Hello!
I'm a passionate web developer and systems architect living in Stockholm, Sweden. I work at Valtech and I am an EMVP.
Read more about me

The Open/Closed Principle – A real world example

Posted November 15 2009 in Software architecture . 7 comments.

Out of the five SOLID principles the Open/Closed Principle is probably the one that I've had the hardest time understanding. However, a while ago though I found some code that I had written years ago that made me think to myself *"Hey, this is clearly violating the Open Closed Principle!"*. I had been thinking of reusing this code in the project I was working on, this blog, but realized that I would have to rewrite it to abide by OCP to satisfy my ambition of writing a flexible and well designed application.

The case – protecting the blog from spam bots

The case was this: I was writing a (ASP.NET MVC) controller that handled adding new comments from visitors to a blog post. Naturally I wanted to stop spam bots from posting comments but I didn't want to force visitors to fill in some CAPTCHA text. So, I decided to check commenting visitors IP address' against Project Honey Pot's IP black list. While that strategy doesn't give bulletproof protection against spam bots it's very easy to implement so I thought I'd start with that and then add other protection, such as Akismet, later on should I need to.

As I had implemented spam bot protection with Project Honey Pot before I dug up that old piece of code. It consisted of a single method that was called when a new comment was about to be saved. My old implementation looked something like this:

```
01. public class EntryController
02. {
03.     public void AddComment()
04.     {
05.         if(ValidateNotSpam())
06.         {
07.             //Save the comment to the database
08.         }
09.     }
10.
11.     private bool ValidateNotSpam(string comment)
12.     {
13.         //Check if the IP-address that made the
               request
14.         //is known as a spammer by Project Honey
               Pot
15.     }
16. }
```

This code has a few problems. First of all it is responsible for doing two things, validating the request and persisting the comment, which isn't optimal. It's dependency on an external service renders it almost impossible to test. It is also clearly violating the Open/Closed principle. Let's say I wanted to extend it so that it also used another service for validating the comment. I would then have no other choice but to modify the ValidateNotSpam method. And if I where to ship the compiled code

Latest posts

The ever-present wisdom of Ayende

Progressive EPiServer Development video from NDC 2011

Page Type Builder 2 Preview 2

Progressive EPiServer Development Slides

Follow me on Twitter

@ampgt Considering we both seem to attend NDC each year, I'll book a date in June 2012 then :) 5 days ago

@ampgt Yeah, just know you'd have at least one appreciative reader if you did :) 6 days ago

Bookmarked: Visualizing the DOM nodes depth and coordinates at Tilt http://bit.ly/pCVqix 6 days ago

view more follow me

Latest comments

applicant tracking software wrote "Joel, huge help. I've been working on a new project to crea..." on A simple example of the Open/Closed Principle

Kostiantyn wrote "This does the job a bit more straightforward and basically w..." on Getting the first day in a week with C#

Kostiantyn wrote "This does the job a bit more straightforward and basically w..." on Getting the first day in a week with C#

About this site

This blog is built with EPiServer Community, EPiServer CMS, ASP.NET MVC and a bunch of other great products. The source code is available for download at the projects page, where you also can read more about this site and my other projects.

read more

My latest bookmarks

Dean Edwards' Packer - Javascript compression .NET library

AnythingSlider | CSS-Tricks

as a product the users of it would have no way to add another service. Hence, it's not possible to extend the class, that is it isn't **open for extension**, and to modify behavior that isn't at the core of what the class should do (persisting comments) requires the actual class to be changed, it isn't **closed for modification**.

A better solution – abiding by the Open Closed Principle

Instead of using that old class I decided to come up with a better solution, which (I think) abides by the Open/Closed principle and which is much easier to extend. First of all I created an interface called ICommentValidator with a single method.

```
1. public interface ICommentValidator
2. {
3.     bool Validate(string authorIP, string
          content);
4. }
```

I then let my controller require an array of instances of ICommentValidator as a constructor parameter.

```
01. public class EntryController : Controller
02. {
03.     private ICommentValidator[]
          _commentValidators;
04.
05.     public EntryController(ICommentValidator[]
          commentValidators)
06.     {
07.         _commentValidators = commentValidators;
08.     }
09. }
```

I also modified the ValidateNotSpam method so that it invoked the Validate method on all of the ICommentValidators supplied to the constructor.

```
01. public class EntryController
02. {
03.     private bool ValidateNotSpam(string comment)
04.     {
05.         foreach (ICommentValidator validator in
              _commentValidators)
06.         {
07.             if
                  (!validator.Validate(Request.ServerV
                  comment))
08.                 return false;
09.         }
10.
11.         return true;
12.     }
13. }
```

Finally I created a new class, ProjectHoneyPotCommentValidator that implemented the ICommentValidator interface and supplied an array with an instance of that class to the EntryControllers constructor. Actually I didn't do that last part myself though, but instead let StructureMap instantiate the EntryController with instances of all ICommentValidator that it could find in the application domain, but that's sort of beside the point.

```
1. public class ProjectHoneyPotCommentValidator :
      ICommentValidator
2. {
3.     public bool Validate(string authorIP, string
          content)
4.     {
5.         //Check if authorIP is blacklisted with
6.         //Project Honey Pot
7.     }
8. }
```

Pikachoose jQuery Image Gallery

Programming is not a craft « DanNorth.net

techrash: An approach to Content Modelling with EPiServer CMS 6

my delicious profile

Favorite blogs

Ayende Rahien

Steven Sanderson

Scott Allen

Herding Code Podcast

CodeBetter.com

Karl Seguin

Conclusion

In order to satisfy the Open/Closed Principle I had to make quite a few changes and add some extra abstraction but as a result my EntryController class is now much more flexible while at the same time it's more stable as it's less likely that it will have to be changed in the future. Choosing this implementation might have cost me about 15 minutes extra development time, but it didn't take me long until that paid off when I had to add more validation functionality which was now a breeze. I also think that this code is much easier to maintain.

Any feedback is much appreciated! Did this example make sense? Was it a good example of the Open Closed Principle?

PS. For updates about newposts, sites I find useful and the occasional rant you can followme on Twitter. You are also most welcome to subscribe to the RSS-feed.

Tags: asp.net | asp.net mvc | building this site | c# | dependency injection | ioc | open closed principle | solid principles

Add a comment!

Related posts

- A simple example of the Open/Closed Principle
- Slides from tonight's EPiServer meetup - An introduction to SOLID
- One small step for EPiServer, one giant leap for our code

Comments

Jeff Doolittle
1 years ago

Good example of OCP, Joel. Although, to clarify for those learning OCP, you do not necessarily need the parameter to be an array in order to apply OCP. Suppose the parameter simply accepted ICommentValidator rather than ICommentValidator[]. If in the future you needed to apply multple comment validators, you could create a CompositeCommentValidator that applied rules from multiple validation providers.

[Pro]: I like the cleaner constructor for the EntryController. Requiring an array feels ugly to me. Of course since StructureMap will scan for all implementors, it certainly makes the hook up easy. But I'm not so sure I want my APIs to be affected by my DI container choice.

Which leads to a [Con]: You could no longer wireup StructureMap by simply scanning for all implementors of ICommentValidator. If CompositeCommentValidator received an array of validators in its constructor, you'd get a StructureMap error since CompositeCommentValidator implements ICommentValidator and this self-reference would be problematic.

My main point is that you don't have to accept array parameters in order to comply with OCP.

Good point Jeff. Thanks for the feedback!

Joel Abrahamsso
1 years ago

Daniel Berg
1 years ago

Great post, Joel! Love the framework mindset. :)

Ted Nyberg
1 years ago

Great post, I'd love to see more writings on patterns! In this case I totally agree with your architectural approach!

I can also see that your approach for comment validation is fairly similar to that of BlogEngine.NET (which I still run on since I don't have a slick EPiServer Community blog like you) ! Oh, the envy... :)

Bob Santos
1 years ago

I am not a .Net fan as I am more of an open source guy, mainly developing using Java but I would like to say you got one good and simple to understand example there Joel. Good job!

Moreover, I agree with Jeff on the con of using arrays as parameters since its better to use interfaces polymorphically.

My take on OCP is that more than usual you will get this if you code against an interface rather than a concrete class.

Again, good post! Hoping for more posts on patterns.

Naveen Prabhu
10 months ago

Good example joel, thanks for sharing :)

Nice example. I am going to show your example as part of my presentation. Thanks, Tarun

Tarun
5 months ago

Add a comment

Name

Email (required, but will not be published)

Website

Comment

Allowed tags: <b>, , <quote cite="">, <code>, <c-sharp-code>, <css-code>, <sql-code>, <xml-code>, <javascript-code>. If you want to display code examples, please remember to write < for < and > for >.

Preview

Notify me via e-mail when new comments are added

Add Comment

Categories

Life

Everyday reports Family Gaming Travel

Software Development

ASP.NET C# EPiServer CMS EPiServer Community Javascript Scala SEO SiteSeeker Software architecture SQL Testing Tools Visual Studio

Tags

.net architecture asp.net asp.net mvc bdd books building this site c# caching code samples conferences configuration custom properties dependency injection devsum download epiabstractions epimvc epimvp episerver episerver cms episerver community episerver meetup functional programming furniture fxcop git github goto hardware humor installation inversion of control ioc javascript jquery json katas keyboard koans log4net lucene lucene.net machine specifications malmö mocking model view presenter moq mspec ndc 2009 ndc 2010 ndc 2011 open closed principle open source øredev øredev 2010 oslo page structure builder page type builder performance phantom presentation progressive episerver development qcon rest rhino mocks ruby scala search engines slides solid principles sql structure map tdd testing travel t-sql tutorial twitter uniform access principle unit testing valtech video visual studio web forms mvp wedding wpf xforms xml sitemap xunit

Latest posts

The ever-present wisdom of Ayende

Progressive EPiServer Development video from NDC 2011

Page Type Builder 2 Preview 2

Progressive EPiServer Development Slides

Going west – Off to NDC again