Overview of CSLA

Framework of Choice

At Magenic, when we begin a software development project, we start with an assumption that we should avoid re-inventing the wheel. For example, when beginning a .NET development project, we start with the .NET Framework's built in abilities before purchasing third party components or creating custom components. We do the same with the business logic of an application.

With this in mind, Magenic makes an assumption that the CSLA .NET Framework will be core to the architecture of a solution. During the early phases of a project, our architects need to justify why they believe CSLA .NET is not the proper Framework for the solution. Although we believe CSLA .NET is a great choice for many solutions, we do know that it does not fit all circumstances.

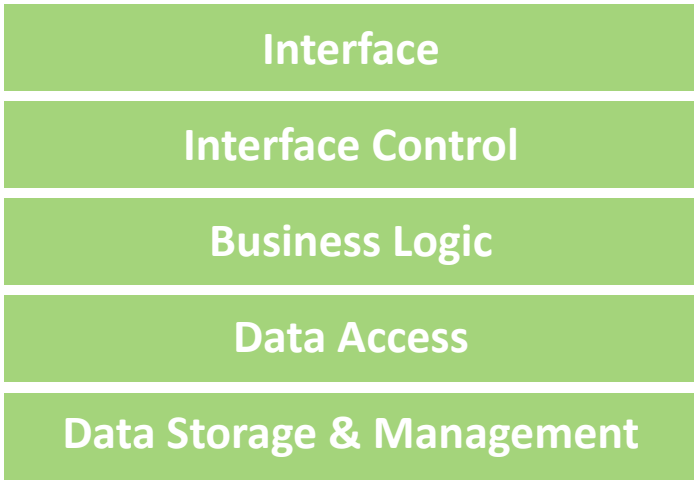
CSLA .NET is the creation of Magenic CTO Rockford Lhotka. While it is not proprietary to Magenic, our consultants are very experienced with CSLA .NET and have assisted Lhotka on the framework. Since 2001, CSLA has evolved and changed in many ways, culminating in its latest release - CSLA 4. It is now one of the most widely used open source development frameworks for the .NET platform. It is covered under a very liberal license and has a vibrant, helpful and friendly community.

Overview of CSLA 4

CSLA 4 is a framework for creating business applications using object-oriented design concepts in a distributed computing environment. CSLA 4 supports development on the .NET, Silverlight, Windows Azure and Windows Phone 7 platforms. The commonality across these platforms allows sharing of business object code between .NET, Silverlight and Windows Phone 7, including running server-side code on Windows Server or Windows Azure. At the same time, CSLA 4 does provide targeted support for each platform where appropriate, allowing developers to exploit the power of the different platforms.

The primary goal of CSLA .NET is to enable creation of a rich, powerful and flexible business layer for an application.

This business layer is composed of business domain objects that encapsulate the business logic (calculations, algorithmic processing, validation and authorization). These objects should be designed based on the business use cases for the application domain. With some reasonable care, it is possible to create a single set of business objects that work within both the .NET and Silverlight environments.



In many cases, Silverlight and Windows Phone 7 business objects may contain the exact same object code used for Windows applications, though there may be some small variations. Generally speaking, more than 90% of the object code will be shared between .NET and Silverlight. The differences are primarily due to the asynchronous nature of Silverlight programming and the more synchronous nature of traditional .NET programming. If developers are willing to apply the same asynchronous designs to .NET implementation, they can achieve 99% or 100% code sharing between the two platforms.

CSLA 4 Core Features

Regardless of platform, CSLA .NET is designed to do two things. First and foremost, it is designed to support developers as they create a powerful business layer based on rich business domain *objects*. Second, it is designed to enable a distributed application architecture centered on a concept called mobile objects.

To support the creation of rich domain objects, CSLA 4 includes subsystems that supply commonly required functionality, including:

- Full support for data binding in all .NET UI technologies
- Object status tracking (Is object new? Changed? Marked for deletion? etc.)
- Standardized business rule processing
- Standardized authorization at the object and property levels

Magenic®
Custom solutions that fit. Guaranteed.

- Integration with, and simplification of, standard .NET authentication models
- Undo capabilities to support implementation of Cancel buttons and complex layered UIs
- Standardized interaction with a data access layer or ORM
- Enhanced support for LINQ queries against business objects
- Numerous general productivity features, useful in many business application scenarios
- Asynchronous data access and asynchronous validation rules

Lhotka's *Using CSLA 4* ebook and video series (available at http://store.lhotka.net/) cover these subsystems in detail.

These subsystems are exposed through a set of base classes which developers inherit to create business objects. These base classes enable a set of object stereotypes:

- Editable root (single or collection) - An object that has read-write properties and can be directly retrieved and stored in a database
- Editable child (single or collection) - An object that has read-write properties and is retrieved and stored in a database as part of some editable root
- Dynamic list - A collection that contains editable root objects, integrating with data grid controls to auto-update each object when the user leaves a row in the grid (not applicable to ASP.NET interfaces)
- Read-only root (single or collection) - An object that has read-only properties and can be directly retrieved from a database
- Read-only child (single or collection) - An object that has read-only properties and is retrieved from a database as part of some read-only root
- Name/value list - A read-only root collection that contains only name/value pairs for use in populating combobox or listbox controls
- Command - An object that executes code on the client and/or the server; often used to execute database code or server-side workflows

The end result of building business objects using CSLA .NET is that the objects are created in a consistent, standardized manner. So not only do the objects automatically gain substantial benefit from all of these subsystems, but the overall maintainability is radically improved thanks to the application's consistent architecture, design and coding.

The concept of *mobile objects* is a technique that supports the use of rich business domain objects in distributed application environments. Specifically, the idea is that business objects are mobile, able to move physically from one computer or device to another to take advantage of the resources available on each.

CSLA .NET includes a component called the data portal which is responsible for providing the mobile object functionality. Using the data portal, business objects may come into existence on an application server so they can efficiently interact with the data access layer (and database). The objects may then physically move across the network to the client workstation or device (web server, Windows client, Silverlight client or Windows Phone 7 device) where they can efficiently interact with the user by being data bound directly to the UI. Once the user is done interacting with the object, the object may then move back to the application server so it can interact with the data access layer to update the database.

Mobile objects are an incredibly powerful technique for exploiting the power of object-oriented design and programming in distributed environments. The CSLA .NET data portal abstracts the complexity of this concept. The result is that applications can switch between 1-tier, 2-tier, 3-tier and even 4-tier models with no change to code - the change is purely one of configuration.

CSLA 4 on Windows

CSLA 4 allows developers to easily build Windows applications using WPF, Silverlight, ASP.NET MVC, ASP.NET Web Forms and Windows Forms user interfaces on top of business objects. It also supports WCF service and asmx web services interfaces, using either SOAP or REST techniques. Technically, *all* of these interfaces could be created on top of the same set of business objects, though most applications require only one or two types of interface (Web Forms and WCF services for example).

CSLA 4 includes some UI controls in each major UI technology. These controls help minimize UI code and maximize productivity. In WPF the following controls are provided:

- ViewModelBase and ViewModel – Simplify the creation of a viewmodel object for use with the MVVM design pattern

Magenic®
Custom solutions that fit. Guaranteed.

- TriggerAction – Enable routing of arbitrary UI events to the viewmodel where behaviors are implemented
- PropertyStatus - Like the Windows Forms ErrorProvider, but manages validation, authorization and busy notification for each property
- BusyAnimation - A control that displays a busy animation; can be bound to an object to automatically show that the object is performing an asynchronous operation

In Web Forms the following controls are provided:

- CslaDataSource - A Web Forms data source control that supports data binding to business objects
- DataMapper - A component that simplifies the copying of form post values into business objects

CSLA 4 supports ASP.NET MVC development with the following controls:

- CslaModelBinder – Enables binding business objects to views with full support for CSLA .NET business, validation and authorization rules, along with existing DataAnnotations rules from .NET
- HtmlExtensions – Adds CSLA-specific extensions to the Html type, making it easy to leverage all features of business objects when creating views
- ViewModelBase – Simplifies the creation of viewmodel objects in cases where the MVVM design pattern is used within an MVC application
- Controller – Base class that helps minimize the code necessary to create a controller than interacts with editable business objects

In Windows Forms the following controls are provided:

- BindingSourceRefresh - Work around for a data binding refresh issue in Windows Forms
- CslaActionExtender - Automate object management behind buttons such as Save and Cancel
- ReadWriteAuthorization - Automatically enable/disable detail controls based on the object's authorization rules
- CSLA .NET also includes functionality to assist in the creation of services and workflow activities

For WCF and asmx services the following components are provided:

- DataMapper - A component that simplifies the copying of data between business objects and data contract objects
- PrincipalCache - A component that temporarily caches .NET principal objects for use when implementing custom authentication/authorization in a WCF service

CSLA 4 on Windows provides a great deal of flexibility in terms of data access. CSLA .NET is not a data access layer or an object-relational mapping (ORM) tool. However, CSLA .NET does provide a level of formalization around how an application interacts with the data access layer or ORM. This formalized flexibility allows developers to use a wide range of data access technologies, including ADO.NET Entity Framework, raw ADO.NET, DataSets, LINQ to SQL, NHibernate, Paul Wilson's ORM mapper and many other technologies.

CSLA 4 on Silverlight

CSLA 4 on Silverlight allows developers to easily build Silverlight user interfaces on top of business objects. By fully supporting Silverlight data binding, along with extra controls provided by CSLA .NET, it is possible to create Silverlight forms with nearly no UI code. Just like with CSLA 4 on Windows, most of the code is encapsulated in the business objects, maintaining clean separation between the presentation and business behaviors.

The Silverlight controls provided by CSLA 4 include:

- ViewModelBase and ViewModel – Simplify the creation of a viewmodel object for use with the MVVM design pattern
- TriggerAction – Enable routing of arbitrary UI events to the viewmodel where behaviors are implemented
- PropertyStatus - Like the Windows Forms ErrorProvider, but manages validation, authorization and busy notification for each property
- BusyAnimation - A control that displays a busy animation; can be bound to an object to automatically show that the object is performing an asynchronous operation

CSLA 4 enables a data access model on Silverlight where business objects invoke remote services to retrieve or update data. This model can be used to implement client/server or service-oriented application designs. For example, ADO.NET Data Services might be used to expose data services from a server, while CSLA 4 would be used to create business objects and a Silverlight UI to interact with those data services.

Magenic®
Custom solutions that fit. Guaranteed.

Overview of CSLA

Developers using Windows Server or Windows Azure can take advantage of some advanced CSLA 4 capabilities. Specifically, a CSLA 4 Silverlight application can interact with CSLA 4 running on the server, enabling 2-, 3- and 4-tier physical deployments of the application. In this model, .NET business objects (perhaps already supporting an ASP.NET MVC UI) are effectively extended directly into the Silverlight client. The standard object persistence models supported by CSLA 4 are now automatically used to support the Silverlight client, providing an incredibly high level of code and functionality reuse across the .NET and Silverlight platforms.

CSLA .NET Deployment Models and Mobile Objects

The CSLA .NET data portal enables the use of mobile object concepts in an application. This is largely transparent to the code, and the code that is written to interact with the data portal is very standardized. The benefit of using the data portal is flexibility. Developers can switch an application from a physical 1-tier deployment to a 3-tier or even 4-tier deployment purely by changing configuration - no coding changes are required.

CSLA 4 supports 1-, 2- and 3-tier physical deployments for .NET applications.

Again, it is possible to switch between these physical models purely by changing configuration. The UI code, business object code and data access code remain entirely intact across all three deployment models.

In the 3-tier model, the business logic layer (the assembly[ies] containing the business object code) is deployed to both the client and application server. Business objects literally move between those two machines through the data portal.

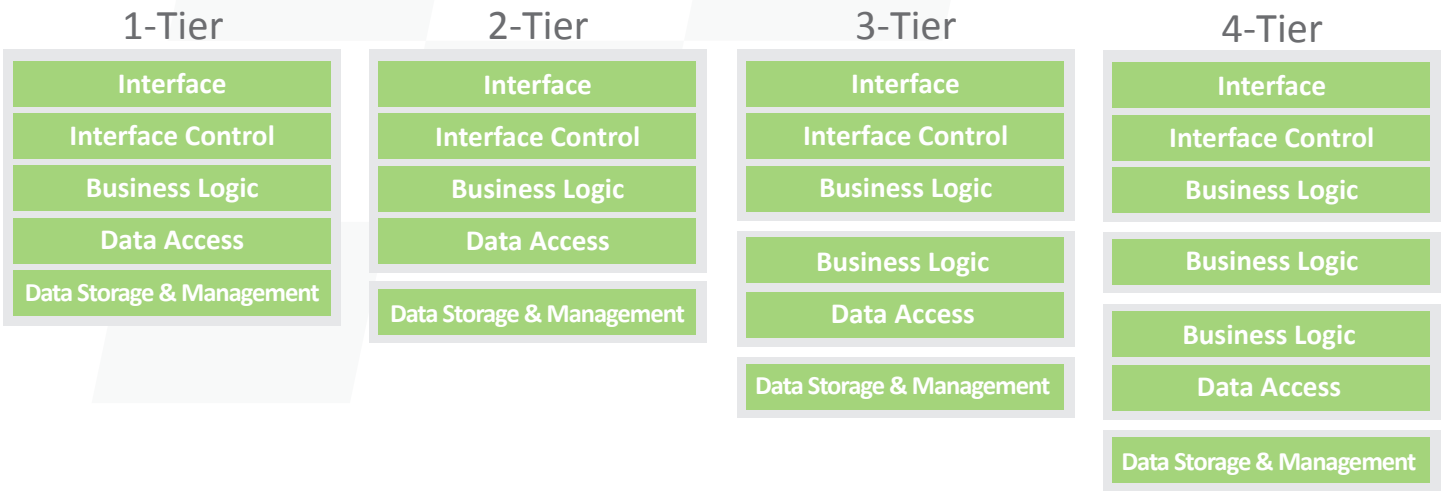
The data portal uses standard .NET technologies such as WCF to manage the network communication. It is implemented using powerful design patterns such as channel adapter, provider and message router.

CSLA 4 supports 1-, 2-, 3- and 4-tier physical deployments for Silverlight and Windows Phone 7 applications. The 1- 2- and 3-tier models employ the same architecture as CSLA 4 on Windows. The 4-tier model is a little different.

When using Silverlight or Windows Phone 7, the data portal is more advanced because business objects literally move between the Silverlight client or Windows Phone 7 device and the .NET server(s). This means the objects are moving between both different platforms and different machines. This is usually entirely transparent to the code, so the resulting functionality and code are the same as in a pure .NET application.

It is important to realize that in the 3- and 4-tier deployment models for .NET, Silverlight and Windows Phone 7, the business code and business objects are fully functional on each machine. This means developers have the flexibility to run logic on the client, the server or both as required to meet the application's needs.

Also keep in mind that the server-side code is the same regardless of whether the code is hosted in Windows Server or Windows Azure.



Overview of CSLA

Standardized Data Access

CSLA .NET is not a data access technology or an object-relational mapping (ORM) tool. However, the data portal (which implements the mobile object concept) does impose a level of standardization and structure around how objects interact with the data access layer or ORM. This standardization remains very flexible, and leaves developers free to use nearly any data access technology they choose, including (but not limited to):

- ADO.NET Entity Framework
- Raw ADO.NET (connections, data readers, etc.)
- DataSet and TableAdapter objects
- LINQ to SQL
- LINQ to XML
- NHibernate and other third-party ORM tools
- Simple file I/O
- Remote XML or JSON services

The data portal supports four models:

- Encapsulated invocation
- Factory implementation
- Encapsulated implementation
- Factory invocation

Perhaps the best model is *encapsulated invocation*, because this enables clean separation between the business and data access layers while maintaining the integrity of the business class by not breaking encapsulation. This is the preferred solution in most cases.

The *factory implementation* model also enables clean separation between the business and data access layers, but requires that the factory object directly interact with private members of the business object, which breaks encapsulation. However, this is a powerful and popular solution.

The simplest approach is *encapsulated implementation*, in which case the data access code is directly contained in the business class. While this is very simple and direct, it doesn't provide clean separation between business and data access layers, which decreases flexibility and testibility.

The most complex approach is *factory invocation*. This is because the object factory concept built into CSLA .NET is already an abstraction layer with another layer of indirection; adding yet another abstraction layer into the mix is typically overkill. However, if an application requires truly extreme flexibility, this solution may make sense.

Summary

CSLA .NET is a powerful, time-tested framework that supports the creation of an object-oriented business layer for distributed application development. It helps developers encapsulate business logic in a set of rich business domain objects, and provides those objects with powerful features around data binding, business logic, validation and authorization.

CSLA 4 supports all common interface types on the .NET, Silverlight, Windows Phone 7, Windows Server and Windows Azure platforms.

More Information

CSLA Consulting – Visit Magenic at http://magenic.com. With five offices spread across the United States, Magenic services the whole country.

Rockford Lhotka's Blog – Visit www.lhotka.net/weblog

CSLA .NET Site – Visit www.lhotka.net/cslanet

CSLA .NET Frequently Asked Questions – Visit www.lhotka.net/cslanet/faq

CSLA Community Forums – Visit http://forums.lhotka.net

Engage Magenic today online at magenic.com
or by calling our sales line at **877.277.1044**

Magenic®
Custom solutions that fit. Guaranteed.