Does anyone have any real-world experience of CSLA? [closed]

Asked 16 years, 4 months ago Modified 10 years ago Viewed 13k times

51

votes

Closed. This question is opinion-based. It is not currently accepting answers.

Closed 3 years ago.

Locked. This question and its answers are locked because the question is off-topic but has historical significance. It is not currently accepting new answers or interactions.

The main web application of my company is crying out for a nifty set of libraries to make it in some way maintainable and scalable, and one of my colleagues has suggested CSLA. So I've bought the book but as :

> *programmers don't read books anymore*

I wanted to gauge the SOFlow community's opinion of it.

So here are my questions:

1. How may people are using CSLA?

2. What are the pros and cons?
3. Does CSLA really not fit in with TDD?
4. What are my alternatives?
5. If you have stopped using it or decided against why?

.net frameworks csla

Share

edited Dec 22, 2014 at 21:51
alex
7,403 ● 10 ● 59 ● 111

asked Aug 18, 2008 at 21:23
Johnno Nolan
29.6k ● 19 ● 113 ● 160

Comments disabled on deleted / locked posts / reviews

23 Answers

Sorted by: Highest score (default)

72 votes

Before I specifically answer your question, I'd like to put a few thoughts down. Is CSLA right for your project? It depends. I would personally consider CSLA for desktop based applications that does not value unit testing as a high priority. CSLA is great if you want to easily scale to an n-tier application. CSLA tends to get some flack because it does not allow pure unit testing. This is true, however like anything in technology, I believe that there is *No One True Way*. Unit testing may not be something you are undertaking for a specific project. What works for one team and one project may not work for another team or other project.

There are also many misconceptions in regards to CSLA. It is not an ORM. it is not a competitor to NHibernate (in fact using CLSA Business Objects & NHibernate as data access fit really well together). It formalises the concept of a *Mobile Object*.

1. How many people are using CSLA?
Based on the CSLA Forums, I would say there are quite a number of CSLA based projects out there. Honestly though, I have no idea how many people are actually using it. I have used it in the past on two projects.

2. What are the pros and cons?
While it is difficult to summarise in a short list, here is some of the pro/con's that come to mind.
Pros:

- It's easy to get new developers up to speed. The CSLA book and sample app are great resources to get up to speed.
- The Validation framework is truly world class - and has been "borrowed" for many many other non-CSLA projects and technologies.
- n-Level Undo within your business objects
- Config line change for n-Tier scalability (Note: not even a recompile is necessary)

- Key technologies are abstracted from the "real" code. When WCF was introduced, it had minimal impact on CSLA code.
- It is possible to share your business objects between windows and web projects.
- CSLA promotes the normalization of *behaviour* rather than the normalization of *data* (leaving the database for data normalization).

Cons:

- Difficulty in unit testing
- Lack of Separation of Concern (generally your business objects have data access code inside them).
- As CSLA promotes the normalization of *behavior*, rather than the normalization of *data*, and this can result in business objects that are named similarly, but have different purposes. This can cause some confusion and a feeling like you are not reusing objects appropriately. That said, once the physiological leap is taken, it more than makes sense - it seems inappropriate to structure objects the "old" way.
- It's not "in fashion" to build applications this way. You may struggle to get developers who are passionate about the technology.

3. After reading this does CSLA really not fit in with TDD?

I haven't found an effective way to do TDD with CSLA.

That said, I am sure there are many smarter people out there than me that may have tried this with greater success.

4. What are my alternatives?
Domain-Driven-Design is getting big push at the moment (and rightfully so - it's fantastic for some applications). There are also a number of interesting patterns developing from the introduction of LINQ (and LINQ to SQL, Entity Framework, etc). Fowlers book PoEAA, details many patterns that may be suitable for your application. Note that some patterns are competing (i.e. Active Record and Repository), and thus are meant to be used for specific scenarios. While CSLA doesn't exactly match any of the patterns described in that book, it most closely resembles Active Record (although I feel it is short-sighted to claim an exact match for this pattern).

5. If you have stopped using it or decided against why?
I didn't fully recommend CSLA for my last project, because I believe the scope of the application is too large for the benefits CSLA provides.
I would *not* use CSLA on a web project. I feel there are other technologies better suited to building applications in that environment.

In summary, while CSLA is anything but a *silver bullet*, it is appropriate for some scenarios.

Hope this helps!

Share

answered Aug 18, 2008 at 22:32

Brad Leach
17k ● 18 ● 74 ● 88

38 Absolutely disagree that its "It's easy to get new developers up to speed" - In my experience people find it very convoluted at first, and its not immediately obvious what the convolution is giving you. – Dan Jul 22, 2009 at 10:22

4 Great answer. Very objective. – Seth Spearman Jan 26, 2010 at 18:08

4 "Difficulty in unit testing" - How so? "Lack of Separation of Concern (generally your business objects have data access code inside them)." - There is nothing preventing you from using the repository pattern with it. "As CSLA promotes the normalization of behavior, rather than the normalization of data, and this can result in business objects that are named similarly, but have different purposes." Correct, but the goal is to avoid tighty coupling of classes. – J.C. Apr 12, 2010 at 19:05

No - not easy if no one in the group doesn't get it, but once there is ONE reason why knows the magic if the DataPortal call, it's easy. – Tony Trembath-Drake Jan 22, 2011 at 11:07

.. and I've unit tested it both core CSLA and from a UI view (WPF) - without too much trouble - perhaps setting up a mokeDB is tricky, but any real TDD of substance will take time. – Tony Trembath-Drake Jan 22, 2011 at 11:08

23 votes

After reading all the answers, I've noticed that quite a few people have some misconceptions about CSLA.

First, **CSLA is not an ORM**. How can I say that so definitely? Because Rockford Lhotka has stated it himself many times in interviews on the *.NET Rocks* and *Hanselminutes* podcasts. Look for *any* episode where Rocky was interviewed and he'll state it in no uncertain terms. I think this is the most critical fact for people to understand, because almost all the misconceptions about CSLA flow from believing that it is an ORM or attempting to use it as one.

As Brad Leach alluded in his answer, CSLA objects model behavior, although it may be more accurate to say that they model the behavior of data, since data is integral to them. CSLA is not an ORM because it's completely agnostic about how you talk to your data store. You *should* use some kind of data access layer with CSLA, perhaps even an ORM. (I do. I now use Entity Framework, which works beautifully.)

Now, on to unit testing. I've never had any difficulty unit testing my CSLA objects, because I don't put my data access code directly into my business objects. Instead, I use some variation of the repository pattern. *The repository is consumed by CSLA, not the other way around.* By swapping in a fake repository for my unit tests and using the local data portal, *BOOM!* it's simple. (Once Entity Framework allows the use of POCOs, this will be even cleaner.)

All of this comes from realizing that CSLA is not an ORM. It might consume an ORM, but it itself is not one.

Cheers.

UPDATE

I thought I'd make a few more comments.

Some people have said that CSLA is verbose compared to things like LINQ to SQL and so on. But here we're comparing apples to oranges. LINQ to SQL is an ORM. It offers some things that CSLA does not, and CSLA offers some things L2S does not, like integrated validation and *n*-tier persistence through the various remote data portals. In fact, I'd say that last thing, *n*-tier persistence, trumps them all for me. If I want to use Entity Framework or LINQ to SQL over the net, I have to put something like WCF in between, and that multiplies the work and complexity enormously, to the point where I think it is *much* more verbose than CSLA. (Now, I'm a fan of WCF, REST and SOA, but use it where you really need it, such as when you want to expose a service to third parties. For most line-of-business apps, it isn't really needed, and CSLA is a better choice.) In fact, with the latest version of CSLA, Rocky provides a `WCFDataPortal`, which I've used. It works great.

I'm a fan of SOLID, TDD, and other modern software development principles, and use them wherever practical. But I think the benefits of CSLA outweigh some of the objections of those orthodoxies, and in any case I've managed to make CSLA work quite well (and easily) with TDD, so that's not an issue.

Share

edited Aug 3, 2009 at 21:57

answered Aug 2, 2009 at 17:40

Gregory Higley
16.5k ● 9 ● 70 ● 100

2 You have hit many misconceptions right on the head! Well put! – J.C. Apr 12, 2010 at 19:08

1 Heh, it is no coincidence that you also hit an ORM question I had right on the head. stackoverflow.com/questions/2625098/… – J.C. Apr 13, 2010 at 21:49

I've moved from the MS world to the Mac/Unix world, but before I left I discovered the joy of using CSLA with Dapper as the ORM. Using a fat ORM like EF works well, but is just overkill for CSLA. – Gregory Higley Oct 7, 2014 at 18:08

19 votes

Yes, I (um, we) used it extensively to model our business process logic that was primarily databound forms in a windows forms application. The application was a trading system. CSLA is designed to be at that layer just below the UI.

If you think about your standard complex line-of-business application you may have a form with many fields, many rules for those fields (including cross-field validation rules), you may invoke a modal dialog to edit some child object, you may want to be able to be able to cancel such dialogs and revert back to a previous state. CSLA supports this.

It's cons are that it has a bit of a learning curve.

The key thing to remember is to use CSLA to model how a **user** interacts with forms on some application. The most efficient way for me was to design the UI and understand it's flows, behaviour and validation rules before building the CSLA objects. Don't have your CSLA objects drive UI design.

We also found it very useful to be able to use CSLA business objects server side to validate objects sent from clients.

We also had built in mechanisms to perform validation asynchronously against web service (i.e. checking the credit limit range of a counterparty against a master).

CSLA enforces a strong seperation between your UI, BusinessLogic and Persistance and we wrote a load of unit tests against them. It may not be strictly TDD because you are driving it from UI design, that doesn't mean it isn't testable.

The only real alternative is creating your own model \ business objects, but pretty soon you end up implementing features that CSLA offers out of the box (INotifyPropertyChanged, IDataErrorInfo, PushState, PopState etc.)

Share

answered Aug 18, 2008 at 22:50

user1010

13 votes

I have used CSLA for one project and it worked great and make things much simpler and neater.

Instead of having your team writing business objects in their own different personal style, we know have a common standard to work against.

//andy

Share

answered Oct 30, 2008 at 11:35

Andy
194 ● 2 ● 2

1 I found this to be the greatest pro for it as well. It enforces a standard design and way of thinking across the team.
– DancesWithBamboo Nov 24, 2008 at 14:50

11 votes

I had experience with it several years ago. It is a brilliant architecture, but very complex, difficult to understand or change, and it's solving a problem that most of us developing web based applications don't necessarily have. It was developed more for windows based applications and handling multi-level undo, with a heavy emphasis on transactional logic. You will probably hear people say that since web applications are request-response at the page level, it is inappropriate, but with AJAX-style web apps maybe this argument doesn't hold so much water.

It has a very deep object model, and it can take a while to really wrap your brain around it. Of course, a lot can change in a few years. I would be interested to hear other recent opinions.

All things considered, it would not be my first choice of architecture.

Share

answered Aug 18, 2008 at 21:48

Eric Z Beard
38.4k ● 27 ● 101 ● 147

2 +1. I came on the team a couple of versions into the *very* large win forms CSLA project. It is complex. You should choose CSLA only if "the problems it solves" clearly override all else for you. You have been warned. IMHO lack of unit testability makes CSLA a non starter for critical systems. We have only about 4% test coverage. CSLA is impervious to dependency injection and your custom code is necessarily tightly coupled to the framework with all the inheritance and overriding.
– radarbob Jun 11, 2012 at 14:45

8 votes

In defence of the CSLA, although I do agree with many of the comments that have been made particularly the unit testing one...

My company used it extensively for a Windows Forms data entry application, with a high degree of success.

- It provided out of the box functionality that we didn't have the time or expertise to write ourselves.
- It standardised all of our business objects making maintenance easy and reducing the learning curve for our new developers.

On the whole I would say that any issues that it caused were more than outwayed by the benefits.

UPDATE: Further to this we are still using it for our windows forms app but experiments with using it for other applications such as web sites have shown that it is perhaps to cumbersome when you don't need much of its functionality and we are now investigating lighter weight options for these scenarios.

Share

edited Feb 23, 2009 at 8:56

answered Aug 28, 2008 at 14:42

Simon Keep
10k ● 10 ● 64 ● 79

We are doing something similar, and I'd be interested to hear about some of the lighter weight options you looked at. – Span Feb 15, 2011 at 22:13

@Span - We are using NHibernate. – Simon Keep Feb 23, 2011 at 14:38

7
votes

I joined a team where CSLA is mandatory. We don't use the remote data portal which is the only reason I could agree for usage of this framework. I never bought into the idea of CSLA so maybe that's why I have nothing but issues with it, sorry.

A couple of the issues:

I don't need a road block between my code and the .NET framework which is what this framework felt like to me. I had a limited option of list objects, while I just had to ignore the rich list objects in the .NET framework.

Ii is totally ridiculous that we had these read-only lists and then non read-only lists. So if I had to add an item to the list I had to recreate the entire list...are you serious?

Then csla wants to manage my object state which is fine but nothing is really exposed. Sometimes I want to change an object state manually instead of fetching it again which seems like what csla wants me to do. I basically end up creating many properties to expose options csla didn't think I should have direct access to.

Why can't I just instantiate an object? We end up creating static methods which instantiates an object and passes it back...are you kidding me?

Check the framework source code and it looks to heavy on the reflection code to me.

Reasons to use csla:

- the straight .net framework is too powerful for you.
- your developers are not seasoned and can't grasp the concept of patterns then csla will pretty much have everyone on the same page.
    1. I don't need a road block between my code and the .NET framework...I am stuck with these list

objects.

Share

edited Feb 22, 2011 at 9:07
Johnno Nolan
29.6k ● 19 ● 113 ● 160

answered Feb 22, 2011 at 0:30
Batman Beyond
103 ● 1 ● 1

6
votes

We started using CSLA because we thought it would help with our model layer. Was sort of overkill and mostly all we use now is the SmartDate class, just because we're already linked to the library.

We thought the validation interface would really help us enforce business rules but it didn't work well with WCF and serialization (we're still stuck on version 2.0.3.0, so things might have changed).

Share

answered Aug 18, 2008 at 21:34
slolife
19.9k ● 20 ● 79 ● 126

6
votes

Not to take CSLA of the list, but before using it, research the benefits and make sure they really apply. Will your team be able to correctly/consistently implement it? Remoting and portal dance needed?

I think beyond all the theoretical ponder, it is all about clean/maintainable/extendable/testable code following basic proven patterns.

I counted lines of code needed in a specific domain of a project converted from CSLA. Between all the different CSLA objects(readonly+editable+root+list combinations) and their stored procs it took about 1700 lines, versus a Linq2SQL + Repository implementation that took 180 lines. The Linq2SQL version consisted mostly of generated classes that your team doesn't need to consume book to understand. And yes, I used CodeSmith to generate the CSLA parts, but I now believe in DRY code with single responsibility bits, and the CSLA implementation now looks to me like yesterday's hero.

As an alternative I would like to suggest looking into Linq2Sql/Entity Framework/NHibernate combined with Repository and UnitOfWork patterns. Have a look at http://www.codeplex.com/backgroundmotion

Cheers!

Share

answered Apr 6, 2009 at 7:01

Johannes
189 ● 2 ● 4

6
votes

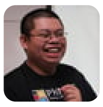
Our company practised CSLA in some of its projects and some of the legacy projects remain to be CSLA. Other projects moved away from it because CSLA violated a plain and simple OOP rule: Single Responsibility Principle.

CSLA objects are self-sustaining, e.g. they retrieve their own data, they manage their own behavior, they save themselves. Unfortunately this meant that your average CSLA object has at least three responsibilities -- representing the domain model, containing business rules, and containing data access definition (not the DAL, or data access implementation, as I previously stated/implied) all at the same time.

Share

edited Sep 22, 2009 at 2:57

answered Aug 18, 2008 at 23:10



Jon Limjap
95.3k ● 15 ● 103 ● 153

3 CSLA encapsulates all logic of retrieval and persistance of itself within the object, but it does not define HOW it is done. You can easily have a DAL that is called by the CSLA object. The purpose of the DataPortal_XYZ methods are for transporting the object across tiers. Last time i checked encapsulation was very OO. – Darren Kopp Aug 19, 2008 at 0:24

1 while you have a point, I find that the fact that you have retrieval and persistence behavior within the object as the violations of SRP (if not OO) *per se*. Unfortunately the argument at hand is more religious than anything else, so I'd settle at agreeing to disagree with you. – Jon Limjap Aug 19, 2008 at 0:51

5 votes

We use CSLA extensively. There are several benefits; first, I believe that every line of business developer should read

Rocky Lhotka's book on Business Objects programming. I've personally found it to be in my top 3 best programming books ever. CSLA is a framework based on this book and using it gives your project access to very high level functionality like n-level undo, validation rules and scalability architecture while providing the details for you. Notice I said "providing" and not "hiding". I've found that the best part of CSLA is that is makes you understand how all of these things are implemented down to the source code without making you reproduce them yourself. You can choose to use as many or few features as you need but I've found that by staying true to the design patterns of the framework, it really keeps you out of trouble. --Byron

Share

answered Jan 28, 2009 at 13:33

Bob Par

161 ● 2 ● 5

Ditto on the benefits of reading the book. Last I read, the new (.NET 3.x) version of the book was not to include the same ground-up tour through the source code that the .NET 2.x version had. That's a shame - I learned a hell of a lot about OOP from that book. Moreso than any other book I've read. – Josh Kodroff Jul 23, 2009 at 4:36

4

votes

We've been using CSLA now for over five years, and we think it works great for constructing business applications. Coupled with code generation you can create business objects in a relative short amount of time and focus your effort on the **meat** of the application.

Share

answered Sep 23, 2008 at 14:22

mattruma
16.7k ● 36 ● 108 ● 174

4
votes

I've been using CSLA since vb5, when it was more of a collection of patterns than it was a framework. With the introduction of.NET, CSLA turned into a full-blown framework, that came with a hefty learning curve. However, the CSLA addresses many things that all business developers tend to write themselves at some point (depending on project scope): validation logic, authentication logic, undo functionality, dirty logic, etc. All of these things you get for free out of the box in one nice framework.

As others have stated, being a framework, it forces developers to write business logic in a similar fashion. It also forces you to provide a level of abstraction for your business logic, so that not using a UI framework such as MVC, MVP, MVVM becomes not so important.

In fact, I would argue that the reason why so many of these UI patterns are so hyped up today (in the Microsoft world) is that people have been doing stuff incredibly wrong for so long (ie., using DataGrids in your UI, sprinkling your business logic everywhere. tisk tisk). Design your middle tier (business logic) correctly from the start, you can reuse your middle tier in ANY UI. Win Form, ASP.NET/MVC, WCF Service, WPF, Silverlight**, Windows Service, ...

But aside from these, the huge payoff for me has been it's built-in ability to scale. The CSLA uses a proxy pattern that is configurable via your config file. This allows your business objects to make remote calls from server to server, without having to write one lick of code. Adding more users to your system? No problem, deploy your CSLA business objects to a new application server, make a config file entry change, and BAM!! Instant scalability needs met.

Compare this to using DTO's, storing your business logic on the client (whatever client that may be), and having to write each of your own CRUD methods as service methods. YIKES!!! Not saying this is a bad approach, but I wouldn't want to do it. Not when there's a framework out there to essentially do it for me.

I'm going to reiterate what other folks have said in that CSLA is NOT an ORM. CSLA forces you to supply your business objects with data. They do not care where you get your data. You can use an ORM to supply your business objects with data. You can also use raw ADO.NET, other services (RESTFUl, SOAP), excel spreadsheets, I can keep going here.

As for your support for TDD, I have never tried using that approach with CSLA either. I have taken the approach where I model my middle tier (ala business objects) using class and sequence diagrams, most often allowing use case, screen and/or process design to dictate. Perhaps a bit old school, but UML has always served me very well in my design and development efforts. I've successfully designed and developed very large and scalable applications still being used today. And until WCF RIA matures, I'll be continuing to use CSLA..

** with some work arounds

Share

answered Mar 5, 2010 at 6:09

Aaron

49 ● 1

4

votes

I'm new to CSLA but I understand the concepts and I already understand that it's not an ORM tool so quit beating that damn drum folks. There are features of CSLA I like but using them feels a bit like there is a magician behind the curtain. I guess if you don't mind not knowing about how it works then you can use the objects and they work fine.

There is a large learning curve for beginners and I think it would benefit greatly by having 5-15 min. videos like Microsoft has for learning the fundamentals. Or how about releasing a companion book with the code instead of getting the code released and taking months to get the book out? Just sayin Mr Lohtka... We started building our stuff before the book and I struggled the whole time. But like I said, I'm new to it.

We used CSLA. We made our objects fit their mold then used 10% of what the framework offered. Object level undo? Didn't use it. NTier flexibility? Didn't use it. We ended up writing enough business rule code that I thought the only thing we were getting out of CSLA was complexity. Some "long in the tooth" developers that know the framework used it as their hammer because they had a nail that needed hitting. CSLA was in their belt and my guess is a lot of proponents of the framework see things from that perspective too.

I guess our seasoned developers are happy because it all makes sense to them. I guess if your organization doesn't have newbie programmers and you guys get bored by writing efficient and simple POCO objects with well formed patterns, then go for it. Use CSLA.

Share

answered Dec 13, 2010 at 23:00

Joe

57 ● 1 ● 1

3

votes

I am using CSLA as the business object framework for a medium size project. The framework has come a long way from the VB6 days and offers an extraordinary amount of flexibility and "out of the box" functionality. CSLA's mobile smart objects makes UI development much easier. However, I agree with others it isn't the right tool for every situation. There is definitely some overhead involved, but also a lot of power. Personally, I am looking forward to using the CSLA Light with Silverlight.

Pros:

- Data technology agnostic[1]
- Large install base and it's FREE!!

- Stable and Logical framework
- Data Access code can be in your objects or in a separate assembly
- Property and Object Validation and Authorization

Cons

- The code can be a lot to maintain[2]
- Probably need a code generator to use effectively
- Learning curve. The structure of CSLA objects are easy to grasp, but the caveats can create headaches.

I'm not sure about test driven design. I don't unit test or test driven design (shame on me), so I don't know if unit tests are different than TDD, but I know that the most recent version of the framework comes with unit tests.

[1] Good thing because data access technologies never stay the same for long.
[2] This has gotten better with recent versions of the framework.

Share

answered Jun 8, 2009 at 4:54

Spirit of '76
41 ● 3

3
votes

A lot of people recommend using Code Generation with CSLA. I'd recommend checking out our set of supported templates as they will increase your ROI immensely.

Thanks -Blake Niemyjski (Author of the CodeSmith CSLA Templates)

Share

answered Aug 5, 2010 at 16:11

Blake Niemyjski
3,577 ● 4 ● 27 ● 41

2
votes

I used it for a project a couple years ago. But when the project was done, I couldn't tell anyone what CSLA did for me. Sure, I inherited from its classes. But I was able to remove that inheritance from almost all classes with no restructuring. We had no use for the N-Tier stuff. The n-level undo was so slow that we couldn't use it. So I guess at the end it only helped us model our classes.

Having said that, other teams have started using it (after a horrid attempt by a team to create their own framework). So there has to be something worthwhile in there, because they're all smarter than me!

Share

answered Sep 23, 2008 at 14:16

Joe
3,847 ● 1 ● 27 ● 38

2
votes

I'm a PHP guy. When we started building comparatively large scale applications with PHP, I started researching on lots of application frameworks and ORMs essentially in PHP world, then in Java and .NET. The reason I also looked at Java and .NET frameworks was not to blindly use any PHP framework, but first try to understand what is really going on, and what kind of enterprise level architectures are there.

Because I haven't used CSLA in a real world application, I can't comment on its pros and cons, but what i can say is Lhotka is one the rare thinkers -I'm not saying just expert- in Software Architecture field. Although the name Domain Driven Design is coined by Eric Evans -by the way his book is also great and i humbly advise to read it- Lhotka was applying domain driven design for years. Having said that, whatever you think about his framework, benefit from his profound ideas in the field.

You can find his talks on dotnetrocks.com/archives.aspx and videos from dnrtv.com/archives.aspx (search for Lhotka).

@Byron What are the other two books you liked?

Share

answered Feb 10, 2009 at 4:12

Ercan

2

John,

votes

We have teams working in CSLA from 2 to 3.5 and have found it a great way to provide a consistant framework so all the developers are "doing it the same way". It is great that most of the low value code is generated and we know when we run unit tests they work out of the box for all the CRUD stuff. We find that our TDD really comes in with the refactoring we do to design, and CSLA doesn't prevent us from doing any of that.

Chris

Share

answered May 6, 2009 at 14:05

ChrisSlee
53 ● 6

2

votes

I last tried to use CSLA in the stone age days of VB6. In retrospect, it would have been more effective if I had used code generation. If you don't have effective code generation tools and a strategy for fitting them into your workflow, they you should avoid frameworks like CSLA, otherwise the features you get from CSLA won't make up for the amount of time you spend writing n lines of code per table, n lines of code per column, etc.

Share

answered May 8, 2009 at 10:51

MatthewMartin
33.1k ● 33 ● 114 ● 171

2
votes

I've used CSLA.NET in few projects now, it was most successfull in a windows forms application which has rich databinding compatabilities (which asp.net application's don't have).

It's main problem is the TDD support like people have been pointing out, this is because of the black-box like behaviour for the Dataportal_XYZ functions and it's inability to allow us to mock the data objects. There have been efforts to work around this issue with this being the best approach

Share

answered Sep 17, 2009 at 12:44

armannvg
1,712 ● 1 ● 15 ● 29

1 Typemock maybe the answer to CSLAs issues with TDD – Johnno Nolan Sep 17, 2009 at 13:53

1
vote

I wanted to use it, but my then lead developer had the idea too much 'magic' was involved...

Share

answered Sep 23, 2008 at 14:19

leppie
117k ● 18 ● 200 ● 300

Was he she referring to Reflection? – Johnno Nolan Sep 24, 2008 at 13:38

0

votes

CSLA is the best application framework that exists. Rocky LHotka is a very but very smart guy. He is writing the history of software development like Martin Fowler, David S Platt, but my favourites writers are Rod Stephens, Mathew mcDonalds Jeff Levinson thearon willis and Louis Davidson alias dr sql. :-) Pros: All design patterns are applied. Cons: Hard to learn, and few samples.

Share

answered Mar 2, 2010 at 6:51

Luis

19 ● 1