


Developing Business Objects With CSLA

David Campbell
Creative Process Solutions
dcampbell@creativeprocess.ca

CLSA Business Objects

- ▶ Encapsulate the application's data along with its related business logic
 - ▶ Enforce validation and authorization
 - ▶ Are portable or “mobile”, they are automatically moved (serialized) across service / transport layer.
 - ▶ The service / transport layer is configurable at runtime using data portal hosts.
- 

Why use business objects?

- ▶ **N-level undo** – Provides functionality to take snapshots of an object's data and then perform undo or accept operations using these snapshots.
- ▶ **Tracking object status** – Keeps track of whether the object is new, old, dirty, clean, valid, or marked for deletion
- ▶ **Root, parent, and child behaviors** – Implement behaviors so the object can function as either a stand-alone object, a parent object, or a child of another object or collection
- ▶ **Validation rules** – Provides functionality to track broken business rules on all editable objects and allow read-only access to user interface developers to the broken rules.
- ▶ **Authorization rules** – Provides functionality to limit access to objects and their properties
- ▶ **User Interface Databinding** – Full support for data binding in both Windows Forms and Web Forms. Implements `IBindingList`, `INotifyProperty`, `IEditableObject`, `IRaiseItemChangedEvents`, `IDataErrorInfo` interfaces to support User Interface data binding
- ▶ **Data Portal** – Provides functionality to encapsulate all data access for object persistence in the object and one configurable point of access to the server. Also provides abstraction of the network transport between Client and Server which enables configurable support for remoting, Web Services, Enterprise Services, and future protocols like WCF

Business Classes for Developers

Editable Classes

BusinessBase(Of T)
Generic MustInherit Class
→ BusinessBase

EditableRootListBase(Of T)
Generic MustInherit Class
→ ExtendedBindingList(Of T)

BusinessListBase(Of T, C)
Generic MustInherit Class
→ ExtendedBindingList(Of C)

Read Only Classes

ReadOnlyBase(Of T)
Generic MustInherit Class

ReadOnlyListBase(Of T, C)
Generic MustInherit Class
→ ReadOnlyBindingList(Of C)

NameValueListBase(Of K, V)
Generic MustInherit Class
→ ReadOnlyBindingList(Of NameValuePair)

NameValueBilingualListBase(Of KeyValue, EnglishValue, FrenchValue)
Generic MustInherit Class
→ ReadOnlyBindingList(Of NameValueEnglishFrench)

Command Classes

CommandBase
MustInherit Class

Utility Classes, Modules and Structures

SortedBindingList(Of T)
Generic Class

FilteredBindingList(Of T)
Generic Class

SafeDataReader
Class

ObjectAdapter
Class

Mapper
Module

SmartDate
Structure

Business Object Receipe

- ▶ Choose a class type for each entity in the domain you will implement
 - Editable
 - Root
 - Parent
 - Child
 - Switchable
 - List
 - Readonly
 - Editable
 - Lookup/Codset/Name Value Pair
 - Collection
 - Editable root
 - Child
 - Command

Use a snippet or a template

▶ Item Templates

- Editable Root
- Editable Parent
- EditableRootParent
- EditableChild
- SwitchableObject
- EditableChildList
- EditableRootList
- NameValueList
- DynamicRootList
- ReadOnlyRoot
- ReadOnlyChild

- ReadOnlyList

- CommandObject

▶ Class snippets

- Cslaclass

- Cslachildclass

- Cslachildcollection

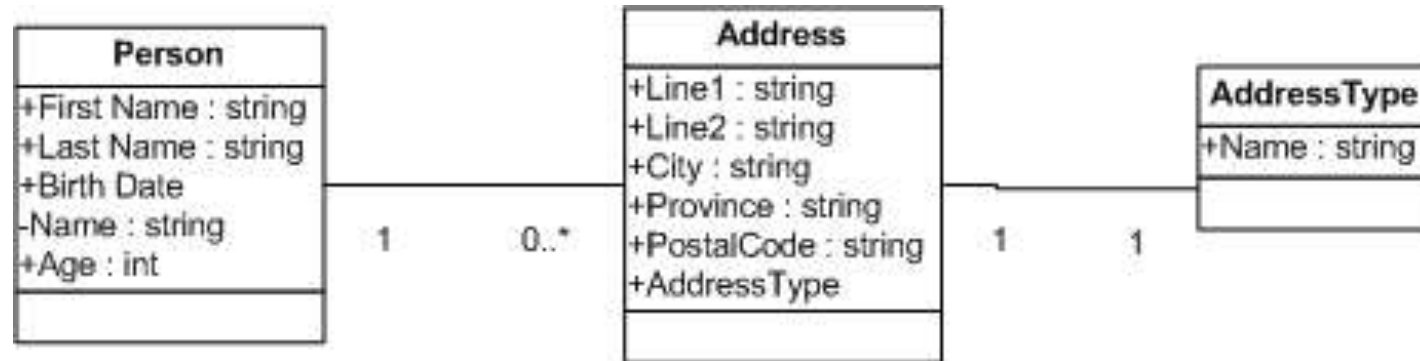
- Csladynamiccollectionclass

- Cslanamevaluelistclass

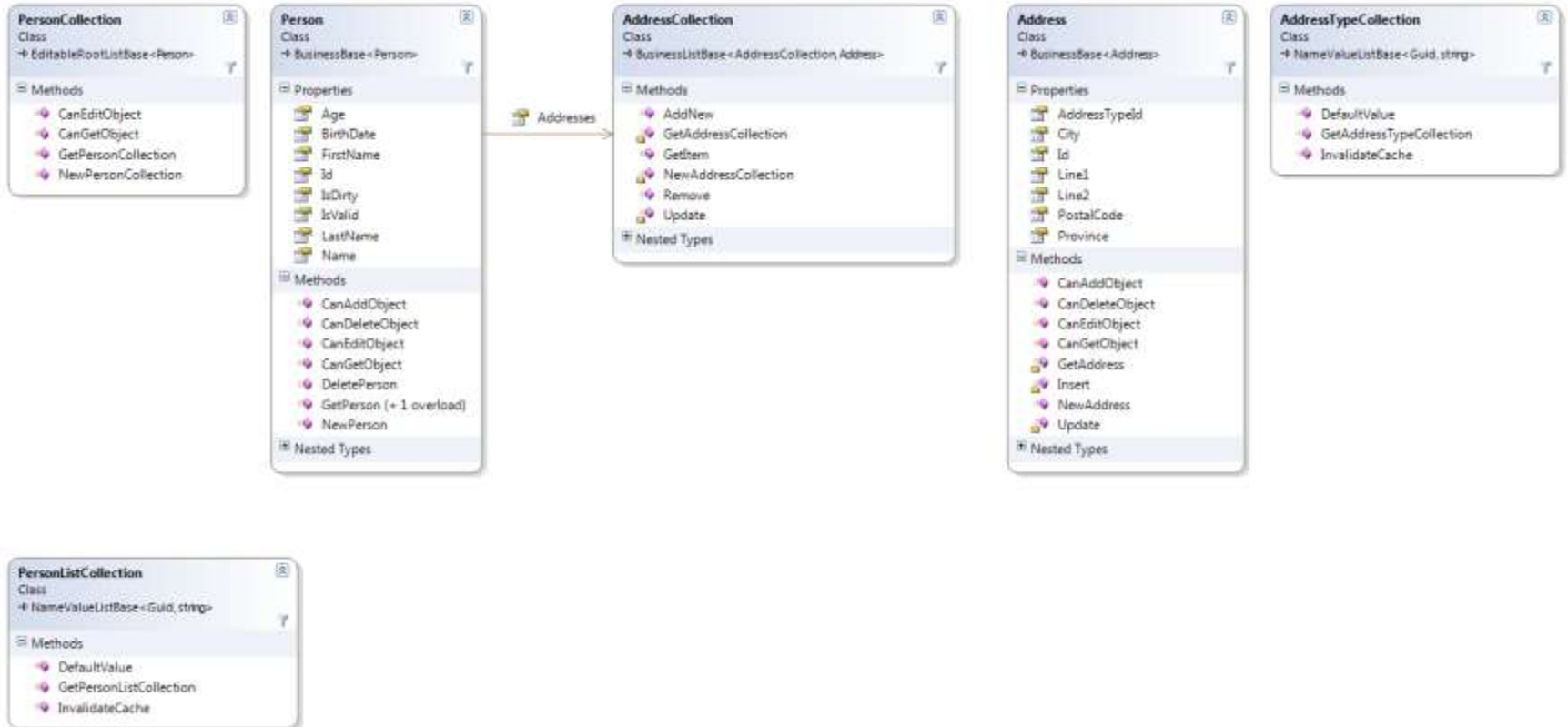
For Each Business Class

- ▶ *In Business Methods*
 - *Add your class specific properties and methods*
- ▶ *In Authorization*
 - *Configure your access rules*
- ▶ *In Validation*
 - *Add your business rules*
- ▶ *In Data Access*
 - *Complete or remove the methods*
 - *DataPortal_Create*
 - *DataPortal_Fetch*
 - *DataPortal_Insert*
 - *DataPortal_Update*
 - *DataPortal_Delete*


Creating the Address Book



Class Diagram



Windows UI

- ▶ Add a reference to your class library
 - ▶ Add an object datasource for your Parent and Root objects
 - ▶ Drag and drop the datasources onto your forms
 - ▶ Add a Bindsource refresh, ReadWriteAuthorisation and ErrorProvider
 - ▶ Write the code to fill your datasource
 - ▶ Write the code to handle events
- 

Form Methods when using an editable Class

- ▶ **In the New Method**
 - Initialize your binding sources and your Private variable.
 - Call BeginEdit() on your Private Variable.
 - Apply Authorization Rules to your form
- ▶ **In the Save method**
 - Using busy As New StatusBusy("Saving...")
 - If your variable has children, set .RaiseListChangedEvents = False on the children
 - Clone your private variable - Dim temp as MyClass = myPrivateVariable.Clone()
 - ApplyEdit to your clone - temp.ApplyEdit()
 - In a Try Catch block call save on your clone - myPrivateVariable = temp.Save()
 - Catch Stc.Framework.DataPortalException
 - Catch Exception
 - In the Finally set .RaiseListChangedEvents = True on the children
- ▶ **In the Cancel Method**
 - call myPrivateVariable.CancelEdit()
- ▶ **Adding and deleting children**
 - Add Child Event - myPrivateVariable.Children.Add (Child.NewChild()) OR myPrivateVariable.Children.AddNew()
 - Delete Child Event - myPrivateVariable.Children.Remove(ChildID)

Web UI

- ▶ Add a reference to your class library
- ▶ Add a csladatasource to your controls or forms for each databound business object
- ▶ For each csladatasource set the TypeName to your business object
- ▶ Write code in the Events for Insert Update Delete and Select
 - Select
 - e.BusinessObject
 - Insert
 - e.Values, e.RowsAffected
 - Update
 - e.Values, e.Keys , e.OldValues , e.RowsAffected
 - Delete
 - e.Keys , e.OldValues , e.RowsAffected

Loading and Saving Data

- ▶ Page_Load
 - If not postback then Apply Authorization Rules
 - Clear error messages
- ▶ Get your Object Event
 - Retrieve from Session("currentObject")
 - If is nothing or not the same type – Load from your library
- ▶ Save your Object Event

```
Dim rowsAffected As Integer
```

```
Try
```

```
    Session("currentObject") = yourObject.Save()
```

```
    rowsAffected = 1
```

```
Catch ex As Stc.Framework.Validation.ValidationException
```

```
    Show messages
```

```
Catch ex As Stc.Framework.DataPortalException
```

```
    Show message
```

```
Catch ex As Exception
```

Authorization and Validation

▶ Authorization


- If yourObject.CanEditObject Then
 - Make user interface editable
- Else
 - Make user interface read only
- End If

▶ Validation

- Catch ex As Stc.Framework.Validation.ValidationException
- If yourObject.BrokenRulesCollection.Count > 0 Then
 - For Each rule As Stc.Framework.Validation.BrokenRule In application.BrokenRulesCollection
 - message.AppendFormat("* {0}: {1}
", rule.Property, rule.Description)

Next

DataPortal Hosts

- ▶ Choose a host type
 - ▶ Set up the host on the server
 - ▶ Make your business object assembly available to the host
 - ▶ Configure the client to use the new host
- 

.net Remoting Host

- ▶ Create an Empty web project
- ▶ Add a reference to your business assembly
- ▶ Ensure that the Stc.framework.dll is in the Bin Directory
- ▶ Add a `<system.runtime.remoting>` element to expose the data portal in the Web.Config
 - `<system.runtime.remoting>`
 - `<application>`
 - `<service>`
 - `<wellknown mode="SingleCall" objectUri="RemotingPortal.rem"`
 - `type="Csla.Server.Hosts.RemotingPortal, Csla"/>`
 - `</service>`
 - `<channels>`
 - `<channel ref="http">`
 - `<serverProviders>`
 - `<provider ref="wsdl"/>`
 - `<formatter ref="soap" typeFilterLevel="Full"/>`
 - `<formatter ref="binary" typeFilterLevel="Full"/>`
 - `</serverProviders>`
 - `</channel>`
 - `</channels>`
 - `</application>`
 - `</system.runtime.remoting>`
- ▶ Configure the client to use the remoting host data portal
 - `<add key="CslaDataPortalProxy" value="Csla.DataPortalClient.RemotingProxy, Csla"/>`
 - `<add key="CslaDataPortalUrl" value="http://YourServer/YourVirtualDirectory/RemotingPortal.rem"/>`

Web Service Host

- ▶ Create a Web Services project in Visual Studio
- ▶ Add a reference to your business assembly
- ▶ Ensure that Stc.Framework.dll is in the Bin directory
- ▶ Edit the asmx file to refer to WebServicePortal class
 - `<%@ WebService Language="vb" Class="Csla.Server.Hosts.WebServicePortal" %>`
- ▶ Configure the client to use the Web Services data portal host
 - `<add key="CslaDataPortalProxy" value="Csla.DataPortalClient.WebServicesProxy, Stc.Framework"/>`
 - `<add key="CslaDataPortalUrl" value="http://YourServer/YourVirtualDirectory/WebServicePortal.asmx"/>`

Enterprise Service Host

- ▶ Create an Enterprise Services proxy/host assembly for your application
 - Create a Class library project
 - Add a reference to System.EnterpriseServices
 - Sign the assembly with a keyfile
 - Add an EnterpriseServicesSetting.vb file
 - `<Assembly: System.EnterpriseServices.ApplicationActivation(System.EnterpriseServices.ActivationOption.Server)>`
 - `<Assembly: System.EnterpriseServices.ApplicationName("Your Application Dataportal Host")>`
 - `<Assembly: System.EnterpriseServices.Description("Your Application Dataportal Host")>`
 - `<Assembly: System.EnterpriseServices.ApplicationAccessControl(False)>`
 - Add a Reference to Csla.dll and your business assembly
 - Use a strong name to sign your proxy/host
 - Create a application.config and application.manifest file
- ▶ Install your proxy/host assembly into COM+ on the server
 - From a Visual Studio command prompt
 - `Regsvcs YourAssembly.dll`
- ▶ Configure the COM+ Application on the server
 - On the Identity tab – configure the account that it will run under
 - On the Activation tab – configure the Application Root Directory
- ▶ Export the COM+ Application to create an install.msi
 - Right Click the node in Component services and Export
- ▶ Configure the client
 - Run the COM+ Application msi
 - Configure the client application to use the host
 - `<add key="CslaDataPortalProxy" value="EnterpriseServicesHost.EnterpriseServicesProxy, YourApplicationHost"/>`

Questions?

- ▶ Creative Process Solutions
 - ▶ David Campbell
 - ▶ dcampbell@creativeprocess.ca
- 