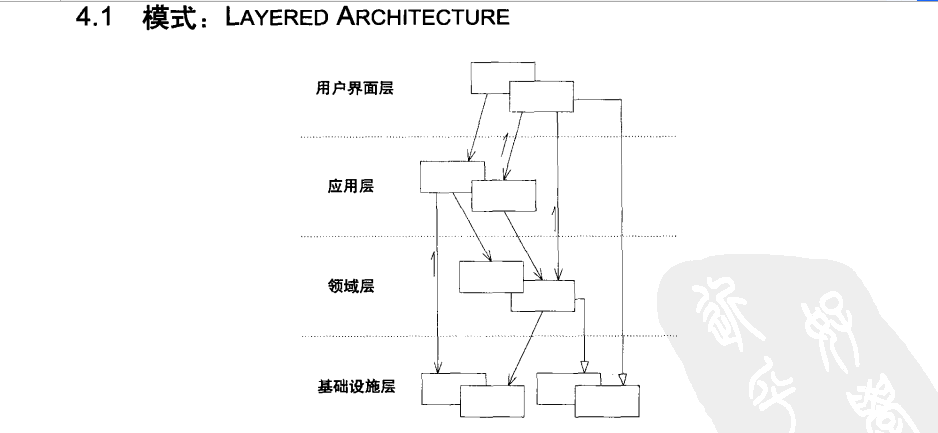
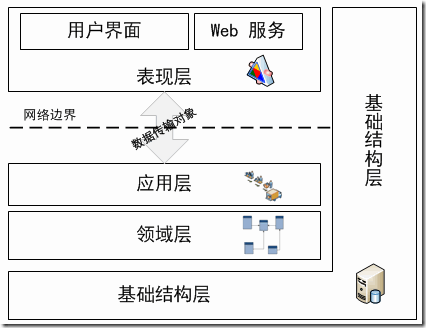
### **1、项目分层**

领域驱动设计将软件系统分为四层：基础结构层、领域层、应用层和表现层。来看看书中的分层：

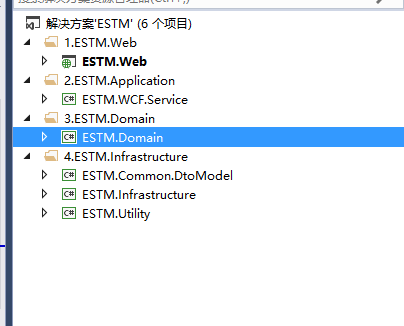


其实在dax.net的系列中这张图更能说明这种架构



### **2、项目架构**

博主打算用权限系统的案例说明的领域驱动设计的项目架构。项目严格按照表现层、应用层、领域层、基础设施层来划分。



表现层：MVC的Web项目，负责UI呈现。

应用层：WCF服务，负责协调领域层的调用，向UI层提供需要的接口。

领域层：定义领域实体和领域逻辑。

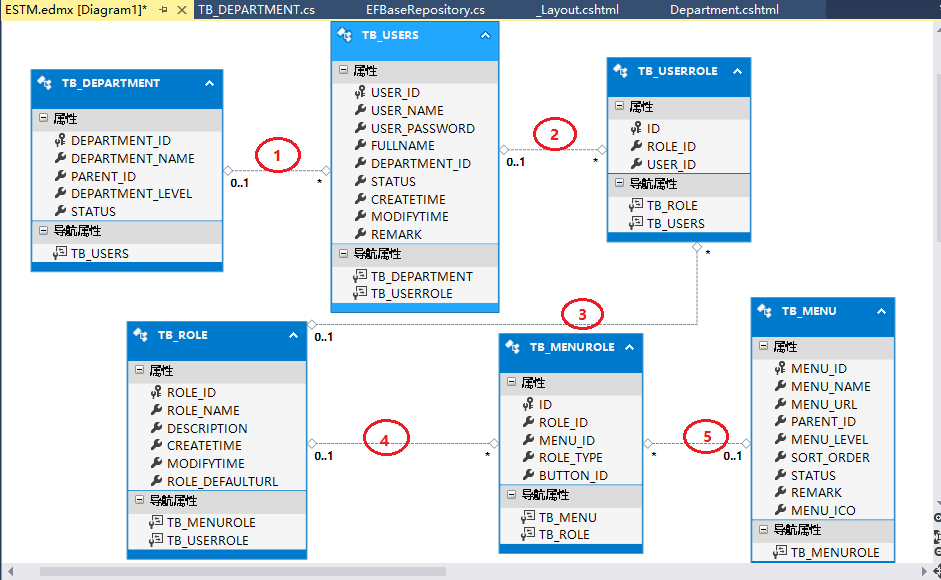
基础设施层：一些通用的技术，比如AOP、MEF注入、通用的工具类、DTO模型层，这里为什么要有一个DTO模型层，DTO是用于UI展现用的纯数据Model，它不包含实体行为，是一种贫血的模型。

整个项目的调用方式严格按照DDD设计来进行，UI层通过WCF服务调用应用层的WCF接口，WCF服务通过仓储调用领域层里面的接口，基础设施层贯穿其他各层，在需要的项目中都可以引用基础设施层里面的内库。

### **3、代码示例**

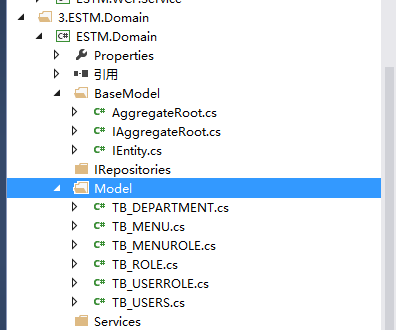
接下来，博主就根据自己的理解，从零开始使用这种架构写一个简单的权限管理系统。由于是领域驱动设计，所以，文章的重点会放在领域层，项目使用了EF的Model First来进行，先设计实体，后生成数据库。

#### **3.1 首先来看看表结构**



#### **3.2 聚合的划分**

在领域层里面我们新建一个BaseModel，里面有三个类



这三个类IEntity、IAggregateRoot、AggregateRoot分别定义了实体的接口、聚合根的接口、聚合根的抽象实现类。

//用作泛型约束，表示继承自该接口的为领域实体

public interface IEntity

{

}

/// <summary>

/// 聚合根接口，用作泛型约束，约束领域实体为聚合根，表示实现了该接口的为聚合根实例，由于聚合根也是领域实体的一种，所以也要实现IEntity接口

/// </summary>

public interface IAggregateRoot:IEntity

{

}

　　 /// <summary>

/// 聚合根的抽象实现类，定义聚合根的公共属性和行为

/// </summary>

public abstract class AggregateRoot:IAggregateRoot

{

}