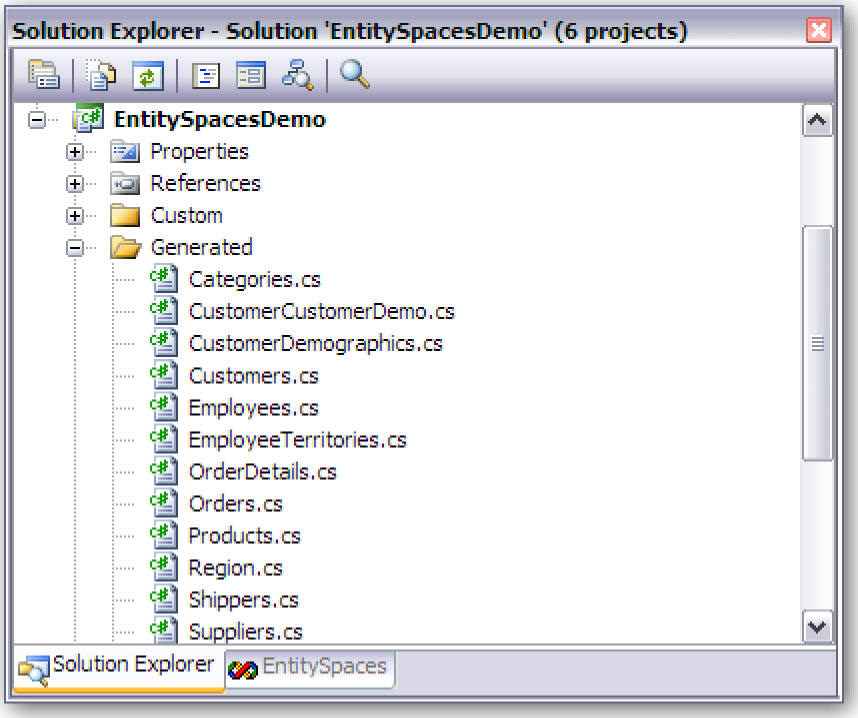
SAVING DEFAULT SETTINGS

You can save the changes you have made to the Settings Connection tab and the Settings File Locations tab, as well as any changes on any of the Settings tabs, by clicking the "Save Default Settings" icon on the toolbar. The next time you start Visual Studio, or the StandAlone, these will be loaded as your the default settings.

Sample Visual Studio Solutions

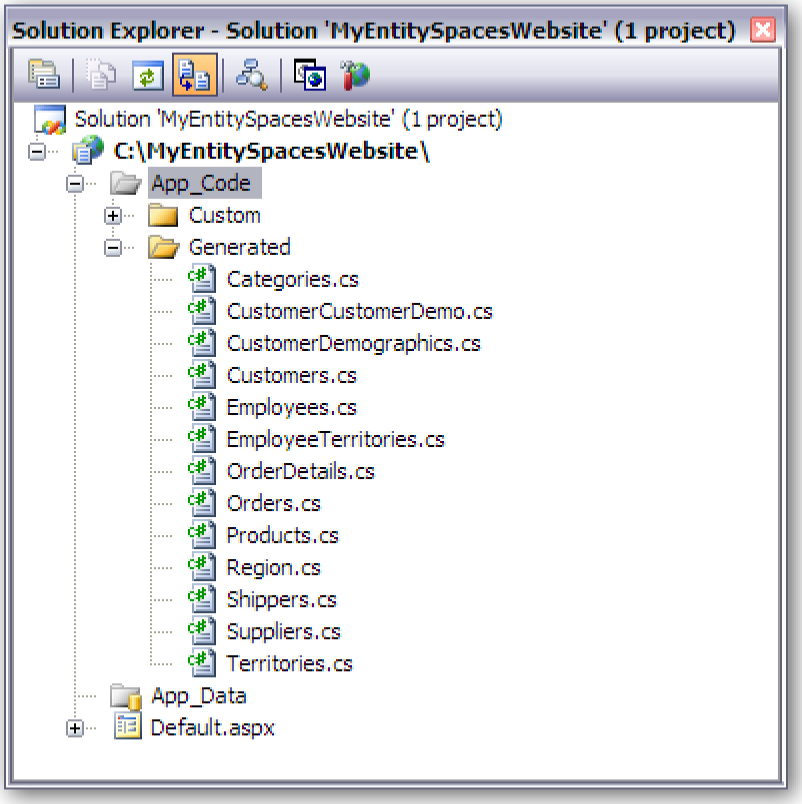
With the Default Output Path set as described in the File Locations Settings section above, you can generate your EntitySpaces classes, and simply include them in your project. They will be in the right folders for you any time your database schema changes and you need to re-generate. If you are using a file-based ASP.NET folder, and you generate right into your App_Code folder, you can simply compile directly after generation. The two screen examples below show the "Custom" and "Generated" folders, and some of the files generated for the Northwind database by the EntitySpaces templates.

An example Windows Form Project


Solution Explorer - Solution 'EntitySpacesDemo' (6 projects)

```
EntitySpacesDemo
    Properties
    References
    Custom
    Generated
        Categories.cs
        CustomerCustomerDemo.cs
        CustomerDemographics.cs
        Customers.cs
        Employees.cs
        EmployeeTerritories.cs
        OrderDetails.cs
        Orders.cs
        Products.cs
        Region.cs
        Shippers.cs
        Suppliers.cs
```

An example file-based ASP.NET website using the App_Code folder


Solution Explorer - Solution 'MyEntitySpacesWebsite' (1 project)

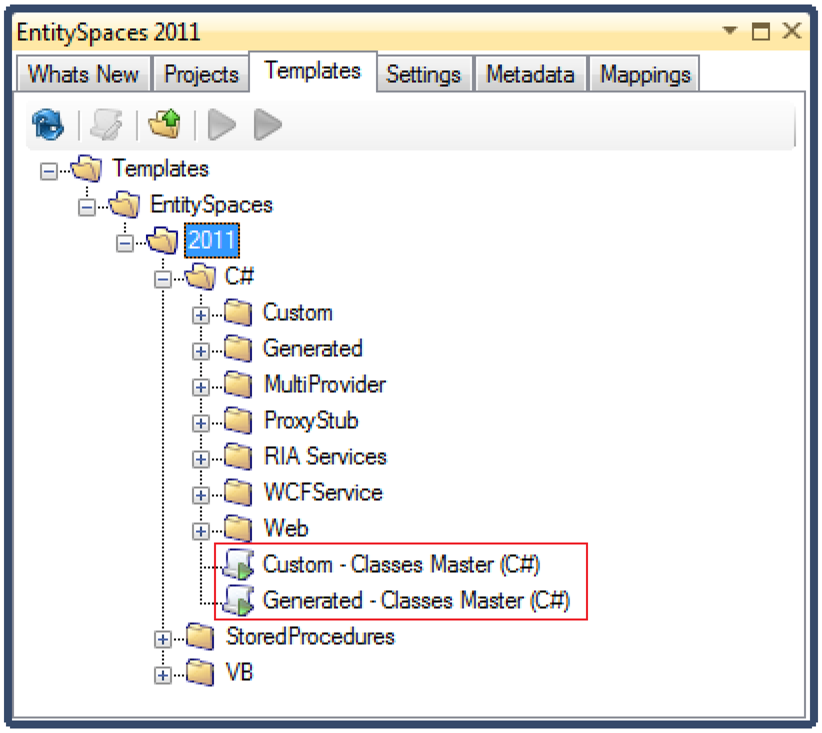
```
Solution 'MyEntitySpacesWebsite' (1 project)
C:\MyEntitySpacesWebsite\
    App_Code
        Custom
        Generated
            Categories.cs
            CustomerCustomerDemo.cs
            CustomerDemographics.cs
            Customers.cs
            Employees.cs
            EmployeeTerritories.cs
            OrderDetails.cs
            Orders.cs
            Products.cs
            Region.cs
            Shippers.cs
            Suppliers.cs
            Territories.cs
    App_Data
    Default.aspx
```

THE TEMPLATES TAB

There are two templates you need to execute to generate your EntitySpaces classes. It only takes a few seconds. The two templates you need to run are the:

1. Custom - Classes Master
2. Generated - Classes Master

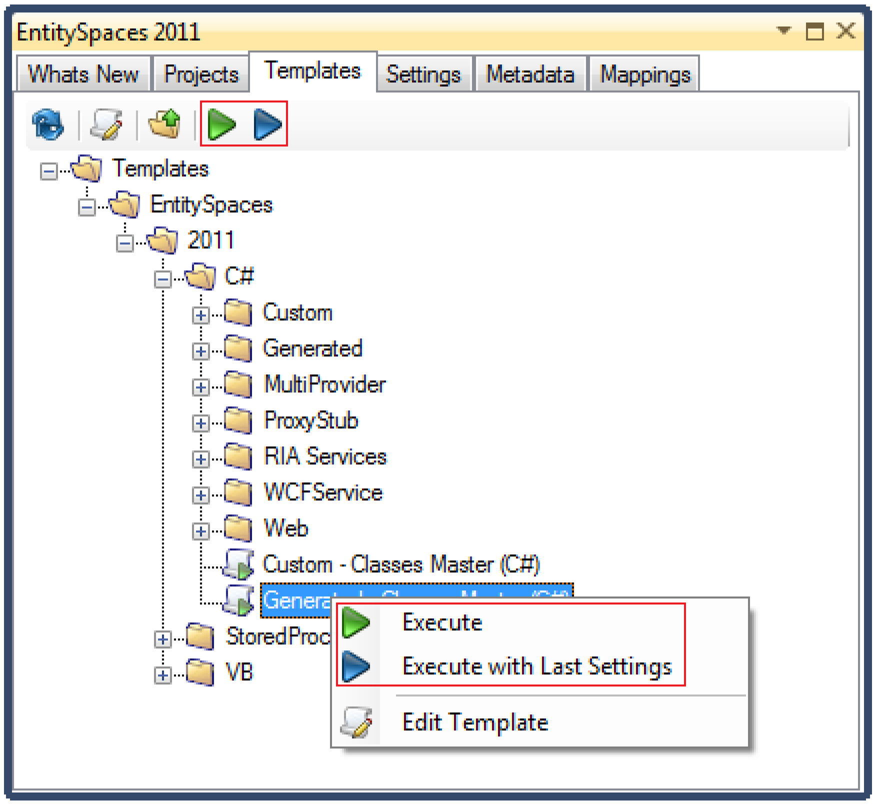
They are available for C# or VB developers. The screen below shows the C# templates.



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

EXECUTING THE ENTITYSPACES TEMPLATES

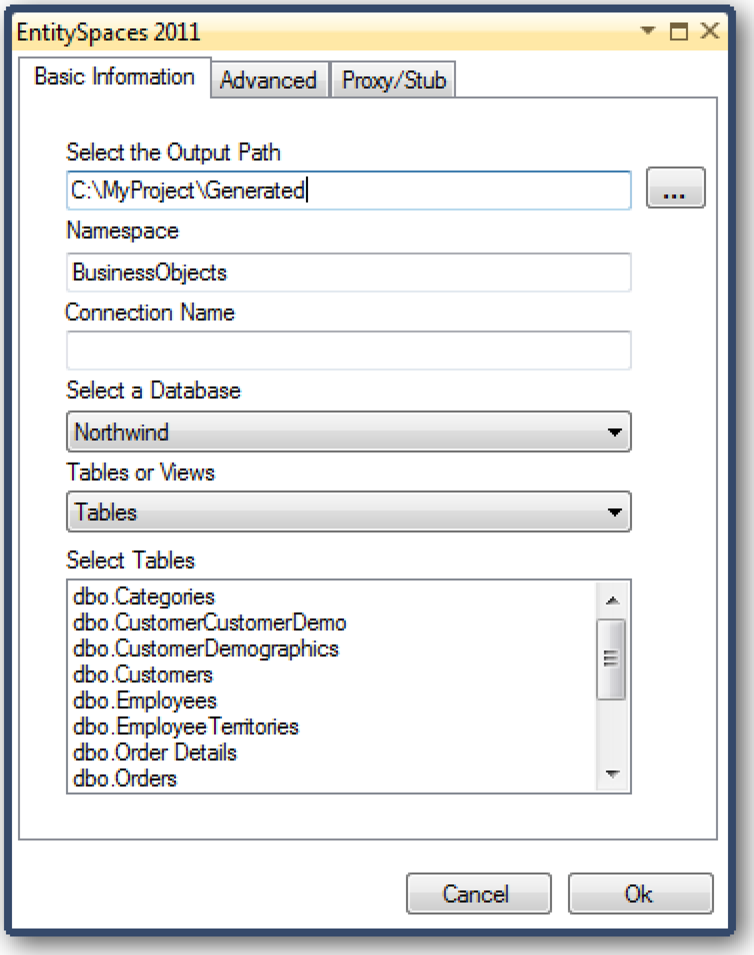
To execute a template, you can either use the “Green” or “Blue” arrows on the toolbar, or on the right-mouse context menu. The “Green” arrow always brings up the template with the default settings. The “Blue” arrow brings up the template user interface with the settings set as they were the last time you ran it (providing you have not closed Visual Studio or the standalone app, if you are using that).



Once you execute the template, you will be presented with the template user interface.

RUNNING THE GENERATED CLASSES MASTER TEMPLATE

The “Generated Classes Master” template is the main workhorse when it comes to creating your EntitySpaces Classes. Notice below that “Generated” is tacked on automatically to our default path, which in this case was C:\MyProject\.



Select the Output Path: C:\MyProject\Generated

Namespace: BusinessObjects

Connection Name:

Select a Database: Northwind

Tables or Views: Tables

Select Tables:
dbo.Categories
dbo.CustomerCustomerDemo
dbo.CustomerDemographics
dbo.Customers
dbo.Employees
dbo.EmployeeTerritories
dbo.Order Details
dbo.Orders

The default namespace for the generated classes is “BusinessObjects”. You can change this if you like.

You can use the Connection Name to make the classes you are about to generate target a particular database. However, we recommend that beginners leave this blank and your classes will use the default connection that you will define later in your config file.

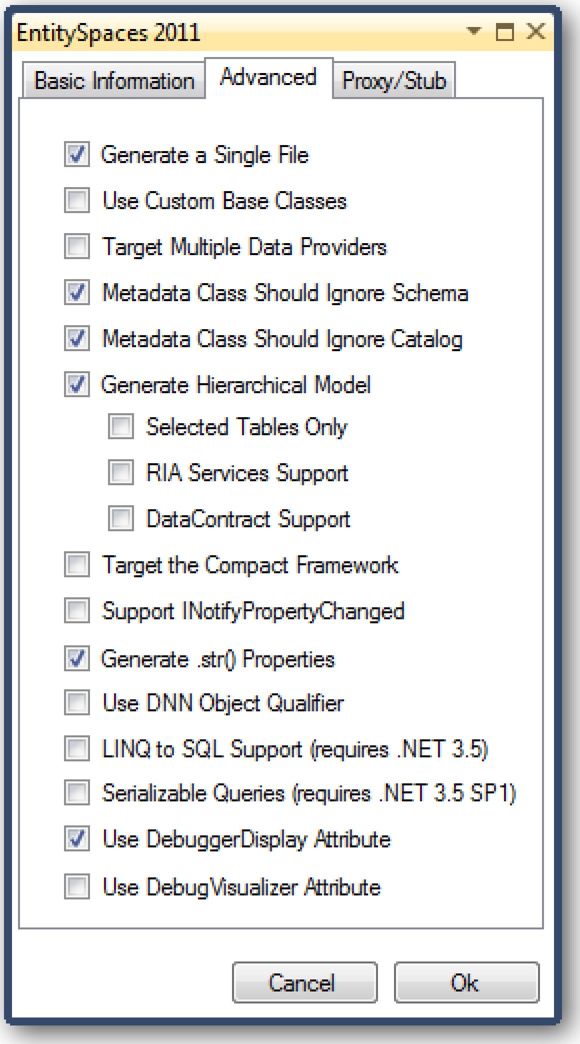
You can pick your database here as well. This is driven by the default catalog if your database supports one in your connection string.

For the “Object Type” you can select Tables or Views.

Finally, you can select one or all of the tables. Use the standard Click/Shift-Click/Ctrl-Click techniques for choosing multiple tables.

Before you press the “OK” button let’s switch to the Advanced Tab and make sure the settings are setup for our particular application.

The Advanced Options tab has many settings that allow you to customize your EntitySpaces application.


EntitySpaces 2011

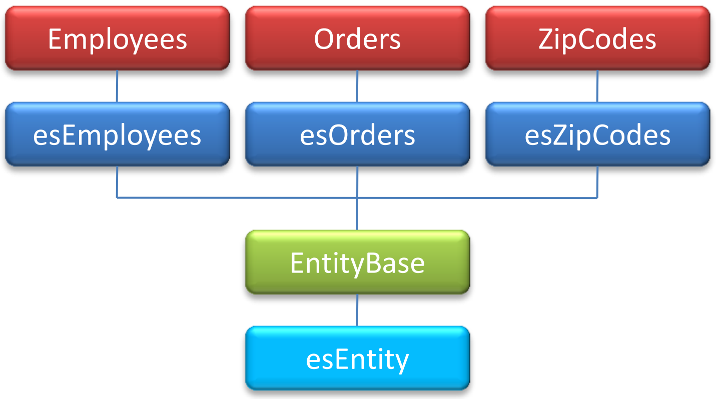
- [✓] Generate a Single File
- [ ] Use Custom Base Classes
- [ ] Target Multiple Data Providers
- [✓] Metadata Class Should Ignore Schema
- [✓] Metadata Class Should Ignore Catalog
- [✓] Generate Hierarchical Model
  - [ ] Selected Tables Only
  - [ ] RIA Services Support
  - [ ] DataContract Support
- [ ] 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

❖ Generate a Single File

This combines all of the generated classes from one table into a single file. We recommend that you check this as it makes maintaining a project much easier. You never hand edit these files anyway.

❖ Use Custom Base Class

If you are just getting started, you should probably not use this option. You can always check it and regenerate later if you need this functionality. If you need to supply base functionality across all of your generated classes, this option will allow you to do just that.



EntityBase in the above diagram is the Custom Class. There is also one for the Collection and Query classes. The empty "Custom Base Classes" are not generated. Prototypes are installed to your EntitySpaces Program Folder. You would need to add them to your project if you check this option.

❖ Target Multiple Data Providers

This is used when you intend to run the same code against multiple types of databases, for instance, Microsoft SQL Server and MySQL. This option is beyond the scope of this document. More information can be found on the Documentation site.

❖ Metadata Class should Ignore Schema

We recommend that you check this for SQL Server and SQL CE. This allows you to drive the Schema from your actual connection string. If un-checked, this is hard-coded into your Metadata class.

❖ Metadata Class should Ignore Catalog

We recommend that you check this for SQL Server and SQL CE. This allows you to drive the Catalog name from your actual connection string. If un-checked, this is hard-coded into your Metadata class.

- Generate Hierarchical Model

This causes your generated classes to implement a fully functional hierarchical model. If you do check this, you will most likely need to generate classes for all of your tables, since they will reference each other. This is no problem since you can select all tables and generate at once. For more on the Hierarchical Model see the Documentation site.

- Selected Tables Only

This causes your hierarchical object model to include hierarchical properties for the tables selected at code generation time. For more on this see the EntitySpaces Hierarchical Object Model Improvements on our blog.

- RIA Services Support

Check this if you want to use your generated entities in a RIA Services application.

- DataContract Support

Check this if you want to use your generated entities in a WCF Services application. This enables object graphs to be transmitted and received via WCF Services.

- Target the Compact Framework

Check this if your application is targeting the Compact Framework. When this is checked, certain features such as serialization are not available.

- Support INotifyPropertyChanged

By checking this checkbox, your EntitySpaces classes will support two-way data binding.

- Generate .str() properties

These are truly only needed if you plan on using the esDataSource control in an ASP.NET scenario. These special properties allow you to work with all of your properties as if they were strings, for instance:

```
Employees entity = new Employees();
entity.str().Age = txtboxAge.Text;  // Will convert the text to an integer for you automatically
```

- Use DNN Object Qualifier

If you are using EntitySpaces in place of the DotNetNuke DAL, then you will want to check this checkbox.

- LINQ to SQL Support (requires .NET 3.5)

This allows you to make LINQ to SQL queries that load EntitySpaces objects. See the help on esEntity.Load() and esEntityCollection.Load() in the compiled help files located in your EntitySpaces Start Menu folder.

- Serializable Queries (requires .NET 3.5 SP1)

If you plan to serialize DynamicQueries in a Silverlight or WCF application, you must check this checkbox.

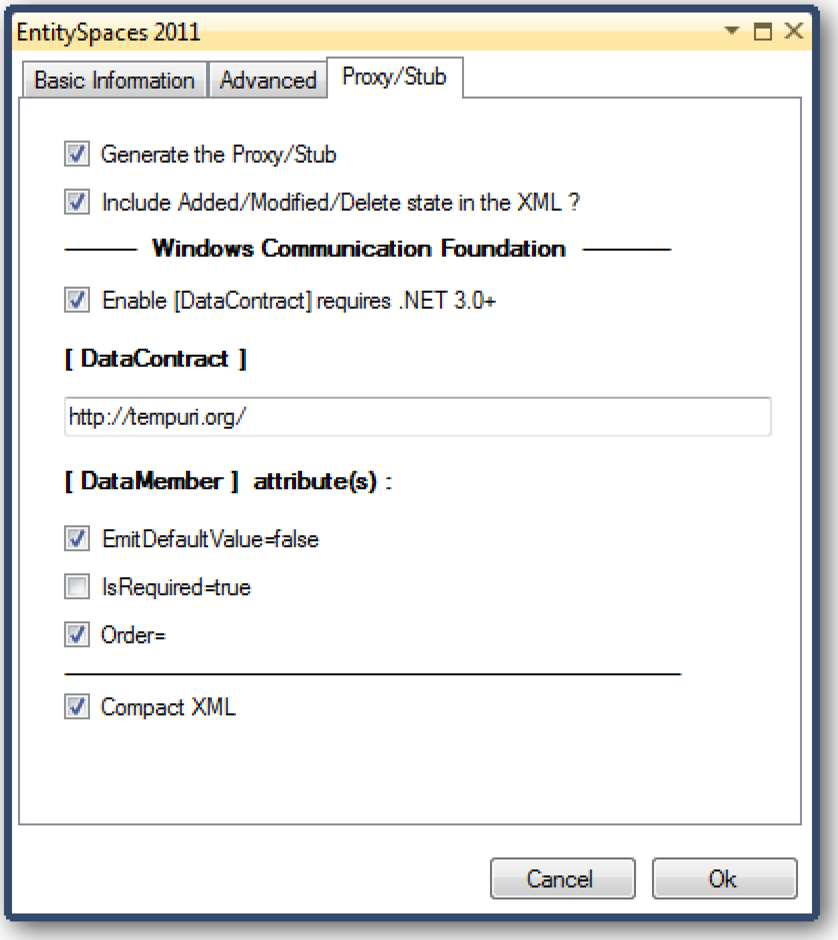
- Use DebuggerDisplay Attribute

We recommend that you leave this on. This makes it nice when inspecting EntitySpaces objects while debugging.

❖ Use DebuggerVisualizer Attribute

Check this if you want to use the EntitySpaces Debugger Visualizer. If you check this, you will need to include a reference to the EntitySpaces.DebuggerVisualizer.dll assembly in your project. When you inspect an EntitySpaces collection, the collection will be displayed in a popup grid. The source to the EntitySpaces.DebuggerVisualizer.dll is included in your C:\Program Files\EntitySpaces 2012\Runtimes folder.

The Proxy/Stub tab has some optional settings that add Web Services support to your EntitySpaces application.



- [✓] Generate the Proxy/Stub
- [✓] Include Added/Modified/Delete state in the XML ?

Windows Communication Foundation

- [✓] Enable [DataContract] requires .NET 3.0+

[DataContract]

http://tempuri.org/

[DataMember] attribute(s) :

- [✓] EmitDefaultValue=false
- [ ] IsRequired=true
- [✓] Order=
- [✓] Compact XML

❖ Generate the Proxy/Stub

If checked, you can use the proxy stub for any form of communication in which you want EntitySpaces on both sides of the conversation. This includes WebServices.

❖ Include Added/Modified/Delete state in the XML

This is checked by default, and in most cases, you should always leave it checked.

❖ Enable [DataContract] (requires .NET 3.0+)

Only check this if you intend to use the proxy's for Windows Communication Foundation scenarios.

❖ [DataContract]

A data contract is a formal agreement between a service and a client that abstractly describes the data to be exchanged. That is, in order to communicate, the client and the service do not have to share the same types, only the same data contracts. A data contract precisely defines, for each parameter or return type, what data is serialized (turned into XML) in order to be exchanged. The default namespace is http://tempuri.org/.

❖ [DataMember] attribute(s):

- EmitDefaultValue=false

  Gets or sets a value that specifies whether to serialize the default value for a field or property being serialized. We recommend you always have this checked.

- IsRequired=true

  Indicates to the serialization engine that the member must be present when reading or deserializing.

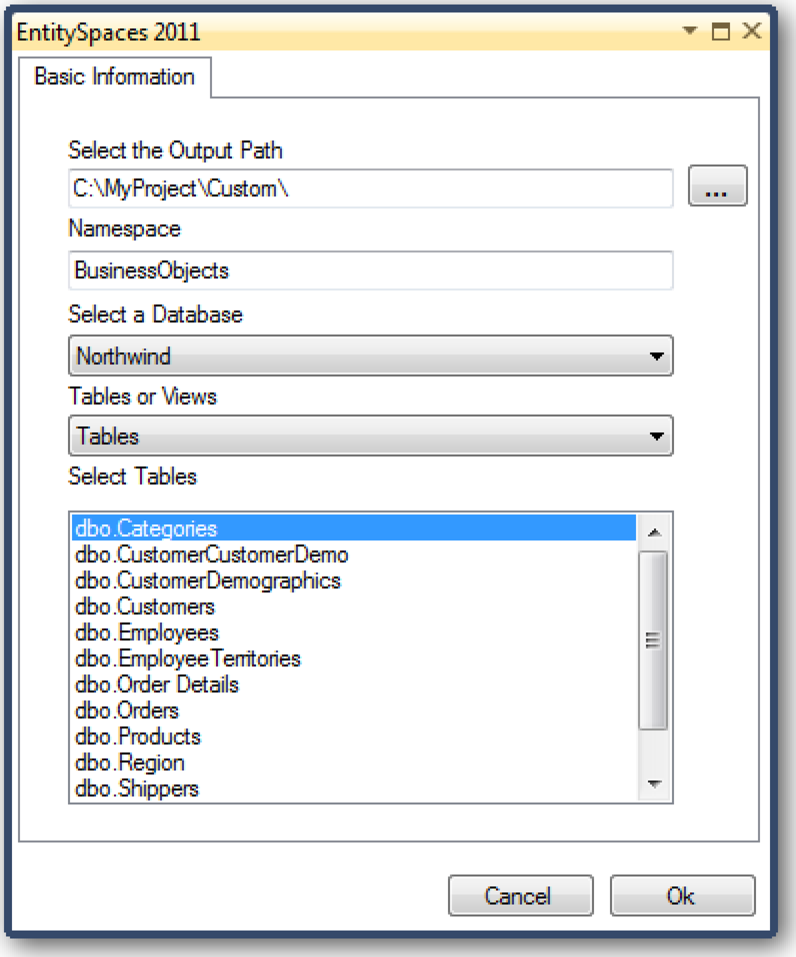
- Order=

  Sets the order of serialization and deserialization of a member. This is also the ordinal of the column in the table or view.

❖ Compact XML

This option will make your property names very tiny and greatly reduce your packet size over the wire, this does not affect your API. Your property names are still the full property names. However, in order to use this you need to use both our Server and Client proxies and not the Visual Studio proxies. This option is valid for WCF and Silverlight and only works with DataContract serialization.

RUNNING THE CUSTOM CLASSES MASTER TEMPLATE



Select the Output Path: C:\MyProject\Custom\
Namespace: BusinessObjects
Select a Database: Northwind
Tables or Views: Tables
Select Tables:
dbo.Categories
dbo.CustomerCustomerDemo
dbo.CustomerDemographics
dbo.Customers
dbo.Employees
dbo.EmployeeTerritories
dbo.Order Details
dbo.Orders
dbo.Products
dbo.Region
dbo.Shippers

The Custom Master template is very simple. There are no options. You run this one time only. The Custom Classes are meant to house your custom business logic. In them, you can override most of the underlying EntitySpaces infrastructure including the generated properties. You also have access to all kinds of powerful protected methods.

Notice that our Output Path has "Custom" tacked onto the default path of C:\MyProject\ that was specified on the Settings tab.

The Namespace should be the same as the one you specified for the Generated Master template. The default is "BusinessObjects".

Select your Database. Choose whether you want to generate against Tables or Views. Select all of the tables or views desired. Then click "OK".

Remember, you only need to run this once. One of the nice features of EntitySpaces is that when you look in your custom classes you only see the true business logic, not all of the underlying infrastructure.

This also means that you are free to regenerate your generated classes whenever you change your database schema without the fear of losing custom code. No worries, they are in separate files.

The output for this template is set to not over-write files with the same name. This prevents you from accidentally wiping out any code you have added to your custom classes. The Custom classes are partial classes to the Generated classes. Both the EntitySpaces Custom and Generated classes must be housed in the same assembly.

THE ENTITYSPACES CONFIGURATION FILE

EntitySpaces allows for both configuration file based settings or configless setup. We are going to look at using a configuration file first. This is a sample app.config file.

THE CONFIG FILE

```
<?xml version="1.0"?>
    <configuration>
      <configSections>
            <sectionGroup name="EntitySpaces"
                    type="EntitySpaces.Interfaces.esConfigSettings,
                    EntitySpaces.Core">
                 <section name="connectionInfo"
                     type="EntitySpaces.Interfaces.esConfigSettings,
                     EntitySpaces.Interfaces"
                     allowLocation="true"
                     allowDefinition="Everywhere"
                     restartOnExternalChanges="true"/>
           </sectionGroup>
    </configSections>

    <EntitySpaces>
      <connectionInfo default="SqlDevServer">
            <connections>
                <add name="SqlDevServer"
                     providerMetadataKey="esDefault"
                     sqlAccessType="DynamicSQL"
                     provider="EntitySpaces.SqlClientProvider"
                     providerClass="DataProvider"
                     connectionString="User ID=sa; . . . "
                     databaseVersion="2005" />
            </connections>
        </connectionInfo>
    </EntitySpaces>
</configuration>
```

The first part in the **configSections** merely registers the ConfigurationSection class that reads the **EntitySpaces** section down below. It is the EntitySpaces section that we are interested in. Let’s look at the setup.

First, we have the connectionInfo element with the default set to “SqlDevServer”. The default specified must match the “name” of one of your connections. You can register any number of connections in the connection section. This config file only registers a single connection. Connections are registered via the **add** element.

- The name is merely a nickname that you can reference the connection by in code. You can “name” them anything you like, but each one must have a unique “name”.
- The providerMetadataKey tells EntitySpaces which MetadataMap to use. This will always be “esDefault” for any classes generated using the Generated Master template, regardless of the back-end database. (Note: Only change this when working with multiple databases such as Microsoft SQL Server and MySQL within the same application, and generating for one of them with the MetadataMap template. This feature is not covered in this document.)
- The sqlAccessType is either “DynamicSQL” or “StoredProcedure” and this governs your CRUD operations CREATE/READ/UPDATE/DELETE.
- The provider indicates which EntitySpaces data provider you want to use with this connection.

- The providerClass is where you indicate whether you want to run with Distributed Transactions or ADO.NET connection based transactions. You use “DataProvider” for ADO.NET connections and “DataProviderEnterprise” for COM+ distributed transactions. Most of the time you will use “DataProvider”.
- The connectionString is the connection string for your database. Be careful if you copy the connection string from the Settings Connections tab in the EntitySpaces UI. EntitySpaces uses ADO.NET within your application, not OLEDB. For example, you would need to delete "Provider=SQLOLEDB.1;".
- The databaseVersion is used only for SQL Server currently. If you are using SQL Server 2005 or 2008, you need to specify this in order for timestamps to be handled correctly. Use databaseVersion="2005" for both SQL Server 2005 and 2008. Do not use databaseVersion="2008".
- The schema can be used to set the database schema for your connection string. **This is optional**.
- The catalog can be used to set the database catalog for your connection string. **This is optional.** This can often can be indicated in the raw connection string such as “*Initial Catalog=Northwind*”.

CONFIGLESS EXECUTION

EntitySpaces also allows for configless execution, that is, no .config file required. You can do it all in code. In fact, certain environments such as SQL CE and DotNetNuke really need this feature. Below we are registering the same settings that are possible in an app.config or web.config file. Note that you only need to do this once for the life of your application. You must remove all EntitySpaces sections from the app/web .config file to use configless execution. You cannot use both.

```
using EntitySpaces.Interfaces;

// --- Manually register a connection
esConnectionElement conn = new esConnectionElement();
conn.Name = "SqlCe";
conn.ConnectionString = @"Data Source=C:\Tests\SqlCe\AggregateDB.sdf;";
conn.Provider = "EntitySpaces.SqlClientProvider.Ce";
conn.ProviderClass = "DataProvider";
conn.SqlAccessType = esSqlAccessType.DynamicSQL;
conn.ProviderMetadataKey = "esDefault";
conn.DatabaseVersion = "2005";

// --- Assign the Default Connection ---
esConfigSettings.ConnectionInfo.Connections.Add(conn);
esConfigSettings.ConnectionInfo.Default = "SqlCe";
```

STANDARD .NET CONNECTION STRINGS

Whether you use a config file, or configless EntitySpaces, you can reference standard .NET connection strings in your config file from the EntitySpaces configuration. The "AppSettings:" keyword, below, tells EntitySpaces to use an existing, standard connectionString named "SqlProductionServer" that you have in your config file. You can also use “ConnectionStrings:” in place of "AppSettings:".

```
conn.ConnectionString = @"AppSettings:SqlProductionServer"
```

DOTNETNUKE CONFIGURATION SETTINGS

We recommend the configless method for DotNetNuke. Here is the suggested sample. This code snippet includes registering the Loader, which is covered in the next section.

```
using EntitySpaces.Interfaces;

private void SetConnection()
{
    if (esConfigSettings.ConnectionInfo.Default != "SiteSqlServer")
    {
        esConfigSettings ConnectionInfoSettings = esConfigSettings.ConnectionInfo;
        foreach (esConnectionElement connection in ConnectionInfoSettings.Connections)
        {
            //if there is a SiteSqlServer in es connections set it default
            if (connection.Name == "SiteSqlServer")
            {
                   esConfigSettings.ConnectionInfo.Default = connection.Name;
                   return;
            }
        }

        // No SiteSqlServer found grab dnn cnn string and create

        string dnnConnection =
          ConfigurationManager.ConnectionStrings["SiteSqlServer"].ConnectionString;

        // Manually register a connection
        esConnectionElement conn = new esConnectionElement();
        conn.ConnectionString = dnnConnection;
        conn.Name = "SiteSqlServer";
        conn.Provider = "EntitySpaces.SqlClientProvider";
        conn.ProviderClass = "DataProvider";
        conn.SqlAccessType = esSqlAccessType.DynamicSQL;
        conn.ProviderMetadataKey = "esDefault";
        conn.DatabaseVersion = "2005";

        // Assign the Default Connection
        esConfigSettings.ConnectionInfo.Connections.Add(conn);
        esConfigSettings.ConnectionInfo.Default = "SiteSqlServer";

        // Register the Loader
        esProviderFactory.Factory = new EntitySpaces.LoaderMT.esDataProviderFactory();
    }
}
```

REGISTERING THE LOADER

The Loader is a simple class that loads the EntitySpaces data providers. EntitySpaces applications are loosely coupled to the data providers, and therefore, can switch between databases at runtime.

The loaders enumerate through all of your registered connection entries loading the appropriate EntitySpaces Providers to support each connection. There are two forms of the loader, the non-medium trust, and the medium trust version. No matter which type of loader you use, you will only need to ship the EntitySpaces Provider(s) that you actually use. Here are the two loaders.

EntitySpaces.LoaderMT = medium trust (no reflection)
EntitySpaces.Loader = (uses reflection)

It is our recommendation that you use the non-medium trust loader. The Medium Trust loader copies all the EntitySpaces Providers into your bin folder at compile time. You only need to deploy the Provider(s) used by your application however. Regardless of the Loader chosen, your application requires a one-time call to assign the proper loader at program startup. Here is how we recommend that you do this in your code.

C# Windows Forms Application

This is how our EntitySpaces Demo Application initializes the Loader.

```
using EntitySpaces.Interfaces;

namespace EntitySpacesDemo
{
   static class Program
   {
      [STAThread]
      static void Main()
      {
         esProviderFactory.Factory =
            new EntitySpaces.LoaderMT.esDataProviderFactory();

         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Demo());
      }
   }
}
```

VB.NET Windows Forms Application

Solutions created for VB.NET generally do not include a Program.vb file. You can add one for program initialization code, but this involves editing the Project options, disabling the Application Framework, and setting the Startup Form. It is simpler just to add the Loader initialization to your Startup Form's constructor.

```
Imports EntitySpaces.Interfaces

Public Class Form1

  Public Sub New()

   ' This call is required by the Windows Form Designer.
   InitializeComponent()
   ' Add any initialization after the InitializeComponent() call.

   esProviderFactory.Factory = _
      New EntitySpaces.LoaderMT.esDataProviderFactory()

  End Sub

End Class
```

ASP.NET Application

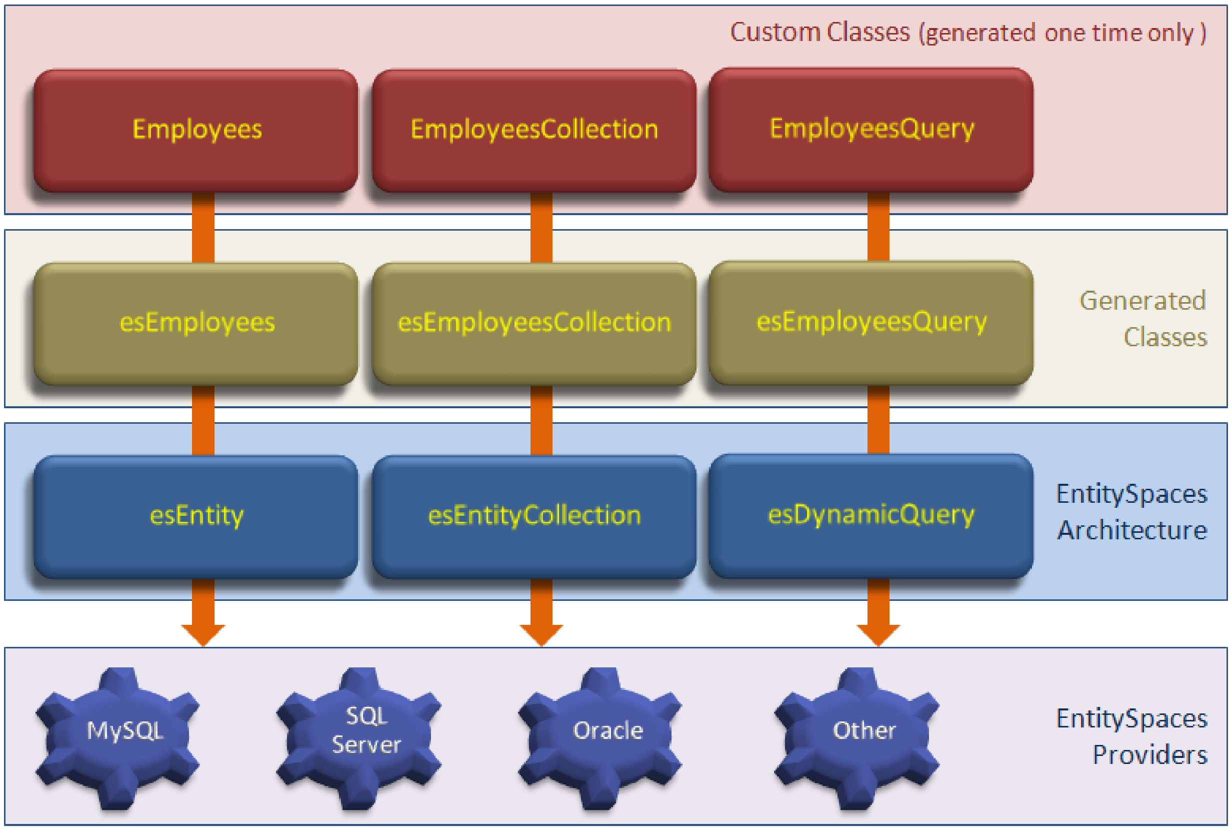
For ASP.NET, the best approach is probably to add a "Global Application Class" or Global.asax file. Here is an example.

```
<%@ Application Language="C#" %>
   <script runat="server">

      void Application_Start(object sender, EventArgs e)
      {
         EntitySpaces.Interfaces.esProviderFactory.Factory =
            new EntitySpaces.LoaderMT.esDataProviderFactory();
      }

   </script>
```

HIGH LEVEL ARCHITECTURAL OVERVIEW

The following is a very high-level overview of the EntitySpaces Architecture. Reading through this section should be enough to get you started with the EntitySpaces API.

- The EntitySpaces DataProviders are loosely coupled to your application.
- The EntitySpaces Architecture provides a wealth of functionality that your classes will leverage.
- The Generated Classes should be re-generated whenever your database schema changes, and are never hand edited.
- The Custom Classes are generated one time only. This is where you add your actual business logic.



Note that the above diagram is also an inheritance diagram. In your Custom classes, you can override most of the generated and base methods and properties to further customize your architecture.

HIGH LEVEL OBJECT USAGE

The EntitySpaces Architecture contains Collections full of Entities; however, the Entities can live on their own as well.

COLLECTIONS

Collections can both load and save many rows of data. Each row is represented with its single entity counterpart.

LOADING A COLLECTION

This sample loads all Employees in the database.

```
EmployeesCollection coll = new EmployeesCollection();
if(coll.LoadAll())
{
    foreach (Employees emp in coll)
    {
        Console.WriteLine(emp.LastName);
    }
}
```

QUERYING A COLLECTION

This sample loads all Employees whose last name starts with “Smi”.

```
EmployeesCollection coll = new EmployeesCollection();
coll.Query.Where(coll.Query.LastName.Like("Smi%")
if(coll.Query.Load())
{
    foreach (Employees emp in coll)
    {
        Console.WriteLine(emp.LastName);
    }
}
```

SAVING A COLLECTION

Here we are transferring all of our Employees to Indianapolis. You never call Save on the single entity when it lives in a collection. Save will commit all modified, added, and deleted rows.

```
EmployeesCollection coll = new EmployeesCollection();
if(coll.LoadAll())
{
    foreach (Employees emp in coll)
    {
        emp.City = "Indianapolis";
    }
    coll.Save();
}
```

ADDING A NEW RECORD THROUGH A COLLECTION

Here we are adding two new Employees and saving them both.

```
EmployeesCollection coll = new EmployeesCollection();

Employees emp = coll.AddNew();
emp.FirstName = "Joe";
emp.LastName = "Smith";

emp = coll.AddNew();
emp.FirstName = "Sue";
emp.LastName = "Smith";

coll.Save();
```

DELETING A RECORD THROUGH A COLLECTION

This example demonstrates the use of FindByPrimaryKey to locate a particular Employee in the collection, then marks it as deleted, and finally saves the collection.

```
EmployeesCollection coll = new EmployeesCollection();
coll.LoadAll();

Employees emp = coll.FindByPrimaryKey(41);
if(emp != null)
{
    emp.MarkAsDeleted();
    coll.Save();
}
```

If you need to delete all of the records from a table, you use the MarkAllAsDeleted method as follows:

```
EmployeesCollection coll = new EmployeesCollection();
coll.LoadAll();
coll.MarkAllAsDeleted();
coll.Save();
```

NOTE: Be very wary of foreaching through a collection and calling MarkAsDeleted. The .NET framework does not allow anything inside a foreach loop that changes the order of the iterated collection.

ENTITIES

Although single entities can live inside of collections, they can also live on their own. A single Entity can never hold more than one record. If you attempt to load more than one record, an exception is thrown.

LOADING A SINGLE ENTITY

This sample simply loads an Employee by its primary key. Don't let the plural "Employees" fool you into thinking this is a collection. This sample uses the Microsoft SQL Northwind Database and the table is named "Employees" and not "Employee".

```
Employees emp = new Employees();
if (emp.LoadByPrimaryKey(42))
{
    Console.WriteLine(emp.LastName);
}
```

QUERYING AN ENTITY

Although this query could easily be done via the LoadByPrimaryKey method we are just demonstrating the query ability of the single object.

```
Employees emp = new Employees();
emp.Query.Where(emp.Query.EmployeeID == 37 || emp.Query.EmployeeID == 38);
if (emp.Query.Load())
{
    Console.WriteLine(emp.LastName);
}
```

SAVING AN ENTITY

Here we save a single entity, no collection needed.

```
Employees emp = new Employees();
if (emp.LoadByPrimaryKey(42))
{
    emp.LastName = "Smith";
    emp.Save();
}
```

DELETING AN ENTITY

Here we delete a single entity, no collection needed.

```
Employees emp = new Employees();
if (emp.LoadByPrimaryKey(42))
{
    emp.MarkAsDeleted();
    emp.Save();
}
```

THE ENTITYSPACES ASSEMBLIES

If you accepted the defaults during installation, as recommended, you will find the EntitySpaces assemblies in the following folders. Note that under 64 bit version of Windows the default installation path will be C:\Program Files (x86)\EntitySpaces 2012.

.NET 3.5 Assemblies

C:\Program Files\EntitySpaces 2012\Runtimes\.NET 3.5
C:\Program Files\EntitySpaces 2012\Runtimes\.NET 3.5\Web
C:\Program Files\EntitySpaces 2012\Runtimes\.NET 3.5\x64

.NET 4.0 Assemblies

C:\Program Files\EntitySpaces 2012\Runtimes\.NET 4.0
C:\Program Files\EntitySpaces 2012\Runtimes\.NET 4.0\Web
C:\Program Files\EntitySpaces 2012\Runtimes\.NET 4.0\x64

Silverlight Assemblies

C:\Program Files\EntitySpaces 2012\Runtimes\Silverlight\3
C:\Program Files\EntitySpaces 2012\Runtimes\Silverlight\4

Windows Phone 7

C:\Program Files\EntitySpaces 2012\Runtimes\Windows Phone 7

Compact Framework

C:\Program Files\EntitySpaces 2012\Runtimes\.NET CF 3.5

THE ENTITYSPACES ASSEMBLIES

The following is the list of EntitySpaces assemblies:

EntitySpaces.Core.dll	Contains the esEntity/esEntityCollection classes
EntitySpaces.Interfaces.dll	Providers link to this assembly
EntitySpaces.DynamicQuery.dll	Contains the DynamicQuery API
EntitySpaces.Loader.dll	Loader, uses reflection
EntitySpaces.LoaderMT.dll	Loader, medium trust support
EntitySpaces.Profiler.dll	Profiler, adds EntitySpaces Profiler support to your app
XDMessaging.dll	Profiler, adds EntitySpaces Profiler support to your app
EntitySpaces.MSAccessProvider.dll	Provider - Microsoft Access
EntitySpaces.MySqlClientProvider.dll	Provider - MySQL
EntitySpaces.OracleClientProvider.dll	Provider - Oracle
EntitySpaces.SqlClientProvider.dll	Provider - Microsoft SQL Server
EntitySpaces.SqlServerCeProvider.dll	Provider - Microsoft SQL CE 3.x desktop provider
EntitySpaces.SqlServerCe4Provider.dll	Provider - Microsoft SQL CE 4.x desktop provider
EntitySpaces.SybaseSqlAnywhereProvider.dll	Provider – Sybase SQL Anywhere 11
EntitySpaces.VistaDBProvider.dll	Provider – VistaDB 3.x
EntitySpaces.VistaDB4Provider.dll	Provider – VistaDB 4.x
EntitySpaces.Npgsql2Provider.dll	Provider - PostgreSQL 8.3 + Guid Support
EntitySpaces.SQLiteProvider.dll	Provider – SQLite 3.x
EntitySpaces.EffiProzProvider.dll	Provider - EffiProz
EntitySpaces.Web.Design.dll	esDataSource (design time support for ASP.NET)
EntitySpaces.Web.dll	esDataSource Control for ASP.NET
EntitySpaces.DebuggerVisualizer.dll	An optional DLL that will display collections when debugging

The following is the list of EntitySpaces assemblies for the Compact Framework:

EntitySpaces.Core.CF.dll	Contains the esEntity/esEntityCollection classes
EntitySpaces.Interfaces.CF.dll	Providers link to this assembly
EntitySpaces.DynamicQuery.CF.dll	Contains the DynamicQuery API
EntitySpaces.Loader.CF.dll	Loader, uses reflection
EntitySpaces.LoaderMT.CF.dll	Loader, medium trust support
EntitySpaces.SybaseSqlAnywhereProvider.CF.dll	Provider – Sybase SQL Anywhere 11
EntitySpaces.SqlServerCeProvider.CF.dll	Provider - Microsoft SQL CE 3.x
EntitySpaces.SQLiteProvider.CF.dll	Provider - SQLite 3.x
EntitySpaces.VistaDBProvider.CF.dll	Provider - VistaDB CE (3.x Only)

If you are developing an ASP.NET Application, you should bind to the EntitySpaces.Core.dll in the “Web” folder as it has better binding support for ASP.NET.