


# Introduction to CSLA

David Campbell  
Creative Process Solutions  
[dcampbell@creativeprocess.ca](mailto:dcampbell@creativeprocess.ca)

# Common Development Challenges

- ▶ Complexity of creating and configuring different service and or transport layers can tie an application to a particular technology
- ▶ Time is of the essence. Developers take shortcuts or don't take into account the impact of cross cutting application functionality like security, error handling, and localisation
- ▶ Repetition of code
  - A large percentage of development time is spent on recreating plumbing code.
  - Validation and authorisation rules must be repeated in each user interface of each application as well as each database used by an application.
  - The need to support multiple interfaces for the same application
- ▶ Never have the time to test.
- ▶ Quality of code
- ▶ Changing requirements

# *When to use CSLA*

- ▶ Enterprise level applications that need to be easily scalable and/or distributable.
  - ▶ Applications that may have multiple user interfaces or application interfaces.
  - ▶ Applications that need to be database independent.
  - ▶ Applications that change often.
- 

# Definition of an Application

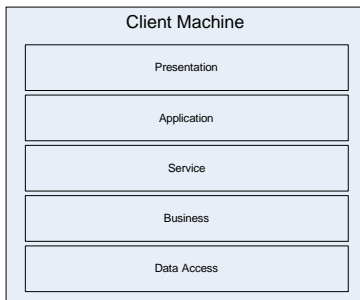
- ▶ “An application is a set of code, objects or components that’s considered to be part of a single logical unit. Even if parts of the application are in different assemblies or installed on different machines, all the code is viewed as being part of a singular application” *Expert C# 2005 Business Objects*

# *Five Layer Application Architecture*

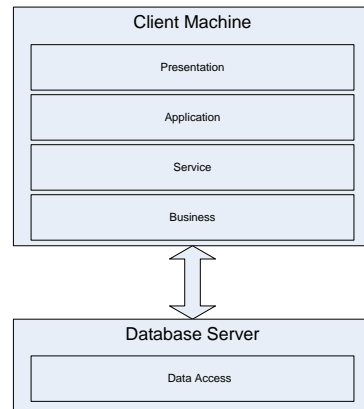
- ▶ **Data Access Layer**
  - Microsoft Enterprise library data access application block from the patterns and practices group
  - Software factory pattern
  - Provider pattern
  - Database independent
  - Consistent methods
- ▶ **Business Layer**
  - Business rules
  - Validation
  - Authorization
  - Uses the data access layer to persist data in a data store
  - Business Logic
  - Business Entities
  - Parent/Child relationships
- ▶ **Service/Transport Layer**
  - Data Portal providers
- ▶ **Application Layer**
  - Model View Presenter or Model View Controller
  - Application Interface specific logic
  - Navigation logic
- ▶ **Presentation Layer**
  - Interface classes and forms
  - Web forms
  - Windows forms
  - Mobile forms
  - XAML (Windows Presentation Foundation)
  - Web Service

# Logical Layer to Physical Tier mapping

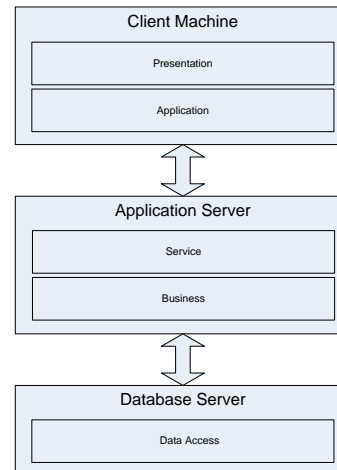
## One Tier



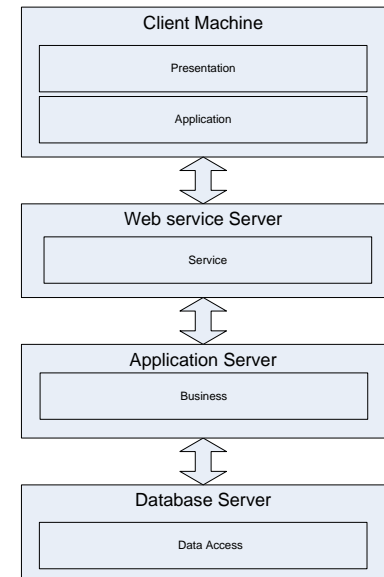
## Two Tier



## Three Tier

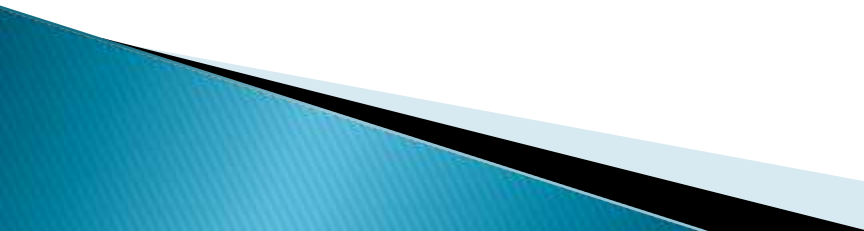


## Four Tier




One to Four physical tiers with 5 logical layers

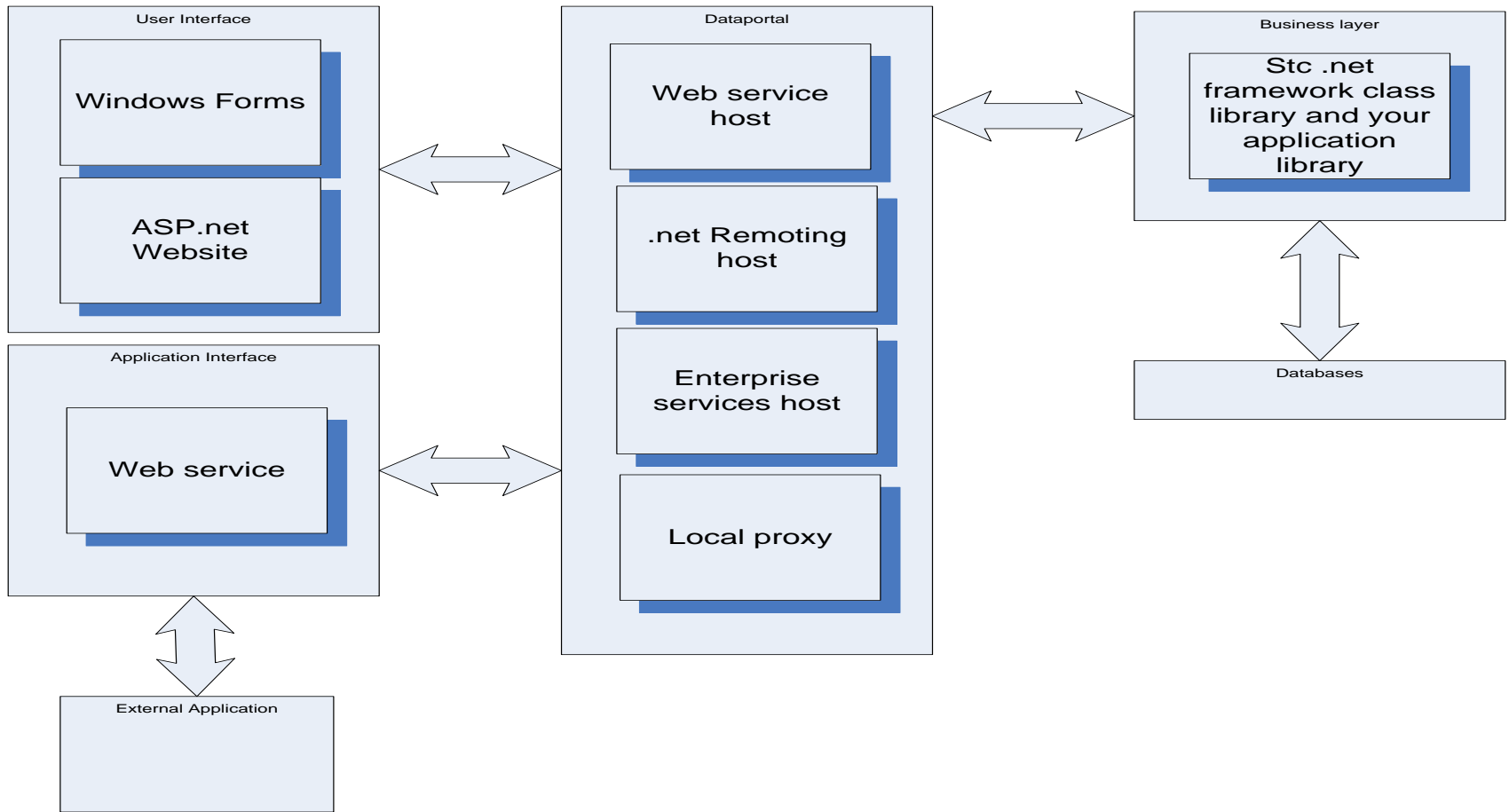
# CSLA to the rescue

- ▶ Open source
    - Source code in C# and VB
  - ▶ Sample Application – Project Tracker
    - Windows Forms User Interface
    - Web Forms User Interface
    - Web Service Interface
    - .net Remoting Host
    - Web service host
    - Enterprise Services Host
    - Windows Forms User Interface that consumes the web service
  - ▶ Includes Unit tests for NUnit and VSTS
  - ▶ Large user community
  - ▶ Expert C# Business Objects and Expert VB Business Objects books available
- 

# 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.
- 

# Mobile Business Objects



# 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

# Creating Business Objects

- ▶ Choose an object type
  - Editable
    - Root
    - Parent
    - Child
    - Switchable (can act as both a child and a parent object)
  - Editable List
    - Must have a root or child object to create an editable list
  - Read only
  - Read only list
  - Name value list
  - Command
- ▶ Use one of the templates or Create a new class and use the snippets

# 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

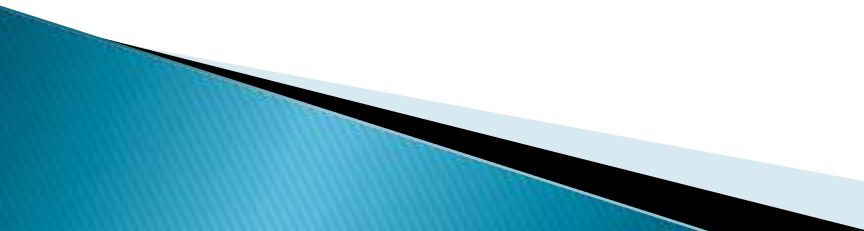
**SafeDataReader**  
Class

**ObjectAdapter**  
Class

**Mapper**  
Module

**SmartDate**  
Structure

# Project Tracker

- ▶ Class library of business objects
  - ▶ 3 DataPortal hosts
    - .net Remoting
    - Web service
    - Enterprise Services
  - ▶ 2 User Interfaces
    - Windows Form
    - ASP.net Website
  - ▶ Web Service
  - ▶ Smart Client
  - ▶ NUnit and VSTS tests for the class library
- 

# Part 2

- ▶ Developing business objects with CSLA
  - ODNC luncheon event March 28, 2007