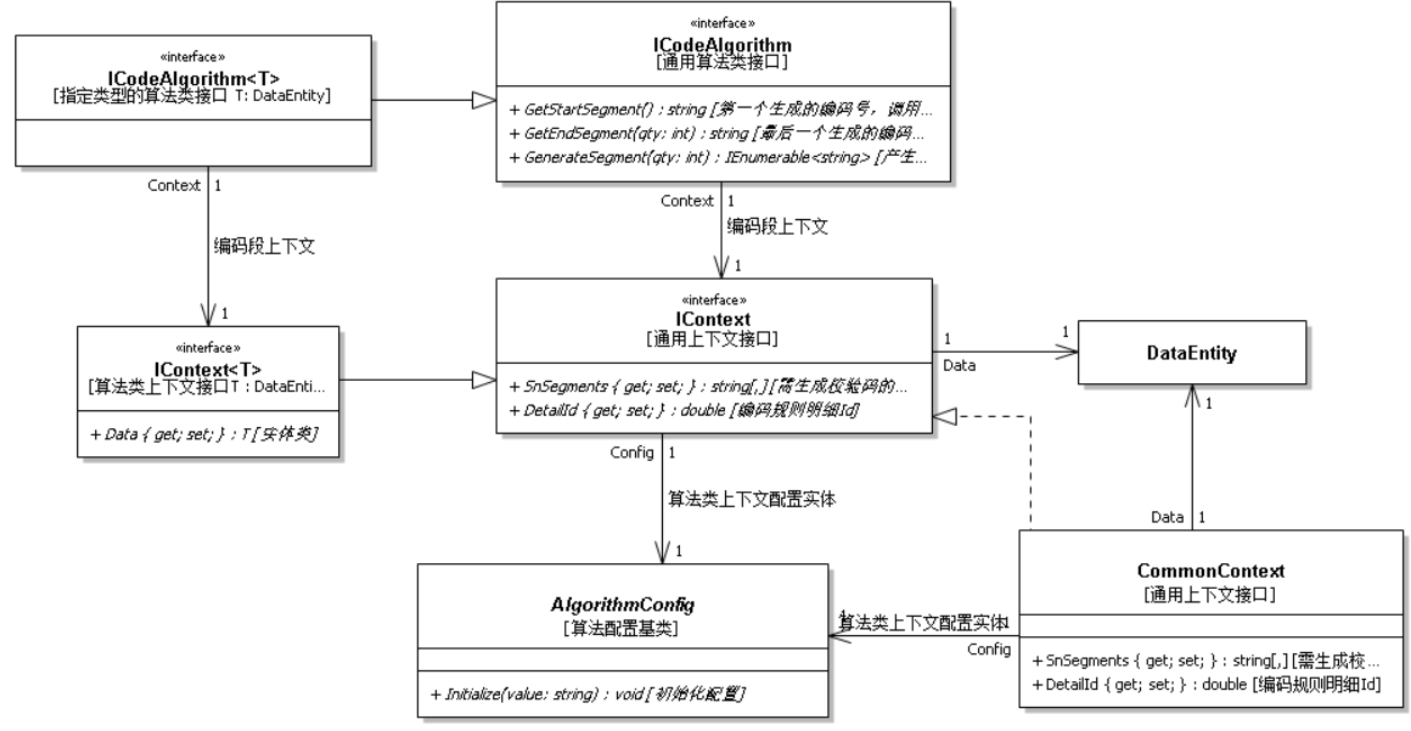
自定义编码算法

**业务场景：**

APS生产订单编号有三种情况：  
(1)正常的订单 MO20180424XXX  
(2)拆分后的订单 CXXX- D 说明：固定+源单号+流水号(区分源单号)  
(3)合并后的订单 HXX\_ XXX 说明：固定+子单数+流水号  
对于正常订单可以通过框架的原有编码算法配置实现，但是拆分跟合并的情况编号跟当前的订单实体有关联，所以针对2、3情况得自定义编码算法。

算法实现接口抽象类类图



# 实现：

## 一、开发

### 算法1：生产订单拆单算法（源单号）

创建算法类实现编码算法泛型基类，实现GetCode(DataEntity data)方法，方法返回源单号。  
注意：标记特性Algorithm，指定算法配置类，算法类型

/// <summary>

/// 生产订单拆单算法

/// </summary>

[Algorithm("生产订单拆单算法", typeof(CodeAlgorithmConfig), AlgorithmType.Other)]

[RootEntity, Serializable]

public class SplitBillAlgorithm : CodeAlgorithm<ProductionOrder>

{

/// <summary>

/// 获取单号片段

/// </summary>

/// <param name="data">生产订单</param>

/// <returns>单号片段</returns>

public override string GetCode(ProductionOrder data)

{

return "{0}-".FormatArgs(data.ProductionOrderCodes);

}

}

### 算法2：生产订单合单算法（子单数量）

/// <summary>

/// 生产订单合单算法

/// </summary>

[Algorithm("生产订单合单算法", typeof(CodeAlgorithmConfig), AlgorithmType.Other)]

[RootEntity, Serializable]

public class CombineBillAlgorithm : CodeAlgorithm<ProductionOrder>

{

/// <summary>

/// 获取单号片段

/// </summary>

/// <param name="data">生产订单</param>

/// <returns>单号片段</returns>

public override string GetCode(ProductionOrder data)

{

return "{0}\_".FormatArgs(data.Children.Count.ToString("D2"));

}

}

### 算法3：序列算法(区分源单号)

（1）区分源单号的序列算法，需要根据源单号来生成序列，也就是源单号不同时序列重新开始计算

/// <summary>

/// 订单单号序列

/// </summary>

[RootEntity, Serializable]

public class NoSequence : SequenceBase

{

#region 单号 No

/// <summary>

/// 单号

/// </summary>

public static readonly Property<string> NoProperty = P<NoSequence>.Register(e => e.No);

/// <summary>

/// 单号

/// </summary>

public string No

{

get { return this.GetProperty(NoProperty); }

set { this.SetProperty(NoProperty, value); }

}

#endregion

}

/// <summary>

/// 订单单号序列实体配置

/// </summary>

internal class NoSequenceConfig : EntityConfig<NoSequence>

{

/// <summary>

/// 配置元数据

/// </summary>

protected override void ConfigMeta()

{

Meta.MapTable("SEQ\_NO").MapAllProperties();

Meta.EnablePhantoms();

Meta.EnableTimeStamp();

}

}

（2）新增算法类SequenceByNo继承自序列计算基类，重写GetSequenceBase()方法，实现对单号序列对象的操作

/// <summary>

/// 序列生成算法(区分生产订单源单号)

/// </summary>

[Algorithm("序列生成算法(区分生产订单源单号)", typeof(SequenceConfig), AlgorithmType.Other)]

[RootEntity, Serializable]

public class SequenceByNo : SequenceAlgorithm

{

/// <summary>

/// 获取序列

/// </summary>

/// <param name="startValue">开始值</param>

/// <returns>序列</returns>

protected override SequenceBase GetSequenceBase(int startValue)

{

var order = Context.Data as ProductionOrder;

return RT.Service.Resolve<ProductionOrderController>().GetNoSequence(Context.DetailId, order?.ProductionOrderCodes, startValue);

}

}

(3)控制器操作方法

/// <summary>

/// 获取序列

/// </summary>

/// <param name="detailId">明细ID</param>

/// <param name="no">单号</param>

/// <param name="startValue">开始值</param>

/// <returns>序列</returns>

public virtual NoSequence GetNoSequence(double detailId, string no, int startValue)

{

var detail = RF.GetById<NumberRuleDetail>(detailId);

if (detail == null)

throw new EntityNotFoundException(typeof(NumberRuleDetail), detailId);

var result = Query<NoSequence>()

.Where(p => p.DetailId == detailId && p.No == no)

.FirstOrDefault();

if (result == null)

{

using (var tran = DB.AutonomousTransactionScope(RF.Find<NumberRuleDetail>()))

{

////先进来Update，进行同步锁

DB.Update<NumberRuleDetail>().Set(p => p.Id, detailId).Where(p => p.Id == detailId).Execute();

////再查一次

result = DB.Query<NoSequence>()

.Where(p => p.DetailId == detailId && p.No == no)

.FirstOrDefault();

if (result == null)

{

using (var seqTran = DB.AutonomousTransactionScope(RF.Find<NoSequence>()))

{

result = new NoSequence();

result.CurrentValue = startValue;

result.No = no;

result.DetailId = detailId;

RF.Save(result);

seqTran.Complete();

}

}

tran.Complete();

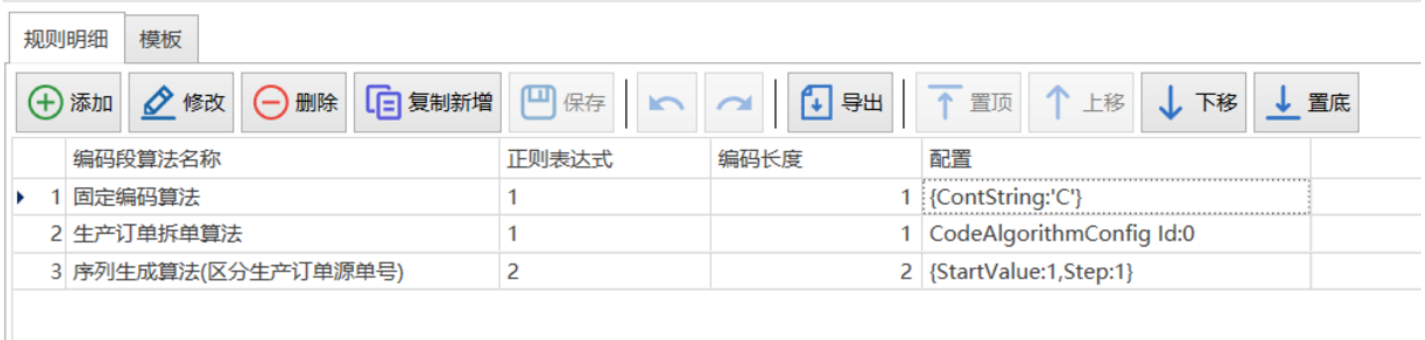
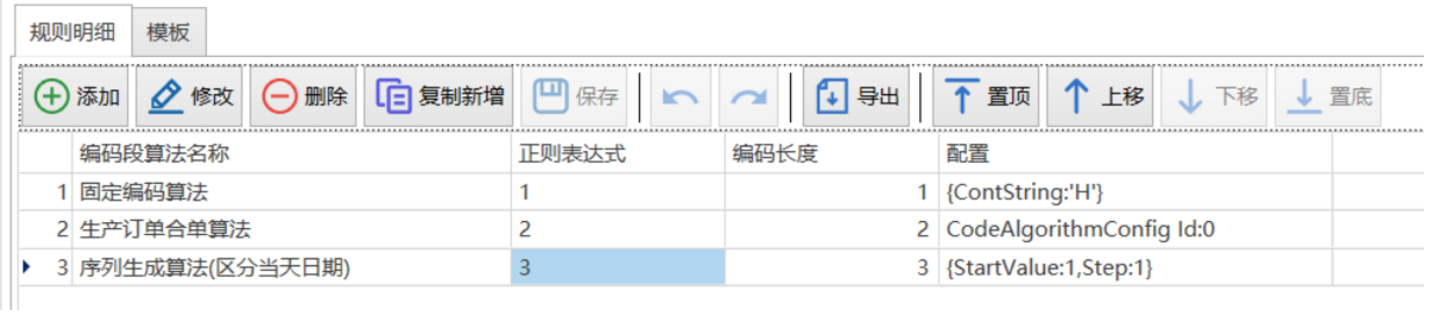
}

}

return result;

}

# 二、配置

1、打开编码段菜单，点击初始化，成功后查看是否存在算法名称：生产订单拆单算法、  
生产订单合单算法、序列生成算法(区分生产订单源单号)三个算法，并启用它们。  
  
2、打开编码规则菜单，添加三个编码规则：  
（1）正常生产订单生产规则： MO20180424XXX  
  
（2）生产订单拆单单号生成规则： CXXX D 说明：固定+源单号+流水号(区分原单号)  
（3）生产订单合单单号生成规则 HXX XXX 说明：固定+子单数+流水号  
  
3、打开生产订单菜单，点击配置项配置单号生产规则

