<https://www.c-sharpcorner.com/blogs/code-first-vs-database-first-vs-model-first-approach1>

|  |  |
| --- | --- |
| Code First vs. Database First vs. Model First Approach   * [Mukesh Kumar](https://www.c-sharpcorner.com/members/mukesh-kumar23)      * [Mukesh Kumar](https://www.c-sharpcorner.com/members/mukesh-kumar23)        * Nov 30, 2015      * 163.2k      * [3](javascript:void(0);)      * [5](javascript:void(0);)     In this blog, you will see the logical differentiation between following approach which is used to handle database activity using entity framework.  **Code First Approach**  It is very popular approach between MVC programmers. It has full control over the code rather than database activity. Using this approach we can do all the database activity from the code. So, we can say manual changes to database have been lost to use this and everything is depended on the code.  In this you need to create POCO entities as data model.  **Database First Approach**  If you have already a designed database and you don’t want to do extra effort then you can go with this approach. If your database is ready then Entity Framework will complete his duty and create POCO entities for you with T4 template.  You can modify the database manually and update model from database. So, we can say, entity framework is able to create your model classes based on tables and columns from relational database.  https://www.c-sharpcorner.com/UploadFile/BlogImages/11292015124156PM/entity-framework-design-approaches.png  **Model First Approach**  In this approach, you need to draw your model first and let workflow to decide to generate your database. If you are going to work with small project then it will be best. You cannot make manual change into database because of you don’t have command on database, but you can made the changes into model classes.  So, using the entity framework designer, you can create your model classes and based on your model classes your database will be generated.  Hope you enjoyed this blog. | Cách tiếp cận mã đầu tiên so với cơ sở dữ liệu đầu tiên so với phương pháp tiếp cận mô hình đầu tiên   * [Mukesh Kumar](https://www.c-sharpcorner.com/members/mukesh-kumar23)      * [Mukesh Kumar](https://www.c-sharpcorner.com/members/mukesh-kumar23)        * Ngày 30 tháng 11 năm 2015      * 163,2k \_      * [3](javascript:void(0);)      * [5](javascript:void(0);)     Trong blog này, bạn sẽ thấy sự khác biệt logic giữa cách tiếp cận sau đây được sử dụng để xử lý hoạt động cơ sở dữ liệu bằng khung thực thể.  **Cách tiếp cận mã đầu tiên**  Đó là cách tiếp cận rất phổ biến giữa các lập trình viên MVC. Nó có toàn quyền kiểm soát mã hơn là hoạt động cơ sở dữ liệu. Sử dụng phương pháp này, chúng ta có thể thực hiện tất cả hoạt động cơ sở dữ liệu từ mã. Vì vậy, chúng ta có thể nói rằng những thay đổi thủ công đối với cơ sở dữ liệu đã bị mất để sử dụng điều này và mọi thứ đều phụ thuộc vào mã.  Trong phần này, bạn cần tạo các thực thể POCO làm mô hình dữ liệu.  **Cách tiếp cận cơ sở dữ liệu đầu tiên**  Nếu bạn đã có một cơ sở dữ liệu được thiết kế sẵn và không muốn tốn nhiều công sức thì bạn có thể thực hiện theo phương pháp này. Nếu cơ sở dữ liệu của bạn đã sẵn sàng thì Entity Framework sẽ hoàn thành nhiệm vụ của mình và tạo các thực thể POCO cho bạn với mẫu T4.  Bạn có thể sửa đổi cơ sở dữ liệu theo cách thủ công và cập nhật mô hình từ cơ sở dữ liệu. Vì vậy, chúng ta có thể nói, khung thực thể có thể tạo các lớp mô hình của bạn dựa trên các bảng và cột từ cơ sở dữ liệu quan hệ.  https://www.c-sharpcorner.com/UploadFile/BlogImages/11292015124156PM/entity-framework-design-approaches.png  **Cách tiếp cận đầu tiên theo mô hình**  Theo cách tiếp cận này, trước tiên bạn cần vẽ mô hình của mình và để quy trình làm việc quyết định tạo cơ sở dữ liệu của bạn. Nếu bạn định làm việc với dự án nhỏ thì điều đó sẽ là tốt nhất. Bạn không thể thực hiện thay đổi thủ công đối với cơ sở dữ liệu vì bạn không có lệnh trên cơ sở dữ liệu, nhưng bạn có thể thực hiện các thay đổi đối với các lớp mô hình.  Vì vậy, bằng cách sử dụng trình thiết kế khung thực thể, bạn có thể tạo các lớp mô hình của mình và dựa trên các lớp mô hình đó, cơ sở dữ liệu của bạn sẽ được tạo. |

<https://stackoverflow.com/questions/5446316/code-first-vs-model-database-first>

**Code first**

* Very popular because hardcore programmers don't like any kind of designers and defining mapping in EDMX xml is too complex.
* Full control over the code (no autogenerated code which is hard to modify).
* General expectation is that you do not bother with DB. DB is just a storage with no logic. EF will handle creation and you don't want to know how it does the job.
* Manual changes to database will be most probably lost because your code defines the database.

**Database first**

* Very popular if you have DB designed by DBAs, developed separately or if you have existing DB.
* You will let EF create entities for you and after modification of mapping you will generate POCO entities.
* If you want additional features in POCO entities you must either T4 modify template or use partial classes.
* Manual changes to the database are possible because the database defines your domain model. You can always update model from database (this feature works quite well).
* I often use this together VS Database projects (only Premium and Ultimate version).

**Model first**

* IMHO popular if you are designer fan (= you don't like writing code or SQL).
* You will "draw" your model and let workflow generate your database script and T4 template generate your POCO entities. You will lose part of the control on both your entities and database but for small easy projects you will be very productive.
* If you want additional features in POCO entities you must either T4 modify template or use partial classes.
* Manual changes to database will be most probably lost because your model defines the database. This works better if you have Database generation power pack installed. It will allow you updating database schema (instead of recreating) or updating database projects in VS.

I expect that in case of EF 4.1 there are several other features related to Code First vs. Model/Database first. Fluent API used in Code first doesn't offer all features of EDMX. I expect that features like stored procedures mapping, query views, defining views etc. works when using Model/Database first and DbContext (I haven't tried it yet) but they don't in Code firs