Roy Adams

Entity Framework

Database first means that you build the database before writing any code. This approach has several benefits and a few drawbacks. It allows one to design the database without having to follow the constraints of a codebase. It also allows the designer to create a database that will be easy to interact with when it comes time to start programming an application. This approach could also streamline development process, as EntityFramework builds all the necessary relationships before the coding begins. This allows the developer to focus on the code structure rather than the code structure *and* the database structure at the same time. It creates a level of separation between the two. Another benefit of database first is EntityFramework’s ability to update the models from the database. This allows the database to be tweaked in the future in case other data storage capabilities are required.

The biggest drawback that I see to this approach is that it requires a lot more careful planning. If the database is not designed carefully with all use cases in mind before writing the code it could make things awkward if the database structure needs to be changed at a later time. The code that is written must be based on what exists in the database. When use cases change, while it will be easy to change the database it will not be as easy to change the code.

Code first, on the other hand, requires less of this careful planning. Once satisfactory code is in place, a database can be generated based on the classes of the domain. This way the use cases are already implemented by the programmer, allowing them to “take their hands off the wheel” and let EntityFramework do the heavy lifting of building the database. Since the database is build based on the code, it will only need to change if the classes in the code change drastically. Developers sometimes do not want to worry about the database as it is only a storage unit for data.

The first problem I see with code first involves stored procedures. Stored procedures can be incredibly useful in many situations, but the code-first approach makes stored procedures more difficult to create. Using SQL to create stored procedures may not be an option since the underbelly of the database is relatively unknown to the developer. Another problem is maintaining the database. While Database-first makes it easy to modify the database, code-first is nothing like this: updates could cause data loss.