CslaExtension for Visual Studio 2010

Quick Setup Guide

CSLA .NET for Silverlight 4

3-tier Application model
WCF Binary Message Encoding
Compression

Lino Zonjić, http://t4csla.codeplex.com/
5/14/2010

Introduction

Thanks to Rockford Lhotka (http://lhotka.net) we have CSLA .NET for Windows and CSLA .NET for Silverlight. Now, question is how to quickly generate CSLA business objects based on database model you created or later updated? Project CslaExtension (http://t4csla.codeplex.com) is initially started with this aim – to help developers to start using CSLA and maintain business layer and data access code quickly and easily. More about how to start using CslaExtension together with demo application you can find at http://t4csla.codeplex.com/documentation. There you will find more information how can you benefit of using CslaExtension for Visual Studio 2010.

> ***This document will guide you to setup all necessary parts to create basic Silverlight 4 application in Visual Studio 2010 by using CSLA 3-tier application model with compression and WCF binary message encoding. Much information presented in this document is collected from lhotka.net site but here are concentrate to help you reuse them.***

Essentially, Visual Studio solution will consist of 4 projects:

Project Name	VS Project Type	Project purpose/description
SilverlightApplication	Silverlight Application	Silverlight user interface
Business.Client	Silverlight Class Library	Business Layer Library on client (Silverlight) side – contains CSLA classes without Data Access Layer
Business.Server	Class Library	Server side of Business Layer Library and Data Access Layer – contains CSLA classes with Data Access Layer code.
SilverlightApplication.Web	ASP.NET Web Application	Hosts Silverlight Application.

Prerequisites

1. Visual Studio 2010 with Silverlight 4 Tools (http://www.silverlight.net/getstarted/)
2. CSLA 4 for Windows and Silverlight (DLLs), basic understanding of CSLA Business objects (http://lhotka.net)
3. CslaExtension for VS2010 and knowledge how to use it (http://t4csla.codeplex.com/documentation)
4. SharpZipLib (http://www.sharpziplib.com/) and its Silverlight port (http://slsharpziplib.codeplex.com)

Creating Silverlight 4 Application using CSLA 3-tier Application model with WCF Binary Message Encoding

1. **Create new Silverlight Application**
   [File\New Project\Silverlight Application (project name: "SilverlightApplication")]
   Solution is now consisted of two projects: SilverlightApplication, SilverlightApplication.Web.

2. **Add Class Library project for Business Layer Library on Web server side**
   [Add\New Project\Class Library (project name: "Business.Server")]

   In Project Properties: set Assembly name and Namespace to "Business" (this name must be the same as in Business.Client project's properties)

Add Reference: CSLA .NET for Windows ("csla.dll")

3. **Add Silverlight Class Library for Business Layer Library on Client side**
   [Add\New Project\Silverlight Class Library (project name: "Business.Client")]

   In Project Properties: set Assembly name and Namespace to "Business" (this name must be the same as in Business.Server project's properties)

   Add References: CSLA .NET for Silverlight ("csla.dll")

4. **Configuring ASP.NET Web Application project ("SilverlightApplication.Web")**

   Add References: (1) Business.Server; (2) CSLA .NET for Windows ("csla.dll")

   Create file "WcfSlPortal.svc" with following content:

```
<% @ServiceHost Service="Csla.Server.Hosts.Silverlight.WcfPortal" %>
```

   Edit "Web.config" file to have following content:

```
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.0" />
  </system.web>

  <connectionStrings>
    <!-- Here put SQL connection string -->
  </connectionStrings>

  <system.serviceModel>
    <bindings>
      <customBinding>
        <binding name="BinaryBinding_IWcfPortal" receiveTimeout="00:10:00" sendTimeout="00:10:00"
openTimeout="00:10:00" closeTimeout="00:10:00" >
          <binaryMessageEncoding maxReadPoolSize="2147483647" maxSessionSize="2147483647"
maxWritePoolSize="2147483647">
            <readerQuotas maxDepth="32" maxStringContentLength="5242880" maxArrayLength="200000"
maxBytesPerRead="4096" maxNameTableCharCount="16384" />
          </binaryMessageEncoding>
          <httpTransport maxBufferPoolSize="2147483647" maxBufferSize="2147483647"
maxReceivedMessageSize="2147483647"/>
        </binding>
      </customBinding>
    </bindings>

    <services>
      <service name="Csla.Server.Hosts.Silverlight.WcfPortal"
behaviorConfiguration="WcfPortalBehavior">
        <endpoint address="" binding="customBinding" contract="Csla.Server.Hosts.Silverlight.IWcfPortal"
bindingConfiguration="BinaryBinding_IWcfPortal">
          <identity>
            <dns value="localhost"/>
          </identity>
        </endpoint>
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior name="WcfPortalBehavior">
          <serviceMetadata httpGetEnabled="true"/>
```

```
            <serviceDebug includeExceptionDetailInFaults="true"/>
          </behavior>
        </serviceBehaviors>
      </behaviors>
    </system.serviceModel>
</configuration>
```

5. **Configure Silverlight Application project ("SilverlightApplication")**

   Add References: (1) Business.Client; (2) CSLA .NET for Silverlight ("csla.dll")

   Add Service Reference or Create file "ServiceReferences.ClientConfig" with following content:

```
<configuration>
    <system.serviceModel>
        <bindings>
            <customBinding>
                <binding name="BinaryBinding_IWcfPortal">
                    <binaryMessageEncoding />
                    <httpTransport maxReceivedMessageSize="2147483647" maxBufferSize="2147483647" />
                </binding>
            </customBinding>
        </bindings>
        <client>
            <!--  For details about 'contract' attribute content
see http://forums.lhotka.net/forums/p/8745/41605.aspx  -->

            <endpoint address="http://localhost:50565/WcfSlService.svc" binding="customBinding"
                bindingConfiguration="BinaryBinding_IWcfPortal" contract="WcfPortal.IWcfPortal"
                name="BinaryBinding_IWcfPortal" />
        </client>
    </system.serviceModel>
</configuration>
```

6. **Add Entity Framework Model and CslaExtension.Template to Business Layer Library on Web server side**
   [Add/New Item/ADO.NET Entity Data Model]

   [Right click on ED surface/Add Code Generation Item/Visual C# Items/CslaExtension.Template]

   Set "**Csla BL Namespace**" and "**Csla DAL Namespace**" properties.

   Set "Csla Multiple Files Output" property to "**CommonAndServerCodeForEachClass**".

   Set "**Csla Class Template**" property for each entity you want to create CSLA class.

   Set right name and path to edmx file in CslaExtension T4 template (Generate method's parameter).

7. **Configure Business.Client project to use generated Business Layer Library from Web server side**
   Add links to files with common business logic (generated files without ".Server.cs" suffix) from Business.Server project to Business.Client project.

8. **Set SQL connection string**
   Copy SQL connection string from App.Config file to Web.config file in "SilverlightApplication.Web" project.

Compressing network traffic in Silverlight Application

CSLA business objects can be quite large and thus produce a lot of network traffic while moving from client to server and vice versa. Decrease of network traffic generated by CSLA movable business objects can greatly speed up application-server communication. Depending on content and compression method, it's possible to achieve decrease of network traffic for around 80% - 90% or even more if you transfer large business object graphs.

To compress network traffic in Silverlight application you have several options but I'll mention only two:

1. extend Csla.Server.Hosts.Silverlight.WcfPortal and Csla.DataPortalClient.WcfProxy classes and override methods in which you can compress/decompress content before/after WCF communication

2. use dynamic HTTP compression in IIS (I didn't try this option, so I'm not sure how it works, details you can find at http://blog.wassupy.com/2009/08/enabling-dynamic-http-compression-in.html)

In this document I'll cover first option.

Prerequisites

Library such is SharpZipLib (http://www.sharpziplib.com/) is required for using compression because System.IO.Compression namespace is not available in Silverlight. At http://slsharpziplib.codeplex.com you can find SharpZipLib version ported for Silverlight 3. With 2 minor modifications in file "FileSystemScanner.cs" it's possible to compile it for Silverlight 4. Probably it's possible to use other libraries such is DotNetZip (http://dotnetzip.codeplex.com).

Compressing network traffic by extending CSLA WcfPortal and WcfProxy classes

1. **Configure Business.Server project**

   Add reference to ICSharpCode.SharpZipLib (standard SharpZipLib compiled for Full .NET Framework)

   Add class CompressionUtility in Business.Server project:

```
using System;
using System.IO;
using ICSharpCode.SharpZipLib;

namespace Business.Compression
{
  public class CompressionUtility
  {
    public static byte[] Compress(byte[] byteData)
    {
      byte[] compressedData = null;
      if (byteData != null)
      {
        using (MemoryStream ms = new MemoryStream())
        {
          ICSharpCode.SharpZipLib.Zip.Compression.Deflater defl =
           new ICSharpCode.SharpZipLib.Zip.Compression.Deflater(9, false);
          using (Stream s =
              new ICSharpCode.SharpZipLib.Zip.Compression.Streams.DeflaterOutputStream(ms, defl))
            s.Write(byteData, 0, byteData.Length);
          compressedData = ms.ToArray();
        }
      }
```

```
      return compressedData;
    }

    public static byte[] Decompress(byte[] byteInput)
    {
      byte[] bytResult = null;
      if (byteInput != null)
      {
        using (MemoryStream ms = new MemoryStream(byteInput, 0, byteInput.Length))
        {
          string strResult = String.Empty;
          byte[] writeData = new byte[4096];
          Stream s2 =
             new ICSharpCode.SharpZipLib.Zip.Compression.Streams.InflaterInputStream(ms);
          bytResult = ReadFullStream(s2);
        }
      }
      return bytResult;
    }

    private static byte[] ReadFullStream(Stream stream)
    {
      byte[] buffer = new byte[32768];
      using (MemoryStream ms = new MemoryStream())
      {
        while (true)
        {
          int read = stream.Read(buffer, 0, buffer.Length);
          if (read <= 0)
            return ms.ToArray();
          ms.Write(buffer, 0, read);
        }
      }
    }

  }
}
```

Add class CompressedHost in Business.Server project

Important: You must change Web.config and WcfSlPortal.svc files according to full name of class derived from WcfPortal (in this example “Business.Compression.CompressedHost”).

```
using System;
using System.Net;
using Csla.Server.Hosts;

namespace Business.Compression
{
  public class CompressedHost : Csla.Server.Hosts.Silverlight.WcfPortal
  {
    protected override Csla.Server.Hosts.Silverlight.CriteriaRequest
ConvertRequest(Csla.Server.Hosts.Silverlight.CriteriaRequest request)
    {
      Csla.Server.Hosts.Silverlight.CriteriaRequest returnValue = new
Csla.Server.Hosts.Silverlight.CriteriaRequest();
      returnValue.ClientContext = CompressionUtility.Decompress(request.ClientContext);
      returnValue.GlobalContext = CompressionUtility.Decompress(request.GlobalContext);
      if (request.CriteriaData != null)
        returnValue.CriteriaData = CompressionUtility.Decompress(request.CriteriaData);
      returnValue.Principal = CompressionUtility.Decompress(request.Principal);
      returnValue.TypeName = request.TypeName;
```

```
      return returnValue;
    }

    protected override Csla.Server.Hosts.Silverlight.UpdateRequest
ConvertRequest(Csla.Server.Hosts.Silverlight.UpdateRequest request)
    {
      Csla.Server.Hosts.Silverlight.UpdateRequest returnValue = new
Csla.Server.Hosts.Silverlight.UpdateRequest();
      returnValue.ClientContext = CompressionUtility.Decompress(request.ClientContext);
      returnValue.GlobalContext = CompressionUtility.Decompress(request.GlobalContext);
      returnValue.ObjectData = CompressionUtility.Decompress(request.ObjectData);
      returnValue.Principal = CompressionUtility.Decompress(request.Principal);
      return returnValue;
    }

    protected override Csla.Server.Hosts.Silverlight.WcfResponse
ConvertResponse(Csla.Server.Hosts.Silverlight.WcfResponse response)
    {
      Csla.Server.Hosts.Silverlight.WcfResponse returnValue = new Csla.Server.Hosts.Silverlight.WcfResponse();
      returnValue.GlobalContext = CompressionUtility.Compress(response.GlobalContext);
      returnValue.ObjectData = CompressionUtility.Compress(response.ObjectData);
      returnValue.ErrorData = response.ErrorData;
      return returnValue;
    }
  }
}
```

2. **Configure SilverlightApplication.Web project**

   Change Web.config file:

   from: `<service name="Csla.Server.Hosts.Silverlight.WcfPortal" behaviorConfiguration="WcfPortalBehavior">`

   to: `<service name="Business.Compression.CompressedHost" behaviorConfiguration="WcfPortalBehavior">`

   Change WcfSlPortal.svc:

   from: `<% @ServiceHost Service="Csla.Server.Hosts.Silverlight.WcfPortal" %>`

   to: `<% @ServiceHost Service="Business.Compression.CompressedHost" %>`

3. **Configure Business.Client project**

   Add reference to SharpZipLib (SharpZipLib ported and compiled for Silverlight)

   Add file link to CompressionUtility.cs from **Business.Server project.**

   Create class CompressedProxy in Business.Client project

```
using System;
using System.Net;
using Csla.DataPortalClient;

namespace Business.Compression
{
  public class CompressedProxy<T> : WcfProxy<T>
    where T : Csla.Serialization.Mobile.IMobileObject
```

```
  {
    protected override Csla.WcfPortal.CriteriaRequest ConvertRequest(Csla.WcfPortal.CriteriaRequest request)
    {
      Csla.WcfPortal.CriteriaRequest returnValue = new Csla.WcfPortal.CriteriaRequest();
      returnValue.ClientContext = CompressionUtility.Compress(request.ClientContext);
      returnValue.GlobalContext = CompressionUtility.Compress(request.GlobalContext);
      if (request.CriteriaData != null)
        returnValue.CriteriaData = CompressionUtility.Compress(request.CriteriaData);
      returnValue.Principal = CompressionUtility.Compress(request.Principal);
      returnValue.TypeName = request.TypeName;
      return returnValue;
    }

    protected override Csla.WcfPortal.UpdateRequest ConvertRequest(Csla.WcfPortal.UpdateRequest request)
    {
      Csla.WcfPortal.UpdateRequest returnValue = new Csla.WcfPortal.UpdateRequest();
      returnValue.ClientContext = CompressionUtility.Compress(request.ClientContext);
      returnValue.GlobalContext = CompressionUtility.Compress(request.GlobalContext);
      returnValue.ObjectData = CompressionUtility.Compress(request.ObjectData);
      returnValue.Principal = CompressionUtility.Compress(request.Principal);
      return returnValue;
    }

    protected override Csla.WcfPortal.WcfResponse ConvertResponse(Csla.WcfPortal.WcfResponse response)
    {
      Csla.WcfPortal.WcfResponse returnValue = new Csla.WcfPortal.WcfResponse();
      returnValue.GlobalContext = CompressionUtility.Decompress(response.GlobalContext);
      returnValue.ObjectData = CompressionUtility.Decompress(response.ObjectData);
      returnValue.ErrorData = response.ErrorData;
      return returnValue;
    }
  }
}
```

Document history

Date	Version	Comments
14.05.2010.	1.0	Initial document's version.