|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **记录名称** |  | | **记录编码** |  | |
| **密 级** |  | | | | |
| **起草/日期** |  | **审核/日期** |  | **批准/日期** |  |

**记录修改履历**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **修订前版本** | **现行**  **版本** | **修订内容概述** | **起草** | **修订日期** |
| 1 | N/A | 1.0 |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**目 录**

[1 3](#_Toc522027021)

[1.1 未处理异常的捕获，定义自己的异常捕获 3](#_Toc522027022)

[1.2 NuGet，联机安装，使用常用的类库。 3](#_Toc522027023)

[1.3 DLL 插件方式的开发MEF,MVVM插件的开发思想。 3](#_Toc522027024)

[1.4 VS自带工具 IL 中间语言的反汇编 4](#_Toc522027025)

[1.5 接口和抽象类有什么区别 4](#_Toc522027026)

[1.6 代理，委托和回调，泛型 4](#_Toc522027027)

[1.7 LOG NLOG 5](#_Toc522027028)

[1.8 线程池 5](#_Toc522027029)

[1.8.1 多个并发的线程，采用线程池的方式。 5](#_Toc522027030)

[1.8.2 同种线程，大并发情况。 5](#_Toc522027031)

[1.9 Dictionary，list，arraylist，hashcode，queue，stack， 5](#_Toc522027032)

[1.10 C#操作符??和?: 6](#_Toc522027033)

[1.11 对象CLONE 7](#_Toc522027034)

[1.12 ORM 7](#_Toc522027035)

[1.13 E-R图 7](#_Toc522027036)

[1.14 构造函数定义为private，protected 9](#_Toc522027037)

[1.15 => 11](#_Toc522027038)

[1.16 As IS 11](#_Toc522027039)

[1.17 P/Invoke 12](#_Toc522027040)

[1.18 IOC控制反转 13](#_Toc522027041)

[1.19 MEF 13](#_Toc522027042)

# **常用知识点**

## **未处理异常的捕获，定义自己的异常捕获**

# **ORM数据库持久层关系映射**

对象关系映射（英语：**(Object Relational Mapping**，简称**ORM**，或**O/RM**，或**O/R mapping**），是一种程序技术，用于实现面向对象编程语言里不同类型系统的数据之间的转换[1]  。从效果上说，它其实是创建了一个可在编程语言里使用的--“虚拟对象数据库”。

面向对象是从[软件工程](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B)基本原则（如耦合、聚合、封装）的基础上发展起来的，而关系数据库则是从数学理论发展而来的，两套理论存在显著的区别。为了解决这个不匹配的现象，对象关系映射技术应运而生。

对象关系映射（Object-Relational Mapping）提供了概念性的、易于理解的模型化数据的方法。ORM方法论基于三个核心原则： 简单：以最基本的形式建模数据。 传达性：[数据库结构](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%BB%93%E6%9E%84)被任何人都能理解的语言文档化。 精确性：基于数据模型创建正确标准化的结构。 典型地，建模者通过收集来自那些熟悉应用程序但不熟练的[数据建模](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BB%BA%E6%A8%A1)者的人的信息开发[信息模型](https://baike.baidu.com/item/%E4%BF%A1%E6%81%AF%E6%A8%A1%E5%9E%8B)。建模者必须能够用非技术企业专家可以理解的术语在概念层次上与数据结构进行通讯。建模者也必须能以简单的单元分析信息，对样本数据进行处理。ORM专门被设计为改进这种联系。

简单的说：ORM相当于中继数据。具体到产品上，例如ADO.NET Entity Framework。DLINQ中实体类的属性[Table]就算是一种中继数据。

ORM不是必要的，要不要用ORM取决于架构师的口味。Java-er习惯于过度设计，一丁点儿逻辑需要弄一堆文件再带上百兆的库不稀奇。

ORM最常解决的问题，就是对于简单的CRUD操作（特别是单表操作，或简单的关联表查询）以对象操作的方式而不是SQL方式来写代码。而复杂的操作，还是建议程序员自己来写SQL。

从一个框架的角度，更好的设计，莫过于对于简单的CRUD操作（包括关联表查询），什么代码都不用写（框架自动解决）；对于复杂的操作，程序员还是自己写，框架提供最简单的调用方式，比如一行代码就可调用并获取结果。

众多厂商和开源社区都提供了持久层框架的实现，常见的有：

JAVA系列：

APACHE OJB

CAYENNE

JAXOR

JPA

MYBATIS

[HIBERNATE](https://baike.baidu.com/item/HIBERNATE)

IBATIS/MYBATIS

JRELATIONALFRAMEWORK

SMYLE

TOPLINK

其中 TOPLINK 是 ORACLE 的商业产品，其他均为开源项目。

其中 HIBERNATE的轻量级 ORM 模型逐步确立了在 JAVA ORM 架构中领导地位，甚至取代复杂而又繁琐的 EJB 模型而成为事实上的 JAVA ORM 工业标准。而且其中的许多设计均被 J2EE 标准组织吸纳而成为最新 EJB 3.0 规范的标准，这也是开源项目影响工业领域标准的有力见证。

.NET系列：

ENTITYSCODEGENERATE

LINQ TO[SQL](https://baike.baidu.com/item/SQL)

GROVE

RUNGOO.ENTERPRISEORM

FIRECODE CREATOR

MYGENERATION

CODESMITH PRO

CODEAUTO ...

//===============================================

## **EF的三种方式**

Code First不是和Database First或Model First

<http://www.cnblogs.com/lsxqw2004/p/4701979.html>

<http://www.cnblogs.com/oppoic/p/welcome_to_ef_code_first.html>

Ef的数据库连接方法。常用介绍

<https://www.cnblogs.com/24la/p/ef6-codefirst-mysql.html>

EF连接Mysql

## **EF的使用建议**

<http://blog.csdn.net/dongliang_shali/article/details/50697559>

在.Net Framework SP1微软包含一个实体框架（Entity Framework），此框架可以理解成微软的一个[ORM](https://www.baidu.com/s?wd=ORM&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAR1P1nsrj6snAw9uyDs0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6Lrj6dP1mY)产品。用于支持开发人员通过对概念性应用程序模型编程（而不是直接对关系存储架构编程）来创建数据访问应用程序。目标是降低面向数据的应用程序所需的代码量并减轻维护工作。Entity Framework 应用程序有以下优点：  
· 应用程序可以通过更加以应用程序为中心的概念性模型（包括具有继承性、复杂成员和关系的类型）来工作。  
· 应用程序不再对特定的数据引擎或存储架构具有硬编码依赖性。  
· 可以在不更改应用程序代码的情况下更改概念性模型与特定于存储的架构之间的映射。  
· 开发人员可以使用可映射到各种存储架构（可能在不同的数据库管理系统中实现）的一致的应用程序对象模型。  
· 多个概念性模型可以映射到同一个存储架构。  
· 语言集成查询支持可为查询提供针对概念性模型的编译时语法验证。  
实体框架Entity Framework 是 ADO.NET 中的一组支持开发面向数据的软件应用程序的技术。在[EF](https://www.baidu.com/s?wd=EF&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAR1P1nsrj6snAw9uyDs0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6Lrj6dP1mY)中的实体数据模型（[EDM](https://www.baidu.com/s?wd=EDM&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAR1P1nsrj6snAw9uyDs0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6Lrj6dP1mY)）由以下三种模型和具有相应文件扩展名的映射文件进行定义。  
· 概念架构定义语言文件 (.[csdl](https://www.baidu.com/s?wd=csdl&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAR1P1nsrj6snAw9uyDs0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6Lrj6dP1mY)) -- 定义概念模型。  
· 存储架构定义语言文件 (.ssdl) -- 定义存储模型（又称逻辑模型）。  
· 映射规范语言文件 (.msl) -- 定义存储模型与概念模型之间的映射。  
实体框架 使用这些基于 [XML](https://www.baidu.com/s?wd=XML&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAR1P1nsrj6snAw9uyDs0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6Lrj6dP1mY) 的模型和映射文件将对概念模型中的实体和关系的创建、读取、更新和删除操作转换为数据源中的等效操作。[EDM](https://www.baidu.com/s?wd=EDM&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAR1P1nsrj6snAw9uyDs0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6Lrj6dP1mY) 甚至支持将概念模型中的实体映射到数据源中的存储过程。它提供以下方式用于查询 [EDM](https://www.baidu.com/s?wd=EDM&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAR1P1nsrj6snAw9uyDs0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6Lrj6dP1mY) 并返回对象：  
· [LINQ](https://www.baidu.com/s?wd=LINQ&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAR1P1nsrj6snAw9uyDs0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6Lrj6dP1mY) to Entities -- 提供语言集成查询 ([LINQ](https://www.baidu.com/s?wd=LINQ&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAR1P1nsrj6snAw9uyDs0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6Lrj6dP1mY)) 支持用于查询在概念模型中定义的实体类型。  
· Entity [SQL](https://www.baidu.com/s?wd=SQL&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAR1P1nsrj6snAw9uyDs0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6Lrj6dP1mY) -- 与存储无关的 [SQL](https://www.baidu.com/s?wd=SQL&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAR1P1nsrj6snAw9uyDs0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6Lrj6dP1mY) 方言，直接使用概念模型中的实体并支持诸如继承和关系等 EDM 功能。  
· 查询生成器方法 --可以使用 [LINQ](https://www.baidu.com/s?wd=LINQ&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAR1P1nsrj6snAw9uyDs0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6Lrj6dP1mY) 风格的查询方法构造 Entity [SQL](https://www.baidu.com/s?wd=SQL&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAR1P1nsrj6snAw9uyDs0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6Lrj6dP1mY) 查询。

## **EF模型的数据库连接初始化方式**

### [Database.SetInitializer的几种参数](http://www.cnblogs.com/brown-birds/p/6039293.html)

**一：数据库不存在时重新创建数据库**

Database.SetInitializer<testContext>(new CreateDatabaseIfNotExists<testContext>());

**二：每次启动应用程序时创建数据库**

Database.SetInitializer<testContext>(new DropCreateDatabaseAlways<testContext>());

**三：模型更改时重新创建数据库**

Database.SetInitializer<testContext>(new DropCreateDatabaseIfModelChanges<testContext>());

**四：从不创建数据库**

Database.SetInitializer<testContext>(null);

## **entity framework的网络教程，英文**

<http://www.entityframeworktutorial.net/code-first/database-initialization-strategy-in-code-first.aspx>

# 常见数据库关系型数据库

1、Microsoft SQL Server Compact 3.5开发笔记

嵌入式本地数据库。

sqlserver

mysql

sqlserver ce

sqlite

oracle

nosql 泛指非关系型数据库

关系型数据库和非关系型数据库对比。

https://blog.csdn.net/liutong123987/article/details/79203963

HAL系统 数据库持久层

# NOSQL和关系数据库

https://blog.csdn.net/xgf415/article/details/75370783

# 存储过程

<https://blog.csdn.net/tojohnonly/article/details/70738629>

EF中的存储过程和使用SQL和存储过程

<https://www.cnblogs.com/lvdongjie/p/5856306.html>

SQL存储过程语法

<https://www.cnblogs.com/chaoa/articles/3894311.html>

# 名词相关

## **T-SQL**

如果用一句话概括的话，SQL作为结构化查询语言，是标准的关系型数据库通用的标准语言；T-SQL是在SQL基础上扩展的SQL Server中使用的语言。

# 数据库和数据仓库

数据库：传统的关系型数据库的主要应用，主要是基本的、日常的事务处理，例如银行交易。

数据仓库：数据仓库系统的主要应用主要是OLAP（On-Line Analytical Processing），支持复杂的分析操作，侧重决策支持，并且提供直观易懂的查询结果。

举个最常见的例子，拿电商行业来说好了。

基本每家电商公司都会经历，从只需要业务数据库到要数据仓库的阶段。

* 电商早期启动非常容易，入行门槛低。找个外包团队，做了一个可以下单的网页前端 + 几台服务器 + 一个MySQL，就能开门迎客了。这好比手工作坊时期。
* 第二阶段，流量来了，客户和订单都多起来了，普通查询已经有压力了，这个时候就需要升级架构变成多台服务器和多个业务数据库（量大+分库分表），这个阶段的业务数字和指标还可以勉强从业务数据库里查询。初步进入工业化。
* 第三个阶段，一般需要 3-5 年左右的时间，随着业务指数级的增长，数据量的会陡增，公司角色也开始多了起来，开始有了 CEO、CMO、CIO，最初非常粗放的：“昨天的收入是多少”、“上个月的 PV、UV 是多少”，逐渐演化到非常精细化和具体的用户的集群分析，特定用户在某种使用场景中，例如“20~30岁女性用户在过去五年的第一季度化妆品类商品的购买行为与公司进行的促销活动方案之间的关系”。

这类非常具体，且能够对公司决策起到关键性作用的问题，基本很难从业务数据库从调取出来。原因在于：

1. 业务数据库中的数据结构是为了完成交易而设计的，不是为了而查询和分析的便利设计的。
2. 业务数据库大多是读写优化的，即又要读（查看商品信息），也要写（产生订单，完成支付）。因此对于大量数据的读（查询指标，一般是复杂的只读类型查询）是支持不足的。

而怎么解决这个问题，此时我们就需要建立一个数据仓库了，公司也算开始进入信息化阶段了。数据仓库的作用在于：

1. 数据结构为了分析和查询的便利；
2. 只读优化的数据库，即不需要它写入速度多么快，只要做大量数据的复杂查询的速度足够快就行了。

那么在这里前一种业务数据库（读写都优化）的是业务性数据库，后一种是分析性数据库，即数据仓库。

最后总结一下：

数据库 比较流行的有：MySQL, Oracle, SqlServer等  
数据仓库 比较流行的有：AWS Redshift, Greenplum, Hive等

这样把数据从业务性的数据库中提取、加工、导入分析性的数据库就是传统的 ETL 工作。现在也有一些新的方法，这展开说又是另一件事情了，有机会再详细说说

看了一下我2003年写的东西，如果不是我失忆了，那就是：“尽管很多东西你早就知道了，却在多年的实践中才不同程度地有所理解。” 所以我不成熟地希望大家还是多多实践吧，尽量少花些时间在这些诸如本质啊区别啊啥啥的问题上面。当然,如果你精通ORACLE,我支持你去了解ORACLE, DB2, MS SQL, MYSQL之间有什么区别。

# 问题和解决办法

## Install-Package MySql.Data.Entity.EF6失败的解决办法

<http://blog.csdn.net/yw1688/article/details/50865837>

如果采用mysql server的安装程序，connector等安装目录

在vs2013中用mysql+EF6时，通常是这样子操作的：

1 Install-Package EntityFramework -Version 6.0.0然后Enter  
2 Install-Package EntityFramework.zh-Hans -Version 6.0.0  
3 Install-Package MySql.Data.Entity.EF6

1和2只要网络是正常的，基本都会成功，但是第三个在执行的时候，通常会报错，如下：

Install-Package : 无法找到程序包“MySql.Data.Entity.EF6”。  
所在位置

行:1 字符: 16  
+ Install-Package

<<<< MySql.Data.Entity.EF6  
+ CategoryInfo

: NotSpecified: (:) [Install-Package], InvalidOperationException  
+ FullyQualifiedErrorId

: NuGetCmdletUnhandledException,NuGet.PowerShell.Commands.InstallPackageCommand

解决办法：

通过右键项目->添加引用->浏览，把MySQL Connector Net 6.8.3\Assemblies\v4.5下的所有dll文件引用进来，

4 在web.config里面添加provider节点即可：

<provider invariantName="MySql.Data.MySqlClient" type="MySql.Data.MySqlClient.MySqlProviderServices, MySql.Data.Entity.EF6"></provider>

经过这些步骤，问题解决了！！

## XMl连接字

### 连接有问题的xml

<!--<?xml version="1.0" encoding="utf-8"?>

<configuration>

<configSections>

--><!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 --><!--

<section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />

</configSections>

<connectionStrings>

<add name="MyContext" connectionString="Data Source=localhost;port=3306;Initial Catalog=my\_oa;user id=sa;password=1234;" providerName="MySql.Data.MySqlClient" />

</connectionStrings>

<entityFramework codeConfigurationType="MySql.Data.Entity.MySqlEFConfiguration, MySql.Data.Entity.EF6">

<defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework" />

</entityFramework>

<system.data>

<DbProviderFactories>

<remove invariant="MySql.Data.MySqlClient" />

<add description=".Net Framework Data Provider for MySQL" invariant="MySql.Data.MySqlClient" name="MySQL Data Provider" type="MySql.Data.MySqlClient.MySqlClientFactory, MySql.Data, Version=6.10.4.0, Culture=neutral, PublicKeyToken=c5687fc88969c44d" />

</DbProviderFactories>

</system.data></configuration>-->

### 成功的xml

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<configSections>

<!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->

<section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false"/>

</configSections>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5"/>

</startup>

<entityFramework codeConfigurationType="MySql.Data.Entity.MySqlEFConfiguration, MySql.Data.Entity.EF6">

<defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">

<parameters>

<parameter value="mssqllocaldb"/>

</parameters>

</defaultConnectionFactory>

<providers>

<provider invariantName="MySql.Data.MySqlClient" type="MySql.Data.MySqlClient.MySqlProviderServices, MySql.Data.Entity.EF6"/>

</providers>

</entityFramework>

<connectionStrings>

<add name="MyContext" connectionString="Data Source=localhost;port=3306;Initial Catalog=my\_oa;user id=root;password=123456;Character Set=utf8;" providerName="MySql.Data.MySqlClient"/>

</connectionStrings>

<system.data>

<DbProviderFactories>

<remove invariant="MySql.Data.MySqlClient"/>

<add name="MySQL Data Provider" invariant="MySql.Data.MySqlClient" description=".Net Framework Data Provider for MySQL" type="MySql.Data.MySqlClient.MySqlClientFactory, MySql.Data, Version=6.9.9.0, Culture=neutral, PublicKeyToken=c5687fc88969c44d"/>

</DbProviderFactories>

</system.data>

</configuration>