EF Model First or Code First Approach?

Asked 13 years, 8 months ago Modified 12 years, 5 months ago
Viewed 16k times

41

I know this question has been asked numerous times before as I've read quite a few posts on the topic about the pros and cons etc but I still can't decide which approach is right for me. I'm very new to web programming and come from a SQL DB Admin / report writing background. I've decided to try build my own website which may end up having 30 -40 tables maybe more in the future.

I've looked at both approaches and I do favour Entity Model approach only because I like simplicity of the designer and I like seeing the whole model in front me, it shows the overall picture in one snapshot. Also, me not being a strong programmer I am impressed with the way it generates the POCO's using the DbContext generator template and does all the linking between the classes.

However, although I like the Model First Approach I feel there are some draw backs, I'm not sure if they are actual drawbacks or I just don't know enough about the model

first approach and code first approach as I’m still very new to this.

The Reasons that I am hesitant to use the Model First approach are:

-Mainly because I am struggling to find tutorials on the Model first approach using MVC 3. The best tutorial I’ve found using the DbContext is by Julie Lerman but she doesn’t cover buddy classes which are important for using data annotations and making other changes that aren’t lost when you regenerate the POCOs . Most tutorials MVC 3 related seem to use the Code first approach. Most people say this is because the tutor doesn’t want to focus on EF but rather show more MVC in the tuts. I personally think it’s because Microsoft is championing the Code First methodology over the others :)

-If it’s good practice to create buddy classes why can’t I find many tutorials showing this for MVC 3? Are Buddy Classes another name for View Models? And why can’t I find any tutorials by Microsoft showing these buddy/view models in use with MVC 3?

-I was trying to do a basic 1 to 1 relationship between 2 tables. In model first you have to set the identity keys of each table to the same field rather than using a FK in one of the tables, which may get a little confusing when you have 3 or more tables linked to each other by 1 to 1 relationships. In code first a way around this is use the model builder and set it up manually. I think in MF you

can change the relationship by going into the XML which I am not keen on doing at all.

-More support/help on code first problems

The reasons I am hesitant to use the Code First approach are:

-I’m a novice coder.

-I see it getting quite difficult to keep track of tables and relationships as the project expands.

-There is no Model diagram and I have to say I really do like this idea.

-Mapping entities to the database via configuration classes I find impossible :).

-Updating a table will require a change to the code and the DB. In Model first only one change to the model which will automatically update the DB and Code having said that if you’re using buddy classes you may have to update these as well.

Also now I see people are somewhat combining Code First and Database first approaches, in that you don’t let Code First generate your database but manually create a database and use code first API to EF to get to it.

My head is spinning with all the options and drawbacks and pros and cons. I just want to get on with creating my website and not ponder on which approach to take. Can

anyone give me some insight on which approach they think is best based on what I've said and/or what they think will be more main stream in the future?

Many thanks dave

.net asp.net-mvc-3 entity-framework-4.1 ef-code-first ef-model-first

Share

Improve this question

Follow

edited Jul 9, 2012 at 13:45

abatishchev

100k ● 88 ● 301 ● 442

asked Apr 26, 2011 at 22:39

davey

1,564 ● 4 ● 26 ● 41

4 Answers

Sorted by: Highest score (default)

37

This is too long question. You should break your problem into multiple separate questions next time.

Code-first x Model-first x Database-first

You are a database guy so the best approach for you is an incremental database-first approach where you define the stuff in DB (or VS Database tools) and update your model from the database. This will give you a big control over your database and allow you building the application and the DB incrementally. Why I think you will like it:

- You did SQL DB Admin before - you probably know something about DB and how to design them for a performance - EF will not do anything from this for you. EF will not create indexes for you etc.
- 30-40 tables means that you will not build the model in one shoot. You will start with the small model and continuously grow it. Once you start making changes in DB or adding initialization data you will not want to lose these changes and the data. Code-first allows only deleting the whole database and recreating it from scratch. Model-first allow building the DB incrementally but you need Entity Designer Database Generation Power pack and VS 2010 Premium or Ultimate ($5.000-$10.000).

More about differences between DB first, Model first and Code first. Another answer describes differences between code-first and working with designer.

DbContext API + Database-first + Fluent mapping

I would call this the hardest way to go. You will the define database first and you will use DbContext fluent API or data annotations to define mapping. This requires good understanding of EF and all principles behind the mapping + understanding of default convention used in DbContext API. It will give you nice and explicit control over mapping but it is the most work to do. It is definitely the hardest way to go. Also it is not supposed usage because DbContext API was primarily created for code-first approach.

DbContext API x ObjectContext API

Once you start using EDMX (entity designer) you have a choice to use either DbContext Generator T4 template or POCO Generator T4 template. Decision is up to you - you can use either DbContext API (first template) or ObjectContext API (second template) which is much better documented and you can also use two great books:

- Programming Entity Framework

- Entity Framework Recepies

All I know about ObjectContext API is from these books, authors' blogs and practice + Reflector.

DbContext API doesn't currently have any book. You can check some main sites to get info about it:

- ADO.NET Team blog
- Julia Lerman's blog
- Morteza Manavi's blog

All I know about DbContext API is from these blogs and practice + Reflector.

Even if you are using code first you can still use class diagram to visualize your class diagram (it is not the same as EDMX but it is enough for getting the big picture).

Searching on Stack Overflow or MSDN forum will give you answers on most problems you will have with both APIs.

MVC 3

There is nothing specific with using entity framework with MVC 3. Buddy classes for data validation annotations are considered bad practice. A buddy class is separate class used as a metadata holder applied on the entity. A view model is the class used to transfer data between

controller and view. View model should be specific per view with its own validation annotations because you usually need different validations in different screens of your application when working with the same entity type - for example edit and insert screen can have different validation requirements.

Despite the fact it is not considered as the good practice, adding validation to entities is possible - you can either create buddy class for each your entity manually or you can try to modify T4 template to generate annotations for you directly (well this is hard).

One-to-one relation

Yes EF requires creating one-to-one relation only on top of primary keys. The reason is that EF doesn't support unique keys / constraints. There is no way around this and using unique keys in database will not change it.

Share Improve this answer Follow

edited May 23, 2017 at 12:16

Community Bot

1 ● 1

answered Apr 27, 2011 at 7:16

Ladislav Mrnka

364k ● 60 ● 665 ● 673

Hi Ladislav, thanks for all the comments! You are spot on, I would love to build my app and Db incrementally. I'll probably have a load of Init data and Type Tables and don't want to lose it every time I change the schema. You mentioned Code-

first allows only deleting the whole database and recreating it from scratch but I came across this article Scott Gu's blog explaining that one can create the database and then use Code First weblogs.asp.net/scottgu/archive/2010/08/03/….
– davey Apr 27, 2011 at 21:55

I see on another post that you used the DB first approach how is that working for you? Have you hit any issues / draw backs? Also do you have any examples of using a View Model with DB or Model first. How does one 'wire' it up to the edmx's generated partial classes in the.tt file? thanks
– davey Apr 27, 2011 at 21:55

@davey: I mentioned code first with existing DB in my answer in the section: **DbContext API + Database-first + Fluent mapping**. I just don't call that code first because it is not code first. – Ladislav Mrnka Apr 27, 2011 at 22:05

@davey: Database first is good if you find that EF generated query is slow. You can always build some stored procedure or view to increase performance of your application. I don't have examples of using View models but if you search Stack overflow you will find a lot of them because it is common question. Answers will more probably mention that you will write view models yourselves and use AutoMapper to map between entities and view models. – Ladislav Mrnka Apr 27, 2011 at 22:07

Apols you did mention it. Youre right it does look tricky when trying to do fluent mapping and will probably be extremely difficult for me as I'm very new to this. The idea of using automapper and creating view models sounds like quite abit of extra work. What I like about codefirst is that you can just add data annotations into your POCO's without the worry of the classes being overwritten. I have never used Automapper and will look into it. I struggled to find any examples on Stack Overflow of "View Models" lots of info on them but no code examples showing how to do this in practice. Thanks
– davey Apr 27, 2011 at 23:06

8

It's relatively simple. If you don't care about the database model, use code first. If you do, use Model first (or Database First). It just depends on where your focus lies, data centric or code centric.

Share Improve this answer

Follow

answered Apr 26, 2011 at 22:41

Erik Funkenbusch
93.4k ● 29 ● 200 ● 292

2 Hi Mystere man,i dont think its a case of caring about a database model(design) or not. At the end of the day dont all approaches build a model? Isnt it more a case of chosing what tools to build the model with be it code, a model designer or via a Database? – davey Apr 27, 2011 at 1:57

@davey - Not really. If you have a very strong opinion of how the database model should be designed, you won't be using code first bacause code first will likely not build the model the way you would want it to be. Code first allows you to build your site without thinking about how the database will physically store it. I can't imagine anyone that is concerned

about how the database is designed working that way.
– Erik Funkenbusch Apr 27, 2011 at 4:49

3

> I've looked at both approaches and I do favour Entity Model approach only because I like simplicity of the designer and I like seeing the whole model in front me, it shows the overall picture in one snapshot. Also, me not being a strong programmer I am impressed with the way it generates the POCO's using the DbContext generator template and does all the linking between the classes.

+

> Mapping entities to the database via configuration classes I find impossible :).

= use model first

> If it's good practice to create buddy classes why can't I find many tutorials showing this for MVC 3? Are Buddy Classes another name for View Models? And why can't I find any tutorials by Microsoft showing these buddy/view models in use with MVC 3?

This might be because the code-first is something like a new kid in the block. That's why there are mostly code-first tutorial for MVC3. Model-first is 'much' older and was probably most favourited solution in times of MVC2.

Btw: You already know my oppinion, that you should use, what you like best or find most comfortable (as I told you last time you asked about this), but I just wanted to add something here :)

Edit after comments:

Take a look at these things, that will help you with code first a lot imo:

Creating an Entity Framework Data Model for an ASP.NET MVC Application (1 of 10) Scaffold your ASP.NET MVC 3 project with the MvcScaffolding package

++ these great videos from MIX11 at channel9:
Scott Hanselman showing new stuff in his awesome way as usually
Steve Sanderson showing power of MvcScaffolding

Share Improve this answer Follow

edited Apr 27, 2011 at 2:29

answered Apr 26, 2011 at 22:47

Damb
14.6k ● 6 ● 48 ● 49

Hey Dampe I know ive asked a summarised version of the question before :) After reading a few threads on the topic and seeing the 3 answers posted here the majority of people seem to lean towards codefirst. Its not something i really wanted to hear but I've never seen anyone big up modelfirst over codefirst so I think im going to follow the norm and go with codefirst. Mainly because im new to this and need all the help I can get. I dont want to be halfway through my project and hit an issue with my model that I cant find a solution to or anyone to turn to as the approach is used by a minority. – davey Apr 27, 2011 at 2:08

Well.. it's also a solution. You will learn someting new if you switch to code first and it's true that it's easier to get hints about codefirst nowadays. + www.asp.net has new huge tutorial with code first and is also covering scenarios like updating m:n relationship (which I didn't get working in my solution, but that's another story). Also take a look at `MvcScaffolding`. I'll try to add few links to my answer. – Damb Apr 27, 2011 at 2:12

Done. If you like relaxing and having fun while learning, I would really recommend watching Scott Hanselman first. Then it's up to you. Probably dive into that new 10 part tutorial at asp.net and later on check second video about MvcScaffolding, which is really usefull when doing code first imo. – Damb Apr 27, 2011 at 2:20

thanks dampe, I will have a look into those videos this weekend. Regarding scaffolding I've had a play with this in the MVC 3 tools update. The Add Controller Dialog now supports full automatic scaffolding of Create, Read, Update, and Delete controller actions and corresponding views. Which is dam sweet ! – davey Apr 27, 2011 at 22:07

0

You can use the model first examples from any version of MVC, if that is your primary issue with model first. The way MVC handles "models" is not really different across versions. Sure, there are enhancements to the view model, etc., but you should be fine with older tutorials.

I prefer the code first, as I feel there are database models and domain models and they serve different purposes. Database organization is for performance and size on the database and not to help your application. Having your own model allows you to focus on state needs in your application, regardless of the database.

Now, you can get to this model from model first, but you are more likely to think about the database than your needs in this manner ... esp. if you are a newbie at this.

Just my 2 cents.

Share Improve this answer

Follow

answered Apr 26, 2011 at 22:45

Gregory A Beamer

17k ● 3 ● 26 ● 32

BTW, the model driven approach creates semi-POCOs, if you ask me, as they are heavily attributed. – Gregory A Beamer Apr 26, 2011 at 22:46

hey greg thanks for the response, ive tried looking at the older tutorials but i would prefer to see tutorials that use MVC 3 from start to finish probably because im lazy and I don't want to jump between versions looking at older versions for data access layer and newer ones for the front end, views etc :) Also in the latest version model first uses dbContext

template to generate the classes and i have only found a handfull of tuts on this. You have raised some interesting points about app needs versus db needs. thanks – davey Apr 27, 2011 at 1:50

The MVC 3 tutorials are very useful in the bits that have changed. Most of them require getting a bit deeper than you are now. But, I agree with you, if you can find exactly what you want for MVC 3, it is a better solution. :-)
– Gregory A Beamer Apr 27, 2011 at 18:55