# ERP开发指南

本文档的说明...

# 对象的设计

在对象的设计中，主要包含ID主健设计，对象封装的设计，主对象与子对象关系的设计。主对象定义为一对多或一对一的关系性数据库的主健对应的对象，反之为子对象。

### **ID主健的设计**

ID为对象的唯一性主健，保证对象的唯一性。一般采用如下两种方式定义，标准的、有数据本身唯一性约束的使用MD5编码，其他的使用自定义算法。

#### MD5编码

例：标准产品的ID设计，使用命名+品牌+包装+封装+批次+生产批次的MD5值作为ID值。

string.Concat(this.Name, this.Manufacturer.Name, this.PackageCase

, this.Packaging, this.Batch, this.DateCode).MD5();

#### **自定义算法**

使用自定义算法时，在本项目下根目录下定义枚举类PKeyType作为各对象的ID其起始值。如NtErp.Wss.Oss.Services项目，需在NtErp.Wss.Oss.Services项目的根目录下定义枚举文件：PKeyType.cs，命名空间为本地项目默认命名空间。

如何设置主健枚举类型，在特性中定义本地数据库，定义主键值的起始字符，是否时间类型，长度。

/// <summary>

/// 入库通知

/// </summary>

[Repository(typeof(ScWmsReponsitory))]

[PKey("ENT", PKeySigner.Mode.Time, 10)]

EntryNotice = 10000

生成的主健值为：ENT201809030000000006

例：订单ID的设计，使用自定义算法作为ID值。

this.ID = Needs.Overall.PKeySigner.Pick(PKeyType.OrderItem)；

使用自定义算法时，在本地数据库创建表：PrimaryKeys，SQL脚本参考PrimaryKeys.Sql文件。

### **主对象设计**

#### **一对一类型的主对象**

在主对象中包含子对象，不包含子对象的主健。在主对象持久化时，同时持久化子对象。

如何持久化主对象及子对象，参考第三节：对象的持久化。

#### **一对多类型的主对象**

在主对象中包含子对象，子对象设计为Items，在主对象持久时，同时持久化子对象。

如何持久化主对象及子对象，参考第三节：对象的持久化。如何进行Items对象的填装，参考，第三节，View视图，子对象的查询。

#### **外键类型的对象**

作为外键类型的对象，不可包含主对象，仅包含主对象主健ID，例：Order与OrderItem的关系为1对多，OrderItem为Order的外键对象，在OrderItem中， 仅设计Order的主健OrderID, 代码如下：

public string OrderID { get; set; }

### **冗余字段在对象中的设计**

例：订单中的总金额Total的设计，需要根据订单中产品的总金额及附加价值计算得出，在对象中计算并在对象持久化时，持久化到到数据库中。

decimal total;

/// <summary>

/// 订单总金额

/// </summary>

public decimal Total

{

get

{

if (this.items == null || this.premiums == null)

{

return total;

}

return this.Items.Total + this.Premiums?.Total ?? 0m;

}

internal set { this.total = value; }

}

### **对象的实例化**

在实例化中，完成默认属性的赋值，例：

public ArrivalItem()

{

this.UpdateDate = this.CreateDate = DateTime.Now;

}

# 二、对象的持久化

### **1、对象的持久化**

对象本身的持久化，使用using 语句中声明和实例化本地的 LinqReponsitory 对象。

例： using (var reponsitory = new Layer.Data.Sqls.CvOssReponsitory())

新增或更新数据库的两种方式，1，在实例化的reponsitory对象中Insert直接添加Linq数据库对象，例：

reponsitory.Insert(new Layer.Data.Sqls.ScSales.Orders

{

ID = this.ID,

AdminID = this.Admin.ID,

CompanyID = this.Company.ID,

ConsigneeID = this.Consignee.ID,

DelivererID = this.Deliverer.ID,

BeneficiaryID = this.Beneficiary.ID,

Status = (int)this.Status,

CreateDate = this.CreateDate,

UpdateDate = DateTime.Now,

Summary = this.Summary

});

2，在Extends文件下，定义对象的扩展方法，返回该对象要持久化的Linq数据库对象。命名空间为该对象的命名空间加Extends，   例：Needs.Uls.Services.Extends，对象的拓展方法名称定义为ToLinq，例：

static public Layer.Data.Sqls.CvOss.UserOutputs ToLinq(this Models.UserOutput entity)

{

return new Layer.Data.Sqls.CvOss.UserOutputs

{

ID = entity.ID,

ClientID = entity.ClientID,

Type = (int)entity.Type,

From = (int)entity.From,

OrderID = entity.OrderID,

UserInputID = entity.UserInputID,

Currency = (int)entity.Currency,

Amount = entity.Amount,

DateIndex = entity.DateIndex,

CreateDate = entity.CreateDate

};

}

reponsitory.Insert(this.ToLinq());

### **2、对象中子对象的持久化**

如果对象中包含需要持久化的子对象，要在该对象持久化的函数中进行子对象的持久化，子对象的重复性验证在子对象中的持久化中进行，例：

public void Enter()

{

this.Consignee.Enter();

}

### **持久化前的验证**

持久化前的验证，在对象的持久化函数中进行，需要访问数据库的验证，需使用实例化本地的 LinqReponsitory 对象进行数据库查询。

例：重复性验证

int count =reponsitory.ReadTable<Layer.Data.Sqls.ScSales.Orders>().Count(item => item.ID == this.ID);

if (count == 0)

新增

else

更新

# 三、View视图

View类的作用是为对象提供统一的查询.？这样解释还需要陈瀚补充

### **1、主视图及命名规范**

定义对象的View时，首先继承：

UniqueView<Models.EntryNotice, ScWmsReponsitory>，

参数分别为:View索引器返回的对象类型，Linq数据访问类型。

例：

public class EntryNoticesView : UniqueView<Models.EntryNotice, ScWmsReponsitory>。

命名规范：对象名+s+View。

关于Linq的使用，自行学习。

### **如何继承**

### **视图的扩展方法**

### **4、View在对象中使用**

主对象中的Items,不在View中进行关联子表查询，在get时进行查询。

如果需要分页查询，在Itembase中继承IQueryable接口

Premiums premiums;

/// <summary>

/// 产品项

/// </summary>

public Premiums Premiums

{

get

{

if (premiums == null)

{

using (var view = new Views.PremiumsView())

{

var query = view.Where(item => item.OrderID == this.ID);

this.Premiums = new Premiums(query);

}

}

return this.premiums;

}

set

{

if (value == null)

{

return;

}

this.premiums = new Premiums(value, delegate (Premium item)

{

item.OrderID = this.ID;

});

}

}

### **5、在View中完成Binding**

# 四、统一调用路径

案例：客户的订单，访问路径： Needs.Erp.Website.MyOrders

百度自行学习链式编程。

# 五、命名空间

框架中各类库创建及命名空的说明，请阅读《项目建立说明》。

项目命名空间的规范如下：

### **1、接口、对象**

     接口、对象的cs文件可用文件Interfaces、Models 文件区分，但是须在同一命名下，例： Needs.Uls.Services.Models

### **2、枚举**

    项目中的枚举cs文件，建议直接定义在项目的根目录中，可用文件Enums区分，但不受命名空间管控， 例：Needs.Uls.Services

### **3、View**

    视图，用文件Views区分， 受命名空间管控，例如：Needs.Uls.Services.Views

### **4、拓展方法**

     对象或View视图的拓展方法，统一定义在项目根目录下的Extends文件夹下，命名规范为项目的默认命名空间+Extends，例：Needs.Uls.Services.Extends。

对象拓展类文件的名称为：对象名+Extend。例：Order对象的拓展方法定义为：OrdersExtends.cs

# 六、Admin对象的使用

### **1、Admin的本地扩展**

如何扩展本地Admin?

### **2、Admin特性**

在本地AdminTop的类上添加特性:

[Needs.Underly.FactoryView(typeof(Views.AdminTopView))]

在前端可以根据系统当前管理员ID获取Ad'min对象，

Var Admin=Needs.Underly.FkoFactory<Needs.Chains.Services.Models.AdminTop>.Create(Needs.Erp.ErpPlot.Current.ID);

# 七、前端开发

前端开发面向对象开发，

### PageBase

页面继承

### **子对象**

在前端一对多或一对一的子对象的，需要在持久化前通过外键值填装，不可仅对外键赋值。

参考代码：

var project = new NtErp.Crm.Services.Models.AdminProject();

project.Company=

Needs.Overall.FkoFactory<NtErp.Crm.Services.Models.Company>.Create(project.CompanyID);

### **枚举**

var project = new NtErp.Crm.Services.Models.AdminProject();

project.JobType = NtErp.Crm.Services.Enums.JobType.Sales;

获取枚举的描述：TypeName = item.Type.GetDescription(),

### **向UI吐数据**

**在linq to sql 的查询数据库中，无法使用，通过使用**

Func<NtErp.Crm.Services.Models.Project, object> convert = item => new

{

item.ID,

item.Name,

ClientName = item.Client.Name,

CompanyName = item.Company.Name,

CurrencyName = item.Currency.GetDescription(),

item.Valuation,

item.ExpectValuation,

StatusName = item.Status.GetDescription(),

AdminName = item.Admin.UserName

};

this.Paging(data, convert);

### **Admin 对象的获取**

获取当前Admin(系统登陆管理员)ID：Needs.Erp.ErpPlot.Current.ID。

Needs.Underly.FkoFactory<Needs.Chains.Services.Models.Admin>.Create(Needs.Erp.ErpPlot.Current.ID);