Should DB layer members be static or instance?

Asked 16 years, 4 months ago Modified 6 years, 5 months ago

Viewed 812 times

3

I've seen projects where the classes in the DB layer have just static functions in them and other projects where those classes need to be instantiated to get access to the member functions.

> Which is "better" and why?

database orm class-design

Share

Improve this question

Follow

edited Jul 21, 2018 at 14:43

Ipsit Gaur

2,927 ● 2 ● 25 ● 38

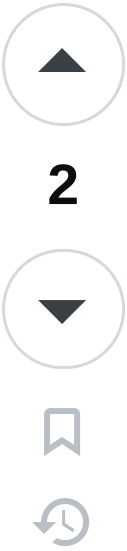
asked Aug 19, 2008 at 14:49

Guy

67.1k ● 101 ● 265 ● 331

6 Answers

Sorted by: Highest score (default)



I like a single object to be correlated to a single record in the database, i.e. an object must be instantiated. This is your basic ActiveRecord pattern. In my experience, the one-object-to-one-row approach creates a much more fluid and literate presentation in code. Also, I like to treat objects as records and the class as the table. For example to change the name of a record I do:

```
objPerson = new Person(id)

objPerson.name = "George"

objPerson.save()
```

while to get all people who live in Louisiana I might do

```
aryPeople = Person::getPeopleFromState("LA")
```

There are plenty of criticisms of Active Record. You can especially run into problems where you are querying the database for each record or your classes are tightly coupled to your database, creating inflexibility in both. In that case you can move up a level and go with something like DataMapper.

Many of the modern frameworks and ORM's are aware of some of these drawbacks and provide solutions for them. Do a little research and you will start to see that this is a problem that has a number of solutions and it all depend on your needs.

Share Improve this answer

Follow

answered Aug 19, 2008 at 15:06

Barrett Conrad

1,878 ● 14 ● 14

1

It's all about the purpose of the DB Layer. If you use an instance to access the DB layer, you are allowing multiple versions of that class to exist. This is desirable if you want to use the same DB layer to access multiple databases for example.

So you might have something like this:

```
DbController acrhive = new DbController("dev");
DbController prod = new DbController("prod");
```

Which allows you to use multiple instances of the same class to access different databases.

Conversely you might want to allow only one database to be used within your application at a time. If you want to do this then you could look at using a static class for this purpose.

Share Improve this answer

Follow

answered Aug 19, 2008 at 14:57

lomaxx

116k ● 58 ● 147 ● 180

0

As lomaxx mentioned, it's all about the purpose of the DB model.

I find it best to use static classes, as I usually only want one instance of my DAL classes being created. I'd rather use static methods than deal with the overhead of potentially creating multiple instances of my DAL classes where only 1 should exist that can be queried multiple times.

Share Improve this answer

Follow

answered Aug 19, 2008 at 15:08

Dan Herbert

103k ● 51 ● 192 ● 221

0

I would say that it depends on what you want the "DB layer" to do...

If you have general routines for executing a stored procedure, or sql statement, that return a dataset, then using static methods would make more sense to me, since you don't need a permanent reference to an object that created the dataset for you.

I'd use a static method as well if I created a DB Layer that returned a strongly-typed class or collection as its result.

If on the other hand you want to create an instance of a class, using a given parameter like an ID (see @barret-conrad's answer), to connect to the DB and get the necessary record, then you'd probably not want to use a

static method on the class. But even then I'd say you'd probably have some sort of DB Helper class that DID have static methods that your other class was relying on.

Share Improve this answer

Follow

answered Aug 19, 2008 at 18:10

Brian G Swanson

1,039 ● 7 ● 17

0

Another "it depends". However, I can also think of a very common scenario where static just won't work. If you have a web site that gets a decent amount of traffic, and you have a static database layer with a shared connection, you could be in trouble. In ASP.Net, there is *one* instance of your application created by default, and so if you have a static database layer you may only get *one* connection to the database for *everyone* who uses your web site.

Share Improve this answer

Follow

answered Aug 30, 2008 at 23:52

Joel Coehoorn

415k ● 114 ● 577 ● 813

-2

It depends which model you subscribe to. ORM (Object Relational Model) or Interface Model. ORM is very popular right now because of frameworks like nhibernate, LINQ to SQL, Entity Framework, and many others. The ORM lets you customize some business constraints around your object model and pass it around with out actually knowing how it should be committed to the database. Everything related to inserting, updating, and

deleting happens in the object and doesn't really have to worry the developer too much.

The Interface Model like the Enterprise Data Pattern made popular by Microsoft, requires you to know what state your object is in and how it should be handled. It also requires you to create the necessary SQL to perform the actions.

I would say go with ORM.

Share Improve this answer

Follow

answered Aug 19, 2008 at 15:00

Nick Berardi

54.8k ● 15 ● 117 ● 136