**Instructions to Build CSLA .NET 2.0 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 VB 2005 Business Objects or Expert C# 2005 Business Objects books; those books remain the primary source of information regarding the framework and how to use it.

1. Download the code from this page and unpack the zip archive to your hard drive. It is recommended that you put the code somewhere other than in your My Documents area, because the ASP.NET user account under IIS will need access to the directory.

    Specifically, you need to put the code into a directory that is accessible by the ASP.NET user account under IIS (assuming you want to run the web UI or Web Service interface). Alternately, you can put the code where you want and set the appropriate NTFS permissions to grant the ASP.NET user account appropriate access.

2. If you plan to run the Web Forms UI or Web Services interface, you must do the following: using the Internet Information Services console, create virtual roots that point to both the `PTWeb` and `PTWebService` directories. You'll need to choose whether to use VB or C#, or name the virtual roots using language-specific names (like `PTWebvb` and `PTWebcs`). The reason for this step is discussed in Chapters 10 and 11 and also here.

    a. Make sure to set the virtual roots to use ASP.NET version 2.0. This is done by setting the virtual root's properties in Windows XP Pro or Windows Server 2003. Under Windows 2000 this is done through `web.config`.
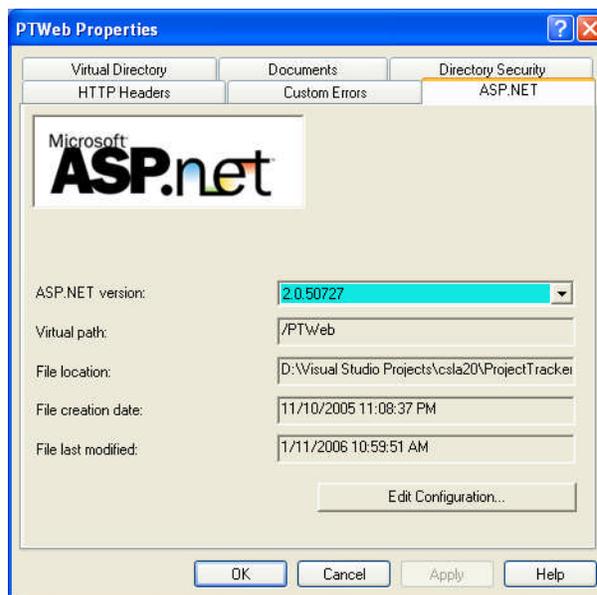
3. Open the `Csla` solution in Visual Studio 2005 and build the solution. There are separate solutions for VB and C# and you can build either or both as you choose. The VB `ProjectTracker` uses the VB framework, the C# `ProjectTracker` uses the C# framework, so make sure to build the right framework(s) for the sample you intend to build.

    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. Though `Csla.dll` includes a `ServicedComponent` and references `System.Enterprise.dll`, it *should not* be registered with COM+. This is discussed in Chapter 4.

    c. 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](#).

4. Close the `Csla` solution and open the `ProjectTracker` solution for your chosen programming language. DO NOT BUILD the project at this time.

5. You may need to remove and re-add the reference to `Csla.dll`. Normally this isn't necessary, but if you've altered the relative directory structures while unpacking the zip file the reference may have been lost.

6. Update the connection strings in each config file to correspond to the location where you unzipped the files. The `ProjectTracker` sample is designed to use SQL Server 2005 Express, and database files are included in the download. SQL Server 2005 Express allows an application to refer to a database directly by its file name, and that is how `ProjectTracker` is configured.

    Most projects in the solution have either a `web.config`, `app.config` or `application.config` file that you'll need to update.

    a. To start with, I *strongly* recommend using the local data portal proxy as discussed in Chapter 4 and in each of the interface chapters (Chapters 9-11). Once you have the code running with that proxy, you can branch out to the remote data portal proxies from Chapter 4.

    To use a remote data portal through Remoting, Enterprise Services or Web Services please refer to Chapter 12 where you'll find complete instructions on setting up each type of application server.

    b. If you don't plan to run a given UI (`PTWin`, `PTWeb` or `PTWebService`) you can ignore the config file in that UI project.

c. If you don't plan to use a given application server host (`RemotingHost`, `WebServicesHost` or `EnterpriseServicesHost`) you can ignore the config file in that project.

d. The `PTWin` client can use the database files directly, just make sure to change the path to the files in `app.config` before building the solution.

e. The `PTWeb` and `PTWebService` web sites typically won't have direct access to the database files. You can either copy the files to the `App_Data` directories in each project, or configure those projects to use a remote data portal server. I typically use the Enterprise Services data portal channel for `PTWeb` and the Remoting channel for `PTWebService` (due to the security complexity with calling a COM+ component from a web service). If you use the Enterprise Services channel, make sure to change the database connection strings in `application.config` as discussed in Chapter 12.

f. If you want to use an external database (like a pre-existing SQL Server 2005 database) that is fine, just change the connection strings in the config files accordingly.

g. If you want to use SQL Server 2000 make sure to read this.

7. The `PTWeb` and `PTWebService` projects are configured to use the `EnterpriseServicesProxy` within the data portal. Before you can run these projects you need to start a <u>Visual Studio 2005 Command Prompt</u> window and run `regsvcs.exe` to register the `EnterpriseServicesHostvb.dll` or `EnterpriseServicesHostcs.dll` file. Alternately, you can change `PTWeb` and `PTWebService` to use either the Remoting or Web Services proxies instead.

If you do use the `EnterpriseServicesProxy`, you must also open the COM+ Applications properties window for the service and set the Application Root Directory in the Activation tab. The path should point to the same directory that contains the `application.config` file you edited earlier.

Please note: the `PTWebService` project is missing its reference to the `EnterpriseServicesHostvb` or `EnterpriseServicesHostcs` project. You will need to add this project reference to `PTWebService` before using the Web Service interface!

8. Set the Start Options for both `PTWeb` and `PTWebService` to use a custom server, pointing to the virtual root URLs you set up earlier.
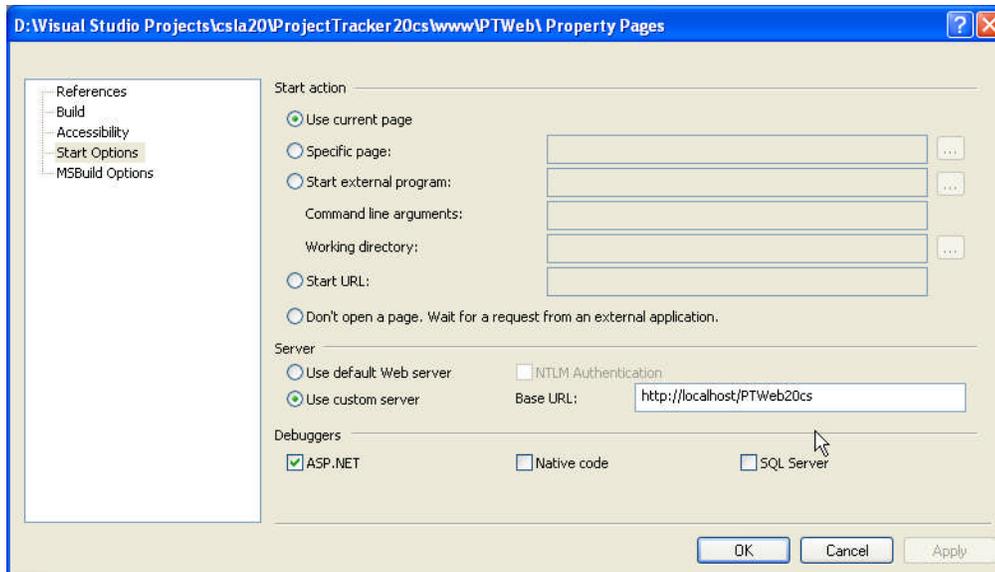
You may optionally set a specific page (such as `Default.aspx`) as the startup page, though this is not necessary. The `PTWeb` pages include functionality to redirect the user to a "safe" page if they attempt to start on a page without proper authorization.

9. 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.

10. Build the `ProjectTracker` solution. You should now be able to run `PTWin`, `PTWeb` and `PTServiceClient`.

    a. To log into the application, you can use

<u>Username</u>	<u>Password</u>	
pm	pm	Log in as a project manager
user	user	Log in as a normal user