Does Domain Driven Design require to implement the business logic outside the domain objects.

Asked 13 years, 11 months ago Modified 10 years, 8 months ago

Viewed 371 times

1

The model of the domain are my entities used as `POCO`s which means no base class, no interfaces around and no Attributes.

So the business logic like validation rules must be outside of the entities. (`Anemic Domain Model`)

Would this comply with Domain Driven Design?

logic domain-driven-design

Share

Improve this question

Follow

edited Apr 3, 2014 at 7:59

Albert Laure
1,722 ● 5 ● 21 ● 51

asked Dec 29, 2010 at 22:44

Elisabeth
21.1k ● 55 ● 208 ● 325

2 Answers

Sorted by: Highest score (default)

2

No. Not really.

Main aim of domain driven design is to capture and encapsulate business domain in model explicitly as possible. Business would always contain behavior, therefore - Your objects are supposed to have behavior too.

> The model of the domain are my entities used as POCOs which means no base class, no interfaces around and no Attributes.

...and no c#, .net clr should be used. that's infrastructure, right? ;)

Those are tools to express Your model. You should try to keep noise level down, separate Your model, but You won't be able to runaway completely because it's a model of real life expressed in programming language and technology around it.

Btw, You might want to investigate idea of never allowing domain object to be in invalid state. And if it feels that this particular kind of validation does not relate to business - it is not supposed to be in domain model first of all.

Share Improve this answer Follow

answered Jan 1, 2011 at 21:55

Arnis Lapsa

47.5k ● 29 ● 118 ● 196

1

That's a really philosophical question. I really want to give an equally philosophical answer, so here goes:

As I have understood domain driven design, the most important thing is that whoever knows something, does things with that knowledge. I believe this to be intertwined with this article.

With this in mind, your plain old objects should have the means of performing their "life or death" - important tasks (which makes your solution wrong).

However, another way of looking at it would be that these plain old objects are the tiniest available sets of data, almost like primitives. What happens then is that the objects owning these data objects, they are the actual model objects within the domain driven design. and they don't have to correlate perfectly (which would make your solution correct).

This could easily happen if the model and the data layer are designed by two completely independent designers, or if one person is capable of switching hats. Or maybe be a little.... wohoooooo D: i'm thinking this could be a good thing though! let me give an example:

A forum

> What do we need? We need users, boards, threads, and posts. The last 3 all have a *"one to many"* relationship in the data layer. One board has many threads, and one thread has many posts. One user also has many posts, and one user starts many threads *(could be derived by finding the author of the first post in a thread, so might not have to be stored in the data layer)*. But what is going on in the presentation layer?
>
> When viewing a board, we will want to see all available threads in that board. but we won't be satisfied with seeing the name of the thread, and the name of the user who started it. We also want to see the number of posts in each thread, plus the name of the last poster in the thread, and the time of that posting.

We are now looking at a model object which is somewhat out of sync with the data layer. It will contain business logic to calculate the needed data from the given data objects, and then it will be able to load some sort of view with the data that the view wants. No getters or setters will be needed in the model, so capsulation is never broken. The model object conforms to the domain, which should be dependant on the usability demands, not the limitations of data storage. The data objects conform to the old data storing style.

This would give us a data abstraction layer (with the pocos), mvc, **and** domain driven design. win? :)

Share Improve this answer Follow

edited Jun 20, 2020 at 9:12
Community Bot
1 ● 1

answered Dec 30, 2010 at 0:03
davogotland
2,777 ● 1 ● 17 ● 20

I guess so... and I have to buy a DDD book :P Any recommendation? – Elisabeth Jan 8, 2011 at 18:49

sorry, no :(there is one book i'd like to recommend, but it's not really a book on domain driven design: "head first design patterns". the whole book is always talking about dividing responsibilities in the model layer, which makes it kind of on topic ^^ – davogotland Jan 9, 2011 at 17:39