Instructions to Build CSLA .NET 3.6 and the ProjectTracker Sample

The following are basic instructions to get CSLA .NET and the `ProjectTracker` sample application built and running. These instructions are not meant to replace the Expert C# 2008 Business Objects book; that book remains the primary source of information regarding the framework and how to use it.

Prerequisites

CSLA .NET for Windows version 3.6 has the following requirements:

- Visual Studio 2008 Professional (or higher)
- Microsoft .NET Framework 3.5 SP1

You must install these prerequisites before attempting to build or use CSLA .NET 3.6.

Building the CSLA .NET Framework

The following instructions cover getting and building the CSLA .NET framework:

1. Download CSLA .NET for Windows and unpack the zip archive to your hard drive. It is recommended that you put the code into the following path structure:

   ```
   c:\Visual Studio Projects\csla\Source
   ```

   The result should be

   ```
   c:\Visual Studio Projects\csla\Source\...
   ```

   where all the code in the zip archive is below the `Source` folder.

   You can put the code elsewhere, but if you do, then you'll need to manually fix up all references to `Csla.dll` in the sample applications.

2. Open the `Csla` solution in Visual Studio 2008 and build the `Csla.sln` solution. This should create a debug build in `\bin\debug`.

   a. The framework solution includes a key file, because `Csla.dll` must have a strong name. If your organization has a pre-existing key store and a policy for managing key files, you may want to use your own key rather than the publicly available one provided with the framework.

   b. Please note that the copyright notices in the `Csla` solution should be left intact. This is one of the few requirements of the CSLA .NET License.

c. I recommend using a file reference to the debug build of `Csla.dll` when building your business applications. This helps reduce the chance of making accidental changes to the framework code, and can avoid some nasty issues with Visual Studio, especially when using custom controls from `Csla.dll`.

d. In general, I recommend using the local data portal proxy for development, and a remote data portal proxy (preferably `WcfProxy`) for testing.

   The local data portal proxy makes debugging and stepping through code much simpler. The remote data portal proxy for testing ensures that you haven't made a mistake in your business object code that prevents the data portal from properly functioning.

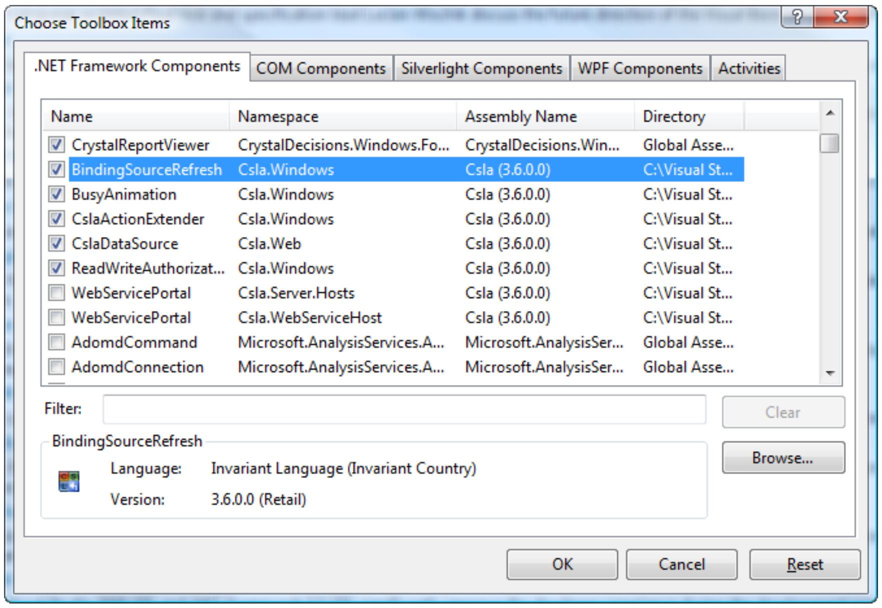
3. For production purposes you will want to create a release build of the framework, which you can do using normal Visual Studio procedures.

4. To make use of the custom controls in `Csla.dll` you will need to add them to your Visual Studio toolbox.

   a. Right-click on the toolbox in Visual Studio and select `Choose items…`.

   b. In the resulting dialog, click the Browse button and select `Csla.dll`.

   c. Sort the items by Assembly Name, then select the items shown in this figure:



Name	Namespace	Assembly Name	Directory
✓ CrystalReportViewer	CrystalDecisions.Windows.Fo...	CrystalDecisions.Win...	Global Asse...
✓ BindingSourceRefresh	Csla.Windows	Csla (3.6.0.0)	C:\Visual St...
✓ BusyAnimation	Csla.Windows	Csla (3.6.0.0)	C:\Visual St...
✓ CslaActionExtender	Csla.Windows	Csla (3.6.0.0)	C:\Visual St...
✓ CslaDataSource	Csla.Web	Csla (3.6.0.0)	C:\Visual St...
✓ ReadWriteAuthorizat...	Csla.Windows	Csla (3.6.0.0)	C:\Visual St...
☐ WebServicePortal	Csla.Server.Hosts	Csla (3.6.0.0)	C:\Visual St...
☐ WebServicePortal	Csla.WebServiceHost	Csla (3.6.0.0)	C:\Visual St...
☐ AdomdCommand	Microsoft.AnalysisServices.A...	Microsoft.AnalysisSer...	Global Asse...
☐ AdomdConnection	Microsoft.AnalysisServices.A...	Microsoft.AnalysisSer...	Global Asse...

BindingSourceRefresh
Language: Invariant Language (Invariant Country)
Version: 3.6.0.0 (Retail)

   Notice that the `WebServicePortal` items have been deselected, as they are not for direct use by business developers.

Building Sample Applications

The following instructions cover getting and building the CSLA .NET for Windows sample applications, focusing on the `ProjectTracker` application.

1. Download the CSLA .NET Samples and unpack the zip archive to your hard drive. It is recommended that you put the code into the following path structure:

   ```
   c:\Visual Studio Projects\csla\Samples
   ```

   The result should be

   ```
   c:\Visual Studio Projects\csla\Samples\...
   ```

   where all the code in the zip archive is below the `Samples` folder.

   You can put the code elsewhere, but if you do, then you'll need to manually fix up all references to `Csla.dll` in the sample applications. In that case you'll also need to fix up all database connection strings in all `app.config` and `web.config` files.

2. Most of the sample apps can be directly built and run in Visual Studio 2008. Please note that not all sample apps are tested with every release, and many are designed to test or illustrate very narrow areas of CSLA .NET functionality.

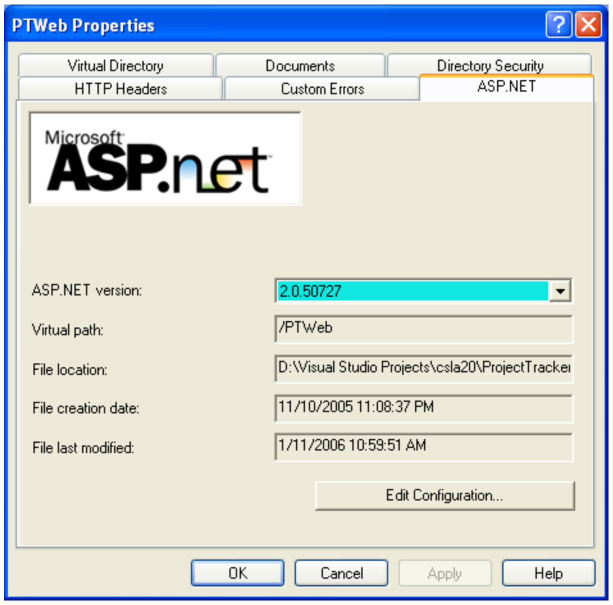
   Only `ProjectTracker` is an "official" demo application.

3. Several of the `ProjectTracker` UI applications have the data portal configured for a 3-tier model, communicating with a WCF data portal service. This WCF service is hosted in the ASP.NET Development Web Server (Cassini) and should work without any effort. If you do have difficulties, ensure that the port number of the `WcfHost` web site matches the port in each UI application's `system.serviceModel` endpoint definition in `app.config` or `web.config`.

4. The `ProjectTracker` Web Forms UI (`PTWeb`), WCF service interface (`PTWcfService`) and Web Services interface (`PTWebService`) web sites will not run within the ASP.NET Development Web Server (Cassini) due to a "feature" of Cassini. Due to this, you must run these web sites in IIS.

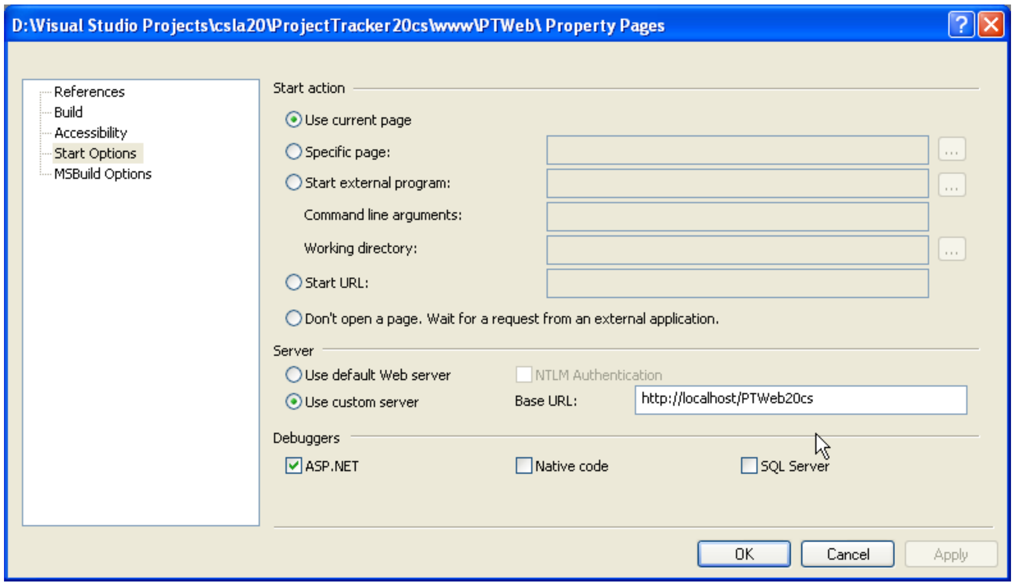
   a. Using the Internet Information Services console, create virtual roots that point to the `PTWeb`, `PTWcfService` and `PTWebService` directories.

   b. Make sure to set the virtual roots to use ASP.NET version 2.0. This is done by setting the virtual root's properties.



ASP.NET version: 2.0.50727
Virtual path: /PTWeb
File location: D:\Visual Studio Projects\csla20\ProjectTracker
File creation date: 11/10/2005 11:08:37 PM
File last modified: 1/11/2006 10:59:51 AM

c. Set the Start Options for PTWeb, PTWcfService and PTWebService to use a custom server, pointing to the virtual root URLs you just set up.



Start action: [✓ Use current page] Specific page: Start external program: Start URL: Don't open a page. Wait for a request from an external application.
Server: Use default Web server [✓ Use custom server]; NTLM Authentication ☐; Base URL: http://localhost/PTWeb20cs
Debuggers: [✓ ASP.NET] ☐ Native code ☐ SQL Server

You may optionally set a specific page (such as Default.aspx) as the startup page, though this is not necessary. The PTWeb pages include functionality to redirect the user to a “safe” page if they attempt to start on a page without proper authorization.

d. The PTWcfClient project is a client for PTWcfService. You’ll need to make sure its app.config file contains the URL for the virtual root you created for `PTWcfService` earlier.

e. The `PTServiceClient` project is a client for `PTWebService`. You'll need to make sure its `app.config` file contains the URL for the virtual root you created for `PTWebService` earlier.

5. Build the `ProjectTracker` solution. You should now be able to run the various UI projects.

   a. To log into the application, you can use

Username	**Password**	
pm	pm	Log in as a project manager
admin	admin	Log in as an admin user
user	user	Log in as a normal user