SOFTWARE ENGINEERING

# CRUD operations in DDD

Asked 4 years, 4 months ago Active 4 years, 4 months ago Viewed 3k times

3

I'm designing an application with DDD. I'm moving from flat POCO objects to strong domain models, so my question is:

Would I have to call my basic CRUD operations (located in my repository layer) from controllers directly, without passing through the domain layer? I can't see any added value to doing that, but I'm not sure if it's inside the DDD practices make that direct call.

design-patterns mvc domain-driven-design asp.net-mvc patterns-and-practices

edited Nov 10 '15 at 3:31
Robert Harvey
182k ●48 ●419 ●629

asked Nov 10 '15 at 2:51
Franco
133 ●1 ●5

## 3 Answers

Active Oldest Votes

The typical entry point for this in DDD is an Application Service. Application services orchestrate calls to repositories and domain objects. They also know about the current execution state and often control the overarching business transaction through a unit of

5

work that is committed at the end of the service method.

For example :

```
Create new domain object
Add it to Repository
Commit UoW

or

Get domain object from Repository
Modify it
Commit UoW

etc.
```

The application service can be called from a Controller. In some implementations it *is* the controller, when people don't want to bother an additional abstraction layer. But that can lead to a Fat Controller.

> my basic CRUD operations (located in my repository layer)

While C, R and D are part of a Repository interface, U doesn't have to if you have a Unit of Work. Update of all changed domain entities in the UoW will be done automatically on `UoW.Save()`.

edited Nov 10 '15 at 9:38

answered Nov 10 '15 at 9:31
guillaume31
7,447 ● 15 ● 29

I forgotten the role of Application Services here. So, I think that, for now, the best option is making calls to that layer, and let it manage the transactions. For that purpose, do you recommend exposing a SaveChanges() method in the Repository interface to call it from App Services? (it would be the "Commit UoW" you used in your example) Also, I ussually had two separated methods for Create and Update, but I think I'm going to implement a Save method for the sake of simplicity. – Franco Nov 10 '15 at 13:19

No, I wouldn't do that, I put `SaveChanges()` in my UoW interface instead. I know some people do have it in their repository, but I feel they are mixing two very different responsibilities inside it. To me a Repository shouldn't be about "changes" and shouldn't expose a way to conclude a transaction by flushing things to a persistent store. It's supposed to behave like an in-memory collection. – guillaume31 Nov 10 '15 at 14:22

I see the point, thanks. I'll investigate the UoW more deeply to improve my Application Service layer. However, the main answer about the direct calls to Repository is answered. – Franco Nov 10 '15 at 14:47

0

> Would I have to call my basic CRUD operations (located in my repository layer) from controllers directly?

No, there's nothing that forces or requires you to do that. You can add as many layers between the controller and your database as you wish, or split it into separate repositories if you like.

edited Nov 10 '15 at 3:31

answered Nov 10 '15 at 3:22
Robert Harvey
182k ● 48 ● 419 ● 629

Maybe I shouldn't have specified that I'm using Microsoft technologies. What I mean is if, from a DDD point of view, is that correct. – Franco Nov 10 '15 at 3:26

What do you mean by "correct?" Make sure you're structuring your questions in a way that can't simply be answered "yes" or "no." I edited my answer to conform to your question edit. – Robert Harvey Nov 10 '15 at 3:29

0

> Would I have to call my basic CRUD operations (located in my repository layer) from controllers directly, without passing through the domain layer?

I think this is ok. Just take a note, that if you have your application split into three layers (UI, domain logic and storage), in your case UI will depend both on domain logic and storage layers.

If persistence is used only from domain logic layer, UI layer is left depending on domain logic layer only.

Without seeing your actual application it's hard to say, if that is good or bad.

answered Nov 10 '15 at 9:24
Vladislav Rastrusny
1,824 ● 10 ● 14