Repository vs DAO

published on 01 November 2012 in Best practices

You're right, it's confusing! You have the repository which abstracts the persistence access details and you have the Data Access Object (DAO) which used to .. abstract persistence access details. So, is there a difference between those two, are they the same thing with different name?

Well, once again the devil is in the details. In many apps, especially data-centric, DAO and a repository are interchangeable as they do the same job. But the difference starts to show up when you have more complex apps with complex business behavior. While the Repository and DAO will strict abstract the data access they have different intentions in mind.

A DAO is much closer to the underlying storage , it's really data centric. That's why in many cases you'll have DAOs matching db tables or views 1 on 1. A DAO allows for a simpler way to get data from a storage, hiding the ugly queries. But the important part is that they return data as in **object state**.

A repository sits at a higher level. It deals with data too and hides queries and all that but, a repository deals with** business/domain objects**. That's the difference. A repository will use a DAO to get the data from the storage and uses that data to restore a business object. Or it will take a business object and extract the data that will be persisted. If you have an anemic domain, the repository will be just a

We have the definition: "**Mediates between the domain and data mapping layers** using a collection-like interface for accessing domain objects. A system with a complex domain model often benefits from a layer [...] that **isolates domain objects** from details of the database access code".

So the intention of the repository is to isolate the domain objects from the database access concerns. This is the important part. The fact that the repository is using a collection-like interface, doesn't mean you MUST treat it as a collection and you MUST have ONLY Add/Remove/Get/Find functionality. Really, it's not a dogma and you don't have to apply it by the letter.

Let's think a bit. The pattern is used to acheive separation of concerns and to simplify things. If you force yourself to use a collection and a criteria and some unit of work (things that made ORMs very popular) in probably 99% of cases you just complicate things. And it's quite ironic that people are using ORMs or some other Unit of Work approach to apply the repository pattern forcing their domain objects to include data access related stuff (like making properties virtual for NHibernate or needing to provide a parameterless constructor etc). What's the point in using the repository pattern if you couple your domain objects to data access details?

Back to Repository and DAO, in conclusion, they have similar intentions only that the Repository is a higher level concept dealing directly with business/domain objects, while DAO is more lower level, closer to the database/storage dealing only with data. A (micro)ORM is a DAO that is used by a

;

Sapiens Works

Converting tech into business advantage

Topics About

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

ALSO ON SAPIENSWORKS

CRUD vs DDD – Sapiens Works – Maintainable code is a business advantage

2 comments • 3 years ago

Avatar **Brian Sanders** — I realize this post is over 4 years old, but the whole "replace instances of 'pre' with 'code'" thing is pretty annoying

DDD Decoded - Bounded Contexts Explained – Sapiens Works – Maintainable code is a business

3 comments • 3 years ago

Avatar **Mike** — An aggregate is a bunch of rules and aggregates don't overlap so data doesn't belong to one, the 'data' is a part of the business state and with CQRS it's part of a read

DDD Decoded - The Aggregate and Aggregate Root Explained (Part 3) – Sapiens Works – Maintainable

22 comments • 3 years ago

Avatar **Mike** — I'm the opposite of dogmatic and in this case, the DDD dogma doesn't even exist, because it's all about how one identifies the domain abstraction. If there's a dogma,

Microservices - These Are Not The Droids You're Looking For

2 comments • a year ago

Avatar **Mike** — Thanks for your kind words. For now, my blog is the book :) . Unfortunately, the DevTeach 2016 talks weren't recorded.

Sapiens Works

Converting tech into business advantage

Topics About