Why the domain model should not be used as resources in REST API?

Asked 4 years, 7 months ago Active 1 year, 6 months ago Viewed 4k times

13

5

I came across a statement that the domain model designed in accordance with DDD should not be used as resources in a REST API (source).

It is clear that a REST API is a contract of the application while the domain model is part of the implementation and therefore it is the best to keep these two things separate, so that a change in the domain model does not automatically imply a change in the REST API.

However, I think in case of small projects (where the REST API has just one consumer - the javascript frontend, developed by one team) the benefits of having separate models does not justify the cost of separating the models (different classes - domain model and resource representations and mapping code between the models). Obviously the domain layer cannot have any references to REST specific infrastructure code (to keep separation of concerns).

Should the domain and the REST models be separated?

api rest domain-driven-design domain-model

edited Nov 28 '15 at 12:19
theDmi
14.8k 6 59 118

asked Nov 28 '15 at 11:21
Adam Siemion
13.8k 4 40 80

5 Answers

Active Oldest Votes

11

When using DDD, the REST API should always be separated from the domain model.

The main reason for this is simplification - you don't want to leak the complexity of the domain model through the API to the clients. Otherwise, clients need to know about the nuances and intricacies of your domain, which most probably makes the API hard to use.

And the main driver for using DDD is a complex problem domain, so this is always a problem.

> However, I think in case of small projects (…) the benefits of having separate models does not justify the cost of separating the models (…).

I agree that there are projects where a separated domain model and REST API is over-engineering. However, these cases aren't candidates for DDD, because you will not benefit from DDD enough to justify its cost.

14.8k 6 59 118

5

> Why the domain model should not be used as resources in REST API?

Because the web is a totally different world than your core domain layer. The methods in your entities are especially hard to translate since HTTP only has a handful of verbs. If you want to expose your application via REST, you have to shoehorn your domain processes into HTTP and that usually means making compromises and designing resources that are different from your domain entities.

Of course you should find terms from the Ubiquitous Language in the messages exchanged between the HTTP client and server and in the Domain Application Protocol if you're doing HATEOAS, but the web will necessarily distort your domain representations.

The point of REST is not to re-create a high fidelity model of your domain and its processes, but to deliver them in an HTTP compliant way while losing as little as possible in translation. Yet it remains a translation.

answered Dec 1 '15 at 16:22
guillaume31
12.3k 26 41

1

I think another thing to consider is who uses your REST API. If you are developing an frontend for an app then you could say that everything is still happening within 1 bounded context. Just some part lives in client/javascript. In that case i think it does make sense to expose your model in your rest api.

in that case the REST api might just be a means of communication with your domain services for example.

answered Nov 29 '15 at 7:19
Batavia
2,370 10 16

1

You can hook your business logic into your REST resources based on your domain model. For example, whenever someone sets is_published = 1 you can notify an admin, do extra validation, etc by hooking into an event or a mutator. Sometimes things may be too complicated and weird to do that way, so you can flag certain attributes as unmodifiable then create custom actions to modify them that you expose, if that makes sense. I think that if you design properly you don't even need these "custom actions" though. Facebook doesn't use any with the Graph API I don't think. I'm thinking about developing a framework based around just exposing the Model layer, I still think it's a good idea.

answered Nov 4 '16 at 3:49

1

I think the main benefit of REST APIs is providing a service for (typically server side and not SPA) 3rd party REST clients. If you use HATEOAS and other self-descriptive message solutions like RDF, then the REST clients will break a lot harder due to changes in the REST API. For small projects - *"where the REST API has just one consumer - the javascript frontend, developed by one team"* - I don't think it's worth the effort to have a proper REST API. Most people use the simplified version of it, which I call CRUD API, those might be good for these projects.

There can be an 1 to 1 mapping between the CRUD resources and the domain objects of an anaemic domain model. If we are talking about real objects (instead of data structures) with more than just CRUD methods, then you have to translate between resource.verb and object.method, for example:

```
POST /dogs/{id}/barking
 -> domain.dog.bark()
```

If we are talking about more complex things involving multiple domain objects and unit of work (transactions), then you need to add another layer for the application services, otherwise you would move the whole complex operation including the transaction handling to the client. In those cases you translate between resource.verb and applicationService.operation for example:

```
POST /dogs/{id1,id2,..}/barking
 -> dogService.multiDogBark(...)
 -> UnitOfWork{domain.dogs[ids[i]].bark()}
```

I think most of the developers confuse this CRUD services + anaemic domain model approach with the REST services + domain model approach, that's why this question is asked and that's why there are many "REST" frameworks which add 1:1 domain object - CRUD resource mapping or maybe even ORM entity - CRUD resource mapping. I find this trend very destructive and I think the main cause that developers learn certain technologies only superficially from short articles or Q&A sites instead of reading books and dissertations where they can get deep knowledge in the actual topic. I think this is an Y+ generation problem, that we are losing the ability to read long texts because of the usage of the digital technology. We are conditioned to instant rewards instead of the delayed reward a long text gives...

answered Dec 13 '18 at 13:54

inf3rno
18.8k 9 86 159