Entity FrameWork

WhatIsEf ?

چهارچوبی برای دسترسی به داده است که به صورت lightweight و CrossPlatform بوده و برای کار با .net طراحی شده است و یک ORM را ارائه می دهد Object Relational Mapper و به توسعه دهندگان امکان دسترسی به داده را می دهد.

WhatIsOrm ?

اختصار Object Relational Mapper است که یک تکنیک برنامه نویسی که به برنامه نویس ها کمک می کند که Data را بین سیستم های incompatible تبدیل کنند (ناسازگار) به خصوص بین سیستم های oop مثل c# و relational database مثل sql server , my sql

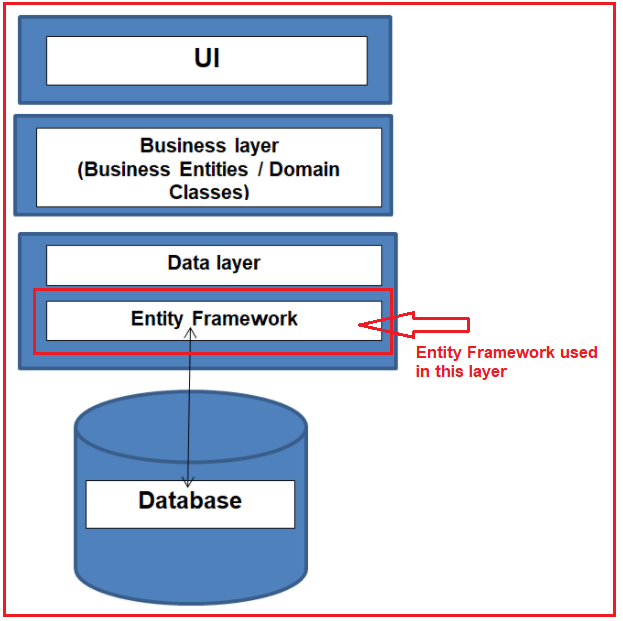
این امکان را به توسعه دهندگان می دهد که بر اساس قوانین object با Data کار کنند به جای table , columns به این معنی که orm به صورت خودکار کلاس ها را بر اساس database table می سازد همچنین برعکس این نیز صادق است یعنی می تواند sql لازم برای ایجاد DataBase بر اساس Class ها را نیز تولید کند.

از جمله کارکرد های آن به ساده سازی عملیات crud می توان اشاره کرد یا کار های مثل openingConnection یا executing command یا مدریت transaction ها و اطمینان از data integrity

نکته : به کلاس های در سطح اپ خود domain object یا business object می گویند.

نکته : از ef می توانیم در DataBase های non-relational نیز استفاده کرد.

نکته : قبل از نسخه ی .net 3.5 توسعه دهندگان از Ado.net استفاده می کردند

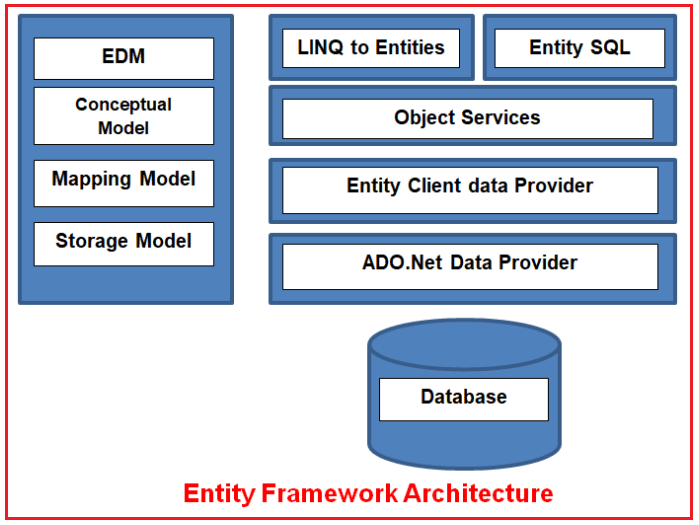


Ef داده های ذخیره شد در property های مربوط به کلاس ها را در DataBase ذخیره می کند

**Entity Framework Architecture**

**معماری entity framework به صورت :**

1. **The Entity Data Model**
2. **LINQ to Entities**
3. **Entity SQL**
4. **The Object Services Layer**
5. **Entity Client Data Provider**
6. **ADO.NET Data Provider**

****

**EDM (Entity Data Model):**

طرحواره منطقی یا رابطه ای انتزاعی است که و این مدل را از طریق یک رویکرد سه لایه نمایش می دهد.

1. The Conceptual Model (C- Space),
2. Mapping model (C-S Space)
3. Storage model (S – Space)

**Conceptual Model:**

شامل مدل های ما classes یا entity به همراه روابط بین آن ها است و بستگی به طراحی دیتا بیس ما دارد.

**Mapping Model**

این وظیفه را دارد که مشخص کند چگونه conceptual model به storage mode مپ شود. کار مپ کردن مدل business به همراه روابط بین آن ها که در لایه conceptual است به سطح table ها و روابط منطقی بین آن ها است.

**Storage Model / Logical Model**

مدل طراحی دیتابیس است که شامل table , views, keys , sp و ... روابط بین آن ها است که شامل 3 نوع فایل xml است C-Space, C-S Space, and the S-Space respectively

**.CSDL (Conceptual Schema Definition Language)**:

برای مپ کردنtype entity با استفاده از conceptual model است.

**.MSL (Mapping Schema Language)**

برای مپ کردن logical mode به conceptual model کاربرد دارد.

**.SSDL (Store Schema Definition Language)**

برای مپ کردن schema information of logical layer کاربرد دارد

**LINQ to Entities:**

برای کوئری نوشتن بر روی object model است و entity برمیگرداند که در conceptual model تعریف شده اند.

**Entity SQL:**

زبان کوئری نویسی دیگری است که به صورت داخلی به کوئری دیتابیس تبدیل می شود.

**Object Service**

در بسیاری از موارد باید با entity های که به صورت in-memroy هستند کار کنیم با استفاده از object service

**کارکرد ها :**

کاربرد های مثل انجام عملیات های مثل add , update , delete , و ... است.

لایه orm که نتیجه داده ها را به نمونه های object ما تبدیل می کند.

قابلیت استفاده از امکاناتی مثل primary key و changeTracking و ... را به توسعه دهندگان می دهد

**سرویس های ارائه شده توسط object service**

1. **Change tracking**
2. **Lazy loading**
3. **Inheritance**
4. **Optimistic concurrency**
5. **Merging data**
6. **Identity resolution**
7. **Support for querying data using Entity SQL and LINQ to Entities**

**Entity Client Data Provider:**

کار اصلی این لایه تبدیل linq به sql query است و با ado.net تعامل دارد که کار ارسال و بازیابی داده از database را بر عهده دارد.

**ADO.Net Data Provider:**

لایه ای است که برای تعامل با DataBase طراحی شده است.

**Entity Framework Context Class**

Ef به ما این امکان را می دهد که عملیات crud را بر روی داده ها با استفاده از CLR یا Common language runtime انجام بدهیم که به عنوان Entity یا .net class شناخته می شوند و ef کلاس ها را و روابط بین آن ها را به مدل Data Base مپ می کند.

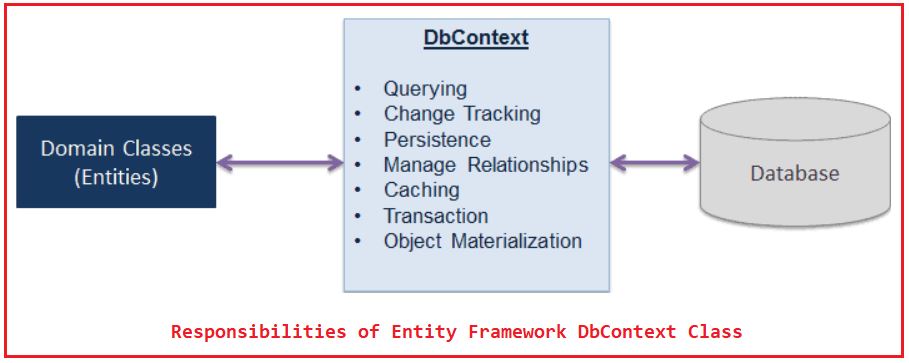
**Context**

کلاس اصلی جهت برقرارای ارتباط با database کلاس Context است. این یک کلاس معمولی بوده که از DbContext باید ارث بری کند. این کلاس یک session مرتبط با database را ارائه می دهد.

یک instance از dbContext می تواند ترکیبی از الگو های unitOfWork و Repository باشد و برای کوئری زدن به database کاربرد داشته باشد.

**What are the Responsibilities of DbContext in Entity Framework?**

در واقع یک پل میان domain class و database است



1. تبدیل کوئری linq به sql و ارسال آن ها به database
2. ثبت وقایع یا change tracking بر روی entity ها بعد از این که دیتا از دیتابیس fetch میشود
3. انجام عملیات های crud
4. ارائه fist-level-caching به صورت default
5. ارائه ویژگی transaction
6. مدیریت relation های میان entity ها در حالت code first و dbFirst
7. تبدیل داده های خام دریافت شده از database به entityObjects

**Entities in Entity Framework**

هر چیزی که در کلاس dbContext به صورت یک property و به شکل public DbSet<User> User {get;set;} تعریف شده باشد را می توانیم entity بنامیم.

Ef هر کدام از entity ها را به یک جدول در دیتابیس مپ می کند و هر کدام از property های مربوط به آن entity را به یک column در دیتابیس مپ می کند.

**انواع Property در یک entity کلاس**

**Scalar property**

در واقع property های است که نوع داده آن از نوع Primitive است مثل int string یا انواع nullable ها

**Navigation property**

این نوع property ها ارائه دهنده یک relationship میان entity ها هستند که می توانند از 2 نوع زیر باشند.

**Reference navigation property**

اگر یک entity دارای یک property از نوع یک entity دیگر باشد از این نوع است که بیان گر رابطه ی 1 به 1 در Ef است در کلاس Employee یک کلاس از نوع Department وجود دارد.

**public virtual Department Department { get; set; }**

**Collection navigation property**

اگر دارای یک Collection از نوع یک entity به عنوان property در یک entity دیگر باشیم که بیان گر رابطه ای one to many یا یک به چند است

**public virtual ICollection<Employee> Employees { get; set; }**

**Types of Entities in Entity Framework**

انواع entity در ef

**Poco entities**

مخفف plain old clr objects که می توان به عنوان یک domain object موجود در Data model باشد.

این نوع کلاس مپ می شوند به entity های تعریف شده در dataModel. این نوع entity یک کلاس است **که به هیچ یک از framework base class ها وابسته یا Depend نیست.**

این نوع تمام عملیات های insert update delete را پشتیبانی می کند درست همانند entity های generate شده توسط Data Model.

**Sample**

**public class Employee**

**{**

**public int ID { get; set; }**

**public string Name { get; set; }**

**public Nullable<int> Salary { get; set; }**

**public Nullable<int> DepartmentId { get; set; }**

**public virtual Department Department { get; set; }**

**}**

**Dynamic Proxy Entities**

زمانی که از یک موجودیت poco یک instance می سازیم به عنوان یک proxy عمل می کند برای آن entity. همچنین می توان گفت که در زمان runtime پروکسی مانند یک wrapper یا کلاس پوششی برای موجودیت poco خواهد بود

**Requirements of poco proxy entity**

کلاس باید public باشد.

نباید sealed باشد

نباید Abstract باشد

تمامی navigation property ها باید public و virtual باشند.

تمام کالکشن ها باید از نوع ICollection<> باشد.

در context مقدار proxyCreationEnabled باید مقدار false داشته باشد.

**public** **class** Employee

**{**

**public** **int** ID **{** **get**; **set**; **}**

**public** **string** Name **{** **get**; **set**; **}**=

**public** Nullable**<int>** Salary **{** **get**; **set**; **}**

**public** Nullable**<int>** DepartmentId **{** **get**; **set**; **}**

**public** **virtual** Department Department **{** **get**; **set**; **}**

**}**

**نکته : فقط با یک تغییر virtual و حذف آن کلاس از حالت dynamic proxy به حالت poco تبدیل می شود**

**When to use Dynamic Proxy in Entity Framework?**

زمانی که به lazyLoad یا Change tracking نیاز داشته باشیم.

**Lazy**

به صورت پیش فرض در Ef فعال است به این صورت که related ها را تا زمانی که درخواست مستقیم ندهیم load نمی کند.

**changeTracking**

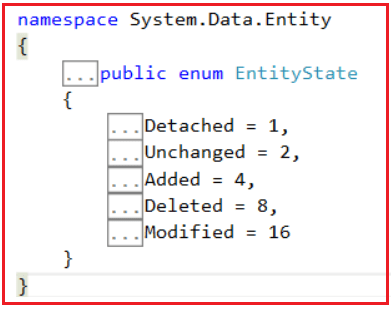
**به صورت خودکار ef از automic change tracking of loaded entites پشتیبانی می کند در طول زمان context life time و کلاس DbChangeTracker به صورت خودکار اطلاعات entity های track شده را می دهد**

**Entity States in Entity Framework**

چرخه حیاط یا life cycle در ef فرایندی را توصیف می کند که در آن یک entity ایجاد اضافه تغییر و حذف می شود.

یک entity در طول عمر خود دارای state های مختلفی است و ef کار نگه داری state هر یک از entity ها را در طور life time خود انجام می دهد.

هر entity دارای یک state بر مبنای عملی که روی ان انجام می شود از طریق کلاس context که خود دارای یک enum است.



نکته : یک entity همیشه در یکی از State ها قرار دارد.

**Added**: The entity is marked as added. The entity is being tracked by the context but does not yet exist in the database.

**Deleted**: The entity is marked as deleted. The entity is being tracked by the context and exists in the database, but has been marked for deletion from the database the next time SaveChanges is called.

**Modified**: The entity has been modified. The entity is being tracked by the context and exists in the database, and some or all of its property values have been modified.

**Unchanged**: The entity hasn’t been modified. The entity is being tracked by the context and exists in the database, and its property values have not changed from the values in the database.

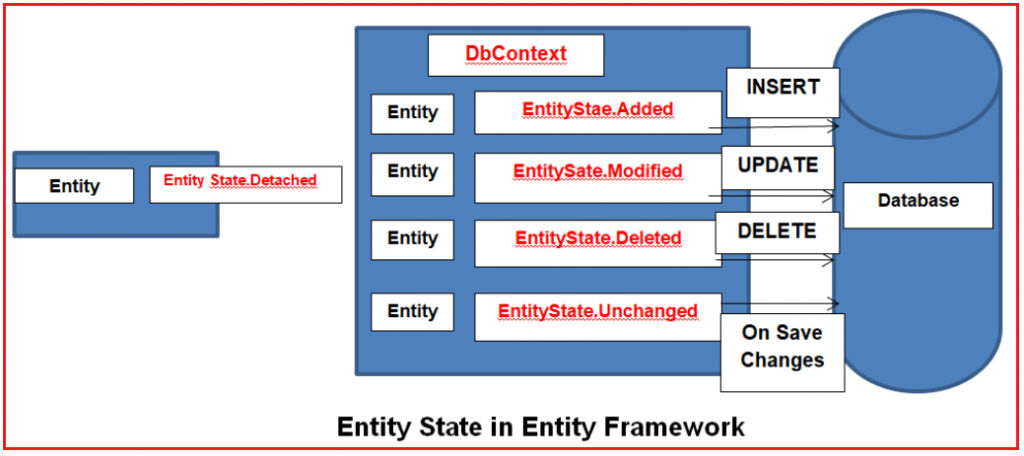
**Detached**: The entity is not being tracked by the context. An entity is in this state immediately after it has been created with the new operator or with one of the System.Data.Entity.DbSet Create methods

**changeTracking**

شی context نه تنها رفرنس تمامی entity ها را در خود نگه داری می کند بلافاصله بعد از این که از database بازخوانی می شوند بلکه کار ردیابی یا tracking و نگه داری state و تغییرات اعمال شده بر روی هر یک از property های entity را نیز انجام می دهد.

**Entity Lifecycle in Entity Framework**

تغییر حالت یک Entity از **حالت unchanged به حالت modified** تنها حالت تغییر وضعیت است که به صورت **automatic توسط context** کلاس مدیریت می شود.



**Unchanged State**

**نکته: property** مربوط به یک entity تغییر نکرده است از زمانی که از database داده retrieved شده است در این حالت زمانی که saveChange فراخوانی شود این entity **توسط context نادیده گرفته می شود** **یا ignore**

**وضعیت unchanged وضعیت پیش فرض در این حالت ها است. یعنی یک entity را دریافت کرده و کاری نکینم و save کنیم.**

**Attaching an Existing Entity to the context**

اگر یک entity داشته باشیم که اطمینان داریم در database ما موجود است اما توسط context رهگیری نمی شود track می توانیم به context بگوییم که entity را رهگیری کند با استفاده از Attach() بر روی همان dbSet در این حالت به صورت unchanged قرار می گیرد.

//Creating an Existing Department Entity

Department ExistingDepartment = new Department**()**

**{**

ID = 1,

Name = "IT",

Location = "BBSR"

**}**;

//Attaching the Entity to the Context

context.Departments.Attach**(**ExistingDepartment**)**;

//Or you can also attach an existing entity as follows

//context.Entry(ExistingDepartment).State = EntityState.Unchanged;

**Detached State of an Entity in Entity Framework**

می توانیم وضعیت یک entity را به حالت detached در بیاوریم در صورتی که این کار را انجام دهیم آن entity توسط chageTracker دنبال نمی شود

//Changing the Entity State to Detached

context.Entry**(**firstDepartment**)**.State = EntityState.Detached;

**Added State of an Entity in Entity Framework**

زمانی که با استفاده از متد Add مربوط به context یک entity جدید را add می کنیم در واقع state آن را به حالت Added تغییر داده ایم. Add به معنای این است که Entity در context وجود دارد اما در database وجود ندارد و زمانی که saveChange ما invoke شد یک کوئری insert ایجاد می شود و زمانی که عملیات با موفقیت انجام شود و داده insert شود state مربوط به entity به حالت unchanged تغییر پیدا میکند

**Modified State of an Entity in Entity Framework**

یک entity زمانی در حالت modified قرارمیگیرد که یکی از scalar property های آن تغییر پیدا کرده باشد و به این معنی است که entity تغییر کرده است اما نه در database و همچنین به این معنی است که entity در database موجود است و بعد saveChange کوئری update ایجاد می شود و state به حالت unchanged تغییر پیدا میکند.

نکته : دستور update فقط برای property های که تغییر داشته اند ایجاد می شود.

**Deleted State of an Entity in Entity Framework**

زمانی که بر روی یک entity متد remove را فراخونی می کنیم به حالت deleted تغییر State می دهد و از context حذف می شود و زمانی که saveChange فراخوانی می شود از database هم ردیف مربوطه حذف می شود این State به این معنی است که دیگر در context وجود ندارد و در Db وجود دارد البته حذف از context زمانی صورت می گیرد که حذف از db با موفقیت انجام شود و به حالت detached در می آید

**What SaveChanges Method Does in Entity Framework?**

اگر entity ها در حالت unchanged باشد هیچ کاری نمی کند.

اگر entity ها در حالت Added باشد کوئری insert برای هر entity می سازد

اگر entity ها در حالت modified باشند کوئری update برای هر کدام می سازد

اگر entity ها در حالت delete باشند از context به حالت detached می روند و بعد موفقیت کار در Db از context حذف شده و دیگر state آن نگه داری نمی شود

بعد از انجام عملیات db در مورد 1 تا 3 به حالت unchanged تغییر وضعیت می دهند.