Code First, Database First y Model First ¿En qué consiste cada uno?

Un ORM es un tipo de software puede funcionar de varias maneras diferentes a la hora de "mapear" las clases de nuestro programa orientado a objetos y las tablas en la base de datos. Cada una tiene un enfoque diferente y es interesante para ciertos casos concretos, además de tener sus beneficios y problemas.

Vamos a dar un **repaso rápido a los 3 modos de trabajo principales de un software ORM** para ver en qué consisten y sus ventajas e inconvenientes.

Database First

En este modo se parte de una base de datos pre-existente con la que queremos trabajar. Es decir, tenemos la base de datos ya diseñada y probablemente con datos y lo que queremos es que el ORM se encargue de generar las clases necesarias y toda la "fontanería" interna para trabajar con ella.

Se suele utilizar si **aprovechamos una base de datos existente y queremos crear una nueva aplicación sobre esta**. Por ejemplo, vamos a modernizar una aplicación antigua y queremos aprovechar todo lo que hay en la capa de datos, o si le añadimos una nueva API por encima a una aplicación que usa otra tecnología para el acceso a datos.

En este caso el ORM creará las entidades orientadas a objetos (las clases) de manera automática, y las actualizará en caso de que haya cambios en la base de datos subyacente. Esto lo realizará a partir de unas **plantillas de clases**, que pueden ser personalizadas.

Ventajas o beneficios de Database First

El trabajo es muy visual. Son muy sencillas de implementar, pues casi todo va en automático y parten de la base de datos.

Si tienes costumbre de trabajar con bases de datos, llevando a cabo primero el diseño Entidad-Relación, y te sientes menos cómodo con código de programación, entonces estarás como en casa.

Se lleva bien con proyectos de datos grandes, con muchas tablas, convenientemente repartidas en varios modelos.

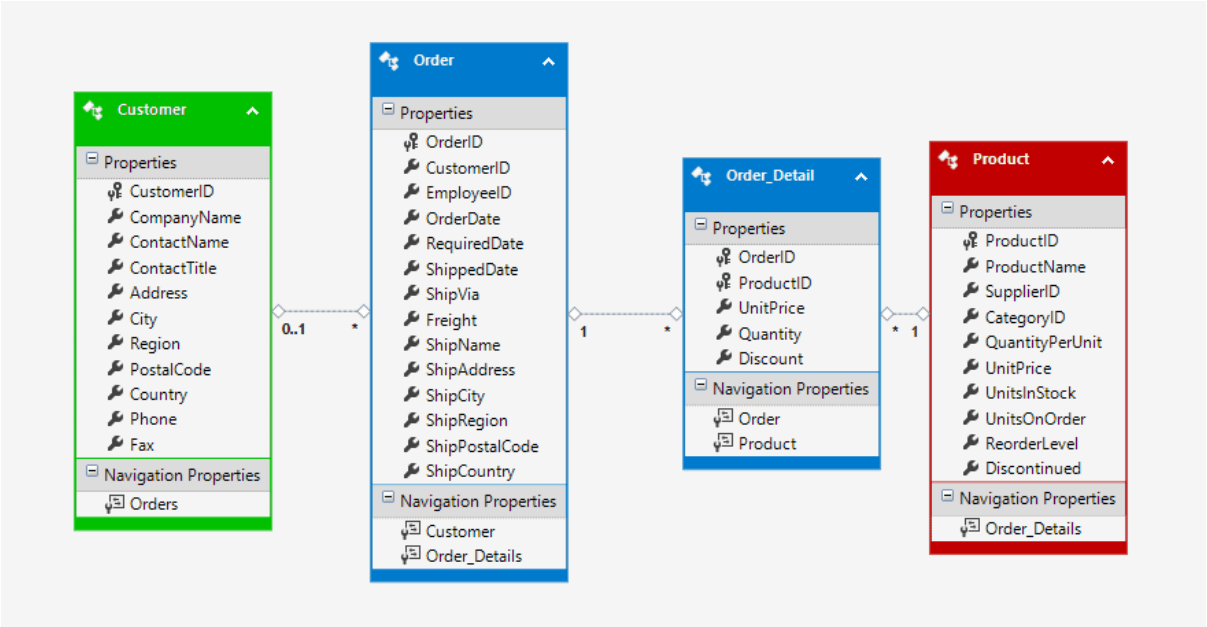
Es muy complicado que pierdas datos al hacer modificaciones en el modelo, ya que las harás en la base de datos, no en el código de la aplicación.

Problemas o inconvenientes de Database First

- El código resultante para los modelos tiene infinidad de código auto-generado sobre el que tenemos poco control y que puede acabar "pesando" mucho. Si hay algún problema te costará más ver de dónde te viene.
- Si necesitas personalizar las clases o su comportamiento debes extenderlas con clases parciales o bien tocando las plantillas. Dependiendo de lo que quieras hacer puedes acabar escribiendo bastante código, que probablemente es lo que querías evitar usando este modelo.
- Cada vez que hay cambios en la base de datos se debe regenerar el archivo de mapeo que la representa, lo cual puede llevar bastante tiempo dependiendo de lo grande que sea. Si estás haciendo y probando muchos pequeños cambios puede llegar a desesperarte.

Model First

Este nombre puede despistarte, NO tiene que ver nada con el patrón MVC en programación, en el que el "Modelo" de la "M" en su nombre, es código que escribes para modelar tus datos. En este caso se refiere a **crear tu modelo de datos visualmente**, usando un Diseñador de Modelos. No tienes que escribir código alguno.



Describes visualmente el modelo de objetos que quieres crear, con las entidades, sus propiedades, relaciones entre ellas (que generan propiedades de navegación entre las entidades), etc... y luego **el ORM se encarga de generar la base de datos** subyacente o modificarla **y también las entidades POJO** (*Plain Old Java Object)*, o sea, clases normales y corrientes de Java) que representan dicho modelo.

Ventajas o beneficios de Model First

-El diseñador visual facilita enormemente el diseño del modelo de datos.
-Es un sistema interesante porque te evita tanto escribir código como definir la base de datos (aunque puedes generar el modelo inicial de una base de datos preexistente si quieres). Si no eres de las personas que disfrutan escribiendo código y mucho menos definiendo bases de datos, te ahorrará muchos desvelos.
-Es muy productivo en proyectos pequeños, pues te olvidas de definir clases y de tocar bases de datos.

Problemas o inconvenientes de Model First

-Pierdes el control tanto de la base de datos como de las clases generadas. Se encarga de todo el ORM, lo cual es estupendo si no necesitas nada "especial", pero te quita control en otros casos.
-Si quieres extender las clases, como antes, debes tocar las plantillas de clases o definir clases parciales.
-Por defecto los cambios que hagas en el modelo no generan ***scripts* SQL** incrementales, sino completos. Esto quiere decir que si haces un cambio y regeneras la base de datos, se elimina la actual y se crea de nuevo. Esto puede significar que pierdas los datos que ya tuvieses en la misma, así que mucho cuidado.

Code First

Si en *Database First* empezábamos por la base de datos generando el ORM todo lo demás, y en *Model First* estábamos en el "medio" definiendo el modelo para que el ORM generase tanto las clases como la base de datos... en *Code First* nos situamos en el otro extremo del espectro: **definimos nuestras clases mediante código, y el ORM se encarga de generar la base de datos y todo lo necesario para encajar las clases en ellas**.

Se trata de un enfoque muy orientado al programador. Es para gente a la que le gusta escribir código y tener control total sobre cómo se comportan sus clases. En este caso el ORM no genera código, ni necesita complejos y pesados archivos de configuración para *mapear* objetos a la base de datos, por lo que todo es más ágil.

El programador define su modelo usando clases (POJO). Relaciona unas con otras simplemente haciendo referencia entre ellas en propiedades. A mayores puede decorar algunas propiedades de estas clases de forma especial (las llamadas "Annotations", "Anotaciones" en inglés) si quiere indicar cosas concretas, como por ejemplo el nombre que quiere darle al campo correspondiente en la base de datos, y cosas así. Esto es lo que hemos utilizado en prácticas para ORMLite.

Ventajas o beneficios de Code First

-Máximo control sobre el código de tu modelo de datos en java, ya que son clases que construyes desde cero.
-Te ofrece control sobre las colecciones de objetos que quieres que se carguen de modo "perezoso" (es decir, a medida que se vayan necesitando, no creamos todo de golpe).
-No tocas una base de datos ni con un palo. El código es el que manda y el "tooling" se encarga de generar la base de datos en función de tu código.
-La estructura de la base de datos es muy fácil de mantener bajo control de código (con Git o similar) ya que no se guarda en absoluto: se guarda el código de nuestras clases del modelo, y se genera la base de datos bajo demanda.

Problemas o inconvenientes de Code First

-Debes dominar bastante más el ORM que con los enfoques anteriores.
-Cualquier cosa que necesites persistir u obtener de tu base de datos la tienes que implementar en código de programación. La base de datos que se genera por detrás no la puedes tocar.
-Si tu base de datos es muy grande y con muchas tablas, la gestión de la base de datos se puede convertir en un pequeño dolor de muelas.

¿Cuál escoger?

Ahora que hemos repasado las opciones, conociendo en qué consisten y sus ventajas e inconvenientes... ¿Cuál sería la mejor opción?

Bueno, si lo tuyo es la programación, te quedas sin opciones: deberás usar Code First, sí o sí.

En el caso de algunos ORM tienes acceso a todo, por lo que para elegir, como todo en la vida, dependerá de muchos factores.

Los más puristas prefieren el enfoque de *Code First* frente a los demás. Les da el máximo rendimiento y control y además desde el punto de vista de un programador "puro", la base de datos es solo un mal necesario y lo importante es nuestro código. Además podrás aplicar casi todo a los ORM más ligeros o básicos, como ORMLite, ya que Code First es una característica básica de cualquier ORM.

Los que **dominan y valoran** las bondades de las **bases de datos**, por el contrario, suelen preferir alguno de los anteriores, especialmente el ***Database First***.

Si tu **proyecto es pequeño** y no va a tener muchas tablas/entidades que manejar, ***Code First*** puede ser muy cómodo y ágil, con control total por tu parte. Si por el contrario hay **muchas tablas** y el proyecto de datos es grande, ***Database First*** puede ser una gran opción aunque algún "purista" se rasgue las vestiduras.

En caso de que no te gusten demasiado las bases de datos ni tampoco tener que escribir todo el código de "fontanería" a mano, **el enfoque *Model First* es una buena opción para los más vagos** (dicho sea desde el cariño), ya que te permite diseñar visualmente las entidades/datos que quieres manejar y se encarga automáticamente de todo lo demás: generar las clases del modelo y las tablas en la base de datos, *mapeándolos* entre sí de manera transparente.

¡Espero que te resulte de ayuda!