EntityFramework 7 - маленькая революция.

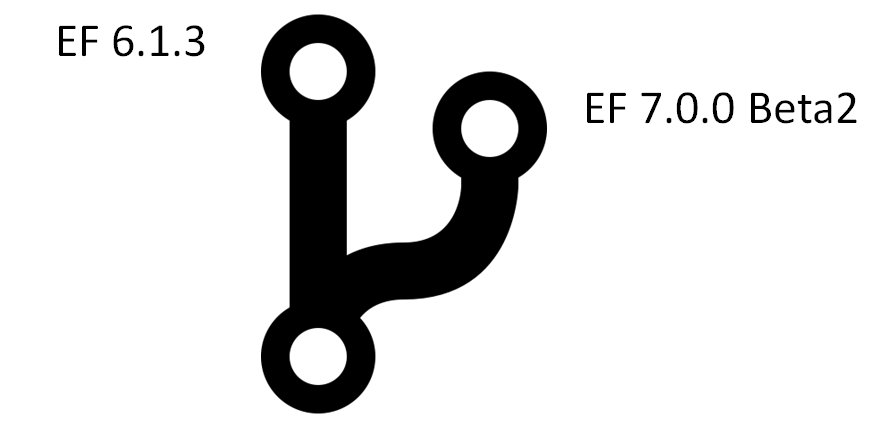
В начале лета 2014 года, Microsoft начал разрабатывать EntityFramework7 <https://github.com/aspnet/EntityFramework>. Этот фреймворк до сих пор в стадии разработки, но поскольку он разрабатывается на github, а команда публикует свои записи с архитектурных встреч, то уже сейчас можно рассказать про некоторые изменения, которые будут в EF7.

Некоторые из них я уже упоминал в статье **EntityFramework 7 + Redis**<http://habrahabr.ru/post/111542/> , но после определенных offline-обсуждений стало понятно, что надо написать более подробно про фундаментальные изменения, а не просто про поддержку 2 NoSQL хранилищ.

<habracut/>

**Версия и история**

7-ая версия в конце весны отбранчевалась от 6-ой версии. Я бы даже сказал, это был fork.



Разработчики и на teched, и в своих записках <http://blogs.msdn.com/b/adonet/archive/2014/10/27/ef7-v1-or-v7.aspx> говорят одно и тоже.

EF6 по прежнему продолжает дорабатываться. Не все фичи, созданные после форка, будут доступны в EF7 с самого начала. EF7- это достаточно революционный проект, и были даже мысли: «А 7-ая ли это версия в принципе или это новый продукт похожий на EF по основному API, но сильно разный внутри?» Не все заявленные в EF7 фичи будут доступны на всех платформах сразу же.

Команда выбрала путь более частых обновлений EF7, чтобы не заставлять всех ждать полного комплекта изменений.

**Основные цели EF7**

Обозначены 2 вектора движения EF7:

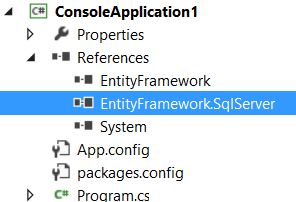
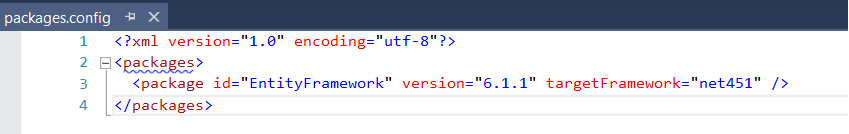
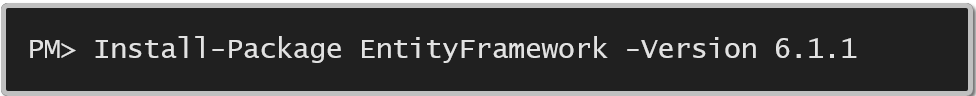
* NewMobilePlatforms;
* NoSQL.

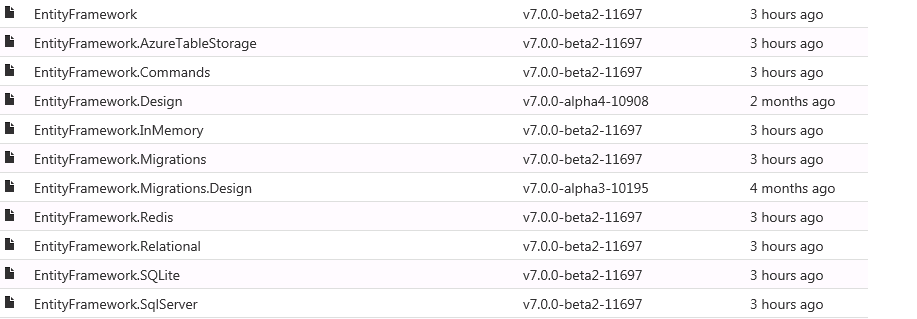
Все дальнейшие изменения надо рассматривать через эту призму.

Уменьшение потребления ресурсов - это обязательное условие для нормальной работы на мобильных девайсах, где объем RAM – 2 Гб на топовых моделях и 512 Мб на стоковых. Рефакторинг и разбиение проекта на разные сборки – это поддержка обоих векторов одновременно. Можно привести достаточно много других примеров.

**Рефакторинг и разбиение проекта на модули**

Раньше был один nuget-пакет EntityFramework, который тянул 2 сборки. Больше ничего.



В EF7 за счет поддержки множества платформ и баз данных, проект разделился на 10 сборок. 

**Оптимизация использования RAM/CPU**

**DetectChanges**

В EFдостаточно тяжеловесный framework. Его часто сравнивают с более легким BLToolkit или чем-то типа того. Основной претензией было то, что за счет отслеживания состояний объектов и вызова на каждый чих DetectChanges, он жутко тормозил. Это все решалось установкой настройки AutoDetectChanges=false при массовых операциях вставки, но об этом надо было знать.

В 7-ой версии настройку AutoDetectChanges удалили за ненадобностью, т.к. DetectChanges перестал вызываться постоянно, только на inEF7 DbContext.SaveChanges, DbSet.Local, DbContext.Entry, DbChangeTracker.Entries. Учитывая, что все это не самые часто используемые команды, то рост скорости вставки очень значительный.<https://github.com/aspnet/EntityFramework/wiki/Design-Meeting-Notes:-October-30,-2014#automatic-detectchanges>

По поводу снижения потребления памяти я ничего в записях не нашел.

**Валидация**

Те, кто раньше использовал EF, помнят проблемы, когда весь код переставал работать, а проблема была всего в одной сущности. В EF7 валидацию сильно упростили. Частично из-за использования различных провайдеров, а это значит, что валидировать модель без привязки к провайдеру, стало достаточно бесполезным занятием. <https://github.com/aspnet/EntityFramework/wiki/Entity-Framework-Design-Meeting-Notes---July-17,-2014#model-validation>

**No EDMX**

Те, кто работал с Model First, Database First, помнят, какие проблемы были с EDMX. Сделать Merge двух веток было крайне нетривиальным занятием, т.к. почти всегда это приводило к куче конфликтов или невалидному edmx-файлу. Любое обновление модели из базы ломало наши кастомизации (в частности, удаление лишних navigation property).

Для самой команды EF это выражалось в необходимости поддержки двух разных моделей с одной стороны Code First, с другой - DB/Model First. Подумав, команда решила отказаться от поддержки EDMX в 7-ой версии. Code First only.

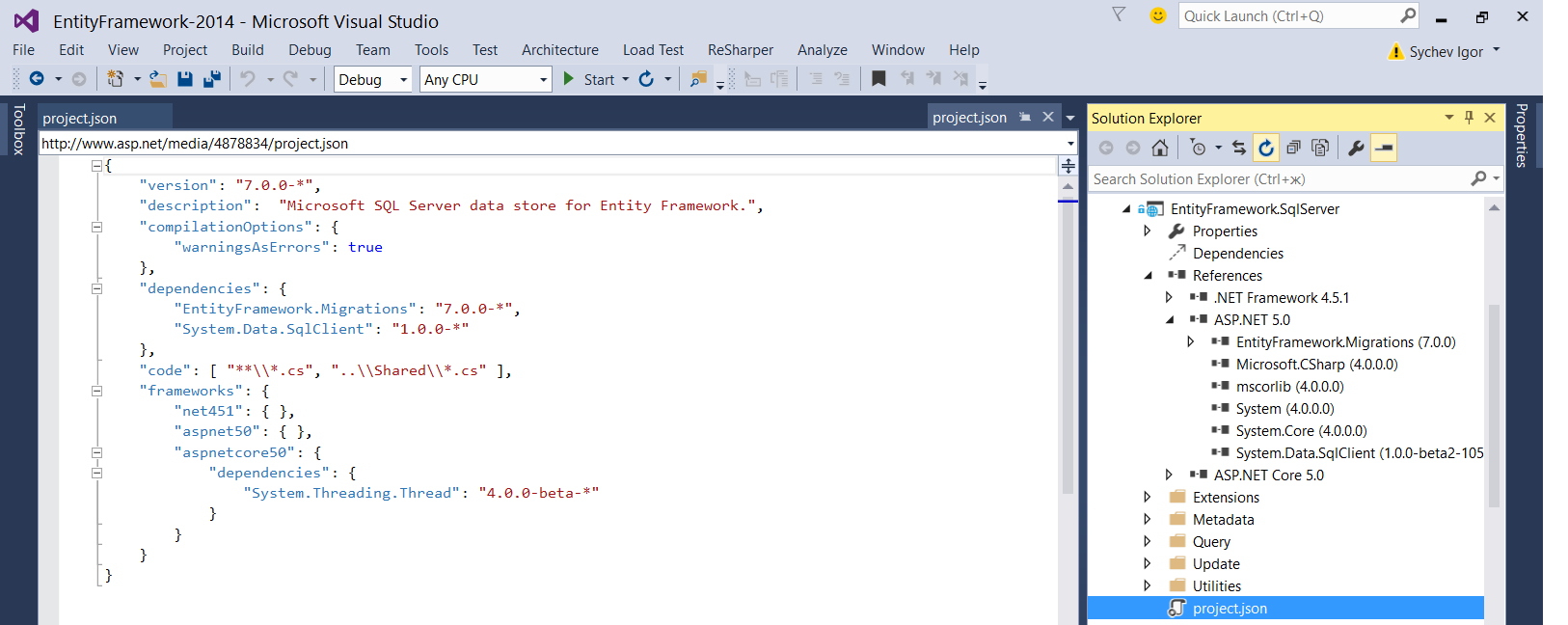
Не нужно пугаться, хотя каждый из нас может придумать причину опасности использования Code First. Команда EF разъясняет, что Code First – не совсем корректное имя, и его не совсем правильно понимают. Правильнее было бы назвать Modeling using Code. Когда мы описываем модель непосредственно в C# коде, то разработчик код лучше всего понимает. Графический интерфейс Visual Studio теперь будет работать именно с Code First, а edmx более не поддерживается.

Если вам страшно, что code first может попытаться обновить вашу боевую базу, удалив нужные данные или как-то еще напортачить, то вам нужно просто запретить это в настройке и не заниматься обновлением базы, а как познавшим дзен, вести проект базы данных, писать скрипты миграции схемы и данных отдельно и иметь при этом полный контроль над базой данных, а не отдавать все на откуп миграциям из EF.

В VisualStudio 2015 preview можно использовать и ef6 и ef5 и, следовательно, db/model first. Но что будет дальше пока не понятно.

**Asp.NetVNext**

EntityFramework пишет под тем же руководством, что и asp.net. Мы уже слышали каким революционным будет VNext. Как сказал мой знакомый: “После анонса VNext, нам всем заново придется учить asp.net”. Это сказывается и на самом EF. Если мы заглянем в исходный код, то увидим там:



В версии для 2015 студии Json-файлы с описанием зависимостей и настроек. В 2013 студии все по старинке, правда…

Еще интереснее, что даже строку подключения можно будет в Json-файле написать. <https://github.com/aspnet/EntityFramework/wiki/Configuring-a-DbContext>



В общем EntityFramework 7 будет достаточно революционным на мой взгляд.