

Yigo语言开发指南

目录

1. Yigo语言简介	1
1.1. 数据模型	1
1.2. 界面模型	2
1.3. 业务流程	3
2. Yigo运行环境	4
2.1. 服务器端环境	4
3. Yigo应用框架	11
3.1. 工程结构	11
3.2. 典型的应用结构	12
4. 界面模型	17
4.1. Form结构定义	17
4.2. 基本组件Component定义	17
4.3. 基本布局类组件定义	18
4.4. 功能入口定义	18
5. 面板组件	20
5.1. 列式布局面板ColumnLayoutPanel	20
5.2. 边界布局面板BorderLayoutPanel	22
5.3. 流式布局面板FlowLayoutPanel	24
5.4. 选项卡面板TabPanel	27
5.5. 拆分面板SplitPanel	29
5.6. 网格布局面板GridLayoutPanel	31
5.7. 流式表布局面板FluidTableLayoutPanel	32
5.8. 弹性流布局面板FlexFlowLayoutPanel	34
5.9. 面板组合应用实例	36
6. 功能类组件	41
6.1. 基本组件	41
6.2. 视图组件	49
6.3. 图表	51
6.4. 字典(Dict)	53
6.5. 表格(Grid)	57
6.6. 其他	62
7. 表单逻辑处理	66
7.1. 数据逻辑	66
7.2. 计算值	68
7.3. 检查规则	71
7.4. 可见性计算	73
7.5. 可用性计算	74
8. Android平台应用配置	75
8.1. 移动应用组成结构	75
8.2. 定义AndroidDef.xml 文件	75
8.3. 尺寸定义	77
8.4. 移动布局面板	77
8.5. 移动基本组件	80
8.6. 综合示例	82
9. 布局及选择器(Layout)	87
9.1. 布局介绍	87
9.2. 实例	88
10. 表达式	93
10.1. 运算符	93
10.2. 标识符、常量及保留字	95
10.3. 注释	95
10.4. 函数	95
10.5. 控制语句	97
数据对象基础	105
1. 数据对象定义	105

2. 使用表单管理数据对象	107
数据对象处理	110
1. 数据对象中的保存	110
2. 数据对象载入的处理	111
3. 中间层检查规则	112
数据对象高级应用	113
1. 数据过滤	113
2. 复合字典	114
3. 表单报表	115
4. 查询条件	120
11. 数据迁移	123
11.1. 基本介绍	123
11.2. 入库实例	125
11.3. 出库实例	130
11.4. 退单实例	134
11.5. 期间实例	135
12. 数据映射	137
12.1. 基本介绍	137
12.2. 示例1(普通映射)	141
12.3. 示例2(最大可映射值&多种值映射)	144
12.4. 示例3(可用映射量计算)	145
12.5. 示例4(数据反填&反填条件)	147
12.6. 示例5(跨级反填)	148
12.7. 示例6(上引下推)	150
13. 业务流程基础	154
13.1. 基本概念	154
13.2. 业务流程元素	156
13.3. 工作台	159
14. 业务流程任务处理	163
14.1. 审批	163
14.2. 状态	170
14.3. 用户任务	171
14.4. 系统任务	173
14.5. 选择	174
14.6. 分支与合并	175
14.7. 会签	176
14.8. 子流程	180
14.9. 数据映射	181
14.10. 加签	182
14.11. 流程重启/重提交	184
14.12. 状态机操作	188
15. 自由流程	191
15.1. 背景	191
15.2. 案例分析	191
16. 工作项系统	195
16.1. 工作项	195
16.2. 参与者	195
17. 任务的操作实现	202
17.1. 任务转移	202
17.2. 任务的撤消(回滚模式)	206
18. 打印报表基础	210
18.1. 报表的结构	210
18.2. 一般明细报表	211
18.3. 分组	214
18.4. 列扩展示例-单层扩展	218
18.5. 列扩展示例-嵌套扩展	220
18.6. 列扩展示例-列扩展合计	220
18.7. 内容的溢出处理	221

18.8. 页眉和页尾	224
18.9. 自动拆行	225

第 1 章 Yigo语言简介

目录

1.1. 数据模型	1
1.2. 界面模型	2
1.3. 业务流程	3

Yigo语言(以后简称为Yigo)是面向信息管理系统软件开发语言,主要的目的是快速建立并生成用户可使用的信息管理系统,并为些提供了一整套模型用于帮助用户建立自己的个性化系统。Yigo语言的历史介绍...,后面补。

从软件开发的历史来看,无论是早期的过程化编程还是面向对象的编程,瀑布式软件过程是迭代式软件过程,信息系统的开发都意味着大量的人力、物力以及大规模的代码工作量。传统的软件开发过程包括需求、设计、实现、测试等阶段,每个阶段需要不同的角色来完成,各个阶段均容易造成理解的偏差,导致软件后期的返工,意味着大量的人力物力浪费。另外在实现过程中,大量的代码开发也需要大量高成本的编程人员,变更和错误均意味着代码的重新编写,导致传统软件开发周期漫长,适应需求变化的能力弱。同时信息系统具有大量的共性,这些代码需要重复编写,极大的降低了软件生产力。

在此背景下,Yigo语言通过不断的创新,提出了信息系统开发的语言模型框架,并提供了一整套完整的信息系统的运行平台。

Yigo提供了三个主要的模型框架和整合这些模型的脚本和函数集;同时Yigo运行平台还提供了其它的信息系统建设所需要的组件,包括报表系统、全文检索等。

1.1:数据模型

数据模型定义Yigo语言处理的信息的结构和规则,数据是信息系统主要的处理对象,Yigo语言的数据模型描述了数据的结构、数据操作、数据约束以及数据之间的关系。

1.1.1:数据结构

Yigo语言的主要数据结构基于关系模型,主要的处理数据称为数据对象,每个数据对象均由多个表组成;数据结构主要定义每个数据对象由哪些表组成以及表包含了哪些列,数据表定义了表的基本属性,包括表的标识,表的持久化特征,表同数据库表的映射;列定义了列的标识、类型等,列的持久化特征以及如何同数据表列映射;

数据结构的定义同时还定义了数据的权限控制,在列的定义上说明数据如何同数据权限系统对应。

1.1.2:数据操作

Yigo提供了数据的基本操作,包括对数据的新建、修改和删除等机制,信息系统需要编写大量的数据操作的代码,Yigo提供了完整的数据操作的功能,避免大量的重复性代码,同一般性的持久化框架不同,Yigo提供了从前端到后端的完全的解决方案,同时也提供了在数据操作过程中数据的版本冲突问题,通过定义不同的存储级别,有效的控制在并发情况下数据的冲突问题。

1.1.3:数据约束

数据的约束是为了防止不规划和不符合业务要求的数据进入数据库,对于每个用户数据,数据各组成部分之间均有相应的关联关系和约束条件,不满足规则的数据一旦进入数据库会对业务的进行产生极大的干扰,因此给数据添加约束是软件系统开发中必不可少的部分,这部分的工作占有较大的工作量,Yigo将这些规则描述成统一的形式,方便应用的数据根据自己的需要独立定义自己的不同的规

则，Yigo运行平台在运行时根据这些规则对数据的约束进行检查；运行平台同时也处理了数据库系统的完整性规则，将这部分的工作向应用的开发者屏蔽，应用的工作仅需要考虑软件业务上的约束条件。

1.1.4: 数据关系

1.1.4.1: 数据迁移

数据迁移表示数据之间的归集关系，表示如何从数据对象通过分组关键字生成汇总数据，将对汇总数据的更新分布到每个数据存储中，以减少产生汇总数据的时间压力。同时迁移的目标表也作为信息系统中总量控制的依据。数据迁移通过定义数据主表同迁移的目标之间的数据转换关系，通过计算原数据对象上的分组关键字的值确定数据应该归集到目标迁移表中的哪条数据上。

1.1.4.2: 数据映射

数据映射表示数据之间的生成关系，用于从一个数据对象生成其它数据对象，在业务系统中，经常需要从某个单据生成其它单据，并且需要跟踪单据之间的关系；数据映射正是为了解决如何生成数据对象以及如何跟踪数据对象的来源关系。Yigo定义数据对象之间的映射关系和规则来生成目标数据对象，并通过关系树来记录数据对象的生成关系；

1.2: 界面模型

信息系统的开发需要大量的展现信息和编辑数据的界面，可以说，用户同信息系统的交互绝大多数都是通过用户界面来进行，离开了用户界面，信息系统的功能几近于零，因此用户界面的开发在整个信息系统开发中占有极大的工作量。在实践中，对于用户界面的需求承受着不同的用户角色的变化而变化，甚至对于单个的用户也有自己的不同偏好，因此满足界面的不同需要是个工作量很大，繁琐而变化极大的工作。随着时间的推移，在一个项目的进展过程中，界面也可能会出现很大的变化，为了应付界面的变动，传统的软件开发人员需要花费很大的精力编码来调整界面。Yigo在总结多年的信息系统开发经验的基础上，抽象出一套信息系统的界面模型，将软件开发人员从繁重的界面开发工作中解放出来。

界面模型包含界面组成、布局和样式选择器、事件模型等。

1.2.1: 界面组成

界面组成指组成用户界面的组件构成，Yigo将界面组件分为两类，一类是布局类组件，一类为功能类组件；布局类组件主要用于对其包含的组件进行布局，功能类组件提供用户的交互能力。Yigo提供了众多内置的布局类组件，包括列式布局、边界布局、网格布局、流布局、弹性流式布局、选项卡、流式表布局、线性布局等布局类组件；功能类组件包括文本编辑框、数值编辑框、按钮等组件，这里不一一列举。

1.2.2: 布局和样式选择器

布局及样式选择器用于在不同的运行平台上转换界面的位置布局和显示样式；Yigo界面运行在从普通的本地应用程序到Web应用再到移动应用程序，由于物理尺寸的限制，不同的运行平台对于界面的布局和样式要求均不同，因此需要在不同的平台间适应界面的转换需求，因此针对不同的运行平台及物理尺寸提供不同的布局和样式转换策略。

1.2.3: 事件模型

Yigo包装了底层平台的事件模型，为用户提供了跨平台一致的界面事件模型，在需要定义界面交互逻辑时，应用的开发者只需要关心一致的事件列表，不必关注于运行平台的细节；Yigo提供了丰富的事件模型，包括点击事件、值改变事件、表单载入事件、行单击事件、双击事件等。对于事件的处理，提供了Yigo的表达式、JavaScript和Java代码三种方式处理。

1.3: 业务流程

业务流程定义应用如何处理其各个业务的过程，应用由大量的处理过程组成，每个处理过程都有相应的处理任务和步骤，Yigo提供了针对过程的开发方法，定义了一套流程的定义语言，使信息系统的开发者可以通过Yigo的过程定义语言描述其过程、任务和任务的处理顺序。

1.3.1: 过程

过程定义一个完整且相对独立的业务处理，每一个用户的事务都有一些相对独立的处理步骤，过程定义将这些处理步骤和处理顺序和规则组织成一个完整的过程定义，用于描述一个用户处理事务。用户可以升级其业务处理过程，这样可以不断的适应新的环境需求，Yigo提供了流程的版本控制，方便用户升级和转换其业务处理过程，这个过程不需要通过代码去实现。

1.3.2: 任务

任务描述过程中的处理步骤，一个过程由多个任务组成，任务包括用户任务、手工任务和系统任务；用户任务是需用户操作系统来完成，比如审批、修改合同金额等；手工任务为系统外任务处理，比如打包包裹；系统任务为系统自动处理的任務，比如发邮件；后面再详细介绍。

1.3.3: 路由

路由描述了过程中任务是否需要处理和以怎样的顺序处理，路由包括选择、并行、互斥、合并等，后面再详细介绍。

第 2 章 Yigo运行环境

目录

2.1. 服务器端环境	4
-------------------	---

获取yigo2.zip文件后解压缩，会获取两个目录

2.1:服务器端环境

教程中的环境基于Tomcat 7.0搭建，请从tomcat.apache.org上下载，发布包中内置了tomcat 7.0的环境。免安装的服务器端环境目录结构如下：

```
<Tomcat root>
  <webapp>
    <yigo>
      <WEB-INF>
        <classes>
          ...
```

yigo目录为Yigo应用程序目录，配置可运行的应用程序，需要修改classes下的以下文件。

2.1.1:core.properties

```
SolutionPath=
DSN=
DEFAULT=
LOGSVR=
```

core.properties为Yigo Web应用的基础属性定义文件。

描述如下：

- SolutionPath 定义应用的配置路径，指向应用配置的根目录；
- DSN 定义数据源标识，数据源可以指向定义一个数据库连接的数据库，也可以是指向一个数据集群。关于数据源后面会单独介绍；
- DEFAULT 在定义多个数据源的情况下，定义默认的数据源；
- LOGSVR 定义日志服务，当前取值为空或log4j，如果定义为空的时候，日志在控制台中输出，定义为log4j时通过log4j输出。

2.1.2:DSN.properties

DSN.properties为数据源定义文件，其中DSN的取值为数据源的标识，同core.properties中定义的数据源标识相对应。比如core.properties文件是定义了数据源SQL，那么其描述文件为SQL.properties。数据源定义文件分为单一数据源描述和集群数据源描述。

2.1.2.1:单一数据源描述

描述指向一个数据库连接的描述文件，结构如下：

```
DSNTag=
Name=
ConnectionType=
DBType=
Driver=
URL=
```

```
User=
Password=
ExtConfig=
```

属性说明如下:

- DSNTag 为数据源的标记, 目前未使用, 留作未来扩展使用;
- Name 数据源的名称, 同文件名相同;
- ConnectionType 连接类型, 取值为jdbc和dbcp; jdbc为一般的jdbc数据库连接, dbcp为连接池;
- DBType 数据库的类型, 取值为: SqlServer、Oracle、DB2、MySQL、Other, 其中Other表示其它类型数据库, 仅用于测试;
- Driver 数据库驱动, 根据数据库实例而定, 可能的取值及URL的形式如下:
 - com.microsoft.sqlserver.jdbc.SQLServerDriver 为SqlServer数据库驱动, URL的形式为jdbc:sqlserver://host:1433;databaseName=xxx;SelectMethod=cursor, 其中host表示数据库服务器的机器名, 1433为端口号(可能会不同), xxx表示数据库的名称;
 - oracle.jdbc.driver.OracleDriver 为Oracle数据库驱动, URL的形式为jdbc:oracle:thin:@host:1521:xxx, 其中host为数据库服务器的机器名, 1521为端口号(可能会不同), xxx为数据库实例名称;
 - com.mysql.jdbc.Driver 为Mysql数据库驱动, URL的形式为jdbc:mysql://host:3306/xxx?useUnicode=true&characterEncoding=UTF-8, 其中host为数据库服务器的机器名, 3306为端口号(可能会不同), xxx为数据库名称;
 - DB2驱动暂缺;
- URL 为数据库连接路径, 取值形式见Driver部分的说明;
- User 为数据库的用户名;
- Password 为用户指定用户的密码;
- ExtConfig 为dbcp连接时的扩展属性文件名;

ExtConfig.properties文件结构如下, 其中ExtConfig为扩展属性的文件名:

2.1.3: ehcache.xml

ehcache.xml为平台缓存配置文件, 包括服务端session的缓存以及字典的缓存。单机使用及分布式部署配置如下:

2.1.3.1: 单机使用

功能相当于map集合, 主要节点配置如下:

```
<!-- 设置缓存在硬盘上的存储位置 -->
<diskStore path="java.io.tmpdir"/>

<!--当在程序中新建cache时, 需要有此默认缓存参数设置-->
<defaultCache maxElementsInMemory="10000"
    eternal="true"
        timeToIdleSeconds="0"
        timeToLiveSeconds="0"
        overflowToDisk="true"
        diskSpoolBufferSizeMB="30"
        maxElementsOnDisk="10000000"
        diskPersistent="false"
        diskExpiryThreadIntervalSeconds="120"
```

```

        memoryStoreEvictionPolicy="LRU">
        <cacheEventListenerFactory
            class="net.sf.ehcache.distribution.RMICacheReplicatorFactory" />
        <bootstrapCacheLoaderFactory class="net.sf.ehcache.distribution.RMIBootstrapCacheLoaderFactory"/>
    </defaultCache>

<!-- session的缓存及参数设置 -->
<cache
    name="sessionCache"
    eternal="true"
    maxElementsInMemory="10000"
    overflowToDisk="true"
    clearOnFlush = "false">
</cache>

<!-- 手机端session缓存及参数配置 -->
<cache
    name="mobileCache"
    eternal="true"
    maxElementsInMemory="10000"
    overflowToDisk="true"
    clearOnFlush = "false">
</cache>

```

属性说明如下：

- name: Cache的名称, 必须唯一;
- maxElementsInMemory:: 内存中缓存元素的最大个数;
- eternal: 对象是否永久有效;
- timeToIdleSeconds: 设置对象在失效前的允许闲置时间, 只对eternal为false的有效, 默认值为0, 表示一直可以访问;
- timeToLiveSeconds: 设置对象在失效前允许存活时间, 只对eternal为false的有效, 默认值为0, 表示一直可以访问(创建时间为基准);
- overflowToDisk: 当内存中的对象数达到maxElementsInMemory时, 是否将对象写入磁盘;
- diskPersistent: 是否在磁盘上持久化, 指重启JVM后, 数据是否有效, 默认为false;
- maxElementsOnDisk: DiskStore中保持的对象数量, 默认是0, 表示无限制
- diskSpoolBufferSizeMB: 设置磁盘的缓存区大小, 默认是30MB, 每个Cache都有自己的一个缓存区, 不超过2GB;
- diskExpiryThreadIntervalSeconds: 对象检测线程运行的时间间隔, 默认是120秒;
- memoryStoreEvictionPolicy: 当达到maxElementsInMemory限制时, Ehcache将会根据指定的策略去清理内存, 默认策略是LRU(最近最少使用), 可以设置为FIFO(先进先出)或是LFU(较少使用);
- clearOnFlush: 内存数量最大时是否清除;

2.1.3.2: 分布式

ehcache分布式下可以有多种同步方式, 这里采用多播的同步方式。所有的同步在cache的put/remove/update等事件触发后被立即执行。需要配置的节点及属性如下:

```

<!-- 设置缓存在硬盘上的存储位置 -->
<diskStore path="java.io.tmpdir"/>

<!-- 手动成员发现, 只能同步特定机器的特定缓存, 有一定的局限性, 如字典的使用 -->
<!--<cacheManagerPeerProviderFactory
    class="net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory"
    properties="peerDiscovery=manual,
    rmiUrls=//192.168.1.48:40001/sessionCache"/>-->

<!-- 自动成员发现, 集群中的机器往同一地址进行广播, 交换数据 -->

```

```

<cacheManagerPeerProviderFactory
  class="net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory"
  properties="peerDiscovery=automatic,
  multicastGroupAddress=230.0.1.1,
  multicastGroupPort=4446,
  timeToLive=32"/>

<!-- 监听其他集群成员发向当前CacheManager的信息 -->
<cacheManagerPeerListenerFactory
  class="net.sf.ehcache.distribution.RMICacheManagerPeerListenerFactory"
  properties="hostname=192.168.1.30,port=40001,socketTimeoutMillis=2000"
/>
<!-- 每个需要同步的cache需要增加节点如下-->
<defaultCache maxElementsInMemory="10000"
  eternal="true"
    timeToIdleSeconds="0"
    timeToLiveSeconds="0"
    overflowToDisk="true"
    diskSpoolBufferSizeMB="30"
    maxElementsOnDisk="10000000"
    diskPersistent="false"
    diskExpiryThreadIntervalSeconds="120"
    memoryStoreEvictionPolicy="LRU">
  <cacheEventListenerFactory
    class="net.sf.ehcache.distribution.RMICacheReplicatorFactory"
    properties="replicateAsynchronously=true,
    replicatePuts=true,
    replicateUpdates=true,
    replicateUpdatesViaCopy=false,
    replicateRemovals=true"/>
</defaultCache>
<bootstrapCacheLoaderFactory class="net.sf.ehcache.distribution.RMIBootstrapCacheLoaderFactory"/>

```

节点说明如下：

- `cacheManagerPeerProviderFactory` 指定除自身之外的网络群体中其他的提供同步的机器；
`peerDiscovery`：集群中机器之间相互识别的方式，平台采用自动发现方式`automatic`；
`multicastGroupAddress`：为广播地址，所有机器向此地址发送心跳，5s一次。
`multicastGroupPort`：用于广播的端口号；
`timeToLive`：搜索范围，0 是同一台服务器，1 是同一个子网，32 是同一站点，64 指同一块区域，128 是同一块大陆；
- `cacheManagerPeerListenerFactory`：监听其他集群成员发向当前CacheManager的信息，为机器本身配置；
`hostname`：指的是本机。不要使用 `localhost`；
`socketTimeoutMillis`：超时时间默认 2000ms；
- `cacheEventListenerFactory`：用于监听缓存内的数据变化，触发同步；
- `bootstrapCacheLoaderFactory`：用于机器启动时，缓存的预热，从其他机器同步已有数据；
`class`：指定用于启动同步的类对象。

2.1.4:quartz.properties与quartz_jobs.xml

`quartz.properties`定义调度器整体运行的相关参数，`quartz_jobs.xml`定义需要调度的任务及触发器设置。两者搭配使用，构成完整的作业调度机制。

2.1.4.1:quartz.properties

功能相当于map集合，主要节点配置如下：

```

org.quartz.scheduler.instanceName:YigoQuartzScheduler
org.quartz.scheduler.instanceId:AUTO

```

```
org.quartz.threadPool.class = org.quartz.simpl.SimpleThreadPool
org.quartz.threadPool.threadCount = 5
org.quartz.threadPool.threadPriority = 5
org.quartz.jobStore.class=org.quartz.simpl.RAMJobStore

org.quartz.plugin.triggHistory.class = org.quartz.plugins.history.LoggingJobHistoryPlugin
org.quartz.plugin.jobInitializer.class = org.quartz.plugins.xml.XMLSchedulingDataProcessorPlugin
org.quartz.plugin.jobInitializer.fileNames = quartz_jobs.xml
org.quartz.plugin.jobInitializer.failOnFileNotFound = true
org.quartz.plugin.jobInitializer.scanInterval = 10
org.quartz.plugin.jobInitializer.wrapInUserTransaction = false

# close auto update
org.quartz.scheduler.skipUpdateCheck = true
```

属性说明如下：

- org.quartz.scheduler.instanceName: 调度器的名称, 任意字符串表示, 一般选取有意义的表示方法, 如YigoQuartzScheduler;
- org.quartz.scheduler.instanceId: 调度器唯一的ID标示, 可以是任意字符串值, 确保唯一性即可。自动生成使用AUTO;
- org.quartz.threadPool.class: 定义调度器使用的线程池, 一般使用org.quartz.simpl.SimpleThreadPool就可以满足系统要求;
- org.quartz.threadPool.threadCount: 线程池中的线程数量, 根据调度任务的多少加以确定;
- org.quartz.threadPool.threadPriority: 执行线程的优先级, 根据实际需求确定, 默认的是默认的优先级;
- org.quartz.jobStore.class: 将调度器的相关信息全部存储到RAM内存中, 加快执行速度, 但是应用重启后信息会丢失;
- org.quartz.plugin.triggHistory.class: 调度器的插件, 用于记录执行的历史, 使用默认配置即可;
- org.quartz.plugin.jobInitializer.class: 调度器的插件, 用于在调度器初始化时, 扫描quartz_jobs.xml文件, 加入需要调度的任务信息;
- org.quartz.plugin.jobInitializer.fileNames: 定义了任务调度信息的文件。由XMLSchedulingDataProcessorPlugin在启动时扫描;
- org.quartz.plugin.jobInitializer.failOnFileNotFound: 扫描不到quartz_jobs.xml文件立刻抛出异常;
- org.quartz.plugin.jobInitializer.scanInterval: 扫描quartz_jobs.xml的间隔, 是否可以动态修改quartz_jobs.xml文件, 等待以后验证;
- org.quartz.plugin.jobInitializer.wrapInUserTransaction: 具体作用不详, 按照默认配置即可;
- org.quartz.scheduler.skipUpdateCheck : 关闭自动更新, 提高运行效率;

2.1.4.2: quartz_jobs.xml

定义了需要执行的调度任务信息及触发器信息。需要配置的节点及属性如下：

```
<!-- 同一个group中多个job或者group的那么不能相同, 若未设置group则所有未设置group的job为同一个分组 -->
<schedule>
  <job>
    <name>TestQuartz_A</name>
    <group>DEFAULT</group>
    <description>quartz测试A</description>
    <job-class>com.bokesoft.yigo.mid.job.TestScheduleJobA</job-class>
```

```

</job>
<trigger><!--simple类型的Trigger -->
  <simple>
    <name>TestQuartzSimpleTrigger</name>
    <group>DEFAULT</group>
    <job-name>TestQuartz_A</job-name><!-- 把Job和Trigger关联 -->
    <job-group>DEFAULT</job-group>
    <!-- <start-time>2015-12-08T11:35:00</start-time> -->
    <repeat-count>-1</repeat-count><!-- 设置为-1,表示重复无数次 -->
    <repeat-interval>5000</repeat-interval><!-- 单位毫秒 -->
  </simple>
</trigger>
<job>
  <name>TestQuartz_B</name>
  <group>DEFAULT</group>
  <description>quartz测试B</description>
  <job-class>com.bokesoft.yigo.mid.job.TestScheduleJobB</job-class>
</job>
<trigger>
  <cron><!--cron类型的Trigger -->
    <name>TestQuartzCronTrigger</name>
    <group>DEFAULT</group>
    <job-name>TestQuartz_B</job-name><!-- 把Job和Trigger关联 -->
    <job-group>DEFAULT</job-group>
    <cron-expression>0/10 * * * * ?</cron-expression>
  </cron>
</trigger>
</schedule>

cron-expression表达式示例, 不全, 具体使用时请参考quartz配置说明:
" 0 0 12 * * ? " 每天中午12点触发
" 0 15 10 ? * * " 每天上午10:15触发
" 0 15 10 * * ? " 每天上午10:15触发
" 0 15 10 * * ? * " 每天上午10:15触发
" 0 15 10 * * ? 2005 " 2005年的每天上午10:15触发
" 0 * 14 * * ? " 在每天下午2点到下午2:59期间的每1分钟触发
" 0 0/5 14 * * ? " 在每天下午2点到下午2:55期间的每5分钟触发
" 0 0/5 14, 18 * * ? " 在每天下午2点到2:55期间和下午6点到6:55期间的每5分钟触发
" 0 0-5 14 * * ? " 在每天下午2点到下午2:05期间的每1分钟触发
" 0 10, 44 14 ? 3 WED " 每年三月的星期三的下午2:10和2:44触发
" 0 15 10 ? * MON-FRI " 周一至周五的上午10:15触发
...

```

节点说明如下:

- job 调度任务信息节点

name : 调度任务名称, 在同一个分组中不重复;

group: 调度任务的分组;

description: 调度任务的文字描述;

job-class: 调度任务的执行类, 从DefaultScheduleJob继承;

- trigger 调度任务触发器节点, 触发器有simple及cron两种:

simple:

name: 触发器的名称, 在同一分组中不重复

group: 触发器的组别;

job-name: 与该触发器绑定的Job名称;

job-group: 与该触发器绑定的Job组别;

start-time: 首次调度时间, 若没有设置此属性或者start-time设置的时间比当前时间早, 则服务启动后会立即执行一次调度, 若比当前时间晚, 那么要等到设置的时间才会进行第一次调度(慎用)

repeat-count: 调度任务执行几次;

repeat-interval: 调度任务的执行间隔, 单位是毫秒;

cron: (推荐使用, 对执行时间的要求很高的情况下)

name: 触发器的名称, 在同一分组中不重复

group: 触发器的组别;

job-name: 与该触发器绑定的Job名称;

job-group: 与该触发器绑定的Job组别;

cron-expression: 表达式的形式定义调度任务的执行时间

第 3 章 Yigo应用框架

目录

3.1. 工程结构	11
3.2. 典型的应用结构	12

Yigo应用程序按照应用与工程结构组成，每个应用由一个或多个工程组成，其目录结构如下：

```
<Root>
- Solution.xml
- setting.xml 应用设置文件
CommonDef.xml - 应用公共定义
<Resource>
...
<Data> - 应用的数据存储目录(一般为附件)
...
<Project>
...
```

其中<Root>为应用的根目录。Solution.xml为应用的基础属性及应用列表；setting.xml为应用的设置文件。

CommonDef.xml为应用的公共定义，为所有工程所使用，详细内容请参见Yigo语言参考。

<Resource>目录包含了应用的所有资源文件，通常是图片文件；

<Data>目录为应用默认的数据存储目录；

<Project>目录包含一个工程定义。

3.1:工程结构

工程的基本结构如下：

```
<Root>
- Project.xml - 工程描述文件
- setting.xml - 工程设置文件，定义同应用中的同名文件
- CommonDef.xml - 项目公共定义
- Schema.xml - 数据库定义集合
- <Schema> - 数据库定义存储目录
  - Table1.xml - 具体的表的描述文件
- <DataObject> - 数据库对象存储目录
  <Sub1>
    DataObject1.xml - 具体的数据对象的描述文件
    ...
  <Sub2>
    - DataObject2.xml - 具体的数据对象的描述文件
    ...
- <DataMap> - 数据映射存储目录
  <Sub1>
    DataMap1.xml
    ...
- <DataMigration> - 数据迁移存储目录
  <Sub1>
    DataMigration1.xml
    ...
- <Form> - 窗口对象存储目录
  <Sub1>
    Form1.xml - 具体的窗口对象的描述文件
    ...
  <Sub2>
    - Form2.xml - 具体的窗口对象的描述文件
    ...
- <Template> -- 模板存储路径
  ...
- <Report> - 报表的存储目录
  ...
```


- BPM.xml - 流程部署信息和与表单的关联信息
- <BPM> - 流程对象存储目录
 - BPM1.xml - 具体的流程对象的描述文件
- Entry.xml - 功能入口定义描述

<Root>所指向的为工程的根目录，目录中包含以下内容：

- Project.xml，工程文件，定义工程的全局属性；
- setting.xml，工程的设置文件，定义同应用中的同名文件，只是优先级更高；
- CommonDef.xml，工程中的公共设置，定义同应用中的同名文件，只是优先级更高；
- Schema.xml，工程的数据描述文件，目前没有使用，保留；
- <Schema>目录，定义工程的数据描述文件集合，目前没有使用，保留；
- <DataObject>目录，数据对象定义存储目录，可包含子目录；
- <DataMap>目录，数据映射关系存储目录，可包含子目录；
- <DataMigration>目录，迁移关系存储目录，可包含子目录；
- <Form>目录，表单定义存储目录，可包含子目录；
- BPM.xml，工程中流程对象定义集合文件；定义工程中流程部署信息和与表单的关联信息；
- <BPM>目录，流程定义存储目录，可包含子目录；
- Entry.xml，功能入口定义文件；

3.2: 典型的应用结构

一个典型的应用分为公共工程的其它事务工程，公共工程放置应用的公共表单及定义，包括Yigo语言预置的一些定义及应用自身的公共定义。本章的例子中我们以Common命名公共工程，以App命名应用工程。本例中我们通过一个请假单列表和请假单来说明如何开始做一个Yigo应用。

3.2.1: Yigo语言预置的定义

包括一些公共字典和其它一些通用表单，这里仅列出其名称，具体内容请参见FRAME应用中的具体内容。

公共的表单

- Role 角色字典；
- Operator 操作员字典；
- DictEdit 字典编辑表单；
- DictQuery 字典模糊查询表单；
- Dialog 通用对话框；

公共表单的定义见源代码文件，在具体的项目中可以扩充其定义。

3.2.2: 状态定义

在系统中用到的实体表单通常都会定义状态，表单的状态集合以表单内定义、工程定义、应用定义这样的优先顺序来获得，本例中的状态定义集合定义在应用中，如下：

```
<StatusCollection>
```

```
<Status Key="Init" Caption="初始" Value="0"/>
<Status Key="Auditing" Caption="审批中" Value="1" />
<Status Key="Audited" Caption="审批通过" Value="2" />
<Status Key="Denied" Caption="否决" Value="-1" />
</StatusCollection>
```

3.2.3: 具体的表单例子

具体的应用通常情况下由表单列表和表单组成，这里我们以请假单列表和请假单来说明。

3.2.3.1: 请假单

定义一个完整的表单包括几个步骤。

3.2.3.1.1: 数据源

首先定义请假单的数据源，定义一个表单的数据源以两种方法，一种是定义在表单的定义内部，放在表单的<DataSource>标签内部，另外一种是用外部的数据源(定义在工程的数据Object目录下)。在本例子中我们定义在表单的DataSource标签下，如下：

```
<DataSource>
  <DataObject Key="VacationRequest" Caption="请假单" PrimaryTableKey="VacationRequest" PrimaryType="Entity">
    <TableCollection>
      <Table Key="VacationRequest" Caption="基本信息" DBTableName="VacationRequest" Persist="true">
        <Column Key="OID" Caption="对象标识" DBColumnName="OID" DataType="Long" Persist="true"/>
        <Column Key="POID" Caption="父对象标识" DBColumnName="POID" DataType="Long" Persist="true"/>
        <Column Key="SOID" Caption="主对象标识" DBColumnName="SOID" DataType="Long" Persist="true"/>
        <Column Key="VERID" Caption="对象版本" DBColumnName="VERID" DataType="Integer" Persist="true"/>
        <Column Key="DVERID" Caption="对象明细版本" DBColumnName="DVERID" DataType="Integer" Persist="true"/>
        <Column Key="STATUS" Caption="单据状态" DBColumnName="STATUS" DataType="Integer" Persist="true"/>
        <Column Key="IDNumber" Caption="工号" DataType="Varchar" Length="50" Persist="true"/>
        <Column Key="Name" Caption="姓名" DBColumnName="Name" DataType="Varchar" Length="50" Persist="true"/>
        <Column Key="FromTime" Caption="开始时间" DataType="DateTime" Persist="true"/>
        <Column Key="ToTime" Caption="结束时间" DataType="DateTime" Persist="true"/>
        <Column Key="Reason" Caption="事由" DataType="Varchar" Length="250" Persist="true"/>
        <Column Key="Type" Caption="类型" DataType="Integer" Persist="true"/>
        <Column Key="DayCount" Caption="总计" DataType="Integer" Persist="true"/>
      </Table>
    </TableCollection>
  </DataObject>
</DataSource>
```

其中OID、POID、SOID、VERID、DVERID是需要存储的表必须有的字段，在设计器中会默认加进去。

下面解释一下数据源中的一些主要属性，详细的属性列表请见Yigo语言参考。

数据对象属性：

- Key 数据对象的标识，对于需要存储的数据对象，这个标识必须在整个应用全局唯一；
- Caption 数据对象的名称，只是对数据对象进行说明；
- PrimaryType 为数据对象的主类型，取值为Entity和Virtual，Entity表示为实体数据对象，可以持久化；为Virtual时为虚拟数据对象，不支持持久化；
- SecondaryType 为数据对象的辅助类型，在PrimaryType为Entity时用于区分实体数据对象，取值包括Normal(普通实体数据)、Dict(字典数据)、Migration(迁移表)、CompDict(复合字典)、ChainDict(链式字典数据)。默认值为Normal，本例子中没有定义即为Normal。
- PrimaryTableKey 定义实体数据对象的主表，主表用于控制数据的版本，在实体数据对象中必须定义，虚拟数据对象不需要定义。

数据表属性：

- Key 表的标识；
- Caption 表的名称，仅对表的说明；

- DBTableName 对应的数据库表的名称，如果没有定义，则对应的数据库表的名称取的是Key对应的名称；
- Persist 为持久化标志，定义表需要持久化，对于实体数据对象的主表来说，必须持久化；

列属性：

- Key 列的标识；
- Caption 列的名称，仅对列的说明；
- DBColumnName 对应的数据库列的名称，如果没有定义，则对应的数据库列的名称取的是Key对应的名称；
- DataType 列的类型，取值为Long(64位长整型)、Integer(32位整型)、Varchar(变长字符串)、Numeric(数值类型)、DateTime(日期类型)、Binary(二进制)；
- Length 在列的类型为Varchar和Binary时定义长度；
- Precision 列的类型为数值时定义数据的精度；
- Scale 列的类型为数值时定义数据的小数位精度；

3.2.3.1.2: 表单

第二步定义表单，主要是定义界面的结构和同数据对象的绑定关系。这里我们只列出界面定义的源代码，具体的界面定义方法请见界面部分的教程。

表单的属性的源代码如下(节选)：

```
<Form Key="VacationRequest" Caption="请假单" FormType="Entity">
  <DataSource>
    ...
  </DataSource>
  ...
  <Body>
    ...
  </Body>
</Form>
```

表单一般由数据源和表单界面组成，其中DataSource部分定义数据源，在上面已经介绍过，Body为表单的界面组成。表单包含一些基本属性，主要包括：

- Key 表单的标识，在整个应用中必须唯一；
- Caption 表单的名称，一般会显示在打开的工作区的标签栏上；
- FormType 表单的类型，取值为Normal(普通表单)、Entity(实体表单)、Dict(字典表单)、View(视图表单)、Condition(查询条件界面)、Detail(明细表单)。

表单的界面组成源代码如下(节选)：

```
<Body>
  <Block>
    <FlexFlowLayoutPanel Key="main">
      <ToolBar Key="main_toolbar" Height="pref" IsDefault="true"/>
      <GridLayoutPanel Key="content" Height="100%" Padding="5px">
        <RowDefCollection RowGap="5" RowHeight="25">
          <RowDef/>
          <RowDef/>
          <RowDef/>
          <RowDef/>
          <RowDef/>
          <RowDef/>
          <RowDef Height="24px"/>
        </RowDefCollection>
        <ColumnDefCollection>
          <ColumnDef Width="50%">
            <ColumnDef Width="50%">

```

```
<Label Key="L_IDNumber" Caption="工号" X="0" Y="0"/>
<TextEditor Key="IDNumber" Caption="工号" X="1" Y="0">
  <DataBinding TableKey="VacationRequest" ColumnKey="IDNumber"/>
</TextEditor>
<Label Key="L_Name" Caption="姓名" X="0" Y="1"/>
...
</GridLayoutPanel>
</FlexFlowLayoutPanel>
</Block>
</Body>
```

关于界面的结构这里不作说明，这里只解释一下DataBinding，DataBinding为绑定同数据源的绑定关系，其中TableKey为绑定的数据表的标识(数据源表的Key，不是DBTableName)，ColumnKey为绑定的数据列的标识(数据源列的Key，不是DBColumnName)。

运行界面效果如下图所示：

工号	<input type="text"/>
姓名	<input type="text" value="请输入姓名"/>
开始时间	<input type="text"/> 
结束时间	<input type="text"/> 
事由	<input type="text"/>
类型	<div><div></div><div></div></div>
总计	<input type="text"/>

3.2.3.1.3: 界面操作

定义对表单的一些操作，这里我们只增加“编辑”、“取消”和“保存”，源代码如下：

```
<OperationCollection>
  <Operation Key="Save" Caption="保存" Visible="!ReadOnly()">
    <Action>
      <![CDATA[
SaveData();UpdateView();
]]>
    </Action>
  </Operation>
  <Operation Key="Edit" Caption="编辑" Visible="ReadOnly()">
    <Action>
      <![CDATA[
Edit();
]]>
    </Action>
  </Operation>
  <Operation Key="Cancel" Caption="取消" Visible="!ReadOnly()">
    <Action>
      <![CDATA[
Cancel();
]]>
    </Action>
  </Operation>
</OperationCollection>
```

操作的主要属性如下：

- Key 为操作的标识；
- Caption 操作的名称，名称会出现在运行界面生成的工具栏按钮上；
- Visible 可见性。
- 操作的内容，定义在Action节点中，为操作的具体执行内容的表达式。

3.2.3.2: 请假单列表

在应用中对每个实体表单，通常需要一个列表界面展现其数据。因此这里我们定义一个视图表单 (FormType类型为View)。

数据源部分关联请假单的头表，界面定义一个列表用于显示请假单的列表，这里不再一一说明，源代码请见FRAME部分的VacationRequestView.xml，这里只列出运行结果。

请假单列表					
新增					
对象...	工号	姓名	开始时间	结束时间	事由
10001	001	张三	2015-06-19	2015-06-19	处理家务

在请假单列表界面上通过“新增”操作可以打开新增请假单界面，源代码如下：

```
<Operation Key="New" Caption="新增">
  <Action>
    <![CDATA[
New(' VacationRequest')
]]>
  </Action>
</Operation>
```

函数的具体内容请参见Yigo语言参考的函数部分。

通过双击列表中的行可以打开相应的请假单，源代码如下(节选)：

```
<ListView Key="list" TableKey="VacationRequest" Height="100%">
  <RowDbClick>
    <![CDATA[
Open(' VacationRequest', OID)
]]>
  </RowDbClick>
  ...
</ListView>
```

第 4 章 界面模型

目录

4.1. Form结构定义	17
4.2. 基本组件Component定义	17
4.3. 基本布局类组件定义	18
4.4. 功能入口定义	18

4.1:Form结构定义

Yigo平台把交互界面又称为表单(Form)，表单的集合构成了整个用户应用的用户交互界面集合。表单由多个组件组成，分为布局类组件和功能类组件。表单Form下嵌套布局类组件，布局类组件下嵌套功能类组件。

定义：

```
<Form Key="FormKey" Caption="FormCaption" FormulaCaption="" AbbrCaption="" FormulaAbbrCaption="" FormType="FormType"
Platform="" Shell="">
  <DataSource>           //数据对象
    <DataObject>
      ...
    </DataObject>
  </DataSource>
  <Body> //主体内容
    <Block>
      <Panel> //布局组件
        <Component/> //功能组件
        <Component/>
        ...
      </Panel>
    </Block>
  </Body>
</Form>
```

说明：

<DataSource> 定义数据对象，包括数据对象的数据类型、可持久性等等。

<Body>定义主体内容，包括高度、宽度、边距、对齐方式等。

<Panel>定义布局类组件。

<Component>定义功能类组件

4.2:基本组件Component定义

```
<Component Key="" Caption="" BuddyKey="" ForeColor="" BackColor="" Visible="" Enable="" EmbedSub="" EmbedKey="" X="" Y=""
Width="" Height=""
HAlign="" VAlign="" Class="" DisplayType="" Position="" Area="" Tip=""
Padding="" LeftPadding="" TopPadding="" RightPadding="" BottomPadding=""
Margin="" LeftMargin="" TopMargin="" RightMargin="" BottomMargin="" HasBorder="" TabOrder="">
  <DataBinding Required="" TableKey="TableKey" ColumnKey="ColumnKey" ValueChanged="" DefaultValue=""
DefaultFormulaValue="" ValueDependency="" CheckRule="" CheckDependency="" ErrorInfo="">
  </DataBinding>
</Comonent>
```

下面为组件的公共属性定义：

- Width - 宽度，见尺寸定义，取值可以为比例、固定值。
- Height - 高度，见尺寸定义，取值可以为比例、固定值。

- PopWidth - 弹出窗口宽度，取值为固定值；
- PopHeight - 弹出窗口高度，取值为固定值；
- HAlign - 对齐方式，取值为Left, Center和Right。
- OverflowX - 水平超出时的显示属性，取值为Visible, Scroll, Hidden, Auto。
- OverflowY - 垂直超出时的显示属性，取值为Visible, Scroll, Hidden, Auto。
- TopMargin 顶边距；
- BottomMargin 底边距；
- LeftMargin 左边距；
- RightMargin 右边距。

4.3: 基本布局类组件定义

```
<Panel OverflowX="OverflowX" OverflowY="OverflowY" BackImage="BackImage" BackImagePosition="" BackImageRepeat="">
</Panel>
```

下面为面板的公共属性定义：

- OverflowX - 水平超出时的显示属性，取值为Visible, Scroll, Hidden, Auto。
- OverflowY - 垂直超出时的显示属性，取值为Visible, Scroll, Hidden, Auto。
- BackImage - 背景图片；
- BackImagePosition - 背景图片位置，取值为Left, Top, Right, Botom, Center；默认为Center；
- BackImageRepeat - 背景图片是否垂直填充；取值为True和False，默认为False；

4.4: 功能入口定义

功能入口定义由Entry.xml文件定义。

```
<Entry Key="" Caption="" Visible="" Open="">
  <EntryItem Key="" ShortKeys="" Caption="" Type="" FormKey="" Enable="" Visible="" Parameters="">
    <Action>
      <![CDATA[ ]]>
    </Action>
  </EntryItem>
  ...
</Entry>
...
</Entry>
```

示例：

```
<Entry Key="Root" Caption="根" Open="True">
  <Entry Key="Dict" Caption="字典" Open="True" Icon="Dict.png">
    <EntryItem Key="Unit" Caption="计量单位" Type="Form" FormKey="DictEdit" Parameters="ItemKey=Unit"/>
    <EntryItem Key="Material" Caption="物料" Type="Form" FormKey="DictEdit" Parameters="ItemKey=Material"/>
    <EntryItem Key="Warehouse" Caption="仓库" Type="Form" FormKey="DictEdit" Parameters="ItemKey=Warehouse"/>
    <EntryItem Key="Area" Caption="地区" Type="Form" FormKey="DictEdit" Parameters="ItemKey=Area"/>
    <EntryItem Key="Department" Caption="部门" Type="Form" FormKey="DictEdit" Parameters="ItemKey=Department"/>
    <EntryItem Key="Employee" Caption="人员" Type="Form" FormKey="DictEdit" Parameters="ItemKey=Employee"/>
    <EntryItem Key="Dpt-Emp" Caption="部门-人员" Type="Form" FormKey="DictEdit" Parameters="ItemKey=Dpt-Emp"/>
    <EntryItem Key="ProductReport" Caption="产品报表" Type="Form" FormKey="ProductReport"/>
    <EntryItem Key="FruitReport" Caption="水果销量" Type="Form" FormKey="FruitReport"/>
    <EntryItem Key="Operator" Caption="操作员" Type="Form" FormKey="DictEdit" Parameters="ItemKey=Operator"/>
  </Entry>
</Entry>
```

```
<EntryItem Key="Role" Caption="角色" Type="Form" FormKey="DictEdit" Parameters="ItemKey=Role"/>
</Entry>
```

效果显示：



第 5 章 面板组件

目录

- 5.1. 列式布局面板ColumnLayoutPanel 20
- 5.2. 边界布局面板BorderLayoutPanel 22
- 5.3. 流式布局面板FlowLayoutPanel 24
- 5.4. 选项卡面板TabPage 27
- 5.5. 拆分面板SplitPanel 29
- 5.6. 网格布局面板GridLayoutPanel 31
- 5.7. 流式表布局面板FluidTableLayoutPanel 32
- 5.8. 弹性流式布局面板FlexFlowLayoutPanel 34
- 5.9. 面板组合应用实例 36

本章节对面板组件进行逐个介绍。面板组件就是为其它组件进行定位的一类组件，其包含以下几种面板：列式布局面板 (ColumnLayoutPanel)、边界布局面板 (BorderLayoutPanel)、流式布局面板面板 (FlowLayoutPanel)、选项卡面板 (TabPage)、拆分面板 (SplitPanel)、网格布局面板 (GridLayoutPanel)、线性布局面板 (LinearLayoutPanel)、流式表布局面板 (FluidTableLayoutPanel) 以及弹性流式布局面板 (FlexFlowLayoutPanel)。

5.1:列式布局面板ColumnLayoutPanel

列式布局面板是指在行内按列进行布局，其将一行划分为12列，通过定义组件在列位置及列跨度来布局其内部的子组件，

0	1	2	3	4	5	6	7	8	9	10	11
组件1			组件2				组件3				

示例：定义一个“人员档案”的列式布局面板，其中包含4个子组件，子组件为网格布局面板，每个网格布局面板下包含多个功能类组件，具体按照下图进行定义；

子组件1：基本信息，列跨度为3； 子组件2：补充信息，列跨度为2；

子组件3：技能信息，列跨度为1； 子组件4：备注信息，列跨度为7

姓名	<input type="text"/>	教育经历	<input type="text"/>
性别	<input checked="" type="radio"/> 男 <input type="radio"/> 女	工作经历	<input type="text"/>
家庭住址	<input type="text"/>		
电话	<input type="text"/>		
学历	<input type="text"/>		
专业	<input type="text"/>		
备注	<input type="text"/>		

其主要源代码如下：（详细的源代码可以去src \Form目录查看）

```
<ColumnLayoutPanel Key="ColumnLayoutPanel" Caption="列式布局" >
  <GridLayoutPanel Key="Grid1" Caption="基本信息" X="0" Y="0" XSpan="3">
    <RowDefCollection RowHeight="30" RowGap="3">
      <RowDef/>
      <RowDef/>
      <RowDef/>
      <RowDef/>
      <RowDef/>
      <RowDef/>
    </RowDefCollection>
    <ColumnDefCollection ColumnGap="10">
      <ColumnDef Width="50px"/>
      <ColumnDef Width="33%"/>
      <ColumnDef Width="33%"/>
      <ColumnDef Width="33%"/>
    </ColumnDefCollection>
    <Label Key="Label1" Caption="姓名" X="0" Y="0" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <TextEditor Key="Name" Caption="姓名" MaxLength="8" Trim="True" X="1" Y="0" XSpan="2" YSpan="1">
    </TextEditor>
    <Label Key="Label2" Caption="性别" X="0" Y="1" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <RadioButton Key="RadioButton1_1" GroupKey="RadioButton1" IsGroupHead="true" Caption="男" Value="1" X="1" Y="1"
    XSpan="1" YSpan="1"></RadioButton>
    <RadioButton Key="RadioButton1_2" GroupKey="RadioButton1" Caption="女" Value="2" X="2" Y="1" XSpan="1" YSpan="1"></
    RadioButton>
    <Label Key="Label3" Caption="家庭住址" X="0" Y="2" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <TextEditor Key="address" Caption="家庭住址" MaxLength="20" Trim="True" X="1" Y="2" XSpan="2" YSpan="1">
    </TextEditor>
    <Label Key="Label4" Caption="电话" X="0" Y="3" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <NumberEditor Key="phoneNumber" Caption="数值框-长度11位" Precision="11" UseGroupingSeparator="false" Scale="0" X="1"
    Y="3" XSpan="2" YSpan="1" >
    </NumberEditor>
    <Label Key="Label5" Caption="学历" X="0" Y="4" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <ComboBox Key="ComboBox" Caption="学历" Editable="False" X="1" Y="4" XSpan="2" YSpan="1" >
      <Item Key="combox1" Caption="专科" Value="1"/>
      <Item Key="combox2" Caption="本科" Value="2"/>
      <Item Key="combox3" Caption="硕士" Value="3"/>
    </ComboBox>
    <Label Key="Label6" Caption="专业" X="0" Y="5" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <TextEditor Key="Subject" Caption="专业" MaxLength="20" Trim="True" X="1" Y="5" XSpan="2" YSpan="1" >
    </TextEditor>
  </GridLayoutPanel>

  <GridLayoutPanel Key="Grid2" Caption="补充信息" X="3" Y="0" XSpan="2">
    <RowDefCollection RowHeight="30" RowGap="3">
      <RowDef/>
      <RowDef/>
      <RowDef/>
      <RowDef/>
      <RowDef/>
      <RowDef/>
    </RowDefCollection>
    <ColumnDefCollection>
      <ColumnDef Width="100%"/>
    </ColumnDefCollection>
    <Label Key="Label7" Caption="教育经历" X="0" Y="0" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <TextArea Key="Education" Caption="教育经历" MaxLength="50" X="0" Y="1" XSpan="1" YSpan="2" >
    </TextArea>
  </GridLayoutPanel>
</ColumnLayoutPanel>
```

```
<Label Key="Label8" Caption="工作经历" X="0" Y="3" XSpan="1" YSpan="1" HAlign="Left"></Label>
<TextArea Key="Experience" Caption="教育经历" MaxLength="50" X="0" Y="4" XSpan="1" YSpan="2" >
</TextArea>
</GridLayoutPanel>

<GridLayoutPanel Key="Grid3" Caption="技能" X="6" Y="0" XSpan="1">
  <RowDefCollection RowHeight="30" RowGap="3">
    <RowDef/>
    <RowDef/>
    <RowDef/>
    <RowDef/>
    <RowDef/>
  </RowDefCollection>
  <ColumnDefCollection>
    <ColumnDef Width="100%"/>
  </ColumnDefCollection>
  <Label Key="Label9" Caption="技能(语言)" X="0" Y="0" XSpan="1" YSpan="0" HAlign="Left"></Label>
  <CheckBox Key="CheckBox1" Caption="JAVA" Checked="False" X="0" Y="1" XSpan="1" YSpan="0" />
  <CheckBox Key="CheckBox2" Caption="C++" X="0" Y="2" XSpan="1" YSpan="0" />
  <CheckBox Key="CheckBox3" Caption="SQL" X="0" Y="3" XSpan="1" YSpan="0"/>
  <CheckBox Key="CheckBox4" Caption=".Net" X="0" Y="4" XSpan="1" YSpan="0"/>
</GridLayoutPanel>

<GridLayoutPanel Key="Grid4" Caption="备注" X="0" Y="1" XSpan="7">
  <RowDefCollection RowHeight="30" RowGap="3">
    <RowDef/>
    <RowDef/>
    <RowDef/>
  </RowDefCollection>
  <ColumnDefCollection>
    <ColumnDef Width="100%"/>
  </ColumnDefCollection>
  <Label Key="Label10" Caption="备注" X="0" Y="0" XSpan="1" YSpan="1" HAlign="Left"></Label>
  <TextArea Key="Comments" Caption="备注" MaxLength="50" X="0" Y="1" XSpan="1" YSpan="2" >
  </TextArea>
</GridLayoutPanel>
</ColumnLayoutPanel>
```

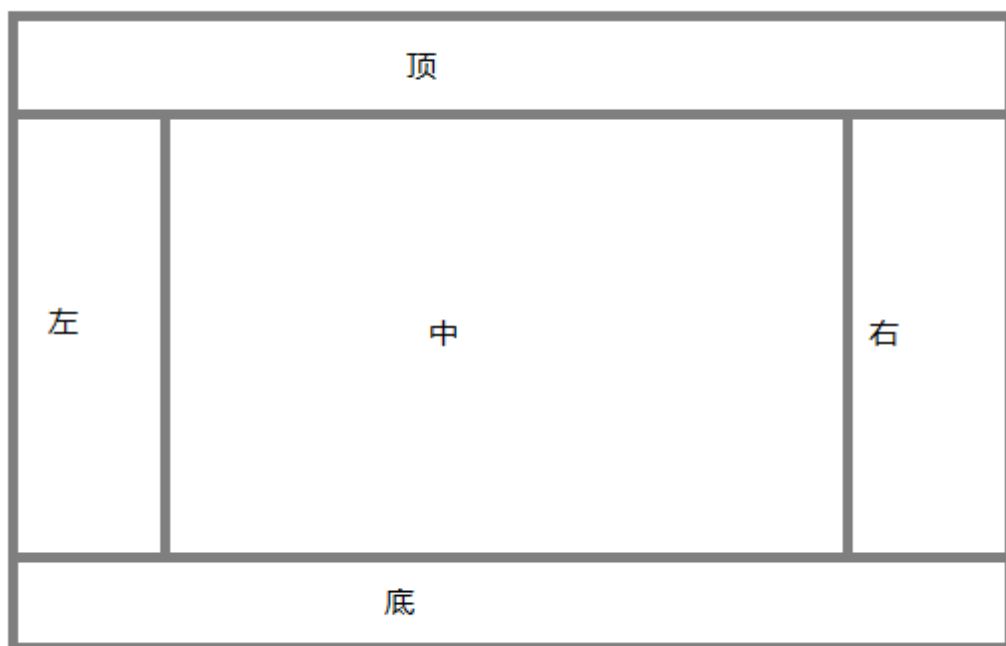


注意

- 列式布局面板没有自己的属性，其计算主要依赖于子组件中的跨度定义（X行位置，Y列位置，XSpan行跨度）；并且组件不能跨行定义。
- 子组件通常会是一个布局面板，而不会是一个单独的功能类组件。

5.2: 边界布局面板BorderLayoutPanel

边界布局面板将整个区域划分为左、右、顶、底、中五个区域，子组件在这5个区域中进行布局。



示例：定义一个“请假查询”的边界布局面板，面板的左区域进行按字段查询，面板的中间显示请假信息。顶部是一个工具条可以进行新建、修改、删除操作。

其源代码如下：

```
<BorderLayoutPanel Key="BorderLayoutPanel" Caption="边界布局">
  <ToolBar Key="main_toolbar" Height="pref" IsDefault="True" Area="Top" />
  <FlexFlowLayoutPanel Key="Find" Caption="查询" Area="Left">
    <Label Key="Label1" Caption="查询记录" HAlign="Left"></Label>
    <Label Key="Label2" Caption="按人员查询:" HAlign="Left" ></Label>
    <TextEditor Key="Find_Name" Caption="姓名" MaxLength="10" Trim="True">
    </TextEditor>
    <Button Key="button1" Caption="查询" >
      <OnClick>
        <![CDATA[
          DBNamedQueryValue('','');
        ]]>
      </OnClick>
```

```

</Button>
<Label Key="Label3" Caption="按工号查询: " HAlign="Left" ></Label>
<TextEditor Key="Find_id" Caption="工号" MaxLength="10" Trim="True" >
</TextEditor>
<Button Key="button2" Caption="查询" >
    <OnClick>
        <![CDATA[
            DBNamedQueryValue(' ',);
        ]]>
    </OnClick>
</Button>
</FlexFlowLayoutPanel>

<FlexFlowLayoutPanel Key="Leave" Caption="请假信息" Area="Center">
    <ListView Key="list" TableKey="VacationInfo" Height="100%">
        <RowDbClick>
            <![CDATA[
                Open(' ', )
            ]]>
        </RowDbClick>
        <ListViewColumnCollection>
            <ListViewColumn Key="OID" Width="100px" Caption="记录号" ColumnType="Label" DataColumnKey="OID">
            </ListViewColumn>
            <ListViewColumn Key="IDNumber" Width="100px" Caption="工号" ColumnType="Label" DataColumnKey="IDNumber">
            </ListViewColumn>
            <ListViewColumn Key="Name" Width="100px" Caption="姓名" ColumnType="Label" DataColumnKey="Name">
            </ListViewColumn>
            <ListViewColumn Key="FromTime" Width="100px" Caption="开始时间" ColumnType="Label" DataColumnKey="FromTime">
            </ListViewColumn>
            <ListViewColumn Key="ToTime" Width="100px" Caption="结束时间" ColumnType="Label" DataColumnKey="ToTime">
            </ListViewColumn>
            <ListViewColumn Key="Reason" Width="100px" Caption="事由" ColumnType="Label" DataColumnKey="Reason">
            </ListViewColumn>
        </ListViewColumnCollection>
    </ListView>
</FlexFlowLayoutPanel>
</BorderLayoutPanel>

```



注意

边界布局依靠Area属性来定位子组件位置，取值为Left、Right、Top、Bottom和Center。区域的大小依赖组件自身的大小。

5.3: 流式布局面板FlowLayoutPanel

流布局面板按照从上到下的顺序对组件进行定位，子组件的宽度为面板的宽度，高度为子组件的内容高度或定义的高度；



示例：定义一个“挂号信息”的流布局面板，包含2个网格布局面板：病人信息和挂号信息两部分。如下图所示：

病人信息					
门诊号	<input type="text"/>	姓名	<input type="text"/>	性别	<input type="text"/>
年龄	<input type="text"/>	地址	<input type="text"/>		
挂号信息					
性质	<input type="text"/>	门诊科室	<input type="text"/>	挂号类型	<input type="text"/>
医生	<input type="text"/>	挂号费	<input type="text"/>	诊疗费	<input type="text"/>
专家费	<input type="text"/>	挂号编号	<input type="text"/>	减免费	<input type="text"/>
病历费	<input type="text"/>				

主要代码如下：（详细的源代码可以去src\Form目录查看）

```
<FlowLayoutPanel Key="FlowLayoutPanel" Caption="流式布局" >
  <Label Key="Label1" Caption="病人信息" />
  <GridLayoutPanel Key="Grid1" Caption="病人信息" >
    <RowDefCollection RowHeight="30" RowGap="3">
      <RowDef/>
      <RowDef/>
    </RowDefCollection>
    <ColumnDefCollection ColumnGap="10" >
      <ColumnDef Width="50px"/>
      <ColumnDef Width="150px"/>
      <ColumnDef Width="50px"/>
      <ColumnDef Width="150px"/>
      <ColumnDef Width="50px"/>
      <ColumnDef Width="150px"/>
    </ColumnDefCollection>
    <Label Key="Label2" Caption="门诊号" X="0" Y="0" XSpan="1" YSpan="1" ></Label>
    <NumberEditor Key="id" Caption="门诊号" Precision="8" UseGroupingSeparator="false" Scale="0" X="1" Y="0" XSpan="1"
YSpan="1" >
    </NumberEditor>
  </GridLayoutPanel>
</FlowLayoutPanel>
```

```

<Label Key="Label3" Caption="姓名" X="2" Y="0" XSpan="1" YSpan="1" ></Label>
<TextEditor Key="Name" Caption="姓名" MaxLength="8" Trim="True" X="3" Y="0" XSpan="1" YSpan="1"/>

<Label Key="Label4" Caption="性别" X="4" Y="0" XSpan="1" YSpan="1" ></Label>
<TextEditor Key="sexual" Caption="性别" MaxLength="4" Trim="True" X="5" Y="0" XSpan="1" YSpan="1" />

<Label Key="Label5" Caption="年龄" X="0" Y="1" XSpan="1" YSpan="1"></Label>
<NumberEditor Key="Age" Caption="年龄" Precision="3" UseGroupingSeparator="false" Scale="0" X="1" Y="1" XSpan="1"
YSpan="1" >
</NumberEditor>
<Label Key="Label6" Caption="地址" X="2" Y="1" XSpan="1" YSpan="1"></Label>
<TextEditor Key="Address" Caption="地址" MaxLength="20" Trim="True" X="3" Y="1" XSpan="3" YSpan="1"/>
</GridLayoutPanel>

<Separator>
</Separator>

<Label Key="Label12" Caption="挂号信息" ></Label>
<GridLayoutPanel Key="Grid2" Caption="挂号信息" >
  <RowDefCollection RowHeight="30" RowGap="3">
    <RowDef/>
    <RowDef/>
    <RowDef/>
    <RowDef/>
  </RowDefCollection>
  <ColumnDefCollection ColumnGap="10" >
    <ColumnDef Width="50px"/>
    <ColumnDef Width="150px"/>
    <ColumnDef Width="50px"/>
    <ColumnDef Width="150px"/>
    <ColumnDef Width="50px"/>
    <ColumnDef Width="150px"/>
  </ColumnDefCollection>
  <Label Key="Label2" Caption="性质" X="0" Y="0" XSpan="1" YSpan="1" ></Label>
  <ComboBox Key="DropDownButton1" Caption="性质" X="1" Y="0" XSpan="1" YSpan="1" >
    <Item Key="item1" Caption="自费" Value="0"/>
    <Item Key="item2" Caption="公费" Value="1"/>
  </ComboBox>
  <Label Key="Label3" Caption="门诊科室" X="2" Y="0" XSpan="1" YSpan="1" ></Label>
  <ComboBox Key="DropDownButton2" Caption="门诊科室" X="3" Y="0" XSpan="1" YSpan="1" >
    <Item Key="item1" Caption="内科" Value="0"/>
    <Item Key="item2" Caption="儿科" Value="1"/>
    <Item Key="item3" Caption="骨科" Value="2"/>
  </ComboBox>
  <Label Key="Label4" Caption="挂号类型" X="4" Y="0" XSpan="1" YSpan="1" ></Label>
  <ComboBox Key="DropDownButton3" Caption="挂号类型" X="5" Y="0" XSpan="1" YSpan="1">
    <Item Key="item1" Caption="普通门诊" Value="0"/>
    <Item Key="item2" Caption="专家门诊" Value="1"/>
    <Item Key="item3" Caption="特需门诊" Value="2"/>
  </ComboBox>
  <Label Key="Label5" Caption="医生" X="0" Y="1" XSpan="1" YSpan="1"></Label>
  <ComboBox Key="DropDownButton4" Caption="医生" X="1" Y="1" XSpan="1" YSpan="1">
    <Item Key="item1" Caption="张医生" Value="0"/>
    <Item Key="item2" Caption="李医生" Value="1"/>
    <Item Key="item3" Caption="王医生" Value="2"/>
  </ComboBox>
  <Label Key="Label6" Caption="挂号费" X="2" Y="1" XSpan="1" YSpan="1"></Label>
  <NumberEditor Key="cost1" Caption="挂号费" Precision="8" UseGroupingSeparator="false" Scale="2" X="3" Y="1"
XSpan="1" YSpan="1" >
</NumberEditor>
  <Label Key="Label7" Caption="诊疗费" X="4" Y="1" XSpan="1" YSpan="1"></Label>
  <NumberEditor Key="cost2" Caption="诊疗费" Precision="8" UseGroupingSeparator="false" Scale="2" X="5" Y="1"
XSpan="1" YSpan="1" >
</NumberEditor>
  <Label Key="Label8" Caption="专家费" X="0" Y="2" XSpan="1" YSpan="1"></Label>
  <NumberEditor Key="cost3" Caption="专家费" Precision="8" UseGroupingSeparator="false" Scale="2" X="1" Y="2"
XSpan="1" YSpan="1" >
</NumberEditor>
  <Label Key="Label9" Caption="挂号编号" X="2" Y="2" XSpan="1" YSpan="1"></Label>
  <NumberEditor Key="cost4" Caption="挂号编号" Precision="8" UseGroupingSeparator="false" Scale="2" X="3" Y="2"
XSpan="1" YSpan="1" >
</NumberEditor>
  <Label Key="Label10" Caption="减免费" X="4" Y="2" XSpan="1" YSpan="1"></Label>
  <NumberEditor Key="cost5" Caption="减免费" Precision="8" UseGroupingSeparator="false" Scale="2" X="5" Y="2"
XSpan="1" YSpan="1" >
</NumberEditor>
  <Label Key="Label11" Caption="病历费" X="0" Y="3" XSpan="1" YSpan="1"></Label>
  <NumberEditor Key="cost6" Caption="病历费" Precision="8" UseGroupingSeparator="false" Scale="2" X="1" Y="3"
XSpan="1" YSpan="1" >
</NumberEditor>
</GridLayoutPanel>
</FlowLayoutPanel>

```

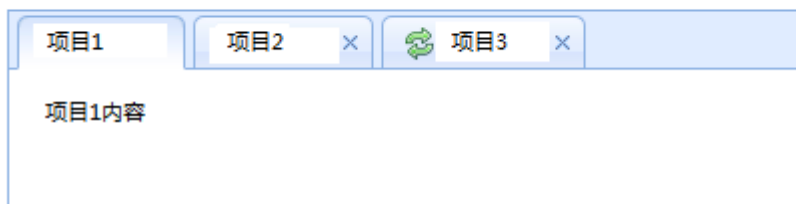


注意

面板按照从上到下的顺序对子组件进行定位，Label1 ->Grid1->分割线->Label12->Grid2
子组件的宽度为面板的宽度，高度为子组件的内容高度或定义的高度；

5.4: 选项卡面板TabPaneI

选项卡面板按卡片式对子组件进行布局，如下图所示：



示例：将上个示例的“挂号信息”定义为选项卡面板，则代码定义如下：

```
<TabPanel >
  <GridLayoutPanel Key="Grid1" Caption="病人信息" >
    <RowDefCollection RowHeight="30" RowGap="3">
      <RowDef/>
      <RowDef/>
    </RowDefCollection>
    <ColumnDefCollection ColumnGap="10" >
      <ColumnDef Width="50px"/>
      <ColumnDef Width="150px"/>
      <ColumnDef Width="50px"/>
      <ColumnDef Width="150px"/>
      <ColumnDef Width="50px"/>
      <ColumnDef Width="150px"/>
    </ColumnDefCollection>
    <Label Key="Label2" Caption="门诊号" X="0" Y="0" XSpan="1" YSpan="1" ></Label>
    <NumberEditor Key="id" Caption="门诊号" Precision="8" UseGroupingSeparator="false" Scale="0" X="1" Y="0" XSpan="1"
YSpan="1" >
      </NumberEditor>
    <Label Key="Label3" Caption="姓名" X="2" Y="0" XSpan="1" YSpan="1" ></Label>
    <TextEditor Key="Name" Caption="姓名" MaxLength="8" Trim="True" X="3" Y="0" XSpan="1" YSpan="1"/>

    <Label Key="Label4" Caption="性别" X="4" Y="0" XSpan="1" YSpan="1" ></Label>
    <TextEditor Key="sexual" Caption="性别" MaxLength="4" Trim="True" X="5" Y="0" XSpan="1" YSpan="1" />

    <Label Key="Label5" Caption="年龄" X="0" Y="1" XSpan="1" YSpan="1"></Label>
    <NumberEditor Key="Age" Caption="年龄" Precision="3" UseGroupingSeparator="false" Scale="0" X="1" Y="1" XSpan="1"
YSpan="1" >
      </NumberEditor>
    <Label Key="Label6" Caption="地址" X="2" Y="1" XSpan="1" YSpan="1"></Label>
    <TextEditor Key="Address" Caption="地址" MaxLength="20" Trim="True" X="3" Y="1" XSpan="3" YSpan="1"/>
    </GridLayoutPanel>

    <GridLayoutPanel Key="Grid2" Caption="挂号信息" >
      <RowDefCollection RowHeight="30" RowGap="3">
        <RowDef/>
        <RowDef/>
        <RowDef/>
        <RowDef/>
      </RowDefCollection>
      <ColumnDefCollection ColumnGap="10" >
        <ColumnDef Width="50px"/>
        <ColumnDef Width="150px"/>
        <ColumnDef Width="50px"/>
      </ColumnDefCollection>
    </GridLayoutPanel>
  </TabPanel>
```



```

        <ColumnDef Width="150px"/>
        <ColumnDef Width="50px"/>
        <ColumnDef Width="150px"/>
    </ColumnDefCollection>
    <Label Key="Label2" Caption="性质" X="0" Y="0" XSpan="1" YSpan="1" ></Label>
    <ComboBox Key="DropDownButton1" Caption="性质" X="1" Y="0" XSpan="1" YSpan="1" >
        <Item Key="item1" Caption="自费" Value="0"/>
        <Item Key="item2" Caption="公费" Value="1"/>
    </ComboBox>
    <Label Key="Label3" Caption="门诊科室" X="2" Y="0" XSpan="1" YSpan="1" ></Label>
    <ComboBox Key="DropDownButton2" Caption="门诊科室" X="3" Y="0" XSpan="1" YSpan="1" >
        <Item Key="item1" Caption="内科" Value="0"/>
        <Item Key="item2" Caption="儿科" Value="1"/>
        <Item Key="item3" Caption="骨科" Value="2"/>
    </ComboBox>
    <Label Key="Label4" Caption="挂号类型" X="4" Y="0" XSpan="1" YSpan="1" ></Label>
    <ComboBox Key="DropDownButton3" Caption="挂号类型" X="5" Y="0" XSpan="1" YSpan="1">
        <Item Key="item1" Caption="普通门诊" Value="0"/>
        <Item Key="item2" Caption="专家门诊" Value="1"/>
        <Item Key="item3" Caption="特需门诊" Value="2"/>
    </ComboBox>
    <Label Key="Label5" Caption="医生" X="0" Y="1" XSpan="1" YSpan="1"></Label>
    <ComboBox Key="DropDownButton4" Caption="医生" X="1" Y="1" XSpan="1" YSpan="1">
        <Item Key="item1" Caption="张医生" Value="0"/>
        <Item Key="item2" Caption="李医生" Value="1"/>
        <Item Key="item3" Caption="王医生" Value="2"/>
    </ComboBox>
    <Label Key="Label6" Caption="挂号费" X="2" Y="1" XSpan="1" YSpan="1"></Label>
    <NumberEditor Key="cost1" Caption="挂号费" Precision="8" UseGroupingSeparator="false" Scale="2" X="3" Y="1"
XSpan="1" YSpan="1" >
    </NumberEditor>
    <Label Key="Label7" Caption="诊疗费" X="4" Y="1" XSpan="1" YSpan="1"></Label>
    <NumberEditor Key="cost2" Caption="诊疗费" Precision="8" UseGroupingSeparator="false" Scale="2" X="5" Y="1"
XSpan="1" YSpan="1" >
    </NumberEditor>
    <Label Key="Label8" Caption="专家费" X="0" Y="2" XSpan="1" YSpan="1"></Label>
    <NumberEditor Key="cost3" Caption="专家费" Precision="8" UseGroupingSeparator="false" Scale="2" X="1" Y="2"
XSpan="1" YSpan="1" >
    </NumberEditor>
    <Label Key="Label9" Caption="挂号编号" X="2" Y="2" XSpan="1" YSpan="1"></Label>
    <NumberEditor Key="cost4" Caption="挂号编号" Precision="8" UseGroupingSeparator="false" Scale="2" X="3" Y="2"
XSpan="1" YSpan="1" >
    </NumberEditor>
    <Label Key="Label10" Caption="减免费" X="4" Y="2" XSpan="1" YSpan="1"></Label>
    <NumberEditor Key="cost5" Caption="减免费" Precision="8" UseGroupingSeparator="false" Scale="2" X="5" Y="2"
XSpan="1" YSpan="1" >
    </NumberEditor>
    <Label Key="Label11" Caption="病历费" X="0" Y="3" XSpan="1" YSpan="1"></Label>
    <NumberEditor Key="cost6" Caption="病历费" Precision="8" UseGroupingSeparator="false" Scale="2" X="1" Y="3"
XSpan="1" YSpan="1" >
    </NumberEditor>
    </GridLayoutPanel>
</TabPanel>

```

效果如下图，图1为“病人信息”Tab选项卡，图2为“挂号信息”Tab选项卡。

病人信息		挂号信息	
门诊号	<input type="text"/>	姓名	<input type="text"/>
年龄	<input type="text"/>	性别	<input type="text"/>
		地址	<input type="text"/>

(图

1)

病人信息		挂号信息	
性质	<input type="text"/>	门诊科室	<input type="text"/>
医生	<input type="text"/>	挂号费	<input type="text"/>
专家费	<input type="text"/>	挂号编号	<input type="text"/>
病历费	<input type="text"/>	减免费	<input type="text"/>

(图2)

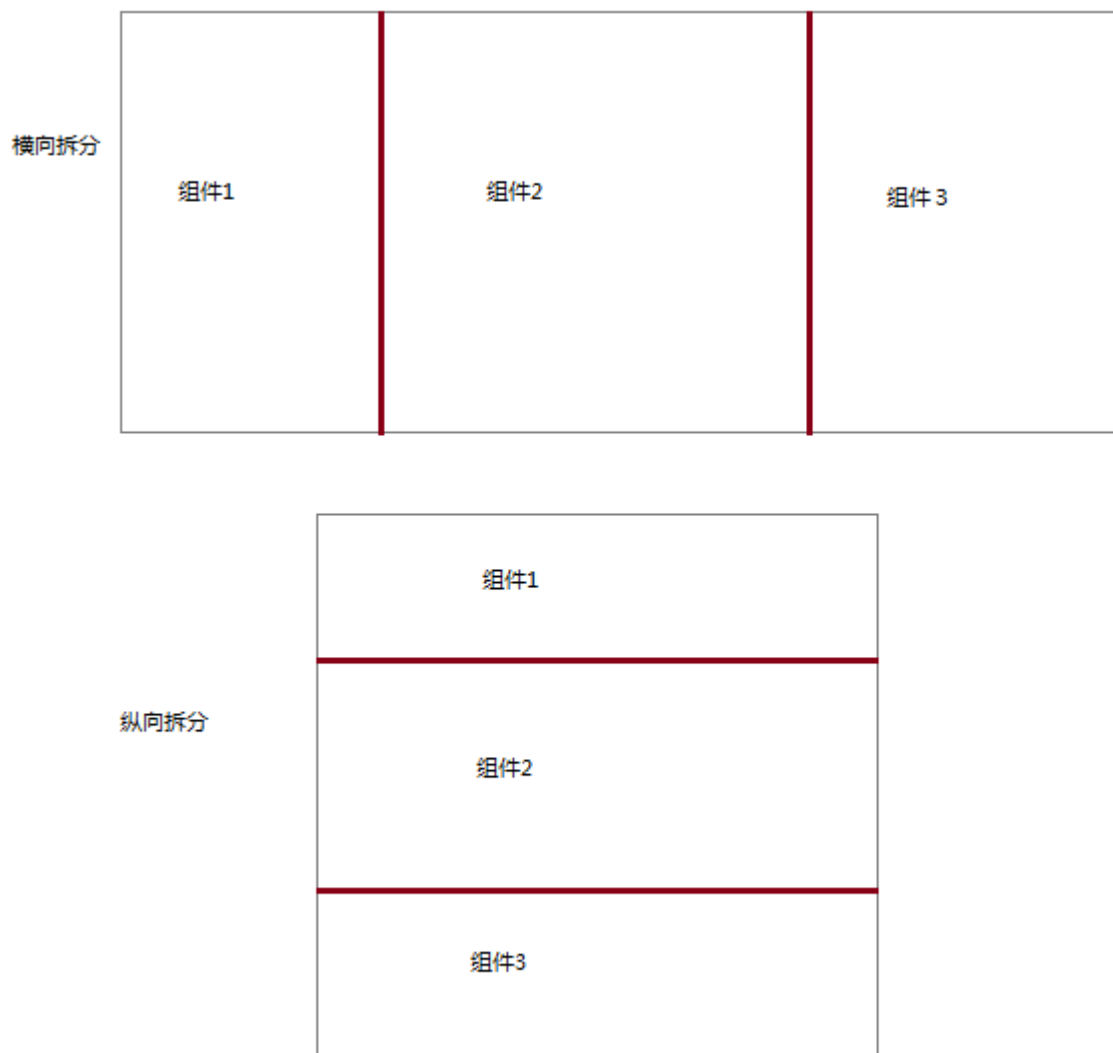


注意

每个选项卡面板相当于一个单独的子表单，它可以且必须嵌套其他形式的布局类组件。

5.5: 拆分面板SplitPanel

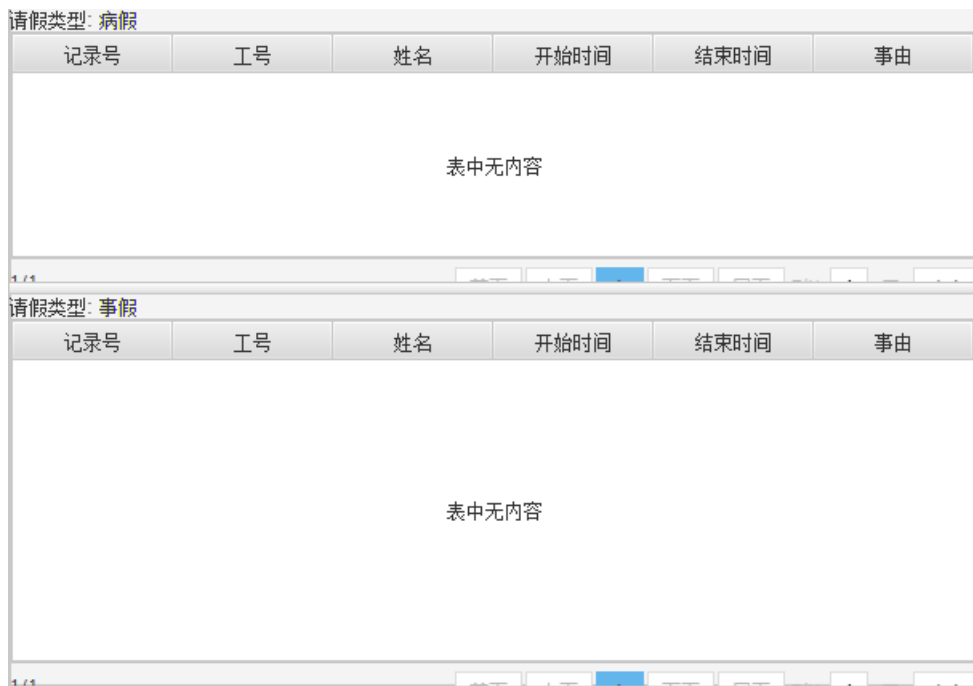
拆分面板将区域按横向或纵向拆分为n个区域，每个区域可以单独定义大小，两个区域之间有拆分栏可以用来改变区域的大小，如下图所示：



示例：定义一个“请假信息表”的纵向拆分面板，拆分为2个区域。第一个区域是病假信息汇总，定义高度为40%；第二个区域是事假信息汇总，定义高度为60%。具体代码定义如下：

```
<SplitPanel Key="Split" Orientation="Vertical">
  <SplitSize Size="40%" >/SplitSize>
  <SplitSize Size="60%" >/SplitSize>
  <FlexFlowLayoutPanel Key="Leave" Caption="请假信息" >
    <Label Key="Label1" Caption="请假类型: 病假" HAlign="Left">/Label>
    <ListView Key="list" TableKey="VacationInfo" Height="100%">
      <ListViewColumnCollection>
        <ListViewColumn Key="OID" Width="100px" Caption="记录号" ColumnType="Label" DataColumnKey="OID">
        </ListViewColumn>
        <ListViewColumn Key="IDNumber" Width="100px" Caption="工号" ColumnType="Label" DataColumnKey="IDNumber">
        </ListViewColumn>
        <ListViewColumn Key="Name" Width="100px" Caption="姓名" ColumnType="Label" DataColumnKey="Name">
        </ListViewColumn>
        <ListViewColumn Key="FromTime" Width="100px" Caption="开始时间" ColumnType="Label" DataColumnKey="FromTime">
        </ListViewColumn>
        <ListViewColumn Key="ToTime" Width="100px" Caption="结束时间" ColumnType="Label" DataColumnKey="ToTime">
        </ListViewColumn>
        <ListViewColumn Key="Reason" Width="100px" Caption="事由" ColumnType="Label" DataColumnKey="Reason">
        </ListViewColumn>
      </ListViewColumnCollection>
    </ListView>
  </FlexFlowLayoutPanel>
  <FlexFlowLayoutPanel Key="Leave2" Caption="请假信息" >
    <Label Key="Label2" Caption="请假类型: 事假" HAlign="Left">/Label>
    <ListView Key="list" TableKey="VacationInfo" Height="100%">
      <ListViewColumnCollection>
        <ListViewColumn Key="OID" Width="100px" Caption="记录号" ColumnType="Label" DataColumnKey="OID">
        </ListViewColumn>
        <ListViewColumn Key="IDNumber" Width="100px" Caption="工号" ColumnType="Label" DataColumnKey="IDNumber">
        </ListViewColumn>
        <ListViewColumn Key="Name" Width="100px" Caption="姓名" ColumnType="Label" DataColumnKey="Name">
        </ListViewColumn>
        <ListViewColumn Key="FromTime" Width="100px" Caption="开始时间" ColumnType="Label" DataColumnKey="FromTime">
        </ListViewColumn>
        <ListViewColumn Key="ToTime" Width="100px" Caption="结束时间" ColumnType="Label" DataColumnKey="ToTime">
        </ListViewColumn>
        <ListViewColumn Key="Reason" Width="100px" Caption="事由" ColumnType="Label" DataColumnKey="Reason">
        </ListViewColumn>
      </ListViewColumnCollection>
    </ListView>
  </FlexFlowLayoutPanel>
</SplitPanel>
```

效果实现



注意

- Orientation – 表示拆分方向，取值为Horizontal横向或Vertical纵向。默认值为Horizontal。

- SplitSize中的Size - 表示子组件的大小，取值为固定值或百分比，比如固定值200px，比例值100%。
- 每个拆分面板可以嵌套其他布局面板

5.6: 网格布局面板GridLayoutPanel

网格布局面板将整个区域划分多行多列，定义行高和列宽，子组件按照行、列位置和行、列跨度进行布局。

它和列式布局最大的不同在于，网格布局面板可以跨行定义，而列式布局面板不能跨行定义。

如下图所示：

	0	1	2	3
0	组件1	组件2		
1				
2		组件3		
3				
4				
5				
6				

示例：定义一个“求职信息表”的网格布局面板，包含一些相关的功能类组件，如下图所示：

求职信息表

姓名

性别

家庭住址

电话

学历

专业

备注

教育经历

工作经历

```
<GridLayoutPanel Key="Grid" Caption="求职信息">
  <RowDefCollection RowHeight="30" RowGap="3">
    <RowDef/>
    <RowDef/>
    <RowDef/>
    <RowDef/>
    <RowDef/>
    <RowDef/>
    <RowDef/>
    <RowDef/>
  </RowDefCollection>
  <ColumnDefCollection ColumnGap="10">
    <ColumnDef Width="50px"/>
    <ColumnDef Width="14%"/>
    <ColumnDef Width="14%"/>
    <ColumnDef Width="14%"/>
    <ColumnDef Width="14%"/>
  </ColumnDefCollection>
</GridLayoutPanel>
```

```

        <ColumnDef Width="14%" />
        <ColumnDef Width="14%" />
        <ColumnDef Width="14%" />
    </ColumnDefCollection>
    <Label Key="Label1" Caption="姓名" X="0" Y="0" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <TextEditor Key="Name" Caption="姓名" MaxLength="8" Trim="True" X="1" Y="0" XSpan="2" YSpan="1">
    </TextEditor>
    <Label Key="Label2" Caption="性别" X="0" Y="1" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <RadioButton Key="RadioButton1" GroupKey="RadioButton1" IsGroupHead="true" Caption="男" Value="1" X="1" Y="1"
XSpan="1" YSpan="1"></RadioButton>
    <RadioButton Key="RadioButton1_2" GroupKey="RadioButton1" Caption="女" Value="2" X="2" Y="1" XSpan="1" YSpan="1"></
RadioButton>
    <Label Key="Label3" Caption="家庭住址" X="0" Y="2" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <TextEditor Key="address" Caption="家庭住址" MaxLength="20" Trim="True" X="1" Y="2" XSpan="2" YSpan="1">
    </TextEditor>
    <Label Key="Label4" Caption="电话" X="0" Y="3" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <NumberEditor Key="phonenumber" Caption="数值框-长度11位" Precision="11" UseGroupingSeparator="false" Scale="0" X="1"
Y="3" XSpan="2" YSpan="1">
    </NumberEditor>
    <Label Key="Label5" Caption="学历" X="0" Y="4" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <ComboBox Key="ComboBox" Caption="学历" Editable="False" X="1" Y="4" XSpan="2" YSpan="1">
        <Item Key="combox1" Caption="专科" Value="1"/>
        <Item Key="combox2" Caption="本科" Value="2"/>
        <Item Key="combox3" Caption="硕士" Value="3"/>
    </ComboBox>
    <Label Key="Label6" Caption="专业" X="0" Y="5" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <TextEditor Key="Subject" Caption="专业" MaxLength="20" Trim="True" X="1" Y="5" XSpan="2" YSpan="1">
    </TextEditor>

    <Label Key="Label7" Caption="教育经历" X="3" Y="0" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <TextArea Key="Education" Caption="教育经历" MaxLength="50" X="3" Y="1" XSpan="3" YSpan="2">
    </TextArea>
    <Label Key="Label8" Caption="工作经历" X="3" Y="3" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <TextArea Key="Experience" Caption="教育经历" MaxLength="50" X="3" Y="4" XSpan="3" YSpan="2">
    </TextArea>

    <Label Key="Label9" Caption="技能(语言)" X="6" Y="0" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <CheckBox Key="CheckBox1" Caption="JAVA" Checked="False" X="6" Y="1" XSpan="1" YSpan="1" />
    <CheckBox Key="CheckBox2" Caption="C++" X="6" Y="2" XSpan="1" YSpan="1" />
    <CheckBox Key="CheckBox3" Caption="SQL" X="6" Y="3" XSpan="1" YSpan="1" />
    <CheckBox Key="CheckBox4" Caption=".Net" X="6" Y="4" XSpan="1" YSpan="1" />

    <Label Key="Label10" Caption="备注" X="0" Y="6" XSpan="1" YSpan="1" HAlign="Left"></Label>
    <TextArea Key="Comments" Caption="备注" MaxLength="50" X="1" Y="6" XSpan="5" YSpan="2">
    </TextArea>
</GridLayoutPanel>

```



注意

此示例实现的效果和列式布局面板实现的最终效果是一样的，但是列式布局面板不能够跨行定义，要通过嵌套网格布局来实现多行的效果。而网格布局面板则不需要通过嵌套其他布局就可以轻松实现跨行定义。

5.7: 流式表布局面板FluidTableLayoutPanel

流式表布局面板，以表格形式来定义面板的区域，流式表布局面板定义列的数量及列宽，但不定义行的数量，行的数量根据列向的重复数和包含的组件数量来计算。流式的意义表示布局的列向的重复数量根据不同的定义而变化；如下图所示：

列向重复数为1时

列间距

行间距

标题1	组件1
标题2	组件2
标题3	组件3
标题4	组件4
标题5	组件5
标题6	组件6
标题7	组件7
标题8	组件8

列向重复数为2时

列向间距

列间距

标题1	组件1	标题2	组件2
标题3	组件3	标题4	组件4
标题5	组件5	标题6	组件6
标题7	组件7	标题8	组件8

示例：定义一个“公司客户联系表”的流式布局面板，包含以下20个功能组件，列向重复数为6：

公司客户联系表

公司简称	<input type="text"/>	公司全称	<input type="text"/>
公司电话1	<input type="text"/>	公司电话2	<input type="text"/>
公司地址	<input type="text"/>	业务关系	<input type="text"/>
备注	<input type="text"/>		

源代码如下：

```
<FluidTableLayoutPanel Key="ContactList" Height="100%" RepeatCount="6" RepeatGap="10" RowGap="2" ColumnGap="2" RowHeight="30"
>
  <TableColumnCollection >
    <TableColumn Width="100px"/>
    <TableColumn Width="100%"/>
  </TableColumnCollection>
  <Label Key="Label1" Caption="公司简称" ></Label>
  <TextEditor Key="Name1" Caption="公司简称" MaxLength="10" />
  <Label Key="Label2" Caption="公司全称" ></Label>
  <TextEditor Key="Name2" Caption="公司全称" MaxLength="30" />
  <Label Key="Label3" Caption="公司英文" ></Label>
  <TextEditor Key="Name3" Caption="公司英文" MaxLength="30" />
  <Label Key="Label4" Caption="公司电话1" ></Label>
  <NumberEditor Key="Phone1" Caption="公司电话1" Precision="11" UseGroupingSeparator="false" Scale="0" />
  <Label Key="Label5" Caption="公司电话2" ></Label>
  <NumberEditor Key="Phone2" Caption="公司电话2" Precision="11" UseGroupingSeparator="false" Scale="0" />
  <Label Key="Label6" Caption="公司联系人" ></Label>
  <TextEditor Key="contact" Caption="公司联系人" MaxLength="10" />
  <Label Key="Label7" Caption="公司地址" ></Label>
  <TextEditor Key="address" Caption="公司地址" MaxLength="50" />
  <Label Key="Label8" Caption="业务关系" ></Label>
  <TextEditor Key="relationship" Caption="业务关系" MaxLength="30" />
  <Label Key="Label9" Caption="公司邮箱" ></Label>
  <TextEditor Key="email" Caption="公司邮箱" MaxLength="50" />
  <Label Key="Label10" Caption="备注" ></Label>
  <TextEditor Key="remark" Caption="备注" MaxLength="50" />
</FluidTableLayoutPanel>
```

当RepeatCount定义为4时（RepeatCount="4"），如下图所示：

公司客户联系表		
公司简称	<input type="text"/>	公司全称
公司英文	<input type="text"/>	公司电话1
公司电话2	<input type="text"/>	公司联系人
公司地址	<input type="text"/>	业务关系
公司邮箱	<input type="text"/>	备注

5.8:弹性流布局面板FlexFlowLayoutPanel

弹性流布局面板为按照从上到下的顺序对组件进行定位，子组件的宽度为面板的宽度，高度为子组件的内容高度或定义的高度； 如下图所示：



示例：定义一个“挂号信息”的弹性流布局面板，其中包括2个网格布局面板，具体同流式布局面板的定义相同。并且一个子组件高度定义为60%，一个定义为40%。

```
<FlexFlowLayoutPanel key="main" >
  <Label Key="Label1" Caption="病人信息" />
  <GridLayoutPanel Key="Grid1" Caption="病人信息" Height="60%" >
    <RowDefCollection RowHeight="30" RowGap="3">
      <RowDef/>
      <RowDef/>
    </RowDefCollection>
    <ColumnDefCollection ColumnGap="10" >
      <ColumnDef Width="50px"/>
      <ColumnDef Width="150px"/>
      <ColumnDef Width="50px"/>
    </ColumnDefCollection>
  </GridLayoutPanel>
</FlexFlowLayoutPanel>
```

```

        <ColumnDef Width="150px"/>
        <ColumnDef Width="50px"/>
        <ColumnDef Width="150px"/>
    </ColumnDefCollection>
    <Label Key="Label2" Caption="门诊号" X="0" Y="0" XSpan="1" YSpan="1" ></Label>
    <NumberEditor Key="id" Caption="门诊号" Precision="8" UseGroupingSeparator="false" Scale="0" X="1" Y="0"
XSpan="1" YSpan="1" >
    </NumberEditor>
    <Label Key="Label3" Caption="姓名" X="2" Y="0" XSpan="1" YSpan="1" ></Label>
    <TextEditor Key="Name" Caption="姓名" MaxLength="8" Trim="True" X="3" Y="0" XSpan="1" YSpan="1"/>

    <Label Key="Label4" Caption="性别" X="4" Y="0" XSpan="1" YSpan="1" ></Label>
    <TextEditor Key="sexual" Caption="性别" MaxLength="4" Trim="True" X="5" Y="0" XSpan="1" YSpan="1" />

    <Label Key="Label5" Caption="年龄" X="0" Y="1" XSpan="1" YSpan="1"></Label>
    <NumberEditor Key="Age" Caption="年龄" Precision="3" UseGroupingSeparator="false" Scale="0" X="1" Y="1"
XSpan="1" YSpan="1" >
    </NumberEditor>
    <Label Key="Label6" Caption="地址" X="2" Y="1" XSpan="1" YSpan="1"></Label>
    <TextEditor Key="Address" Caption="地址" MaxLength="20" Trim="True" X="3" Y="1" XSpan="3" YSpan="1"/>
</GridLayoutPanel>
<Separator>
</Separator>
<Label Key="Label12" Caption="挂号信息" ></Label>

<GridLayoutPanel Key="Grid2" Caption="挂号信息" Height="40%" >
    <RowDefCollection RowHeight="30" RowGap="3">
        <RowDef/>
        <RowDef/>
        <RowDef/>
        <RowDef/>
    </RowDefCollection>
    <ColumnDefCollection ColumnGap="10" >
        <ColumnDef Width="50px"/>
        <ColumnDef Width="150px"/>
        <ColumnDef Width="50px"/>
        <ColumnDef Width="150px"/>
        <ColumnDef Width="50px"/>
        <ColumnDef Width="150px"/>
    </ColumnDefCollection>
    <Label Key="Label2" Caption="性质" X="0" Y="0" XSpan="1" YSpan="1" ></Label>
    <ComboBox Key="DropDownButton1" Caption="性质" X="1" Y="0" XSpan="1" YSpan="1" >
        <Item Key="item1" Caption="自费" Value="0"/>
        <Item Key="item2" Caption="公费" Value="1"/>
    </ComboBox>

    <Label Key="Label3" Caption="门诊科室" X="2" Y="0" XSpan="1" YSpan="1" ></Label>
    <ComboBox Key="DropDownButton2" Caption="门诊科室" X="3" Y="0" XSpan="1" YSpan="1" >
        <Item Key="item1" Caption="内科" Value="0"/>
        <Item Key="item2" Caption="儿科" Value="1"/>
        <Item Key="item3" Caption="骨科" Value="2"/>
    </ComboBox>
    <Label Key="Label4" Caption="挂号类型" X="4" Y="0" XSpan="1" YSpan="1" ></Label>
    <ComboBox Key="DropDownButton3" Caption="挂号类型" X="5" Y="0" XSpan="1" YSpan="1">
        <Item Key="item1" Caption="普通门诊" Value="0"/>
        <Item Key="item2" Caption="专家门诊" Value="1"/>
        <Item Key="item3" Caption="特需门诊" Value="2"/>
    </ComboBox>
    <Label Key="Label5" Caption="医生" X="0" Y="1" XSpan="1" YSpan="1"></Label>
    <ComboBox Key="DropDownButton4" Caption="医生" X="1" Y="1" XSpan="1" YSpan="1">
        <Item Key="item1" Caption="张医生" Value="0"/>
        <Item Key="item2" Caption="李医生" Value="1"/>
        <Item Key="item3" Caption="王医生" Value="2"/>
    </ComboBox>
    <Label Key="Label6" Caption="挂号费" X="2" Y="1" XSpan="1" YSpan="1"></Label>
    <NumberEditor Key="cost1" Caption="挂号费" Precision="8" UseGroupingSeparator="false" Scale="2" X="3" Y="1"
XSpan="1" YSpan="1" >
    </NumberEditor>
    <Label Key="Label7" Caption="诊疗费" X="4" Y="1" XSpan="1" YSpan="1"></Label>
    <NumberEditor Key="cost2" Caption="诊疗费" Precision="8" UseGroupingSeparator="false" Scale="2" X="5" Y="1"
XSpan="1" YSpan="1" >
    </NumberEditor>
    <Label Key="Label8" Caption="专家费" X="0" Y="2" XSpan="1" YSpan="1"></Label>
    <NumberEditor Key="cost3" Caption="专家费" Precision="8" UseGroupingSeparator="false" Scale="2" X="1" Y="2"
XSpan="1" YSpan="1" >
    </NumberEditor>
    <Label Key="Label9" Caption="挂号编号" X="2" Y="2" XSpan="1" YSpan="1"></Label>
    <NumberEditor Key="cost4" Caption="挂号编号" Precision="8" UseGroupingSeparator="false" Scale="2" X="3" Y="2"
XSpan="1" YSpan="1" >
    </NumberEditor>
    <Label Key="Label10" Caption="减免费" X="4" Y="2" XSpan="1" YSpan="1"></Label>
    <NumberEditor Key="cost5" Caption="减免费" Precision="8" UseGroupingSeparator="false" Scale="2" X="5" Y="2"
XSpan="1" YSpan="1" >

```



```
</NumberEditor>
<Label Key="Label11" Caption="病历费" X="0" Y="3" XSpan="1" YSpan="1"></Label>
<NumberEditor Key="cost6" Caption="病历费" Precision="8" UseGroupingSeparator="false" Scale="2" X="1" Y="3"
XSpan="1" YSpan="1" >
</NumberEditor>
</GridLayoutPanel>
</FlexFlowLayoutPanel>
```



注意

弹性流布局面板和流布局面板的区别在于弹性流布局面板可以定义子组件的高度，而流布局不能定义子组件的高度，依赖于子组件本身的高度。如下图所示：

病人信息

门诊号

姓名

性别

年龄

地址

病人信息

门诊号

年龄

挂号信息

性质

医生

专家费

病历费

挂号信息

性质

门诊科室

挂号类型

医生

挂号费

诊疗费

专家费

挂号编号

减免费

病历费

(右图: 流布局面

(左图: 弹性流布局面板)

5.9: 面板组合应用实例

通常制作表单时，会同时应用到多种面板格式，下面将介绍多种面板嵌套使用的方法。

示例1：使用选项卡面板（TabPanel）、弹性流布局面板（FlexFlowLayoutPanel）、拆分面板（SplitPanel）、流式表布局面板（FluidTableLayoutPanel）制作物流表单。

基础信息

货装信息

付款信息

保存

取消

公司名称

公司英文

邮递方式

联系人

公司地址

业务关系

到货时间

付款方式

	物料	规格 (ML)	单位
1			

-

+

```
<TabPanel Key="main_tab" Height="100%" Padding="5px">
    <FlexFlowLayoutPanel Key="main" Caption="弹性流式布局" Height="100%" Padding="5px">
        <ToolBar Key="main_toolbar" Height="pref" IsDefault="True"/>
        <SplitPanel Key="main_split" Orientation="Vertical" Height="100%">
            <SplitSize Size="50%"/>
            <SplitSize Size="50%"/>
            <FluidTableLayoutPanel Key="OrderList" Height="100%" RepeatCount="6" RepeatGap="20" RowGap="2" ColumnGap="2"
RowHeight="30" Padding="10px">
                <TableColumnCollection>
                    <TableColumn Width="10px"/>
                    <TableColumn Width="100%"/>
                </TableColumnCollection>
                <Label Key="Label1" Caption="公司名称"></Label>
                <TextEditor Key="Company" Caption="公司名称" MaxLength="30"/>
                <Label Key="Label2" Caption="公司英文"></Label>
                <TextEditor Key="English" Caption="公司英文" MaxLength="30"/>
                <Label Key="Label3" Caption="物流订单号"></Label>
                <TextEditor Key="Number" Caption="物流订单号" MaxLength="30"/>
                <Label Key="Label4" Caption="邮递方式"></Label>
                <ComboBox Key="ComboBox1" Caption="邮递方式" Editable="False">
                    <Item Key="combox1" Caption="送货上门" Value="1"/>
                    <Item Key="combox2" Caption="自取" Value="2"/>
                    <Item Key="combox3" Caption="普通快递" Value="3"/>
                    <Item Key="combox4" Caption="加急快件" Value="4"/>
                </ComboBox>
                <Label Key="Label5" Caption="联系人"></Label>
                <TextEditor Key="Contact" Caption="联系人" MaxLength="30"/>
                <Label Key="Label6" Caption="联系电话"></Label>
                <NumberEditor Key="Phone" Caption="联系电话" DecPrecision="11" UseGroupingSeparator="false" DecScale="0"/>
                <Label Key="Label7" Caption="公司地址"></Label>
                <TextEditor Key="Address" Caption="公司地址" MaxLength="50"/>
                <Label Key="Label8" Caption="业务关系"></Label>
                <TextEditor Key="relationship" Caption="业务关系" MaxLength="30"/>
                <Label Key="Label9" Caption="公司邮箱"></Label>
                <TextEditor Key="email" Caption="公司邮箱" MaxLength="50"/>
                <Label Key="Label10" Caption="到货时间"></Label>
                <DatePicker Key="TIME" Caption="到货时间">

```

```

        Enable="!ReadOnly()" OnlyDate="True" Visible="true">
    </DatePicker>
    <Label Key="Label11" Caption="付款方式" ></Label>
    <ComboBox Key="ComboBox2" Caption="付款方式" Editable="False">
        <Item Key="combox1" Caption="支付宝" Value="1" />
        <Item Key="combox2" Caption="货到付款" Value="2"/>
        <Item Key="combox3" Caption="银行卡" Value="3"/>
    </ComboBox>
    <Label Key="Label12" Caption="备注" ></Label>
    <TextEditor Key="remark" Caption="备注" MaxLength="50" />
</FluidTableLayoutPanel>
<Grid Key="Detail" Caption="物料信息" Enable="!ReadOnly()" ShowRowHead="True"
    PageRowCount="5" PageLoadType="DB">
    <GridColumnCollection>
        <GridColumn Key="c1" Caption="物料" Width="200px" Sortable="true" Visible="true" ColumnType="Fix">
    </GridColumn>
        <GridColumn Key="c2" Caption="规格 (ML)" Width="200px" Sortable="true" Visible="true" ColumnType="Fix">
    </GridColumn>
        <GridColumn Key="c3" Caption="单位" Width="200px" Sortable="true" Visible="true" ColumnType="Fix">
    </GridColumn>
        <GridColumn Key="c4" Caption="数量" Width="200px" Sortable="true" Visible="true" ColumnType="Fix">
    </GridColumn>
        <GridColumn Key="c5" Caption="重量 (Kg)" Width="200px" Sortable="true" Visible="true" ColumnType="Fix">
    </GridColumn>
        <GridColumn Key="c6" Caption="条形码" Width="200px" Sortable="true" Visible="true" ColumnType="Fix">
    </GridColumn>
    </GridColumnCollection>
    <GridRowCollection>
        <GridRow Key="R1" RowType="Detail">
            <GridCell Key="Material" Caption="物料" CellType="TextEditor">
                <DataBinding ColumnKey="Material"/>
            </GridCell>
            <GridCell Key="Vol" Caption="规格" CellType="NumberEditor" DecScale="0" DecPrecision="10">
                <DataBinding ColumnKey="Vol"/>
            </GridCell>
            <GridCell Key="Unit" Caption="单位" CellType="TextEditor">
                <DataBinding ColumnKey="Unit"/>
            </GridCell>
            <GridCell Key="Amount" Caption="数量" CellType="NumberEditor" DecScale="0" DecPrecision="10">
                <DataBinding ColumnKey="Amount"/>
            </GridCell>
            <GridCell Key="Weight" Caption="重量" CellType="NumberEditor" DecScale="0" DecPrecision="10">
                <DataBinding ColumnKey="Weight"/>
            </GridCell>
            <GridCell Key="Code" Caption="条形码" CellType="TextEditor">
                <DataBinding ColumnKey="Code"/>
            </GridCell>
        </GridRow>
    </GridRowCollection>
</Grid>
</SplitPanel>
</FlexFlowLayoutPanel>
<GridLayoutPanel Key="Grid1" Caption="货装信息" Height="100%" Padding="5px">
</GridLayoutPanel>
<GridLayoutPanel Key="Grid2" Caption="付款信息" Height="100%" Padding="5px">
</GridLayoutPanel>
</TabPanel>

```

示例2: 使用边界布局面板 (BorderLayoutPanel)、列式布局面板 (ColumnLayoutPanel)、网格布局面板 (GridLayoutPanel)、流式布局面板 (FlowLayoutPanel) 制作门诊挂号表单。

门诊挂号

保存

修改

删除

病人信息

门诊号

姓名

性别

年龄

地址

挂号信息

挂号编号

挂号类型

医生

费用

记录号	挂号编号	挂号类型	科室
表中无内容			

1/6

首页

上页

1

2

3

4

```
<BorderLayoutPanel Key="BorderLayoutPanel" Caption="边界布局">
  <ToolBar Key="main_toolbar" Height="pref" IsDefault="True" Area="Top" />
  <ColumnLayoutPanel Key="ColumnLayoutPanel1" Caption="列式布局" Area="Left">
    <GridLayoutPanel Key="Grid1" Caption="网格布局1" X="0" Y="0" Padding="5px">
      <RowDefCollection RowHeight="30" RowGap="3">
        <RowDef/>
        <RowDef/>
        <RowDef/>
        <RowDef/>
        <RowDef/>
        <RowDef/>
      </RowDefCollection>
      <ColumnDefCollection ColumnGap="10">
        <ColumnDef Width="50px"/>
        <ColumnDef Width="100px"/>
        <ColumnDef Width="50px"/>
      </ColumnDefCollection>
      <Label Key="Label1" Caption="病人信息" X="0" Y="0" XSpan="1" YSpan="1">
        <Format ForeColor="#0000FF"/>
      </Label>
      <Label Key="Label2" Caption="门诊号" X="0" Y="1" XSpan="1" YSpan="1"/>
      <NumberEditor Key="id" Caption="门诊号" X="1" Y="1" XSpan="1" YSpan="1"> </NumberEditor>
      <Label Key="Label3" Caption="姓名" X="0" Y="2" XSpan="1" YSpan="1"/>
      <TextEditor Key="Name" Caption="姓名" MaxLength="8" Trim="True" X="1" Y="2"
        XSpan="1" YSpan="1"/>
      <Label Key="Label4" Caption="性别" X="0" Y="3" XSpan="1" YSpan="1"/>
      <TextEditor Key="sexual" Caption="性别" MaxLength="4" Trim="True" X="1" Y="3"
        XSpan="1" YSpan="1"/>
      <Label Key="Label5" Caption="年龄" X="0" Y="4" XSpan="1" YSpan="1"/>
      <NumberEditor Key="Age" Caption="年龄" Precision="3"
        UseGroupingSeparator="false" Scale="0" X="1" Y="4" XSpan="1" YSpan="1"> </NumberEditor>
      <Label Key="Label6" Caption="地址" X="0" Y="5" XSpan="1" YSpan="1"/>
      <TextEditor Key="Address" Caption="地址" MaxLength="20" Trim="True" X="1"
        Y="5" XSpan="2" YSpan="1"/>
    </GridLayoutPanel>
  </ColumnLayoutPanel>
  <TableLayoutPanel Key="TableLayoutPanel1" Caption="表中无内容" X="0" Y="0" XSpan="1" YSpan="1">
    <Table>
      <thead>
        <tr>
          <th>记录号</th>
          <th>挂号编号</th>
          <th>挂号类型</th>
          <th>科室</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td data-cs="4" data-kind="parent">表中无内容</td>
          <td data-kind="ghost"></td>
          <td data-kind="ghost"></td>
          <td data-kind="ghost"></td>
        </tr>
      </tbody>
    </Table>
  </TableLayoutPanel>
</BorderLayoutPanel>
```

39

```

</GridLayoutPanel>
<GridLayoutPanel Key="Grid2" Caption="网格布局2" X="0" Y="2" Padding="5px">
  <RowDefCollection RowHeight="30" RowGap="3">
    <RowDef/>
    <RowDef/>
    <RowDef/>
    <RowDef/>
    <RowDef/>
  </RowDefCollection>
  <ColumnDefCollection ColumnGap="10">
    <ColumnDef Width="50px"/>
    <ColumnDef Width="100px"/>
    <ColumnDef Width="50px"/>
  </ColumnDefCollection>
  <Label Key="Label_Grid2" Caption="挂号信息" X="0" Y="0" XSpan="1" YSpan="1" HAlign="Left">
    <Format ForeColor="#0000FF"/>
  </Label>
  <Label Key="Label_IDNumber" Caption="挂号编号" X="0" Y="1" XSpan="1" YSpan="1" HAlign="Left"/>
  <TextEditor Key="IDNumber" Caption="挂号编号" MaxLength="50" X="1" Y="1" XSpan="2"
    YSpan="1"> </TextEditor>
  <Label Key="Label_Type" Caption="挂号类型" X="0" Y="2" XSpan="1" YSpan="1" HAlign="Left"/>
  <TextEditor Key="Type" Caption="挂号类型" MaxLength="50" X="1" Y="2" XSpan="1" YSpan="1"
    > </TextEditor>
  <Label Key="Label_Doctor" Caption="医生" X="0" Y="3" XSpan="1" YSpan="1" HAlign="Left"/>
  <TextEditor Key="Doctor" Caption="医生" MaxLength="50" X="1" Y="3" XSpan="1"
    YSpan="1"> </TextEditor>
  <Label Key="Label_Fee" Caption="费用" X="0" Y="4" XSpan="1" YSpan="1" HAlign="Left"/>
  <TextEditor Key="Fee" Caption="费用" MaxLength="50" X="1" Y="4" XSpan="2"
    YSpan="1"> </TextEditor>
</GridLayoutPanel>
</ColumnLayoutPanel>
<FlowLayoutPanel Key="FlexFlowLayoutPanel" Caption="流式布局" Area="Center" Padding="5px">
  <ListView Key="list">
    <ListViewColumnCollection>
      <ListViewColumn Key="OID" Width="100px" Caption="记录号" ColumnType="Label"
        DataColumnKey="OID"> </ListViewColumn>
      <ListViewColumn Key="IDNumber" Width="100px" Caption="挂号编号"
        ColumnType="Label" DataColumnKey="IDNumber"> </ListViewColumn>
      <ListViewColumn Key="Type" Width="100px" Caption="挂号类型"
        ColumnType="Label" DataColumnKey="Type"> </ListViewColumn>
      <ListViewColumn Key="Division" Width="100px" Caption="科室"
        ColumnType="Label" DataColumnKey="Division"> </ListViewColumn>
      <ListViewColumn Key="Doctor" Width="100px" Caption="医生"
        ColumnType="Label" DataColumnKey="ToTime"> </ListViewColumn>
      <ListViewColumn Key="Fee" Width="100px" Caption="费用" ColumnType="Label"
        DataColumnKey="Fee"> </ListViewColumn>
    </ListViewColumnCollection>
  </ListView>
</FlowLayoutPanel>
<GridLayoutPanel Key="GridLayoutPanel1" Caption="网格布局" Area="Right">
  <RowDefCollection RowHeight="30" RowGap="3">
    <RowDef/>
    <RowDef/>
    <RowDef/>
    <RowDef/>
  </RowDefCollection>
  <ColumnDefCollection>
    <ColumnDef Width="50px"/>
    <ColumnDef Width="100px"/>
    <ColumnDef Width="50px"/>
  </ColumnDefCollection>
  <Label Key="Label17" Caption="科室信息" X="0" Y="0" XSpan="1" YSpan="1">
    <Format ForeColor="#0000FF"/>
  </Label>
  <Label Key="Label18" Caption="科室" X="0" Y="1" XSpan="1" YSpan="1"/>
  <TextEditor Key="Division" Caption="科室" MaxLength="8" Trim="True" X="1" Y="1"
    XSpan="1" YSpan="1"/>
  <Label Key="Label19" Caption="限额" X="0" Y="2" XSpan="1" YSpan="1"/>
  <TextEditor Key="Limit" Caption="限额" MaxLength="8" Trim="True" X="1" Y="2"
    XSpan="1" YSpan="1"/>
  <Label Key="Label10" Caption="已挂" X="0" Y="3" XSpan="1" YSpan="1"/>
  <TextEditor Key="Used" Caption="已挂" MaxLength="8" Trim="True" X="1" Y="3"
    XSpan="1" YSpan="1"/>
</GridLayoutPanel>
</BorderLayoutPanel>

```

第 6 章 功能类组件

目录

6.1. 基本组件	41
6.2. 视图组件	49
6.3. 图表	51
6.4. 字典(Dict)	53
6.5. 表格(Grid)	57
6.6. 其他	62

本章节介绍的是功能类组件。功能类组件是界面的最基本元素。我将分成5个部分分别介绍，包括基本组件、视图组件、字典、表格、图表这几个部分。

6.1:基本组件

6.1.1:静态文本框

示例：定义一组静态文本框。文本框1：字体红色。文本框2：字体加粗、斜体。

源代码如下：

```
<Label Key="Label1" Caption="文本框-字体红色：" X="6" Y="16" XSpan="1" YSpan="1" >
  <Format ForeColor="#FF4500"/ >
</Label>
<Label Key="Label2" Caption="文本框-字体加粗：" X="7" Y="16" XSpan="1" YSpan="1" >
  <Format >
    <Font Bold="True" Italic="True"/>
  </Format>
</Label>
```



注意

- Key—标识，整个工程中唯一标识，不能重名。默认值为空。
- Caption—名称。默认值为空；
- 组件属性都是区分大小写的，代码编写时请注意！
- Format格式属性定义组件的格式，ForeColor定义字体颜色、Font中定义字体名称、字体大小、字体是否加粗、字体是否倾斜。

6.1.2:按钮

示例：定义一组按钮组件：普通按钮、不可用按钮、字体红色按钮、背景红色按钮、图片按钮。

按钮组件：

普通按钮：

普通按钮

不可用按钮：

不可用按钮

按钮(字体红色)：

按钮(字体红色)

按钮(背景红色)：

按钮(背景红色)

源代码如下：

```
<Label Key="LabelBtn1" Caption="普通按钮：" X="0" Y="1" XSpan="1" YSpan="1" />
<Button Key="Btn1" Caption="普通按钮" X="1" Y="1" XSpan="1" YSpan="1" />

<Label Key="LabelBtn2" Caption="不可用按钮：" X="2" Y="1" XSpan="1" YSpan="1" />
<Button Key="Btn2" Caption="不可用按钮" Enable="False" X="3" Y="1" XSpan="1" YSpan="1" />
```

```

<Label Key="LabelBtn3" Caption="按钮(字体红色): " X="4" Y="1" XSpan="1" YSpan="1"/>
<Button Key="Btn3" Caption="按钮(字体红色)" X="5" Y="1" XSpan="1" YSpan="1">
<Format ForeColor="#FF4500"/>
</Button>

<Label Key="LabelBtn4" Caption="按钮(背景红色): " X="6" Y="1" XSpan="1" YSpan="1"/>
<Button Key="Btn4" Caption="按钮(背景红色)" X="7" Y="1" XSpan="1" YSpan="1">
<Format BackColor="#FF4500"/>
</Button>

```



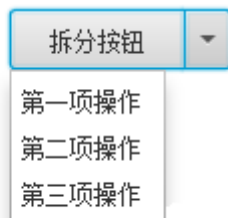
注意

- 按钮2是不可用按钮，可以通过属性 Enable来定义，取值为True可用和False不可用；。
- 按钮3是字体红色按钮，可以通过属性ForeColor来定义，取值为RGB颜色。
- 按钮4是背景红色按钮，可以通过属性BackColor来定义，取值为RGB颜色。

6.1.3: 拆分按钮

示例：定义一个拆分按钮，具有三个下拉菜单项：第一项操作、第二项操作、第三项操作。每个下拉选项具有点击事件，且拆分按钮本身也具有点击事件。

拆分按钮：



源代码如下：

```

<SplitButton Key="SplitButton" Caption="拆分按钮" X="1" Y="3" XSpan="1" YSpan="1" >
<![CDATA[NewDetail(' TestDetailRowEditor', GetOID())]]>
<DropdownItem Key="item1" Text="第一项操作">
<OnClick>
<![CDATA[NewDetail(' TestDetailRowEditor', GetOID())]]>
</OnClick>
</DropdownItem>
<DropdownItem Key="item2" Text="第二项操作">
<![CDATA[NewDetail(' TestDetailRowEditor', GetOID())]]>
</DropdownItem>
<DropdownItem Key="item3" Text="第三项操作">
<![CDATA[NewDetail(' TestDetailRowEditor', GetOID())]]>
</DropdownItem>
</SplitButton>

```



注意

- 拆分按钮本身具有点击事件，事件处理由OnClick属性定义；
- 拆分按钮包含多个下拉菜单项，下拉菜单项由DropdownItem定义；
- 每个下拉菜单都有点击事件，事件处理由OnClick属性定义；

6.1.4: 下拉按钮

示例：定义一个下拉按钮，具有三个下拉菜单项：第一项操作、第二项操作、第三项操作。每个下拉选项具有点击事件。

下拉按钮：



源代码如下：

```
<DropDownButton Key="DropDownButton" Caption="下拉按钮" X="3" Y="3" XSpan="1" YSpan="1">
  <DropDownItem Key="item1" Text="第一项操作">
    <OnClick>
      <![CDATA[NewDetail('TestDetailRowEditor', GetOID())]]>
    </OnClick>
  </DropDownItem>
  <DropDownItem Key="item2" Text="第二项操作">
    <![CDATA[NewDetail('TestDetailRowEditor', GetOID())]]>
  </DropDownItem>
  <DropDownItem Separator="True"/>
  <DropDownItem Key="item3" Text="第三项操作">
    <![CDATA[NewDetail('TestDetailRowEditor', GetOID())]]>
  </DropDownItem>
</DropDownButton>
```



- 注意
- 下拉按钮的下拉项属性同拆分按钮中的属性相同；
 - 下拉按钮和拆分按钮的区别在于：下拉按钮自身没有点击事件，其点击的作用是为了显示下拉菜单；而拆分按钮本身也具有点击事件。
 - DropDownItem中的Separator属性为True时，定义一个菜单项分隔线；

6.1.5: 下拉框

示例：定义一组下拉框组件。下拉框1：类型为静态不可编辑，下拉项为：初始、审批中、审批通过、否决。下拉框2：类型为表达式，下拉项为表达式GetStatusItems。

下拉框3：类型为静态可编辑。下拉框4：类型为查询。

下拉框-静态：



(图1)

下拉框-表达式：



(图2)

下拉框-可编辑：



(图3)

下拉框-Query：



(图4)

源代码如下:

```
<Label Key="LabelComboBox1" Caption="下拉框-静态: " X="0" Y="5" XSpan="1" YSpan="1"/>
<ComboBox Key="ComboBox1" Caption="下拉框-静态" Editable="False" X="1" Y="5" XSpan="1" YSpan="1">
  <Item Key="combox1" Caption="初始" Value="1" />
  <Item Key="combox2" Caption="审批中" Value="2"/>
  <Item Key="combox3" Caption="审批通过" Value="3"/>
  <Item Key="combox4" Caption="否决" Value="4"/>
</ComboBox>

<Label Key="LabelComboBox2" Caption="下拉框-表达式: " X="2" Y="5" XSpan="1" YSpan="1"/>
<ComboBox Key="ComboBox2" Caption="下拉框表达式" SourceType="Formula" X="3" Y="5" XSpan="1" YSpan="1">
  <FormulaItems>
    <![CDATA[
      GetStatusItems()
    ]]>
  </FormulaItems>
</ComboBox>

<Label Key="LabelComboBox3" Caption="下拉框-可编辑: " X="4" Y="5" XSpan="1" YSpan="1"/>
<ComboBox Key="ComboBox3" Caption="下拉框-可编辑" Editable="True" X="5" Y="5" XSpan="1" YSpan="1">
  <Item Key="combox1" Caption="初始" Value="1" />
  <Item Key="combox2" Caption="审批中" Value="2"/>
  <Item Key="combox3" Caption="审批通过" Value="3"/>
  <Item Key="combox4" Caption="否决" Value="4"/>
</ComboBox>

<Label Key="LabelComboBox4" Caption="下拉框-Query: " X="6" Y="5" XSpan="1" YSpan="1"/>
<ComboBox Key="ComboBox4" Caption="下拉框-Query" SourceType="Query" X="7" Y="5" XSpan="1" YSpan="1">
  <QueryDef>
    <Statement>
      <![CDATA[select oid , name from sys_operator where oid > ?]]>
    </Statement>
    <ParameterCollection>
      <Parameter SourceType="Const" Value="0"/>
    </ParameterCollection>
  </QueryDef>
</ComboBox>
```



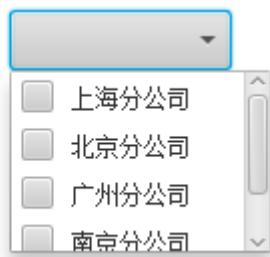
注意

- SourceType定义下拉项的来源类型，取值为Items(项目)、Formula(表达式)、Query(查询)。
- 下拉框通过Editable来定义是否可编辑属性，取值为False时不可编辑下拉框值，True时可编辑下拉框值。

6.1.6: 多选下拉框

示例：定义一个多选下拉框组件。下拉项为：上海分公司、北京分公司、广州分公司、南京分公司。

多选下拉框：



源代码如下:

```
<Label Key="LabelCheckListBox" Caption="多选下拉框: " X="0" Y="6" XSpan="1" YSpan="1"/>
<CheckListBox Key="CheckListBox1" Caption="多选_静态" X="1" Y="7" XSpan="1" YSpan="1">
  <Item Key="1" Caption="上海分公司" Value="1" />
  <Item Key="2" Caption="北京分公司" Value="2"/>
  <Item Key="3" Caption="广州分公司" Value="3"/>
  <Item Key="4" Caption="南京分公司" Value="4"/>
</CheckListBox>
```



注意

- 多选下拉框属性同下拉框基本相同。
- DynamicItems是否是动态下拉项，取值为True或False；如果是动态下拉项，下拉列表会根据绑定的数据源而变化。

6.1.7: 数值编辑框

示例：定义一组数值编辑框。数值框1：长度16位整型，不带千分位。数值框2：长度16位整型，带千分位，分组长度为3。数值框3：长度16位小数2位。数值框4：数值进行四舍五入。

数值框组件：

16位整无千分位

带千分位分组3

小数2位

四舍五入

源代码如下：

```
<Label Key="LabelTestNum" Caption="数值框组件：" X="0" Y="8" XSpan="1" YSpan="1"/>
<Label Key="LabelTestNum1" Caption="16位整无千分位" X="0" Y="9" XSpan="1" YSpan="1"/>
<NumberEditor Key="TestNum1" Caption="16位整无千分位" Precision="16" Scale="0" UseGroupingSeparator="False" X="1" Y="9"
XSpan="1" YSpan="1"/>

<Label Key="LabelTestNum2" Caption="组分隔，分组位3" X="2" Y="9" XSpan="1" YSpan="1"/>
<NumberEditor Key="TestNum2" Caption="组分隔，分组位3" Precision="16" Scale="0" UseGroupingSeparator="True" GroupingSize="3"
X="3" Y="9" XSpan="1" YSpan="1"/>

<Label Key="LabelTestNum3" Caption="小数2位" X="4" Y="9" XSpan="1" YSpan="1"/>
<NumberEditor Key="TestNum3" Caption="小数2位" Precision="16" Scale="2" X="5" Y="9" XSpan="1" YSpan="1"/>

<Label Key="LabelTestNum4" Caption="四舍五入" X="6" Y="9" XSpan="1" YSpan="1"/>
<NumberEditor Key="TestNum4" Caption="四舍五入" Precision="16" Scale="2" RoundingMode="HALF_UP" X="7" Y="9" XSpan="1"
YSpan="1"/>
```



注意

- Precision - 精度，默认值16；如示例数值框-16位整不带千分位
- UseGroupingSeparator - 是否使用组分隔，取值为True或者False，属性GroupingSize定义分组长度。
- Scale - 小数位位数，默认值2；如示例小数2位
- RoundingMode - 四舍五入规则，HALF_UP, ROUND_UP, ROUND_DOWN 默认为HALF_UP

6.1.8: 文本编辑框

示例：定义一组文本编辑框。文本框1：定义最大长度为10。文本框2：转换成大写字母。文本框3：有字符提示。文本框4：不允许输入字符!@#%*。文本框5：去除首尾空格。文本框6：文本框左边有图片，并且内嵌文本。

文本框组件：

最大长度10:

转换大写:

字符提示:

请输入字符

过滤字符:

字符!@#%*不允许输入

去首尾

文本框左图片，内嵌文本:

 用户

源代码如下：

```
<Label Key="LabelTextEditor" Caption="文本框组件：" X="0" Y="10" XSpan="1" YSpan="1"/>
<Label Key="LabelTextEditor1" Caption="最大长度10:" X="0" Y="11" XSpan="1" YSpan="1"/>
<TextEditor Key="TextEditor1" Caption="最大长度10:" MaxLength="10" X="1" Y="11" XSpan="1" YSpan="1"/>

<Label Key="LabelTextEditor2" Caption="转换大写:" X="2" Y="11" XSpan="1" YSpan="1"/>
<TextEditor Key="TextEditor2" Caption="转换大写:" Case="Upper" X="3" Y="11" XSpan="1" YSpan="1"/>
```

```

<Label Key="LabelTextEditor3" Caption="字符提示: " X="4" Y="11" XSpan="1" YSpan="1"/>
<TextEditor Key="TextEditor3" Caption="字符提示: " PromptText="请输入字符" X="5" Y="11" XSpan="1" YSpan="1"/>

<Label Key="LabelTextEditor4" Caption="过滤字符: " X="6" Y="11" XSpan="1" YSpan="1"/>
<TextEditor Key="TextEditor4" Caption="过滤字符: " PromptText="字符!@#%*不允许输入" InvalidChars="!@#%*" X="7" Y="11"
XSpan="1" YSpan="1"/>

<Label Key="LabelTextEditor5" Caption="去首尾多余空格: " X="8" Y="11" XSpan="1" YSpan="1"/>
<TextEditor Key="TextEditor5" Caption="去首尾多余空格: " Trim="True" X="9" Y="11" XSpan="1" YSpan="1"/>

<Label Key="LabelTextEditor6" Caption="文本框左图片, 内嵌文本: " X="0" Y="12" XSpan="2" YSpan="1"/>
<TextEditor Key="TextEditor6" Caption="文本框左图片, 内嵌文本" PreIcon="Resource/User.png" EmbedText="用户" X="2" Y="12"
XSpan="1" YSpan="1" />

```



注意

- **MaxLength** - 可输入文本的最大长度；如示例最大长度10。
- **InvalidChars** - 不允许输入的字符列表。定义的集合中的字符在输入阶段即不能被输入。如示例过滤字符。
- **PromptText** - 内容为空的提示文本，作为用户输入提醒。如示例字符提示。
- **Case** - 是否大小写转换，取值为None不转换，Lower转换为小写，Upper转换为大写，默认值为None，如示例大写。
- **Trim** - 是否去除首尾多余空格。如示例去首尾多余空格。
- **PreIcon**为文本编辑框的左侧图标，取值为相对于Solution中的Resource目录的相对路径；**EmbedText**左侧内嵌文本。如示例文本框左图片，内嵌文本。

6.1.9: 密码编辑框

示例：定义一个密码编辑框。

密码编辑框

密码输入

源代码如下：

```

<PasswordEditor Key="Password" Caption="密码编辑框" EmbedText="密码输入" X="7" Y="12" XSpan="1" YSpan="1"/>

```



注意

密码不能以明文显示

6.1.10: 文本域

示例：定义一个文本域组件，最大字符长度为100。

多行文本：

源代码如下：

```

<TextArea Key="TextArea" Caption="多行文本: " MaxLength="100" X="4" Y="12" XSpan="2" YSpan="2"/>

```



注意

- 文本输入框允许输入多行文本；
- **MaxLength**限制输入的最大字符长度。

6.1.11: 复选框

示例：定义一个仓库复选框，这组复选框包含4个CheckBox：上海仓库、北京仓库、广州仓库、南京。

多选框 - 仓库： ☐ 上海仓库 ☐ 北京仓库 ☐ 广州仓库 ☐ 南京仓库

源代码如下：

```
<Label Key="L_CheckBox" Caption="多选框-仓库:" X="0" Y="14" XSpan="1" YSpan="1"/>
<CheckBox Key="CheckBox1" Caption="上海仓库" X="1" Y="14" XSpan="1" YSpan="1"/>
<CheckBox Key="CheckBox2" Caption="北京仓库" X="2" Y="14" XSpan="1" YSpan="1"/>
<CheckBox Key="CheckBox3" Caption="广州仓库" X="3" Y="14" XSpan="1" YSpan="1"/>
<CheckBox Key="CheckBox4" Caption="南京仓库" X="4" Y="14" XSpan="1" YSpan="1"/>
```



注意

复选框的值是布尔类型，在选中的情况下，取值为True，否则为False，在存储到数据库的时候，存为整型。

6.1.12: 单选框

示例：定义一个性别单选框。这组单选框有3个RadioButton。

单选框 - 性别 ☐ 男 ☐ 女 ☐ 未知

源代码如下：

```
<Label Key="L_RadioButton" Caption="单选框-性别" X="0" Y="15" XSpan="1" YSpan="1"/>
<RadioButton Key="RadioButton1_1" GroupKey="RadioButton1" IsGroupHead="true" Caption="男" Value="1" X="1" Y="15" XSpan="1" YSpan="1"/>
<RadioButton Key="RadioButton1_2" GroupKey="RadioButton1" Caption="女" Value="2" X="2" Y="15" XSpan="1" YSpan="1"/>
<RadioButton Key="RadioButton1_3" GroupKey="RadioButton1" Caption="未知" Value="3" X="3" Y="15" XSpan="1" YSpan="1"/>
```



注意

- GroupKey为组的标识。单选框成组定义，同一个GroupKey的组件为一个单选框组；如示例单选框-性别。
- IsGroupHead 单选框组头标志，默认值为False。一个组内只能有一个组头组件，系统中取值通过这个组头组件获得。
- Value - 选中的值。单选框的值为字符串类型，通过Value属性定义；

6.1.13: 超链接

示例：定义一个超链接。

超链接 [超链接](#)

源代码如下：

```
<Label Key="L_HyperLink" Caption="超链接" X="0" Y="15" XSpan="1" YSpan="1"/>
<HyperLink Key="HyperLink" Caption="超链接" URL="www.baidu.com" TargetShowType="Current" X="1" Y="15" XSpan="1" YSpan="1"/>
```



注意

- TargetShowType - 目标显示方式，取值为新窗口(New)、当前(Current)、新Tab页(NewTab)。
- 在定义OnClick的情况下，超链接的作用相当于按钮，OnClick为点击事件的执行脚本；

6.1.14: 日期选择

示例：定义一个日期选择器，格式为yyyy-MM-dd

日期



源代码如下：

```
<Label Key="L_DatePicker" Caption="日期" X="2" Y="15" XSpan="1" YSpan="1"/>
<DatePicker Format="yyyy-MM-dd" OnlyDate="True" X="3" Y="15" XSpan="1" YSpan="1"/>
```



注意

- 时间的显示格式Format取值为yyyy-MM-dd和yyyy-MM-dd mm:hh:ss；默认值为yyyy-MM-dd。
- 是否仅保留日期部分通过OnlyDate定义，默认值为True。

6.1.15: 分隔线 (Separator)

样式



源代码如下：

```
<Separator>
</Separator>
```



注意

主要是用在列式布局、流式布局、网格布局、弹性流式布局中。（具体再确认）

6.1.16: 工具栏

工具栏下可支持所有的功能类组件，目前支持的是Button。

示例：如定义的默认工具栏，有如下Button组件。

新增 更新 查询

1. 首先在OperationCollection定义相关Button组件。

```
<OperationCollection>
  <Operation Key="New" Caption="新增" Icon="new.png">
```

```

        <Action>
        <![CDATA[
New('')
]]>
        </Action>
    </Operation>
    <Operation Key="Update" Caption="更新" Icon="new.png">
        <Action>
        <![CDATA[
DBUpdate("")
]]>
        </Action>
    </Operation>
    <Operation Key="Query" Caption="查询">
        <Action>
        <![CDATA[
DBNamedQueryValue("");
]]>
        </Action>
    </Operation>

```

2. 再定义ToolBar属性

```
<ToolBar Key="main_toolbar" Height="pref" IsDefault="True"/>
```

IsDefault：用于说明是否是一个表单中的默认工具栏，如果为True，那么表单的操作定义会加到这个工具栏中。

6.1.17: 图片

示例：定义一个图片组件。

图片：



源代码如下：

```

<Label Key="L_Image" Caption="图片：" X="4" Y="16" XSpan="1" YSpan="1"/>
<Image Key="Image_1" SourceType="Resource" Image="Resource/LoginHead.png" Height="200px" X="5" Y="16" XSpan="1" YSpan="1"/>

```



注意

SourceType定义图片的来源类型，取值为Data或Resource，默认值为Data。

6.2: 视图组件

6.2.1: 字典视图 (DictView)

示例：定义一个字典视图，单击行触发事件。

代码	名称
▼ Warehouse	
▶ 01	发货仓(标准品、半...
▶ 04	报废仓
05	呆滞品仓
06	不良品仓(来料不...
11	公司
15	样品仓(新开发标准...
▼ 30	客户
3011	3011
3012	3012
666	666
70	111
71	71
72	72

源代码如下：

```
<DictView Key="DictTree" FormulaItemKey="Para('ItemKey')">
  <RowClick>
    <![CDATA[openDict(GetSelectedValue('DictTree','ItemKey'),GetSelectedValue('DictTree','oid'),'Container')]]>
  </RowClick>
</DictView>
```

6.2.2: 列表视图(ListView)

示例：定义一个列表视图，双击行触发事件，列表包含以下字段：

记录号	工号	姓名	开始时间	结束时间	事由
表中无内容					
1/1	首页	上页	1	下页	尾页
			到第	1	页 确定

1. 首先必须定义对应的DataSource数据源表。

```
<DataSource>
  <DataObject Key="VacationInfo" Caption="请假信息" PrimaryType="Virtual" SecondaryType="Normal">
    <TableCollection>
      <Table Key="VacationInfo" Caption="基本信息" DBTableName="VacationInfo" TableMode="Detail" SourceType="Query"
        Persist="False">
        <Column Key="OID" Caption="记录号" Persist="True" DataType="Long" DBColumnName="OID" />
        <Column Key="IDNumber" Caption="工号" Persist="True" DataType="Varchar" Length="50"/>
        <Column Key="Name" Caption="姓名" Persist="True" DBColumnName="Name" DataType="Varchar" Length="50"/>
        <Column Key="FromTime" Caption="开始时间" Persist="True" DataType="DateTime"/>
        <Column Key="ToTime" Caption="结束时间" Persist="True" DataType="DateTime"/>
        <Column Key="Reason" Caption="事由" Persist="True" DataType="Varchar" Length="250"/>
      </Table>
    </TableCollection>
  </DataObject>
</DataSource>
```

```

</Table>
</TableCollection>
</DataObject>
</DataSource>

```

2. 然后定义ListView，RowDbClick定义行双击事件；RowClick定义行单击事件。

```

<ListView Key="list" TableKey="VacationInfo" Height="45%">
  <RowDbClick>
    <![CDATA[
      Open(' VacationRequest', 0ID)
    ]]>
  </RowDbClick>
  <ListViewColumnCollection>
    <ListViewColumn Key="OID" Width="100px" Caption="记录号" ColumnType="Label" DataColumnKey="OID">
    </ListViewColumn>
    <ListViewColumn Key="IDNumber" Width="100px" Caption="工号" ColumnType="Label" DataColumnKey="IDNumber">
    </ListViewColumn>
    <ListViewColumn Key="Name" Width="100px" Caption="姓名" ColumnType="Label" DataColumnKey="Name">
    </ListViewColumn>
    <ListViewColumn Key="FromTime" Width="100px" Caption="开始时间" ColumnType="Label" DataColumnKey="FromTime">
    </ListViewColumn>
    <ListViewColumn Key="ToTime" Width="100px" Caption="结束时间" ColumnType="Label" DataColumnKey="ToTime">
    </ListViewColumn>
    <ListViewColumn Key="Reason" Width="100px" Caption="事由" ColumnType="Label" DataColumnKey="Reason">
    </ListViewColumn>
  </ListViewColumnCollection>
</ListView>

```

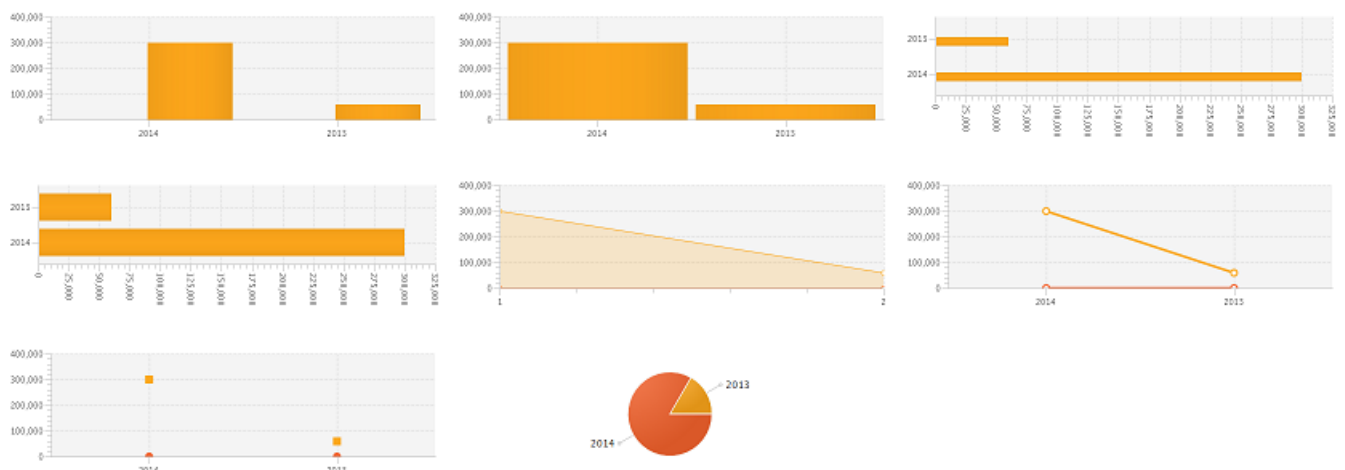
TableKey 是绑定的数据源表，对应刚才定义的DataSource。

ListViewColumn属性：

- Key - 列的标识，整个Window内不重复。
- Caption - 列的名称。
- DataColumnKey - 列绑定的数据对象的列标识。
- ColumnType - 列类型，取值为Label, CheckBox、HyperLink、Button、DatePicker、Dict、ComboBox、NumberEdit、TextEdit类型。
- 其它属性，根据ColumnType的属性而定。

6.3: 图表

图表是数据的一种图像展示。图表的类型有HBar(水平条状图)，StackedHBar(累积水平条状图)，VBar(垂直条状图)，StackedVBar(累积垂直条状图)，Area(面积图)，Line(线图)，Scatter(散点图)，Pie(饼图)。



图表的数据来源由多个系列数据组成；如下图所示：

	2007	2008	2009
苹果	567	1292	1292
柠檬	956	1665	2559
桔子	1154	1927	2774

图表的定义分为系列集合和项目集合

系列定义说明了图表的序列数据如此产生，每个图表可以包含多个系列数据，

项目定义了系列数据中的项目轴来源，每个图表只能有一个项目定义，通过Category定义，DataKey说明了项目的来源列；

示例：定义一个水平条状图，数据项目是年份，数据系列是产量和销售额2个。

还是必须先定义DataSource，然后再定义Chart属性。

```
<DataSource>
  <DataObject Key="ProductReport" Caption="产品报表" MainTableKey="USER_ComputerOccupy" PrimaryType="Virtual"
    SecondaryType="Normal">
    <TableCollection>
      <Table Key="ProductReport" Caption="电脑使用情况列表" DBTableName="ProductReport" TableMode="Detail" SourceType="Query"
        Persist="False">
        <Column Key="Year" Caption="年份" Persist="False" DataType="Integer" DBColumnName="Year"/>
        <Column Key="Amount" Caption="产量" Persist="False" DataType="Numeric" Precision="16" Scale="4" DBColumnName="Amount"/>
        <Column Key="Money" Caption="销售额" Persist="False" DataType="Numeric" Precision="16" Scale="4" DBColumnName="Money"/>
      </Table>
    </TableCollection>
  </DataObject>

  <Chart Key="Chart2" Caption="水平条状图" ChartType="HBar" SourceType="DataObject" X="2" Y="0" XSpan="1" YSpan="1">
    <ChartDataSource BindingKey="ProductReport">
      <Series DataKey="Amount" Title="产量" />
      <Series DataKey="Money" Title="销售额" />
      <Category DataKey="Year"/>
    </ChartDataSource>
  </Chart>
```

- ChartType - 图表类型，包括HBar(水平条状图)，StackedHBar(累积水平条状图)，VBar(垂直条状图)，StackedVBar(累积垂直条状图)，Area(面积图)，Line(线图)，Scatter(散点图)，Pie(饼图)；
- SourceType - 数据源类型，取值为DataObject、List、View、Grid。
- BindingKey-绑定数据表的标识，就是DataObject中定义的数据源表。
- Title - 图表的标题；
- 数据源定义-系列(Series)：DataKey - 系列的数据来源字段
- 数据源定义-项目(Category)：DataKey - 项目的数据来源字段

6.4: 字典(Dict)

字典包括3种：普通字典、动态字典、复合字典。

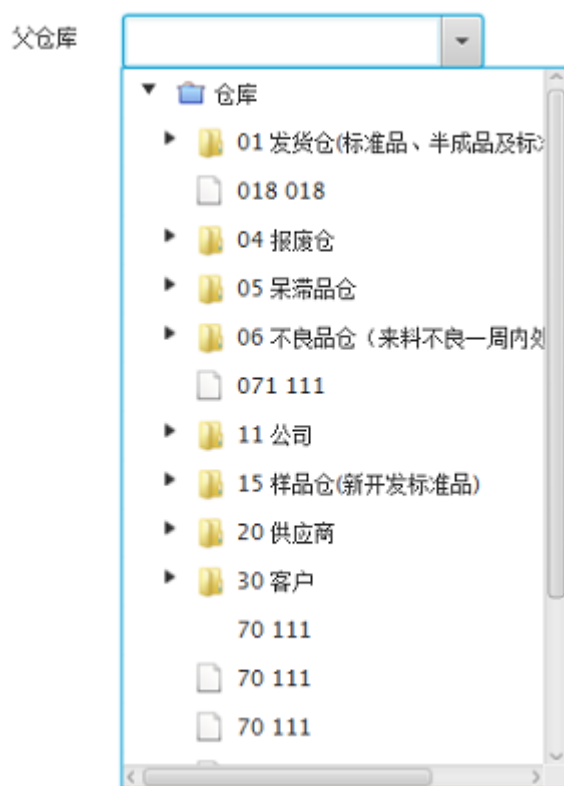
普通字典根据其关联的ItemKey所对应的数据对象的类型不同，又分为层次式字典和链式字典。

6.4.1: 普通字典

6.4.1.1: 层次字典

层次字典适用于数据量较小的情况，控件包含一下拉按钮，通过下拉按钮可以弹出字典树结构：

示例1：定义一个普通字典，显示仓库字典的数据。（以下图示显示的字典内容根据字典定义的数据内容而变化，并非固定不变。）



```
<Label Key="L_Dict1" Caption="普通字典"/>
<Dict Key="Dict1" BuddyKey="L_Dict1" ItemKey="Warehouse" Caption="父仓库" >
</Dict>
```

- ItemKey - 字典对象的标识
- BuddyKey - 关联组件的标识

示例2：定义一个子仓库，子仓库根据父仓库的节点为根节点。



```
<Label Key="L_Dict2" Caption="子仓库" />
<Dict Key="Dict2" BuddyKey="L_Dict2" Root="Dict1" ItemKey="Warehouse" Caption="仓库" X="3" Y="1"/>
```

- Root - 字典树显示的根节点，可以指定一个汇总节点作为字典树根节点

6.4.1.2: 字典过滤

字典的过滤分为值过滤、结果集过滤两种。

值过滤又包括常量Const过滤、表达式Formula过滤。

示例3(值过滤-常量)：定义一个字典包含值过滤，字段为一个常量，过滤字段NodeType为1。

字段过滤-常量 NodeType=1

```
<Label Key="L_Dict3" Caption="字段过滤-常量 NodeType=1" X="4" Y="1"/>
<Dict Key="Dict3" BuddyKey="L_Dict3" ItemKey="Warehouse" Caption="仓库" X="5" Y="1">
  <ItemFilter>
    <Filter Type="FieldValue">
      <FilterValue Type="Const" FieldKey="NodeType" RefValue="1"/>
    </Filter>
  </ItemFilter>
  <DataBinding TableKey="Test_TestDict" ColumnKey="Dict3">
  </DataBinding>
</Dict>
```

当FilterValue类型全为常量时，过滤条件只会计算1次。

Filter 属性：

Type 定义过滤类型，取值为FieldValue值过滤、DataSet结果集过滤；

FilterValue属性

- Type定义值过滤类型，取值为Const常量、Formula表达式
- FieldKey 值过滤的字段；
- RefValue 取值关联的字段

示例4(值过滤-表达式)：定义一个字典包含值过滤，字段为一个表达式，过滤表达式NodeType为1。

字段过滤-公式 NodeType=1

```
<Label Key="L_Dict4" Caption="字段过滤-公式 NodeType=1" X="6" Y="1"/>
<Dict Key="Dict4" BuddyKey="L_Dict4" ItemKey="Warehouse" Caption="仓库" X="7" Y="1">
  <ItemFilter>
    <Filter Type="FieldValue">
      <FilterValue Type="Formula" FieldKey="NodeType" RefValue="1"/>
    </Filter>
  </ItemFilter>
  <DataBinding TableKey="Test_TestDict" ColumnKey="Dict4">
  </DataBinding>
</Dict>
```

```
</DataBinding>
</Dict>
```

过滤表达式NodeType为1，这个1可以看作为一个表达式，而非常量。

当filterValue有类型不为常量时，过滤条件每次都会计算。

示例5(结果集过滤)：定义一个字典包含结果集过滤，根据单选框的值进行过滤，并定义Query结果集的查询语句。

☒ 数据集过滤条件1 ☐ 数据集过滤条件2 根据条件过滤字典

```
<RadioButton Key="Dict6RadioButton_0" GroupKey="Dict6RadioButton" IsGroupHead="true" Caption="数据集过滤条件1" Value="0" X="0"
Y="2">
  <DataBinding DefaultValue="0" TableKey="Test_TestDict" ColumnKey="Dict6RadioButton">
  </DataBinding>
</RadioButton>
<RadioButton Key="Dict6RadioButton_1" BuddyKey="Dict6RadioButton_0" GroupKey="Dict6RadioButton" Caption="数据集过滤条件2"
Value="1" X="1" Y="2">
  <DataBinding TableKey="Test_TestDict" ColumnKey="Dict6RadioButton">
  </DataBinding>
</RadioButton>
<Label Key="L_Dict6_1" Caption="" X="4" Y="2"/>
<Label Key="L_Dict6" Caption="根据条件过滤字典" X="4" Y="2"/>
<Dict Key="Dict6" BuddyKey="L_Dict6" ItemKey="Warehouse" Caption="仓库" X="5" Y="2">
  <ItemFilter>
    <Filter Type="DataSet" Condition="Dict6RadioButton_0==0" Query="select OID from wm_warehousehead2 where parentid=?">
      <FilterValue Type="Const" ParaValue="11505"/>
    </Filter>
    <Filter Type="DataSet" Condition="Dict6RadioButton_0==1" Query="select OID from wm_warehousehead2 where parentid=?">
      <FilterValue Type="Const" ParaValue="11601"/>
    </Filter>
  </ItemFilter>
  <DataBinding TableKey="Test_TestDict" ColumnKey="Dict6">
  </DataBinding>
</Dict>
```

其计算方法为通过Query定义查询出一批字典的集合，只在在查询结果集中出现的字典项才可以在当前字典中出现。

上个示例中，通过Query查询语句：Query="select OID from wm_warehousehead2 where parentid=?";当Dict6RadioButton_0==0时，在warehousehead2表中查询parentid=11505的仓库；当Dict6RadioButton_0==1时，在warehousehead2表中查询parentid=11601的仓库

6.4.1.3:链式字典

链式字典适用于数据量较多的情况，控件包含一查询按钮，通过查询按钮弹出模糊查找窗口。其没有下拉视图。

样式如下：(点击一个链式字典后弹出以下对话框)



6. 4. 2: 动态字典

动态字典的样式同普通字典(Dict)，只是其关联的ItemKey会随着条件的变化而变化：

示例：定义一个动态字典，根据单选框的值确定动态字典的itemKey。

☐ 物料
 ☒ 仓库

动态字典

1. 首先定义1组单选框：物料和仓库。

```
<RadioButton Key="DynamicDict1RadioButton_0" GroupKey="DynamicDict1RadioButton" IsGroupHead="true" Caption="物料"
Value="Material" X="0" Y="2">
  <DataBinding DefaultValue="Material" TableKey="Test_TestDict" ColumnKey="DynamicDict1RadioButton">
  </DataBinding>
</RadioButton>
<RadioButton Key="DynamicDict1RadioButton_1" BuddyKey="DynamicDict1RadioButton_0" GroupKey="DynamicDict1RadioButton"
Caption="仓库" Value="Warehouse" X="1" Y="2">
  <DataBinding TableKey="Test_TestDict" ColumnKey="DynamicDict1RadioButton">
  </DataBinding>
</RadioButton>
```

2. 然后定义动态字典。当选择物料的时候，字典显示物料字典；选择仓库时，字典显示仓库字典。

```
<Label Key="L_dynamicDict1_1" Caption="" X="0" Y="12"/>
<Label Key="L_dynamicDict1" Caption="动态字典" X="0" Y="12"/>
<DynamicDict Key="dynamicDict1" BuddyKey="L_dynamicDict1" RefKey="DynamicDict1RadioButton_0" Caption="物料或仓库" X="1"
Y="12">
  <DataBinding TableKey="Test_TestDict" ColumnKey="dynamicDict1">
  </DataBinding>
</DynamicDict>
```

DynamicDict属性说明

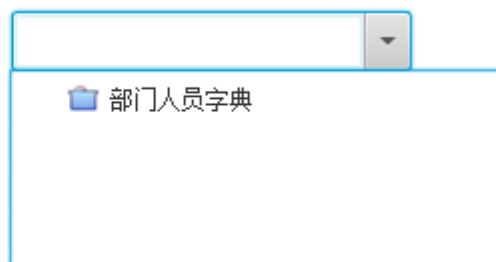
- RefKey - 动态字典ItemKey的关联字段，取ItemKey时，需保证RefKey字段对应的值正确。

6. 4. 3: 复合字典

复合字典为多个普通字典的数据按照从属或包含关系定义生成。

示例：定义一个复合字典 " 部门人员 " 字典。

复合字典:部门-人员



1. 首先在CompDict目录下定义数据对象Dpt-Emp.xml。

```
<DataObject Key="Dpt-Emp" Caption="部门人员字典" PrimaryType="Entity" SecondaryType="CompDict">
  <Relation Caption="部门人员字典" Description="Description">
    <Layer ItemKey="Department">
    </Layer>
    <Layer ItemKey="Employee" Relation="Belong" TableKey="dp_employee" ColumnKey="DepartmentID">
    </Layer>
  </Relation>
</DataObject>
```

- Relation 跟上层的关系，取值为Contain(包含)、Belong(从属)。

2. 再定义CompDict组件

```
<CompDict Key="dptemp" BuddyKey="L_DptEmp" ItemKey="Dpt-Emp" Caption="部门-人员">
  <DataBinding TableKey="Test_TestDict" ColumnKey="dptemp">
  </DataBinding>
</CompDict>
```

6.5: 表格(Grid)

表格是由行集合和列集合组成。

6.5.1: 行数据模型

表格中行分为固定行、分组行、明细行和汇总行。

- 固定行，为表格中的单行数据；固定行只有一行数据。
- 分组行，为数据分组的合计行；分组行根据数据的分组数目而定
- 明细行，表格中关联的表数据行；明细行有多行数据。
- 汇总行，明细数据的汇总行，为表格中的单行数据。汇总行也只有一行数据，其同固定行只是语义不同。

6.5.2: 列数据模型

表格中的列分为固定列、分组列、明细列和汇总列。

- 固定列，表格中的单列数据；固定列只会生成表格中的一列。
- 分组列，表格中的列扩展的分组数据列；分组列根据列扩展的数量会生成多列。
- 明细列，表格中的列扩展字段列；明细列会生成多列数据。
- 汇总列，表格中的列扩展区域的字段汇总数据列。汇总列同固定列一样，语义不同。

6.5.3: 表格的数据模型及定义

表格的数据组成结构基本样式如下表所示：

	固定列	固定列	固定列	分组列	分组列	明细列	分组列	分组列	汇总列
固定行						RootExpandTitle			
固定行	Field0_Title	Field1_Title	Field2_Title	Level0_Head0		Level0_Title		Level0_Tail0	
固定行					Level1_Head0	Field3_Title		Level1_Tail1	
分组行	Group0_Head0					sum(Field3)			
分组行		Group1_Head0				sum(Field3)			
明细行	Field0	Field1	Field2	sum(Fields)	sum(Field3)	sum(Fields)	sum(Field3)	sum(Fields)	sum(Field3)
分组行		Group1_Tail0				sum(Field3)			
分组行	Group0_Tail0					sum(Field3)			
汇总行	Total					sum(Field3)			

6.5.4: 表格分组说明

6.5.4.1: 数据分组

有个表格定义如下，

地区	物料	公司	金额
----	----	----	----

其中地区和物料为分组单元格，表格关联的数据表的数据如下：

地区	物料	型号	数量
上海	电脑	大	10
上海	电脑	大	3
北京	电脑	大	6
北京	电脑	中	30
北京	电脑	大	3
北京	手机	小	9
北京	手机	小	40
上海	手机	小	5
北京	手机	小	6
上海	手机	小	7

数据通过地区和物料分组后，在明细数据分组的基础上，如果表格定义了分组行，那么在相应分组位置，需要生成分组的分组数据汇总行。

将表格的定义改成如下所示：

地区合			sum(数
计			量)
	物料合		sum(数
	计		量)
地区	物料	型号	数量

生成的表格行如下表所示：

地区合计			25
	物料合计		13
上海	电脑	大	10

上海	电脑	大	3
	物料合计		12
上海	手机	小	5
上海	手机	小	7
地区合计			94
	物料合计		39
北京	电脑	大	6
北京	电脑	中	30
北京	电脑	大	3
	物料合计		55
北京	手机	小	9
北京	手机	小	40
北京	手机	小	6

6.5.4.2: 列分组

列扩展的目标是把多行数据，根据选定的关键字转换成一行表格行以展现和编辑数据。

表格数据如下：

地区	物料	部门	型号	数量
上海	电脑	开发部	大	5
上海	电脑	测试部	大	2
上海	电脑	开发部	小	6
上海	电脑	测试部	中	10

"数量"是列扩展，扩展源为"型号"，"地区"和"物料"和"公司"是关键字段，对于给定的数据，生成的表格行如下：

地区	物料	部门	数量		
			大	中	小
上海	电脑	开发部	5		6
上海	电脑	测试部	2	10	

6.5.5: 实例

定义一个表格。列定义 "规格"、"等级" 为明细列，其他定义为固定列。

行定义 "地区" 和 "物料" 为分组行，生成分组的分组数据汇总行：地区合计和物料合计。

	定位				规格			汇总	速度	距离	等级			测试		
	地区	产品			大	中	小				高级	中级	低级	点击		
		物料	部门人员	动态字典										测试按钮	这是超链接	复选框
1	地区合计(头)															
2		物料合计(头)														
3							0		0				测试按钮	超链接	<input type="checkbox"/>	
4																
5																

配置文件如下：

1. 定义GridColumnCollection列集合：

```
<Grid Key="detail_grid" HideGroup4Editing="false" Caption="入库单明细" UsePaging="false" PagingRowCount="6" ShowRowHead="true"
Enable="!ReadOnly()">
<GridColumnCollection>
<GridColumn Key="C11" Caption="定位" ColumnExpand="false" ColumnType="Fix">
<GridColumnCollection>
<GridColumn Key="C1" Caption="地区" Width="79px" ColumnExpand="false" ColumnType="Fix">
</GridColumn>
<GridColumn Key="C31" Caption="产品" ColumnExpand="false" ColumnType="Fix">
<GridColumnCollection>
<GridColumn Key="C2" Caption="物料" Width="75px" ColumnExpand="false" ColumnType="Fix">
</GridColumn>
<GridColumn Key="C3" Caption="部门人员" Width="75px" ColumnExpand="false" ColumnType="Fix">
</GridColumn>
<GridColumn Key="C4" Caption="动态字典" Width="75px" ColumnExpand="false" ColumnType="Fix">
</GridColumn>
</GridColumnCollection>
</GridColumn>
</GridColumnCollection>
</GridColumn>
<GridColumn Key="C4" Caption="规格" Width="" ColumnExpand="true" ColumnType="Detail">
<ColumnExpand ExpandType="title">
</ColumnExpand>
<GridColumnCollection>
<GridColumn Key="C5" Caption="规格" Width="50px" ColumnExpand="true">
<ColumnExpand ExpandType="data" ExpandSourceType="custom" ColumnKey="Size">
<![CDATA[TestExpandSource()]]>
</ColumnExpand>
</GridColumn>
</GridColumnCollection>
</GridColumn>
<GridColumn Key="C601" Caption="汇总" Width="75px" ColumnExpand="false" ColumnType="Fix"/>
<GridColumn Key="C602" Caption="速度" Width="75px" ColumnExpand="false" ColumnType="Fix"/>
<GridColumn Key="C603" Caption="距离" Width="75px" ColumnExpand="false" ColumnType="Fix"/>
<GridColumn Key="C604" Caption="等级" Width="" ColumnExpand="true" ColumnType="Detail">
<ColumnExpand ExpandType="title">
</ColumnExpand>
<GridColumnCollection>
<GridColumn Key="C7" Caption="等级" Width="50px" ColumnExpand="true">
<ColumnExpand ExpandType="data" ExpandSourceType="custom" ColumnKey="Gradation">
<![CDATA[TestExpandSource("Gradation")]]>
</ColumnExpand>
</GridColumn>
</GridColumnCollection>
</GridColumn>
<GridColumn Key="C811" Caption="测试" Width="85px" ColumnExpand="false" ColumnType="Fix">
<GridColumnCollection>
<GridColumn Key="C81" Caption="点击" Width="85px" ColumnExpand="false" ColumnType="Fix">
<GridColumnCollection>
<GridColumn Key="C8" Caption="测试按钮" Width="70px" ColumnExpand="false" ColumnType="Fix"/>
<GridColumn Key="C9" Caption="这是超链接" Width="70px" ColumnExpand="false" ColumnType="Fix"/>
<GridColumn Key="C14" Caption="复选框" Width="50px" ColumnExpand="false" ColumnType="Fix"/>
</GridColumnCollection>
</GridColumn>
<GridColumn Key="C101" Caption="下拉" Width="85px" ColumnExpand="false" ColumnType="Fix">
<GridColumnCollection>
<GridColumn Key="C10" Caption="下拉框" Width="70px" ColumnExpand="false" ColumnType="Fix"/>
<GridColumn Key="C11" Caption="多选下拉框" Width="75px" ColumnExpand="false" ColumnType="Fix"/>
</GridColumnCollection>
</GridColumn>
<GridColumn Key="C12" Caption="测试文本" Width="50px" ColumnExpand="false" ColumnType="Fix"/>
</GridColumnCollection>
</GridColumn>
<GridColumn Key="C13" Caption="输入框2" Width="100px" ColumnExpand="false" ColumnType="Fix"/>
<GridColumn Key="C14" Caption="单选框" Width="100px" ColumnExpand="false" ColumnType="Fix"/>
</GridColumnCollection>
```

2. 定义GridRowCollection行集合：

```
<GridRowCollection>
<GridRow Key="R1" RowType="Group" RowHeight="30" TableKey="" GroupKey="Area">
<GridCell Caption="地区合计(头)"/>
<GridCell />
<GridCell />
<GridCell />
<GridCell Key="AreaSumHead" />
```

```

        <DataBinding DefaultFormulaValue="Sum(' Amount')"/>
    </GridCell>
    <GridCell />
    <GridCell />
    <GridCell />
    <GridCell Key="AreaSumHead1">
    <DataBinding DefaultFormulaValue="Sum(' Price')"/>
    </GridCell>
    <GridCell />
    <GridCell />
    <GridCell />
    <GridCell />
    <GridCell />
    <GridCell />
    <GridCell />
    </GridRow>
    <GridRow Key="R2" RowType="Group" RowHeight="30" TableKey="" GroupKey="Material">
    <GridCell/>
    <GridCell Caption="物料合计(头)"/>
    <GridCell />
    <GridCell />
    <GridCell Key="MeterialSumHead">
    <DataBinding DefaultFormulaValue="Sum(' Amount')"/>
    </GridCell>
    <GridCell />
    <GridCell />
    <GridCell />
    <GridCell Key="MeterialSumHead1">
    <DataBinding DefaultFormulaValue="Sum(' Price')"/>
    </GridCell>
    <GridCell />
    <GridCell />
    <GridCell />
    <GridCell />
    <GridCell />
    <GridCell />
    <GridCell />
    <GridCell />
    </GridRow>
    <GridRow Key="R3" RowType="Detail" RowHeight="30" TableKey="WM_StockInDetail" GroupKey="Material">
    <GridCell Key="Area" Caption="地区(Area)" CellType="Dict" ItemKey="Area" CellGroupType="RowGroup">
    <DataBinding ColumnKey="Area"/>
    </GridCell>
    <GridCell Key="Material" Caption="物料" CellType="Dict" ItemKey="Material" FilterDependency="Memo"
    CellGroupType="RowGroup">
    <DataBinding ColumnKey="Material"/>
    </GridCell>
    <GridCell Key="Emp" Caption="部门人员" CellType="CompDict" ItemKey="Dpt-Emp">
    <DataBinding ColumnKey="Emp"/>
    </GridCell>
    <GridCell Key="DynamicDict" Caption="动态字典" CellType="DynamicDict" RefKey="RadioButton1_Material">
    <DataBinding ColumnKey="DynamicDict"/>
    </GridCell>
    <GridCell Key="Amount" Caption="数量" CellType="NumberEditor">
    <DataBinding ColumnKey="Amount"/>
    </GridCell>
    <GridCell Key="SumExpand" Caption="汇总" CellType="NumberEditor">
    <DataBinding DefaultFormulaValue="SumExpand(' Amount')"/>
    </GridCell>
    <GridCell Key="Speed" Caption="速度" CellType="NumberEditor">
    <DataBinding/>
    </GridCell>
    <GridCell Key="Distance" Caption="距离" CellType="NumberEditor">
    <DataBinding DefaultFormulaValue="Speed*2"/>
    </GridCell>
    <GridCell Key="Price" Caption="单价" CellType="NumberEditor" Scale="2" Precision="10">
    <DataBinding Required="true"/>
    </GridCell>
    <GridCell Key="Test1" Caption="测试按钮" CellType="Button" />
    <GridCell Key="Test2" Caption="超链接" CellType="HyperLink" />
    <GridCell Key="Test3" Caption="单选框" CellType="CheckBox" />
    <GridCell Key="Test7" Caption="下拉框" CellType="ComboBox" DynamicItems="true" ItemsDependency="Test5">
    <FormulaItems>
    <![CDATA[
        GetStatusItems();
    ]]>
    </FormulaItems>
    </GridCell>
    <GridCell Key="Test4" Caption="多选下拉框" CellType="CheckListBox" DynamicItems="true" ItemsDependency="Memo">
    <FormulaItems>
    <![CDATA[
        GetStatusItems();
    ]]>
    </FormulaItems>
    </GridCell>

```

```

]]>
</FormulaItems>
</GridCell>
<GridCell Key="Test5" Caption="测试文本" CellType="TextEditor"/>
<GridCell Key="Test6" Caption="输入框2" CellType="DatePicker" />
<GridCell Key="Test8" Caption="单选框" CellType="CheckBox" />
</GridRow>
</GridRowCollection>
</Grid>

```

添加数据后，效果如下：

	定位				规格			汇总	速度	距离	等级		
	地区	产品			大	中	小				高级	中级	低级
		物料	部门人员	动态字典									
1	地区合计(头)				7.0	10.0	12.0				14.0	17.0	16.0
2		物料合计(头)			7.0	10.0	12.0				14.0	17.0	16.0
3				01 发货仓(...	2.00	3.00	4.00	9.00	0.00	0.00	4.00	4.00	2.00
4				04 报废仓	3.00	5.00	3.00	11.00		0.00	5.00	5.00	5.00
5				11 公司	2.00	2.00	5.00	9.00		0.00	5.00	8.00	9.00

如图所示，明细行数据汇总到物料合计和地区合计。

6.6: 其他

6.6.1: 树形菜单(TreeMenuBar)

```

<TreeMenuBar Type="Type" Source="Source" Tag="Tag" IsDefault="True">
  <TreeMenuItemCollection>
    <TreeMenuItem Key="Key" Caption="Caption" Icon="Icon" Type="Type" FormKey="FormKey" URL="URL" Parameters="Parameters">
      <Action>
        <![CDATA[ ]]>
      </Action>
    </TreeMenuItem>
    ...
  </TreeMenuItemCollection />
  ...
</TreeMenuItemCollection>
<TitleStyle />
<ExpandTitleStyle />
<ItemStyle />
<SelectedItemStyle />
</TreeMenuBar>

```

- Type - 命令视图的样式类型，取值范围为Tree, ListTree, GroupTree, ListGroupTree和Custom;
- Source - 定义命令的来源，如果未定义，表示取系统主入口;
- Tag - 如果Style的取值为Custom，那么Tag用来标记定制所需的区分标记;
- IsDefault - 是否默认菜单，如果是默认菜单，用于加载所有入口。

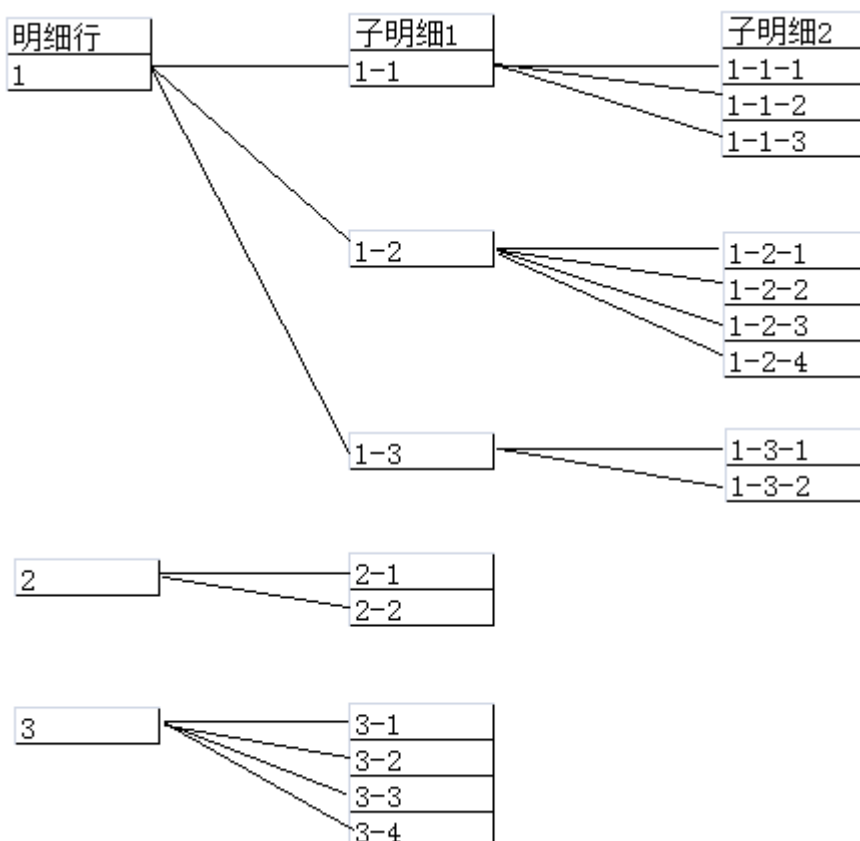
MenuItem菜单项属性

- Key - 菜单项的标识;
- Caption - 菜单项的名称;
- Type - 菜单项的事件类型，取值为Form(链接到表单)、Script(脚本)、URL(链到地URL);
- FormKey - 表单的标识;

- Action - 脚本，参见脚本的说明；
- URL - 超链接地址；
- Parameters - 需要传递给FormKey及URL的参数。

6.6.2: 子明细(EmbedSub)

在数据对象中的结构中有子明细定义，子表单的数据即建立在子明细的基础上，子表单用于处理具有层次结构的数据检索、编辑；子表单的数据结构的一般形式如下图所示：



内嵌子表单就是对数据进行数据过滤和显示。数据过滤通过Grid中的EmbedSub、EmbedKey和MetaComponent的BindingCellKey确定关联子表单。当某行被选中时，过滤其子表格的显示数据，并在需要的关联组件中显示关联的数据。

示例1：第一个表为物料在地区的拣货量总计，第二张表为子表单，是每个地区仓库具体的拣货量。

定义第一张表地区销售额总计为detail_grid，再定义子表单sub_detail_grid，关联子明细表detail_grid。如，选择第一行“上海”的数据时，子表中关联的表格只显示跟“上海”有关的两行数据。

```

<Grid Key="detail_grid" HideGroup4Editing="false" Caption="拣货单明细" UsePage="false" PageRowCount="6" ShowRowHead="true" >
  <GridColumnCollection>
    <GridColumn Key="C1" Caption="地区" Width="75px" ColumnExpand="false" ColumnType="Fix">
    </GridColumn>
    <GridColumn Key="C2" Caption="物料" Width="75px" ColumnExpand="false" ColumnType="Fix">
    </GridColumn>
    <GridColumn Key="C3" Caption="数量" Width="75px" ColumnExpand="false" ColumnType="Fix">
    </GridColumn>
  </GridColumnCollection>
</Grid>
  
```

```

</GridColumnCollection>
<GridRowCollection>
  <GridRow Key="R3" RowType="Detail" RowHeight="30" TableKey="WM_PickDetail" GroupKey="">
    <GridCell Key="Pick_Area" Caption="地区" CellType="Dict" ItemKey="Area">
      <DataBinding ColumnKey="Pick_Area"/>
    </GridCell>
    <GridCell Key="Pick_Material" Caption="物料" CellType="TextEditor">
      <DataBinding ColumnKey="Pick_Material"/>
    </GridCell>
    <GridCell Key="Part_Amount" Caption="数量" CellType="NumberEditor">
      <DataBinding ColumnKey="Part_Amount"/>
    </GridCell>
  </GridRow>
</GridRowCollection>
</Grid>
<SubDetail Key="sub_detail_grid" BindingGridKey="detail_grid">
  <GridLayoutPanel Key="detail_content1" Height="100%" Padding="5px">
    <RowDefCollection>
      <RowDef Height="100%" />
    </RowDefCollection>
    <ColumnDefCollection>
      <ColumnDef Width="100%" />
    </ColumnDefCollection>
    <Grid Key="detail_grid1" HideGroup4Editing="false" Caption="拣货单明细2" UsePage="false" PageRowCount="6"
    ShowRowHead="true" X="0" Y="0">
      <GridColumnCollection>
        <GridColumn Key="C5" Caption="仓库" Width="75px" ColumnExpand="false" ColumnType="Fix">
        </GridColumn>
        <GridColumn Key="C6" Caption="拣货员" Width="75px" ColumnExpand="false" ColumnType="Fix">
        </GridColumn>
        <GridColumn Key="C7" Caption="数量" Width="75px" ColumnExpand="false" ColumnType="Fix">
        </GridColumn>
      </GridColumnCollection>
      <GridRowCollection>
        <GridRow Key="R5" RowType="Detail" RowHeight="30" TableKey="WM_PickDetail2" GroupKey="">
          <GridCell Key="Pick_Warehouse" Caption="仓库" CellType="Dict" ItemKey="Warehouse">
            <DataBinding ColumnKey="Pick_Warehouse"/>
          </GridCell>
          <GridCell Key="Picker" Caption="拣货员" CellType="TextEditor">
            <DataBinding ColumnKey="Picker"/>
          </GridCell>
          <GridCell Key="Pick_Amount" Caption="数量" CellType="NumberEditor">
            <DataBinding ColumnKey="Pick_Amount"/>
          </GridCell>
        </GridRow>
      </GridRowCollection>
    </Grid>
  </GridLayoutPanel>
</SubDetail>

```

6.6.3: 容器(Container)

```

<Container Style="Style" IsDefault="" DefaultFormKey="DefaultFormKey" FormulaFormKey="FormulaFormKey" CustomViewPos="">
</Container>

```

- Style - 样式，取值为栈式(Stack)和选项卡(Tab)。默认为Tab。
- DefaultFormKey - 默认表单对象的标识，默认为空。
- IsDefault - 是否默认容器，默认为True。

6.6.4: 流程图

```

<BPMGraph ViewTag="Classic" TableKey="" ProcessKey="" ProcessVer="">
</BPMGraph>

```

- ViewTag - 视图标记，用于区分流程图的实现；
- TableKey - 关联的流程执行记录表标识；
- ProcessKey - 流程标识，表达式；
- ProcessVer - 流程版本，表达式；

6.6.5: 附件组件 (Attachment)

```
<Attachment Impl="">  
</Attachment>
```

- Impl 为实现标志，默认为空，用于定义用什么方式实现附件组件，二次开发用属性。

在Impl没有定义的情况下，附件组件使用系统默认实现，具体参见附件管理部分。

6.6.6: 定制组件 (Custom)

用于用户自定义一个组件。

```
<Custom Tag="" UserData="" />
```

- Tag 定制组件的标记，界面处理程序根据这个标志来生成相应的自定义组件；其中 Template、Rights为系统保留标记，分别用于模板和权限设置；
- UserData 定制组件的用户数据，用于特定的配置中传递自定义的数据；

第 7 章 表单逻辑处理

目录

7.1. 数据逻辑	66
7.2. 计算值	68
7.3. 检查规则	71
7.4. 可见性计算	73
7.5. 可用性计算	74

表单逻辑处理包括表单的数据规则和事件处理。

7.1:数据逻辑

7.1.1:编号

单据编号由前缀+时间标识+序号三个部分组成。

- 前缀通过数据对象中的NO属性定义；
- 时间标识为表单的发生时间；
- 序号为当前发生的计数值。

示例1：定义一个以“STIN”为前缀的单据编号。

单据编号 STIN20150715000003

源代码如下：

```
<DataSource>
  <DataObject Key="" Caption="" PrimaryTableKey="" PrimaryType="Entity"
    SecondaryType="Normal" NoPrefix="STIN">
    <TableCollection>
      <Table Key="" Caption="" DBTableName="" TableMode="Head"
        SourceType="Table" Persist="True">
        <Column Key="NO" Caption="单据编号" Persist="True" DataType="Varchar" Length="200"
          DBColumnName="NO"/>
        </Table>
      </TableCollection>
    </DataObject>
</DataSource>
<Body>
  <Block>
    <Label Key="L_NO" Caption="单据编号" X="0" Y="4"/>
    <TextEditor Key="NO" Caption="单据编号" X="1" Y="4">
      <DataBinding TableKey="" ColumnKey="NO"> </DataBinding>
    </TextEditor>
  </Block>
</Body>
```

7.1.2:系统字段值

- CREATETIME 创建时间；
- MODIFYTIME 修改时间；
- CREATOR 创建人；
- MODIFIER 修改人。

示例1：定义表单的创建时间。

创建时间 2015-07-16 10:40:09

源代码如下：

```
<DataSource>
  <DataObject Key="" Caption="" PrimaryTableKey="" PrimaryType="Entity"
    SecondaryType="Normal">
    <TableCollection>
      <Table Key="" Caption="" DBTableName="" TableMode="Head"
        SourceType="Table" Persist="True">
        <<Column Key="CREATETIME" Caption="单据日期" Persist="True" DataType="DateTime" DBColumnName="CREATETIME"/>
      </Table>
    </TableCollection>
  </DataObject>
</DataSource>
<Body>
  <Block>
    <Label Key="L_CREATETIME" Caption="单据日期" X="0" Y="1" Visible="true"/>
    <DatePicker Key="CREATETIME" Caption="单据日期" X="1" Y="1" Enable="!ReadOnly()" OnlyDate="True" Visible="true">
      <DataBinding TableKey="" ColumnKey="CREATETIME">
    </DataBinding>
    </DatePicker>
  </Block>
</Body>
```

示例2：定义表单的创建人。

创建人 Admin 系统管理员

源代码如下：

```
<DataSource>
  <DataObject Key="" Caption="" PrimaryTableKey="" PrimaryType="Entity"
    SecondaryType="Normal">
    <TableCollection>
      <Table Key="" Caption="" DBTableName="" TableMode="Head"
        SourceType="Table" Persist="True">
        <Column Key="CREATOR" Caption="创建人" Persist="True" DataType="Long" DBColumnName="CREATOR"/>
      </Table>
    </TableCollection>
  </DataObject>
</DataSource>
<Body>
  <Block>
    <Block>
      <Label Key="L_CREATOR" Caption="创建人" X="0" Y="8" Visible="true"/>
      <Dict Key="CREATOR" ItemKey="Operator" Caption="创建人" X="1" Y="8">
        <DataBinding TableKey="" ColumnKey="CREATOR">
      </DataBinding>
      </Dict>
    </Block>
  </Block>
</Body>
```

7.1.3: 序号

通过在DataTable中定义Sequence，保证数据被修改或表格插入行时，数据按照顺序加载。

示例1：定义明细表的序号。

	地区	计量单位	规格	数量
1				200.00
2				150.00

源代码如下：

```
<Table Key="" Caption="" DBTableName="" TableMode="Detail" SourceType="Table" Persist="True">
  <Column Key="OID" Caption="对象标识" Persist="True" DataType="Long" DBColumnName="OID" />
  <Column Key="POID" Caption="父对象标识" Persist="True" DataType="Long" DBColumnName="POID"/>
  <Column Key="SOID" Caption="主对象标识" Persist="True" DataType="Long" DBColumnName="SOID"/>
  <Column Key="VERID" Caption="对象版本" Persist="True" DataType="Integer" DBColumnName="VERID"/>
  <Column Key="DVERID" Caption="对象明细版本" Persist="True" DataType="Integer" DBColumnName="DVERID"/>
  <Column Key="Sequence" Caption="序号" Persist="True" DBColumnName="Sequence" DataType="Integer"/>
  <Column Key="Area" Caption="地区" Persist="True" DataType="Long" IsPrimary="True"/>
  <Column Key="Unit" Caption="计量单位" Persist="True" DataType="Long" IsPrimary="True"/>
  <Column Key="Size" Caption="规格" Persist="True" DataType="Integer"/>
  <Column Key="Amount" Caption="数量" Persist="True" DataType="Numeric" Precision="16" Scale="2"/>
</Table>
```

7.2: 计算值

表单中的计算值包括默认值和表达式默认值(后面统称为值计算)。

示例1：定义一个计算值为默认值5。

默认值

源代码如下：

```
<Label Key="LabelNb" Caption="默认值" X="0" Y="2" XSpan="1" YSpan="1"> </Label>
<NumberEditor Key="Number" Caption="默认值" X="1" Y="2">
  <DataBinding DefaultValue="5"/>
</NumberEditor>
```

示例2：定义表达式默认值，如数值2等于数值1乘以5，数值3等于数值2除以10，数值4等于数值2与数值3的和。

数值1 数值2 数值3 数值4

源代码如下：

```
<NumberEditor Key="Number1" Caption="数值1" X="1" Y="0">
  <DataBinding Key="Number1" CheckRule="Number1>10"/>
</NumberEditor>
<Label Key="LabelNumber2" Caption="数值2" X="2" Y="0" XSpan="1" YSpan="1"> </Label>
<NumberEditor Key="Number2" Caption="数值2" X="3" Y="0">
  <DataBinding DefaultFormulaValue="Number1*5"/>
</NumberEditor>
<Label Key="LabelNumber3" Caption="数值3" X="4" Y="0" XSpan="1" YSpan="1"> </Label>
<NumberEditor Key="Number3" Caption="数值3" X="5" Y="0">
  <DataBinding Key="Number3" DefaultFormulaValue="Number2/10"/>
</NumberEditor>
<Label Key="LabelNumber4" Caption="数值4" X="6" Y="0" XSpan="1" YSpan="1"> </Label>
<NumberEditor Key="Number4" Caption="数值4" X="7" Y="0">
  <DataBinding Key="Number4" DefaultFormulaValue="Number1+Number2"/>
</NumberEditor>
```

示例3：根据头控件之间进行值计算。

头控件

速度 时间 路程

源代码如下：

```
<Label Key="LabelVelocity" Caption="速度" X="0" Y="1" XSpan="1" YSpan="1"> </Label>
```

```
<NumberEditor Key="Velocity" Caption="速度" X="1" Y="1">
  <DataBinding Key="Velocity"/>
</NumberEditor>
<Label Key="LabelTime" Caption="时间" X="2" Y="1" XSpan="1" YSpan="1"> </Label>
<NumberEditor Key="Time" Caption="时间" X="3" Y="1">
  <DataBinding Key="Time"/>
</NumberEditor>
<Label Key="LabelSpace" Caption="路程" X="4" Y="1" XSpan="1" YSpan="1"> </Label>
<NumberEditor Key="Space" Caption="路程" X="5" Y="1">
  <DataBinding DefaultFormulaValue="Velocity*Time"/>
</NumberEditor>
```

示例4：明细值之间进行值计算。

明细表				
		速度	时间	路程
1		2.00	3.00	6.00
2		3.00	4.00	12.00
3		4.00	5.00	20.00
4				0.00
5	总计	9.00	12.00	38.00

源代码如下：

```
<Grid Key="Grid" Caption="明细表" Enable="!ReadOnly()" ShowRowHead="True"
  PageRowCount="5" PageLoadType="DB">
  <GridColumnCollection>
    <GridColumn Key="c0" Caption="" Width="90px" Sortable="true"
      Visible="true" ColumnType="Detail"> </GridColumn>
    <GridColumn Key="c1" Caption="速度" Width="90px" Sortable="true"
      Visible="true" ColumnType="Detail"> </GridColumn>
    <GridColumn Key="c2" Caption="时间" Width="90px" Sortable="true"
      Visible="true" ColumnType="Detail"> </GridColumn>
    <GridColumn Key="c3" Caption="路程" Width="90px" Sortable="true"
      Visible="true" ColumnType="Detail"> </GridColumn>
  </GridColumnCollection>
  <GridRowCollection>
    <GridRow Key="R1" RowType="Group" GroupKey="Count">
      <GridCell Caption="总计"/>
      <GridCell Key="CountHead0" CellType="NumberEditor">
        <DataBinding DefaultFormulaValue="Sum('v0')"/>
      </GridCell>
      <GridCell Key="CountHead1" CellType="NumberEditor">
        <DataBinding DefaultFormulaValue="Sum('t0')"/>
      </GridCell>
      <GridCell Key="CountHead2" CellType="NumberEditor">
        <DataBinding DefaultFormulaValue="Sum('s0')"/>
      </GridCell>
    </GridRow>
    <GridRow Key="R2" RowType="Detail">
      <GridCell Key="Count" Caption="总计(Count)" CellType="NumberEditor"
        ItemKey="Count" CellGroupType="RowGroup">
        <DataBinding ColumnKey="Count"/>
      </GridCell>
      <GridCell Key="v0" Caption="速度" CellType="NumberEditor" DecScale="0"
        DecPrecision="10">
        <DataBinding ColumnKey="v0"/>
      </GridCell>
      <GridCell Key="t0" Caption="时间" CellType="NumberEditor" DecScale="2"
        DecPrecision="10">
        <DataBinding ColumnKey="t0"/>
      </GridCell>
      <GridCell Key="s0" Caption="路程" CellType="NumberEditor" DecScale="2"
        DecPrecision="10">
        <DataBinding DefaultFormulaValue="v0*t0"/>
      </GridCell>
    </GridRow>
  </GridRowCollection>
</Grid>
```

```
</GridRowCollection>
</Grid>
```

示例5：根据明细值的总量值计算头控件值，如头控值等于明细表某一列数据的总和。

总路程

38.00

明细表

	速度	时间	路程
1	2.00	3.00	6.00
2	3.00	4.00	12.00
3	4.00	5.00	20.00

源代码如下：

```
<Label Key="LabelTotalSpace" Caption="总路程" X="0" Y="2" XSpan="1" YSpan="1"> </Label>
<NumberEditor Key="TotalSpace" Caption="总路程" X="1" Y="2">
  <DataBinding DefaultFormulaValue="Sum(' s0' )" ValueDependency="s0"
  > </DataBinding>
</NumberEditor>

<Grid Key="Grid" Caption="明细表" Enable="!ReadOnly()" ShowRowHead="True"
PageRowCount="5" PageLoadType="DB">
  <GridColumnCollection>
    <GridColumn Key="c0" Caption=" " Width="90px" Sortable="true"
      Visible="true" ColumnType="Detail"> </GridColumn>
    <GridColumn Key="c1" Caption="速度" Width="90px" Sortable="true"
      Visible="true" ColumnType="Detail"> </GridColumn>
    <GridColumn Key="c2" Caption="时间" Width="90px" Sortable="true"
      Visible="true" ColumnType="Detail"> </GridColumn>
    <GridColumn Key="c3" Caption="路程" Width="90px" Sortable="true"
      Visible="true" ColumnType="Detail"> </GridColumn>
  </GridColumnCollection>
  <GridRowCollection>
    <GridRow Key="R1" RowType="Group" GroupKey="Count">
      <GridCell Caption="总计"/>
      <GridCell Key="CountHead0" CellType="NumberEditor">
        <DataBinding DefaultFormulaValue="Sum(' v0' )"/>
      </GridCell>
      <GridCell Key="CountHead1" CellType="NumberEditor">
        <DataBinding DefaultFormulaValue="Sum(' t0' )"/>
      </GridCell>
      <GridCell Key="CountHead2" CellType="NumberEditor">
        <DataBinding DefaultFormulaValue="Sum(' s0' )"/>
      </GridCell>
    </GridRow>
    <GridRow Key="R2" RowType="Detail">
      <GridCell Key="Count" Caption="总计 (Count)" CellType="NumberEditor">
        ItemKey="Count" CellGroupType="RowGroup">
          <DataBinding ColumnKey="Count"/>
        </GridCell>
      <GridCell Key="v0" Caption="速度" CellType="NumberEditor" DecScale="0"
        DecPrecision="10">
          <DataBinding ColumnKey="v0"/>
        </GridCell>
      <GridCell Key="t0" Caption="时间" CellType="NumberEditor" DecScale="2"
        DecPrecision="10">
          <DataBinding ColumnKey="t0"/>
        </GridCell>
      <GridCell Key="s0" Caption="路程" CellType="NumberEditor" DecScale="2"
        DecPrecision="10">
          <DataBinding DefaultFormulaValue="v0*t0"/>
        </GridCell>
    </GridRow>
  </GridRowCollection>
</Grid>
```

7.2.1:DataBinding属性

- TableKey - 数据对象表的标识。
- ColumnKey - 数据对象列的标识。
- Required - 是否必填项，默认值为False；
- ValueChanged - 值改变事件；
- DefaultValue - 默认值；
- DefaultFormulaValue - 表达式默认值；
- ValueDependency - 值关联域集合；（值表达式中依赖关系无法解析时使用）
- CheckRule - 检查规则；
- CheckRuleDependency - 检查规则关联域集合；（依赖关系无法解析时使用）
- ErrorInfo - 错误描述；

7.3:检查规则

检查规则根据定义分为必填项、自定义的检查规则，其中自定义的检查规则通过组件中的CheckRule定义或者通过表单的界面检查规则集合定义。

明细值的检查规则只能依赖头控件的值和当前表格的单元格。

示例1：控件的检查规则，如定义一个检查规则，输入的数值需大于10，否则将显示红色标志提示。

数值

源代码如下：

```
<Label Key="LabelNum" Caption="数值" X="0" Y="0" XSpan="1" YSpan="1"> </Label>
<NumberEditor Key="Num" Caption="数值" X="1" Y="0">
  <DataBinding Key="Num" CheckRule="Num>10"/>
</NumberEditor>
```

示例2：定义表格行的检查规则，如检查每一行速度是否大于50。

明细表				
		速度	时间	路程
1		20.00	3.00	60.00
2		60.00	3.00	180.00

源代码如下：

```
Grid Key="Grid" Caption="明细表" Enable="!ReadOnly()" ShowRowHead="True"
PageRowCount="5" PageLoadType="DB">
  <GridColumnCollection>
    <GridColumn Key="c0" Caption="" Width="90px" Sortable="true"
      Visible="true" ColumnType="Detail"> </GridColumn>
    <GridColumn Key="c1" Caption="速度" Width="90px" Sortable="true"
      Visible="true" ColumnType="Detail"> </GridColumn>
    <GridColumn Key="c2" Caption="时间" Width="90px" Sortable="true"
      Visible="true" ColumnType="Detail"> </GridColumn>
    <GridColumn Key="c3" Caption="路程" Width="90px" Sortable="true"
      Visible="true" ColumnType="Detail"> </GridColumn>
  </GridColumnCollection>
</Grid>
```

```
</GridColumnCollection>
<GridRowCollection>
  <GridRow Key="R1" RowType="Detail">
    <UICheckRuleCollection>
      <UICheckRule Description="" ErrorInfo="速度需大于50">
        <![CDATA[
          v0>50;
        ]]>
      </UICheckRule>
    </UICheckRuleCollection>
    <GridCell Key="Count" Caption="总计 (Count)"
      CellType="NumberEditor" ItemKey="Count"
      CellGroupType="RowGroup">
      <DataBinding ColumnKey="Count"/>
    </GridCell>
    <GridCell Key="v0" Caption="速度" CellType="NumberEditor"
      DecScale="0" DecPrecision="10">
      <DataBinding ColumnKey="v0"/>
    </GridCell>
    <GridCell Key="t0" Caption="时间" CellType="NumberEditor"
      DecScale="2" DecPrecision="10">
      <DataBinding ColumnKey="t0"/>
    </GridCell>
    <GridCell Key="s0" Caption="路程" CellType="NumberEditor"
      DecScale="2" DecPrecision="10">
      <DataBinding DefaultFormulaValue="v0*t0"/>
    </GridCell>
  </GridRow>
</GridRowCollection>
</Grid>
```

示例3：定义表单的检查规则，如检查路程是否大于100。

表单检查.....

保存

取消

头控件

速度

0.00

时间

0.00

路程

0.00

保存

取消

头控件

速度

20.00

时间

6.00

路程

120.00

源代码如下：

```
<UICheckRuleCollection>
  <UICheckRule Description="" ErrorInfo="表单检查.....">
    <![CDATA[
      Space>100;
    ]]>
  </UICheckRule>
</UICheckRuleCollection>

<Body>
```

```
<Block>
  <FlexFlowLayoutPanel Key="main">
    <ToolBar Key="main_toolbar" Height="pref" IsDefault="True"/>
    <GridLayoutPanel Key="GridLayoutPanel4" Caption="头控件" X="0" Y="3" Padding="10px">
      <RowDefCollection RowHeight="30" RowGap="5">
        <RowDef/>
        <RowDef/>
        <RowDef/>
      </RowDefCollection>
      <ColumnDefCollection ColumnGap="10">
        <ColumnDef Width="50px"/>
        <ColumnDef Width="100px"/>
        <ColumnDef Width="50px"/>
        <ColumnDef Width="100px"/>
        <ColumnDef Width="50px"/>
        <ColumnDef Width="100px"/>
      </ColumnDefCollection>
      <Label Key="LabelPanel1" Caption="头控件" X="0" Y="0" XSpan="1" YSpan="1">
        <Format ForeColor="#0000FF"/>
      </Label>
      <Label Key="LabelVelocity" Caption="速度" X="0" Y="1" XSpan="1" YSpan="1"> </Label>
      <NumberEditor Key="Velocity" Caption="速度" X="1" Y="1">
        <DataBinding Key="Velocity"/>
      </NumberEditor>
      <Label Key="LabelTime" Caption="时间" X="2" Y="1" XSpan="1" YSpan="1"> </Label>
      <NumberEditor Key="Time" Caption="时间" X="3" Y="1">
        <DataBinding Key="Time"/>
      </NumberEditor>
      <Label Key="LabelSpace" Caption="路程" X="4" Y="1" XSpan="1" YSpan="1"> </Label>
      <NumberEditor Key="Space" Caption="路程" X="5" Y="1">
        <DataBinding DefaultFormulaValue="Velocity*Time"/>
      </NumberEditor>
    </GridLayoutPanel>
  </FlexFlowLayoutPanel>
</Block>
</Body>
```

7.4: 可见性计算

可见性计算用于组件的可见性以及表格组件(列表视图和网格)中列的可见性。在以下情形下需要计算可见性:

示例1: 当数值1和数值2相等时, 数值3可见。

可见性

数值1

2.00

数值2

3.00

数值1

2.00

数值2

2.00

数值3

源代码如下:

```
<Label Key="Visible" Caption="可见性" X="0" Y="3"> </Label>
<Label Key="LabelNumNum1" Caption="数值1" X="0" Y="4"/>
<NumberEditor Key="NumNum1" Caption="" X="1" Y="4">
</NumberEditor>
<Label Key="LabelNumNum2" Caption="数值2" X="2" Y="4"/>
<NumberEditor Key="NumNum2" Caption="" X="3" Y="4">
</NumberEditor>
<Label Key="LabelNum3" Caption="数值3" X="4" Y="4"/>
<NumberEditor Key="Num3" Caption="" BuddyKey="LabelNum3" X="5" Y="4" Visible="NumNum1==NumNum2">
</NumberEditor>
```

7.5: 可用性计算

可用性计算用于计算组件的可用性、表格列的可用性以及单元格的可用性。

示例1：当数值1不小于数值2时，数值3可用，即可编辑数值。

可用性

数值1

4.00

数值2

3.00

数值3

1,234.00

源代码如下：

```
<Label Key="Enable" Caption="可用性" X="0" Y="5"> </Label>
<Label Key="LabelNum1" Caption="数值1" X="0" Y="6"/>
<NumberEditor Key="NumNum1" Caption="" X="1" Y="6">
</NumberEditor>
<Label Key="LabelNumNum2" Caption="数值2" X="2" Y="6"/>
<NumberEditor Key="NumNum2" Caption="" X="3" Y="6">
</NumberEditor>
<Label Key="LabelNum3" Caption="数值3" X="4" Y="6"/>
<NumberEditor Key="Num3" Caption="" X="5" Y="6" Enable="NumNum1>=NumNum2">
</NumberEditor>
```



注意

可用性值的计算规则如下：

- 如果计算对象自身具体可用性定义以自身的计算结果作为计算对象的可用性；
- 如果计算对象自身没有可用性定义并且表单定义了状态字段，先取得表单的状态值，取得该状态值对应的编辑状态，如果状态值的编辑状态为False，计算对象的可用性为False；否则根据表单的编辑状态来确定计算对象的可用性，如果表单的状态为新增或编辑状态，那么可用性为True，否则为False；
- 如果计算对象自身没有可用性定义并且表单没有定义状态字段，则计算对象的可用性根据表单的状态而定，如果表单为新增或编辑状态，那么可用性为True，否则为False。

第 8 章 Android平台应用配置

目录

8.1. 移动应用组成结构	75
8.2. 定义AndroidDef.xml 文件	75
8.3. 尺寸定义	77
8.4. 移动布局面板	77
8.5. 移动基本组件	80
8.6. 综合示例	82

Yigo软件平台为用户提供了三个不同的客户环境：传统客户端、web浏览器、移动Android客户端。本章主要介绍如何配置Android平台下的应用，以及与PC端的不同部分。

在功能上移动Android客户端上不支持数据迁移、数据映射。

8.1:移动应用组成结构

首先，移动应用需要包含以下目录结构：

```
<Root>
- Solution.xml 应用的基本属性及可用工程列表
- setting.xml 应用设置文件
CommonDef.xml - 应用公共定义，具体定义详见项目中的定义
AndroidDef.xml - 安卓手机及平板的定义
<Resource>
...
<Data> - 应用的数据存储目录(一般为附件)
...
<Project>
...
```

添加了AndroidDef.xml文件的定义，其他与PC端基本一致。

8.2:定义AndroidDef.xml 文件

第二步，定义AndroidDef.xml文件。

8.2.1:基本格式

```
<?xml version="1.0" encoding="utf-8"?>
<AndroidDef>
  <ALaunchDef LaunchForm="" Style="" >
  </ALaunchDef>

  <AMenuDef>
    <AMenuHead Icon="" Title="" />
    <AMenuItemCollection>
      <AMenuItem Icon="" Title="" >
        <![CDATA[]]>
      </AMenuItem>
    </AMenuItemCollection>
    <ASubMenuCollection>
      <ASubMenuDef Title="" >
        <AMenuItemCollection>
          <AMenuItem Icon="" Title="" >
            <![CDATA[]]>
          </AMenuItem>
        </AMenuItemCollection>
      </ASubMenuDef>
    </ASubMenuCollection>
  </AMenuDef>
```



```

<AppCacheDataCollection>
<AppCacheData DataObjKey="">
  <OnLoad>
    <![CDATA[ ]]>
  </OnLoad>
  <OnLoadAfterLogin>
    <![CDATA[ ]]>
  </OnLoadAfterLogin>
</AppCacheData>
</AppCacheDataCollection>

</AndroidDef>

```

8.2.2: 属性说明

8.2.2.1: ALaunchDef 主容器定义:

- LaunchForm 启动表单，可启动多张表单，按顺序用英文逗号分割。
- style 安卓端的主容器选项，类似Container 的概念，。取值为 Tab（默认）、Stack栈式。

Tab 同pc端的tab容器

Stack 同pc端的Stack容器

8.2.2.2: AMenuDef 全局侧滑菜单定义

- AMenuHead 菜单头定义，
- AMenuItem 菜单项定义
- ASubMenuDef 子菜单定义

全局侧滑菜单实例:

```

<AMenuDef>
  <AMenuHead Title="FirstMenu" />
  <AMenuItemCollection>
    <AMenuItem Title="Test01" >
      <![CDATA[]]>
    </AMenuItem>
    <AMenuItem Title="Test02" >
      <![CDATA[]]>
    </AMenuItem>
  </AMenuItemCollection>
  <ASubMenuCollection>
    <ASubMenuDef Title="SecondMenu" >
      <AMenuItemCollection>
        <AMenuItem Title="Test001" >
          <![CDATA[]]>
        </AMenuItem>
        <AMenuItem Title="Test002" >
          <![CDATA[]]>
        </AMenuItem>
        <AMenuItem Title="Test003" >
          <![CDATA[]]>
        </AMenuItem>
      </AMenuItemCollection>
    </ASubMenuDef>
  </ASubMenuCollection>
</AMenuDef>

```

8.2.2.3: AppCacheData 在安卓端内存中保持的数据

- DataObjKey 保持的DataObject key
- OnLoad 初始化函数
- OnLoadAfterLogin 登陆后初始化函数

8.3: 尺寸定义

作为移动平台，这部分说明尺寸的类型和定义，有以下几类：

- 固定像素，为PC像素，在安卓平台是通过英寸换算。定义形式为以px结尾的定义，比如10px，表示10像素。
- 固定Android平台像素，定义形式为以apx结尾的定义，比如10apx，表示10个Android像素；不推荐使用，保证底层功能完整。
- 固定逻辑单位，为Android所使用，称为相对像素。以dp结尾的定义，比如10dp；推荐使用。默认移动端（手机）宽度为320dp。
- 比例，定义形式为以%结尾的定义，比如50%；
- 内容自适应，定义形式为pref；
- 自动分配，定义形式为auto；

8.4: 移动布局面板

8.4.1: 面板

在移动端中，支持的布局面板包括以下几种选项卡面板、 网格布局面板、 线性布局面板、 流式表布局面板。具体定义可以参考PC端的布局面板说明。

- 网格布局面板：在移动端使用方便，推荐使用。定义同PC端一致。
- 选项卡面板：定义同PC端基本一致。但需要注意的是不建议把所有配置都封装在一个Tab中，不利于后期维护。
- 流式表布局面板：推荐使用，以表格形式来定义面板的区域。定义同PC端一致。
- 线形布局面板：（不过，一般更推荐有安卓开发经验的进行使用。）移动应用特有的布局面板LinearLayoutPanel。其以横向或纵向按照线性方式进行布局，从左到右或者从上到下。组件的位置按照其定义的宽或高进行定位；如下图所示：



线性布局面板的定义为：

```
<LinearLayoutPanel Key="" Orientation="VERTICAL">  
...  
</LinearLayoutPanel>
```

- Orientation 线性布局的方向，取值为VERTICAL和HORIZONTAL，默认值为VERTICAL。

- LinearLayoutPanel的尺寸定义只能使用pref自适应、Auto自动分配、固定值px和dp，不能使用比例值%，而是使用weight权重值。

权重的概念

线性布局中的一个权重的概念weight。我们用一个例子来说明：

```
<LinearLayout Key="content" Orientation="VERTICAL">
  <LinearItem Height="pref" Key="orderFormContent" Width="auto"/>
  <LinearItem Height="40dp" Key="makeOrder" Width="auto"/>
  <LinearItem Height="40dp" Key="PayOrder" Width="auto"/>
  <LinearItem Height="0dp" Key="orderFormGoodsLisy" Width="auto" Weight="1"/>
</LinearLayout>
```

说明：

orderFormContent组件高度自适应，宽度自动分配。

makeOrder和PayOrder组件高度为40dp，宽度自动分配。

orderFormGoodsLisy组件宽度自动分配，高度Height="0dp"，Weight="1"权重为1，所以orderFormGoodsLisy组件权重就是100%。

```
<LinearLayout Key="content" Orientation="VERTICAL">
  <LinearItem Height="pref" Key="orderFormContent" Width="auto"/>
  <LinearItem Height="40dp" Key="makeOrder" Width="auto"/>
  <LinearItem Height="40dp" Key="PayOrder" Width="auto"/>
  <LinearItem Height="0dp" Key="orderFormGoodsLisy1" Width="auto" Weight="1"/>
  <LinearItem Height="0dp" Key="orderFormGoodsLisy2" Width="auto" Weight="2"/>
</LinearLayout>
```

有两个组件都使用Weight权重，一个占比例1(大约33%)，另一个占2（大约66%）。

8.4.2: 布局

同PC端相同，移动应用每个面板均可以改变自身的布局。可以定义0个或者多个布局。

推荐定义规则：是将各个基本组件的数据关系、表单逻辑等定义在Pannel中，而不是任何一个LayoutPannel中。把多个布局定义在Layout中。

示例如下：

```
<Body>
  <LayoutCollection>
    <Layout Caption="" Key="all" Media="all">
      <FluidTableLayout Key="content" RowHeight="50">
        <LayoutRowIndex Key="tool" />
        <LayoutRowIndex Key="goodsImg" />
        <LayoutRowIndex Key="goodsName" />
        <LayoutRowIndex Key="goodsPrice" />
        <LayoutRowIndex Key="goodsDetailText" />
        <LayoutColumnCollection>
          <LayoutColumn Width="35%" />
          <LayoutColumn Width="65%" />
        </LayoutColumnCollection>
      </FluidTableLayout>
    </Layout>
  </LayoutCollection>
</StyleCollection>
<Block>
  <Panel Key="content">
    <ToolBar Key="tool" IsDefault="true"/>
    <Label Key="L_goodsImg" Caption="商品图片" BuddyKey="goodsImg"/>
    <Image Key="goodsImg" Source="" SourceType="Data">
      <DataBinding ColumnKey="GoodsImageSrc" TableKey="GoodsDetail"/>
    </Image>
    <Label Key="L_goodsName" Caption="商品名称" BuddyKey="goodsName"/>
    <TextEditor Key="goodsName">
      <DataBinding ColumnKey="GoodsName" TableKey="GoodsDetail"/>
    </TextEditor>
    <Label Key="L_goodsPrice" Caption="商品价格" BuddyKey="goodsPrice"/>
```

```
<NumberEditor Key="goodsPrice" Scale="2">
  <DataBinding ColumnKey="GoodsPrice" TableKey="GoodsDetail"/>
</NumberEditor>
<Label Key="L_goodsDetailText" Caption="商品详情文本" BuddyKey="goodsDetailText"/>
<TextEditor Key="goodsDetailText">
  <DataBinding ColumnKey="GoodsDetailText" TableKey="GoodsDetail"/>
</TextEditor>
</Panel>
</Block>
</Body>
```

8.4.3: 全局样式

移动端可以定义全局样式，如控件本身定义了私有样式则加载私有样式，否则根据属性Media、Rule加载对应的全局样式。

#目前只支持字体大小，其他功能扩展可根据客户需求添加。

```
<Body>
  <StyleCollection>
    <Style Media="all" Rule="" DensityDpi="">
      <StyleItem Key="" Font_Size=""/>
    </Style>
  </StyleCollection>
</Body>
```

属性说明：

DensityDpi：屏幕密度。默认值为AMEDIUM。取值：

ALOW、AMEDIUM、ATV、AHIGH、A280、AXHIGH、A400、AXXHIGH、A560、AXXXHIGH

尺寸参考：

Nexus 5 (5.0", 1080 × 1920: xxhdpi)	360*640
Nexus 7 (7.0", 1200 × 1920: xhdpi)	600*960
Nexus 4 (4.7", 768 × 1280: xhdpi)	384*640
Nexus 10 (10.1", 2560 × 1600: xhdpi)	1280*800
Nexus 7 (2012) (7.0", 800 × 1280: tvdpi)	601*962
Galaxy Nexus (4.7", 720 × 1280: xhdpi)	360*640
Nexus S (4.0", 480 × 800: hdpi)	320*533
Nexus One (3.7", 480 × 800: hdpi)	320*533
10.1" WXGA (Tablet) (1280 × 800: mdpi)	1280*800
7.0" WSVGA (Tablet) (1024 × 600: mdpi)	1024*600
5.4" FWVGA (480 × 854: mdpi)	480*854
5.1" WVGA (480 × 800: mdpi)	480*800
✓ 4.7" WXGA (1280 × 720: xhdpi)	640*320
4.65" 720p (720 × 1280: xhdpi)	360*640
4.0" WVGA (480 × 800: hdpi)	320*533
3.7" FWVGA slider (480 × 854: hdpi)	320*569
3.7" WVGA (480 × 800: hdpi)	320*533
3.4in WQVGA (240 × 432: ldpi)	320*576
3.3" WQVGA (240 × 400: ldpi)	320*533
3.2" QVGA (ADP2) (320 × 480: mdpi)	320*480
3.2" HVGA slider (ADP1) (320 × 480: mdpi)	320*480
2.7" QVGA slider (240 × 320: ldpi)	320*426.6
2.7" QVGA (240 × 320: ldpi)	320*426.6

8.5: 移动基本组件

这里仅介绍移动端特有的组件以及与PC端有差异的组件，其他组件介绍参考PC基本组件的介绍。

- 字典：字典仅支持最基本的查看字典的功能。如下拉->查看字典（层次），点击->查询字典（链式）；不支持复合字典，动态字典等更高级的功能。
- 表格：表格在移动端不支持编辑功能，其功能只用于展示数据，目前基本等同于ListView的功能。

8.5.1: 导航列表(NavigationList)

导航列表用于移动应用。此组件的功能可以用全局侧滑菜单代替，更推荐使用全局侧滑菜单。

定义如下：

```
<NavigationList Style="" ImageScaleType="">
  <NavigationItemCollection>
    <NavigationItem Key="" Caption="" Icon="" HasSenseTip="">
      <OnClick>
        <![CDATA[ ]]>
      </OnClick>
      <SenseTipSource>
        <![CDATA[ ]]>
      </SenseTipSource>
    </NavigationItem>
  </NavigationItemCollection>
</NavigationList>
```

导航列表属性

- Style 样式，取值为List(列表式)、TileList(平铺式)；默认为TileList；
- ImageScaleType：全局图像缩放类型。默认值：center。取值
MATRIX、FIT_XY、FIT_START、FIT_CENTER、FIT_END、CENTER、CENTER_CROP、CENTER_INSIDE

导航项目属性

- Key - 导航项目的标识；
- Caption - 导航项目的名称；
- Icon - 导航项目图标；
- HasSenseTip - 是否有数据敏感提示；
- SenseTipSource - 提示信息来源；公式计算结果为字符串；
- ImageScaleType：图像缩放类型(私有)

示例：展示了一个导航列表，这个导航中包含了移动办公、移动商城、功能演示/测试。导航列表的展示的样式为列表样式。

```
<NavigationList Key="grid_list">
  <NavigationItemCollection>
    <NavigationItem Key="i5" Caption="商品列表" Icon="s_IconGoodsList.png">
      <![CDATA[show("GoodsList")]]>
    </NavigationItem>
    <NavigationItem Key="i6" Caption="购物车" Icon="s_IconGoodsTrolley.png">
      <![CDATA[show("TrolleyList")]]>
    </NavigationItem>
    <NavigationItem Key="i7" Caption="订单列表" Icon="s_IconOrderForm.png">
      <![CDATA[show("OrderFormList")]]>
    </NavigationItem>
  </NavigationItemCollection>
</NavigationList>
```

效果展示：



8.6: 综合示例

示例1：以一个网上商城为例，我们看看在移动端是如何进行配置的。

第一步：定义AndroidDef.xml配置文件：定义了4个表单：简单控件展示、商品列表、购物车列表、订单列表。

```
<AndroidDef>
  <ALaunchDef Style="tab" LaunchForm="SimpleUIDemo, GoodsList, Trolleylist, OrderFormList"/>
  <AppCacheDataCollection>
    <AppCacheData DataObjKey="DataTrolleyList" authenticcate="true">
      <OnLoad>
        <![CDATA[InitTrolley();]]>
      </OnLoad>
    </AppCacheData>
  </AppCacheDataCollection>
</AndroidDef>
```

启动表单都是在ALaunchDef中定义。通常Style都定义为Tab类型。LaunchForm定义启动的表单。

AppCacheDataCollection中定义安卓内存中的数据定义，这里定义了数据列表初始化。

第二步：定义表单(以订单列表为例)

```
<Form Key="OrderForm" Caption="订单">
  <DataSource RefObjectKey="OrderForm"> </DataSource>
  <OnLoad>
    <![CDATA[if(GetOID()<-1){LoadData();}else{InitOrderFormGoodsList();SetValue("formPrice",parent.GetAllGoodsPrice());}]]>
  </OnLoad>
```

```

<Body>
<LayoutCollection>
<Layout Caption="" Key="con_layout" Media="android">
<LinearLayout Key="content" Orientation="VERTICAL">
<LinearLayout Height="pref" Key="orderFormContent" Width="auto"/>
<LinearLayout Height="40dp" Key="makeOrder" Width="auto"/>
<LinearLayout Height="40dp" Key="PayOrder" Width="auto"/>
<LinearLayout Height="0dp" Key="orderFormGoodsLisy" Width="auto" Weight="1"/>
</LinearLayout>
<FluidTableLayout Key="orderFormContent">
<LayoutRowIndex Key="UserID"/>
<LayoutRowIndex Key="formCode"/>
<LayoutRowIndex Key="address"/>
<LayoutRowIndex Key="needBill"/>
<LayoutRowIndex Key="billKind"/>
<LayoutRowIndex Key="billTitle"/>
<LayoutRowIndex Key="formTime"/>
<LayoutRowIndex Key="formPrice"/>
<LayoutRowIndex Key="formState"/>
<LayoutColumnCollection>
<LayoutColumn Width="120dp"/>
<LayoutColumn Width="200dp"/>
</LayoutColumnCollection>
</FluidTableLayout>
<GridLayout Key="orderFormGoodsLisy">
<!--<LayoutSpan Key="OrderGoodsCode" X="1" XSpan="1" Y="0" YSpan="1" />-->
<LayoutSpan Key="OrderGoodsName" X="1" XSpan="3" Y="0"/>
<LayoutSpan Key="OrderGoodsPrice" X="3" Y="1"/>
<LayoutSpan Key="multiply" X="2" Y="1"/>
<LayoutSpan Key="OrderGoodsCount" X="1" Y="1"/>
<LayoutSpan Key="OrderGoodsImageSrc" X="0" Y="0" YSpan="2"/>
<RowDefCollection>
<RowDef Height="50dp"/>
<RowDef Height="50dp"/>
</RowDefCollection>
<ColumnDefCollection>
<ColumnDef Width="100dp"/>
<ColumnDef Width="90dp"/>
<ColumnDef Width="40dp"/>
<ColumnDef Width="90dp"/>
</ColumnDefCollection>
</GridLayout>
</Layout>
</LayoutCollection>
<Block>
<Panel Key="content">
<ToolBar Key="title" ShowTitle="true" Title="商品订单"/>
<Panel Key="orderFormContent">
<TextEditor Key="UserID" Visible="false">
<DataBinding ColumnKey="UserID" DefaultFormulaValue="GetOperator()" TableKey="OrderForm"/>
</TextEditor>
<Label Key="l_formCode" BuddyKey="formCode" Visible="ReadOnly()" VAlign="CENTER" HAlign="right">
<Format BackColor="" ForeColor="">
<Font Bold="" Italic="" Name="订单号" Size=""/>
</Format>
</Label>
<TextEditor Key="formCode" Visible="ReadOnly()" Enable="!ReadOnly()">
<DataBinding ColumnKey="OID" TableKey="OrderForm"/>
</TextEditor>
<Label Key="l_address" BuddyKey="address" VAlign="CENTER" HAlign="right">
<Format BackColor="" ForeColor="">
<Font Bold="" Italic="" Name="送货地址" Size=""/>
</Format>
</Label>
<TextEditor Key="address" Enable="!ReadOnly()">
<DataBinding ColumnKey="SendAddress" TableKey="OrderForm"/>
</TextEditor>
<Label Key="l_needBill" BuddyKey="needBill" VAlign="CENTER" HAlign="right">
<Format BackColor="" ForeColor="">
<Font Bold="" Italic="" Name="是否需要发票" Size=""/>
</Format>
</Label>
<CheckBox Key="needBill" Enable="!ReadOnly()">
<DataBinding ColumnKey="NeedBill" TableKey="OrderForm"/>
</CheckBox>

<Label Key="l_billTitle" BuddyKey="billTitle" VAlign="CENTER" HAlign="right">
<Format BackColor="" ForeColor="">
<Font Bold="" Italic="" Name="发票抬头" Size=""/>
</Format>
</Label>
<TextEditor Key="billTitle" Enable="needBill">
<DataBinding ColumnKey="BillTitle" TableKey="OrderForm"/>

```



```

</TextView>

<Label Key="l_formTime" BuddyKey="formTime" Visible="ReadOnly()" Valign="CENTER" HAlign="right">
  <Format BackColor="" ForeColor="">
    <Font Bold="" Italic="" Name="订单生成时间" Size=""/>
  </Format>
</Label>
<DatePicker Key="formTime" Format="yyyy-MM-dd HH:mm" OnlyDate="false" Visible="ReadOnly()" Enable="false">
  <DataBinding ColumnKey="OrderFormTime" TableKey="OrderForm"/>
</DatePicker>
<Label Key="l_formPrice" BuddyKey="formPrice" Valign="CENTER" HAlign="right">
  <Format BackColor="" ForeColor="">
    <Font Bold="" Italic="" Name="订单总价格" Size=""/>
  </Format>
</Label>
<NumberEditor Key="formPrice" Scale="2" Enable="false">
  <DataBinding ColumnKey="OrderFormPrice" TableKey="OrderForm"/>
  <Format BackColor="" ForeColor="red">
    <Font Bold="" Italic="" Name="" Size=""/>
  </Format>
</NumberEditor>
<Label Key="l_formState" BuddyKey="formState" Visible="ReadOnly()" Valign="CENTER" HAlign="right">
  <Format BackColor="" ForeColor="">
    <Font Bold="" Italic="" Name="订单状态" Size=""/>
  </Format>
</Label>
<ComboBox Key="formState" Visible="ReadOnly()" Enable="false">
  <Item Caption="未支付" Key="" Value="0"/>
  <Item Caption="已支付" Key="" Value="1"/>
  <DataBinding ColumnKey="OrderFormState" TableKey="OrderForm" DefaultValue="0"/>
</ComboBox>
</Panel>

<Button Key="makeOrder" Visible="!ReadOnly()">
  <OnClick>
    <![CDATA[
      if(GetOID()==-1){
        SetValue("formTime",GetTime());
        SaveOrderForm();
        parent.ShowTrolleyList();
      }
    ]]>
  </OnClick>
  <Format BackColor="" ForeColor="">
    <Font Bold="" Italic="" Name="提交订单" Size=""/>
  </Format>
</Button>
<Button Key="PayOrder" Visible="ReadOnly()">
  <OnClick>
    <![CDATA[ ]]>
  </OnClick>
  <Format BackColor="" ForeColor="">
    <Font Bold="" Italic="" Name="付款" Size=""/>
  </Format>
</Button>

<ListView Key="orderFormGoodsLisy" TableKey="OrderFormGoodsLisp">
  <ListViewColumnCollection>
    <ListViewColumn Key="OrderGoodsCode" Caption="商品号" ColumnType="TextEditor" DataColumnKey="OrderGoodsCode"
DefaultFormulaValue="" DefaultValue="" ValueChange="">
      <Format BackColor="" ForeColor="">
        <Font Bold="" Italic="" Name="" Size="13"/>
      </Format>
    </ListViewColumn>
    <ListViewColumn Key="OrderGoodsName" Caption="商品名称" ColumnType="Label" DataColumnKey="OrderGoodsName"
DefaultFormulaValue="" DefaultValue="" ValueChange="">
      <Format BackColor="" ForeColor="red">
        <Font Bold="" Italic="" Name="" Size=""/>
      </Format>
    </ListViewColumn>
    <ListViewColumn Key="OrderGoodsPrice" Caption="商品价格" ColumnType="NumberEditor" Scale="2"
DataColumnKey="OrderGoodsPrice" DefaultFormulaValue="" DefaultValue="" ValueChange="">
      <Format BackColor="" ForeColor="red">
        <Font Bold="" Italic="" Name="" Size=""/>
      </Format>
    </ListViewColumn>
    <ListViewColumn Key="multiply" Caption="乘号" ColumnType="Label">
      <Format BackColor="" ForeColor="">
        <Font Bold="true" Italic="true" Name="*" Size="15"/>
      </Format>
    </ListViewColumn>

    <ListViewColumn Key="OrderGoodsCount" Caption="商品数量" ColumnType="NumberEditor" Scale="0"
DataColumnKey="OrderGoodsCount" DefaultFormulaValue="" DefaultValue="" ValueChange="">
      <Format BackColor="" ForeColor="red">
        <Font Bold="" Italic="" Name="" Size=""/>
      </Format>
    </ListViewColumn>
    <ListViewColumn Key="OrderGoodsImageSrc" Caption="商品图片" ColumnType="Image" DataColumnKey="OrderGoodsImageSrc"
DefaultFormulaValue="" DefaultValue="" ValueChange="">
      <Format BackColor="" ForeColor="red">
        <Font Bold="" Italic="" Name="" Size=""/>
      </Format>
    </ListViewColumn>
  </ListViewColumnCollection>

```

```

<RowClick>
<![CDATA[]]>
</RowClick>
</ListView>
</Panel>
</Block>
</Body>
</Form>

```

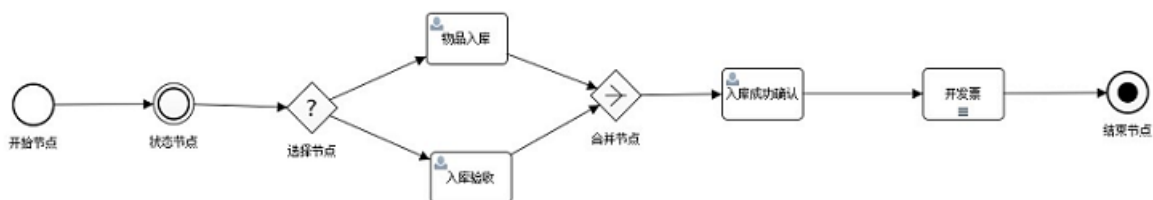
如之前介绍的，表单在定义基本组件的时候，先定义布局Layout，再在Panel中定义各个基本组件的数据关系、表单逻辑等。

配置文件编写基本同PC端类似。

效果展示：

(暂未截图)

示例2：在移动端定义一个 workflow，如下 workflow 图。



```

<Process Caption="入库审批" Key="StockInAudit">
  <Begin Caption="开始节点" ID="1" Key="Begin">
    <TransitionCollection>
      <SequenceFlow Condition="" ID="9" Key="BegintoState" TargetNodeKey="State">
        <TransitionGraphic/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic Height="50" Width="80" X="57" Y="150"/>
  </Begin>
  <State Caption="状态节点" ID="2" Key="State">
    <TransitionCollection>
      <SequenceFlow Condition="" ID="10" Key="StatetoDecision" TargetNodeKey="Decision">
        <TransitionGraphic/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic Height="50" Width="80" X="192" Y="155"/>
  </State>
  <Decision Caption="选择节点" Condition="True" ID="3" Key="Decision">
    <TransitionCollection>
      <SequenceFlow ID="11" Key="DecisionontoTask01" TargetNodeKey="Task01">
        <TransitionGraphic/>
      </SequenceFlow>
      <SequenceFlow ID="12" Key="DecisionontoTask02" TargetNodeKey="Task02">
        <TransitionGraphic/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic Height="50" Width="80" X="328" Y="159"/>
  </Decision>
  <UserTask Caption="入库验收" ID="4" Key="Task01">
    <TransitionCollection>
      <SequenceFlow Condition="" ID="14" Key="Task01toJoin" TargetNodeKey="Join">
        <TransitionGraphic/>
      </SequenceFlow>
    </TransitionCollection>
    <OperationCollection>
      <Operation Action="CommitWorkitem(-1,1,&quot;&quot;)" Caption="提交" Key="op1"/>
    </OperationCollection>
    <AssistanceCollection/>
    <ParticipatorCollection>
      <Query>
        <![CDATA[SELECT OID FROM GP_Operator]]>
      </Query>
    </ParticipatorCollection>
    <TimerItemCollection>
      <TimerAutoPass userInfo="这是一个华丽的自动提交任务!" Key="AutoPass" Peroid="10s"/>
    </TimerItemCollection>
  </UserTask>
  <JoinNode Caption="合并节点" ID="5" Key="Join">
    <TransitionCollection>
      <SequenceFlow Condition="" ID="15" Key="JoinontoConfirmation" TargetNodeKey="Confirmation">
        <TransitionGraphic/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic Height="50" Width="80" X="463" Y="163"/>
  </JoinNode>
  <ConfirmationNode Caption="入库成功确认" ID="6" Key="Confirmation">
    <TransitionCollection>
      <SequenceFlow Condition="" ID="16" Key="ConfirmationontoInvoice" TargetNodeKey="Invoice">
        <TransitionGraphic/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic Height="50" Width="80" X="598" Y="167"/>
  </ConfirmationNode>
  <InvoiceNode Caption="开发票" ID="7" Key="Invoice">
    <TransitionCollection>
      <SequenceFlow Condition="" ID="17" Key="InvoiceontoEnd" TargetNodeKey="End">
        <TransitionGraphic/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic Height="50" Width="80" X="733" Y="171"/>
  </InvoiceNode>
  <EndNode Caption="结束节点" ID="8" Key="End">
    <NodeGraphic Height="50" Width="80" X="868" Y="175"/>
  </EndNode>
</Process>

```

```

</TimerItemCollection>
  <NodeGraphic Height="50" Width="80" X="453" Y="219"/>
</UserTask>
<UserTask Caption="物品入库" ID="5" Key="Task02">
  <TransitionCollection>
    <SequenceFlow Condition="" ID="15" Key="Task02toJoin" TargetNodeKey="Join">
      <TransitionGraphic/>
    </SequenceFlow>
  </TransitionCollection>
  <OperationCollection>
    <Operation Action="CommitWorkitem(-1,1,&quot;&quot;)" Caption="提交" Key="op1"/>
  </OperationCollection>
  <AssistanceCollection/>
  <ParticipatorCollection>
    <Query>
      <![CDATA[SELECT  OID FROM GP_Operator]]>
    </Query>
  </ParticipatorCollection>
  <NodeGraphic Height="50" Width="80" X="466" Y="80"/>
</UserTask>
<Join Caption="合并节点" ID="7" Key="Join">
  <TransitionCollection>
    <SequenceFlow Condition="" ID="13" Key="JointoTask03" TargetNodeKey="Task03">
      <TransitionGraphic/>
    </SequenceFlow>
  </TransitionCollection>
  <NodeGraphic Height="50" Width="80" X="629" Y="136"/>
</Join>
<UserTask Caption="入库成功确认" ID="6" Key="Task03">
  <TransitionCollection>
    <SequenceFlow Condition="" ID="17" Key="Task03toDataMap01" TargetNodeKey="DataMap01">
      <TransitionGraphic/>
    </SequenceFlow>
  </TransitionCollection>
  <OperationCollection>
    <Operation Action="CommitWorkitem(-1,1,&quot;&quot;)" Caption="提交" Key="op1"/>
  </OperationCollection>
  <AssistanceCollection/>
  <ParticipatorCollection>
    <Query>
      <![CDATA[SELECT  OID FROM GP_Operator]]>
    </Query>
  </ParticipatorCollection>
  <NodeGraphic Height="50" Width="80" X="759" Y="134"/>
</UserTask>
<DataMap Caption="开发票" ID="18" Key="DataMap01" SyncTriggerType="InstanceStart">
  <TransitionCollection>
    <SequenceFlow Condition="" ID="19" Key="DataMap01toEnd" TargetNodeKey="End">
      <TransitionGraphic/>
    </SequenceFlow>
  </TransitionCollection>
  <OperationCollection>
    <Operation Action="BPMMap(&quot;StockIn2StockIn&quot;)" Caption="数据映射" Key="op1"/>
  </OperationCollection>
  <AssistanceCollection/>
  <ParticipatorCollection>
    <Query>
      <![CDATA[SELECT  OID FROM GP_Operator]]>
    </Query>
  </ParticipatorCollection>
  <BillDataMapInfoCollection>
    <BillDataMapInfo AutoStartMidMapInstance="false" DataMapKey="StockIn2StockIn"/>
  </BillDataMapInfoCollection>
  <NodeGraphic Height="50" Width="80" X="958" Y="134"/>
</DataMap>
<End Caption="结束节点" ID="8" Key="End">
  <TransitionCollection/>
  <NodeGraphic Height="50" Width="80" X="1141" Y="138"/>
</End>
<SwimlineCollection/>
<BPMInfoCollection/>
<DMTable>
  <Field Key="BillDate" SourceFieldKey="BillDate"/>
</DMTable>
</Process>

```

同样，基本同PC端配置定义相同。

第 9 章 布局及选择器(Layout)

目录

9.1. 布局介绍	87
9.2. 实例	88

每个面板除了其本身的布局以外，均可以改变其自身的布局，定义新的布局样式。本章节就是介绍如何改变其自身布局。这个布局和之前介绍的布局面板意义相同，配置形式不同。

9.1:布局介绍

以下针对每个不同的布局面板描述其布局文件，具体的布局的说明见参见前面关于面板的说明。

1. 列式布局(ColumnLayout)

```
<ColumnLayout>
  <LayoutSpan X="X" Y="Y" XSpan="XSpan" YSpan="YSpan" Key="Key"/> // ColumnLayout跟GridLayout采用相同的存储方式，在列式布局
  ...
  </ColumnLayout>
```

2. 边界布局(BorderLayout)

```
<BorderLayout>
  <LayoutDirection Value="Direction" Key="Key"/>
  ...
</BorderLayout>
```

3. 流布局(FlowLayout)

```
<FlowLayout>
  <LayoutFlowIndex Key="Key"/>
  ...
</FlowLayout>
```

4. 弹性流布局(FlexFlowLayout)

```
<FlexFlowLayout>
  <LayoutFlowIndex Key="Key"/>
  ...
</FlexFlowLayout>
```

5. 选项卡布局(TabLayout)

```
<TabLayout>
  <LayoutTabItem Caption="Caption" Key="Key"/>
  ...
</TabLayout>
```

6. 拆分式布局(SplitLayout)

```
<SplitLayout>
  <LayoutSplitItem Size="Size" Key="Key"/>
  ...
</SplitLayout>
```

7. 网格格式布局(GridLayout)

```
<GridLayout>
  <RowDefCollection />
  <ColumnDefCollection />
  <LayoutSpan X="X" Y="Y" XSpan="XSpan" YSpan="YSpan" Key="Key"/>
  ...
</GridLayout>
```

8. 表布局(TableLayout)

```
<TableLayout>
  <LayoutColumnCollection>
    <LayoutColumn Width="Width"/>
    ...
  </LayoutColumnCollection>
  <LayoutRowCollection>
    <LayoutRow Height="Height"/>
    ...
  </LayoutRowCollection>
  <LayoutSpan X="X" Y="Y" XSpan="XSpan" YSpan="YSpan" Key="Key"/>
  ...
</TableLayout>
```

9. 线性布局(LinearLayout)

```
<LinearLayout Orientation="VERTICAL">
  <LinearItem Key=""/>
  ...
</LinearLayout>
```

10. 流式表布局(FluidTableLayout)

```
<FluidTableLayout RepeatCount="" RepeatGap="" RowGap="" ColumnGap="" RowHeight="">
  <LayoutColumnCollection />
  <LayoutRowIndex Key=""/>
  ...
</FluidTableLayout>
```

9.2: 实例

定义一个布局实例，包含三套不同的布局形式。分别适用于不同系统使用。

布局PC：布局样式定义为选项卡嵌套网格布局的形式，包含三个选项卡item1、item2、item3；结构为4列。

布局Android：布局样式定义为流布局的形式，基本组件只包含item1和item2。

布局Android-300：布局样式定义为流布局嵌套流式表布局的形式，大小为300，基本组件包含item1、item2、item3。内部结构变为两列，组件删减为7个控件

源代码如下：

首先，在Entry.xml要先定义3个不同的布局入口，布局PC、布局Android、布局Android-300。

```
<Entry Key="Form" Caption="表单" Open="True">
  <EntryItem Key="Layout_PC" Caption="布局(PC)" Type="Form" FormKey="Layout"/>
  <EntryItem Key="Layout_ANDROID" Caption="布局(Android)" Type="Form" FormKey="Layout" Media="android"/>
  <EntryItem Key="Layout_ANDROID300" Caption="布局(Android-300)" Type="Form" FormKey="Layout" Media="android" Size="300"/>
</Entry>
```

然后再定义布局内容，

```
<Form Caption="布局测试" FormType="Entity" Key="Layout">
  <Body>
    <LayoutCollection>
      <Layout Key="android1" Media="android">
        <FlowLayout Key="tab">
          <LayoutFlowIndex Key="item1"/>
          <LayoutFlowIndex Key="item2"/>
        </FlowLayout>
      </Layout>
      <Layout Key="android2" Media="android" Size="300">
        <FlowLayout Key="tab">
          <LayoutFlowIndex Key="item1"/>
          <LayoutFlowIndex Key="item2"/>
          <LayoutFlowIndex Key="item3"/>
        </FlowLayout>
        <FluidTableLayout Key="item1" RowHeight="30">
          <LayoutColumnCollection>
            <LayoutColumn Width="120px"/>
          </LayoutColumnCollection>
        </FluidTableLayout>
      </Layout>
    </LayoutCollection>
  </Body>
</Form>
```

```

    <LayoutColumn Width="100%" />
</LayoutColumnCollection>
<LayoutRowIndex Key="I1_1" />
<LayoutRowIndex Key="I2_1" />
<LayoutRowIndex Key="I3_1" />
<LayoutRowIndex Key="I4_1" />
<LayoutRowIndex Key="I5_1" />
<LayoutRowIndex Key="I6_1" />
<LayoutRowIndex Key="I7_1" />
</FluidTableLayout>
<FluidTableLayout Key="item2" RowHeight="30">
    <LayoutColumnCollection>
        <LayoutColumn Width="120px" />
        <LayoutColumn Width="100%" />
    </LayoutColumnCollection>
    <LayoutRowIndex Key="I1_2" />
    <LayoutRowIndex Key="I2_2" />
    <LayoutRowIndex Key="I3_2" />
    <LayoutRowIndex Key="I4_2" />
    <LayoutRowIndex Key="I5_2" />
    <LayoutRowIndex Key="I6_2" />
    <LayoutRowIndex Key="I7_2" />
</FluidTableLayout>
<FluidTableLayout Key="item3" RowHeight="30">
    <LayoutColumnCollection>
        <LayoutColumn Width="120px" />
        <LayoutColumn Width="100%" />
    </LayoutColumnCollection>
    <LayoutRowIndex Key="I1_3" />
    <LayoutRowIndex Key="I2_3" />
    <LayoutRowIndex Key="I3_3" />
    <LayoutRowIndex Key="I4_3" />
    <LayoutRowIndex Key="I5_3" />
    <LayoutRowIndex Key="I6_3" />
    <LayoutRowIndex Key="I7_3" />
</FluidTableLayout>
</Layout>
</LayoutCollection>
    <Block>
        <TabPanel Key="tab">
            <GridLayoutPanel Key="item1" Caption="信息1" Padding="5px" BackColor="#ba97a8">
                <RowDefCollection RowHeight="30" RowGap="5">
                    <RowDef />
                    <RowDef />
                    <RowDef />
                    <RowDef />
                    <RowDef />
                </RowDefCollection>
                <ColumnDefCollection ColumnGap="5">
                    <ColumnDef Width="120px" />
                    <ColumnDef Width="50%" />
                    <ColumnDef Width="120px" />
                    <ColumnDef Width="50%" />
                </ColumnDefCollection>
                <Label Key="L_I1_1" Caption="内容1" X="0" Y="0" />
                <TextEditor Key="I1_1" X="1" Y="0" BuddyKey="L_I1_1" />
                <Label Key="L_I2_1" Caption="内容2" X="2" Y="0" />
                <TextEditor Key="I2_1" X="3" Y="0" BuddyKey="L_I2_1" />
                <Label Key="L_I3_1" Caption="内容3" X="0" Y="1" />
                <TextEditor Key="I3_1" X="1" Y="1" BuddyKey="L_I3_1" />
                <Label Key="L_I4_1" Caption="内容4" X="2" Y="1" />
                <TextEditor Key="I4_1" X="3" Y="1" BuddyKey="L_I4_1" />
                <Label Key="L_I5_1" Caption="内容5" X="0" Y="2" />
                <TextEditor Key="I5_1" X="1" Y="2" BuddyKey="L_I5_1" />
                <Label Key="L_I6_1" Caption="内容6" X="2" Y="2" />
                <TextEditor Key="I6_1" X="3" Y="2" BuddyKey="L_I6_1" />
                <Label Key="L_I7_1" Caption="内容7" X="0" Y="3" />
                <TextEditor Key="I7_1" X="1" Y="3" BuddyKey="L_I7_1" />
                <Label Key="L_I8_1" Caption="内容8" X="2" Y="3" />
                <TextEditor Key="I8_1" X="3" Y="3" BuddyKey="L_I8_1" />
                <Label Key="L_I9_1" Caption="内容9" X="0" Y="4" />
                <TextEditor Key="I9_1" X="1" Y="4" BuddyKey="L_I9_1" />
                <Label Key="L_I10_1" Caption="内容10" X="2" Y="4" />
                <TextEditor Key="I10_1" X="3" Y="4" BuddyKey="L_I10_1" />
            </GridLayoutPanel>
            <GridLayoutPanel Key="item2" Caption="信息2" Padding="5px" BackColor="#baf6f2">
                <RowDefCollection RowHeight="30" RowGap="5">
                    <RowDef />
                    <RowDef />
                    <RowDef />
                    <RowDef />
                    <RowDef />
                </RowDefCollection>

```

```

<ColumnDefCollection ColumnGap="5">
  <ColumnDef Width="120px"/>
  <ColumnDef Width="50%"/>
  <ColumnDef Width="120px"/>
  <ColumnDef Width="50%"/>
</ColumnDefCollection>
<Label Key="L_I1_2" Caption="内容1" X="0" Y="0"/>
<TextEditor Key="I1_2" X="1" Y="0" BuddyKey="L_I1_2"/>
<Label Key="L_I2_2" Caption="内容2" X="2" Y="0"/>
<TextEditor Key="I2_2" X="3" Y="0" BuddyKey="L_I2_2"/>
<Label Key="L_I3_2" Caption="内容3" X="0" Y="1"/>
<TextEditor Key="I3_2" X="1" Y="1" BuddyKey="L_I3_2"/>
<Label Key="L_I4_2" Caption="内容4" X="2" Y="1"/>
<TextEditor Key="I4_2" X="3" Y="1" BuddyKey="L_I4_2"/>
<Label Key="L_I5_2" Caption="内容5" X="0" Y="2"/>
<TextEditor Key="I5_2" X="1" Y="2" BuddyKey="L_I5_2"/>
<Label Key="L_I6_2" Caption="内容6" X="2" Y="2"/>
<TextEditor Key="I6_2" X="3" Y="2" BuddyKey="L_I6_2"/>
<Label Key="L_I7_2" Caption="内容7" X="0" Y="3"/>
<TextEditor Key="I7_2" X="1" Y="3" BuddyKey="L_I7_2"/>
<Label Key="L_I8_2" Caption="内容8" X="2" Y="3"/>
<TextEditor Key="I8_2" X="3" Y="3" BuddyKey="L_I8_2"/>
<Label Key="L_I9_2" Caption="内容9" X="0" Y="4"/>
<TextEditor Key="I9_2" X="1" Y="4" BuddyKey="L_I9_2"/>
<Label Key="L_I10_2" Caption="内容10" X="2" Y="4"/>
<TextEditor Key="I10_2" X="3" Y="4" BuddyKey="L_I10_2"/>
</GridLayoutPanel>
<GridLayoutPanel Key="item3" Caption="信息3" Padding="5px" BackColor="#32bbf7">
  <RowDefCollection RowHeight="30" RowGap="5">
    <RowDef/>
    <RowDef/>
    <RowDef/>
    <RowDef/>
    <RowDef/>
  </RowDefCollection>
  <ColumnDefCollection ColumnGap="5">
    <ColumnDef Width="120px"/>
    <ColumnDef Width="50%"/>
    <ColumnDef Width="120px"/>
    <ColumnDef Width="50%"/>
  </ColumnDefCollection>
  <Label Key="L_I1_3" Caption="内容1" X="0" Y="0"/>
  <TextEditor Key="I1_3" X="1" Y="0" BuddyKey="L_I1_3"/>
  <Label Key="L_I2_3" Caption="内容2" X="2" Y="0"/>
  <TextEditor Key="I2_3" X="3" Y="0" BuddyKey="L_I2_3"/>
  <Label Key="L_I3_3" Caption="内容3" X="0" Y="1"/>
  <TextEditor Key="I3_3" X="1" Y="1" BuddyKey="L_I3_3"/>
  <Label Key="L_I4_3" Caption="内容4" X="2" Y="1"/>
  <TextEditor Key="I4_3" X="3" Y="1" BuddyKey="L_I4_3"/>
  <Label Key="L_I5_3" Caption="内容5" X="0" Y="2"/>
  <TextEditor Key="I5_3" X="1" Y="2" BuddyKey="L_I5_3"/>
  <Label Key="L_I6_3" Caption="内容6" X="2" Y="2"/>
  <TextEditor Key="I6_3" X="3" Y="2" BuddyKey="L_I6_3"/>
  <Label Key="L_I7_3" Caption="内容7" X="0" Y="3"/>
  <TextEditor Key="I7_3" X="1" Y="3" BuddyKey="L_I7_3"/>
  <Label Key="L_I8_3" Caption="内容8" X="2" Y="3"/>
  <TextEditor Key="I8_3" X="3" Y="3" BuddyKey="L_I8_3"/>
  <Label Key="L_I9_3" Caption="内容9" X="0" Y="4"/>
  <TextEditor Key="I9_3" X="1" Y="4" BuddyKey="L_I9_3"/>
  <Label Key="L_I10_3" Caption="内容10" X="2" Y="4"/>
  <TextEditor Key="I10_3" X="3" Y="4" BuddyKey="L_I10_3"/>
</GridLayoutPanel>
</TabPanel>
</Block>
</Body>
</Form>

```

效果展示：

布局

信息1信息2信息3

内容1

内容3

内容5

内容7

内容9

内容2

内容4

内容6

内容8

内容10

PC:

布局

内容1

内容3

内容5

内容7

内容9

内容2

内容4

内容6

内容8

内容10

内容1

内容3

内容5

内容7

内容9

内容2

内容4

内容6

内容8

内容10

android:

内容1

内容2

内容3

内容4

内容5

内容6

内容7

内容1

内容2

内容3

内容4

内容5

内容6

内容7

内容1

内容2

内容3

内容4

内容5

内容6

内容7

配置说明:

1. 一个布局下可以包含0个或者多个新的布局。示例中布局android和布局android-300就是布局PC下包含的其他形式的新布局。
2. 新布局下的基本组件可以继承原始布局中的基本组件，但原则是不能超出原始布局中的基本组件范围，只能少不能多。示例中新布局android只包含了item1和item2，新布局android-300每个item中的组件只包含7个。
3. 新布局可以定义和原始布局相同或者不同的布局类型，其中组件的位置可以重新定义，不需要与原始布局中一致。示例中新布局android-300每个item中的组件只有两列（原始布局布局PC组件是四列）

4. 新定义的布局Key。必须和原始布局的Key相同；注意示例红色标注部分。
5. 入口Entry.xml中不同布局对应FormKey是相同的，通过属性Media来控制使用不同的布局类型。

第 10 章 表达式

目录

10.1. 运算符	93
10.2. 标识符、常量及保留字	95
10.3. 注释	95
10.4. 函数	95
10.5. 控制语句	97

Yigo表达式为Yigo系统中处理事件及数据处理的简单的脚本语言。

10.1: 运算符

运算符包括算术运算、关系运算和逻辑运算，同时包括一些其它符号。

10.1.1: 算术运算

算术运算包括+、-、*、/、&，字符串连接为&。

+为加法运算，但也可以用来做字符串连接，比如1+1结果为2，“1”+“1”的结果为“11”。如果加法的两个因数为整型其返回值为也整型，如果两个因数中有一个为浮点型，其结果也为浮点型。

减法、乘法和除法不再介绍，同加法一样，如果两个因子都为整型，其结果也为整数，如果其中一个因子是浮点数，其结果也为浮点数。

&为字符串连接符号，其结果同两个字符串的加法一样。

示例见EXPR中的Expr1表单。

表单界面如下：

表达式1

执行

数值1

10.00

数值2

20.00

文本1

文本2

其中“执行”按钮中的定义如下，其中OUT为控制台输出函数。

```
OUT (Num1+Num2);
OUT (Num1-Num2);
OUT (Num1*Num2);
OUT (Num1/Num2);
OUT (Txt1&Txt2);
```

输出如下：

```
30.00
-10.00
200.0000
0.5000000000
```

This is String

10.1.2: 关系运算

关系运算包括==、<>、!=、>、<、<=;

==为等于，判断两个值是否相等，注意这里为两个连续的=;

示例见EXPR中的Expr2表单。

表单界面如下:

表达式2

执行

数值1	10.00	数值2	20.00
文本1		文本2	

“执行”按钮中的定义如下:

```
OUT (Num1==Num2);  
OUT (Num1<>Num2);  
OUT (Num1>=Num2);  
OUT (Num1>Num2);  
OUT (Num1<Num2);  
OUT (Num1<=Num2);
```

输出如下:

```
false  
true  
false  
false  
true  
true
```

10.1.3: 逻辑运算

逻辑运算包括逻辑与、逻辑或、逻辑非，分别为: &&、||、!;

示例见EXPR中的Expr3表单。

表单界面如下:

表达式3

执行

数值1	10.00	数值2	20.00
文本1		文本2	

☒ 标志

“执行”按钮的定义如下:

```
OUT (Flag && Num1>Num2);
OUT (Flag || Num1>Num2);
OUT (Flag && !Num1>Num2);
```

输出如下:

```
false
true
true
```

10.1.4: 其它符号

包括=、(、)、{、}、;、,，其中“=”用于赋值，“;”用于语句的结束，“,”用于函数中的参数分隔。“{”和“}”定义语句段的区间；

10.2: 标识符、常量及保留字

10.2.1: 标识符

以字母开头，后续符号可以是字母，下划线，点(.)和数字。

10.2.2: 常量

常量包括数值常量、字符串常量以及逻辑常量；

数值常量以数字开头，后续符号是数字和点(.)，如果为包括.则为整型常量，否则为浮点型常量。

字符串常量为以单引号或双引号括起来的字符串；

逻辑常量只有两个，True和False，不区分大小写。

10.2.3: 保留字

if, else, while, var, self, parent, return, break, loop为系统保留字。

10.3: 注释

注释采用java语言的注释方式，支持两种注释，一种是以//注释到行尾，另外一种是以/* */来注释一段，例如：

```
// 这是一行注释
OUT (Flag && Num1>Num2);

/*
这是一段注释
*/
OUT (Flag || Num1>Num2);
OUT (Flag && !Num1>Num2);
```

10.4: 函数

函数包括函数名和参数列表，比如 func(para1, para2)。在函数之前可以增加对象定义，比如self或parent，其中self指向自己，parent指向父界面，self可以不定义；

示例见EXPR中的Expr4。其界面同Expr1相同，在“执行”中定义如下表达式：

```
SetValue('Num1', 30);
SetValue('Num2', 50);
```

```
SetValue('Txt1', 'Cat');
SetValue('Txt2', 'Dog');
parent.SetValue('Num1', 30);
parent.SetValue('Num2', 50);
parent.SetValue('Txt1', 'Cat');
parent.SetValue('Txt2', 'Dog');
```

修改Expr1，在其中增加一个操作，用于打开Expr4。如下：

```
ShowModal('Expr4')
```

打开Expr1后，点击“打开表达式4”，界面如下：

点击“表达式4”中的“执行”按钮后，将“表达式1”和“表达式4”均赋成了相同的值，如下：



10.5: 控制语句

控制语句包括if-else条件判断和while循环。

10.5.1: 条件判断

条件判断的语法结构有两种形式，一种是 `if (E) { SL }` 结构，一种是 `if (E) { SL1 } else { SL2 }` 结构。其中E是条件，SL、SL1和SL2为执行语句系列。

示例见EXPR中的Expr5。在“执行”按钮中定义表达式如下：

```
if (Flag) {  
    Txt1 = Num1;  
}  
if (Flag) {  
    Txt2 = Num1;  
} else {  
    Txt2 = Num2;  
}
```

执行结果如下图所示：

表达式5

执行

数值1

1.00

数值2

2.00

文本1

1.00

文本2

1.00

☒ 标志

在“数值1”和“数值2”中分别输入1和2后，选中“标志”，点执行，结果如下图。

10.5.2:循环

循环语句的形式为while(E) { SL }，其中E是条件，SL为执行语句系列。

示例见EXPR中的Expr6。Expr6的界面如下：

表达式6

执行

数值1

数值2

文本1

文本2

		数1	数2
1	<input type="checkbox"/>	1.00	
2	<input type="checkbox"/>	2.00	
3	<input type="checkbox"/>	3.00	
4	<input type="checkbox"/>	4.00	
5	<input type="checkbox"/>	5.00	
6	<input type="checkbox"/>	6.00	
7	<input type="checkbox"/>		

在“执行”按钮中定义如下表达式：

```
var count = GetRowCount('detail');
var i = 0;
while ( i < count ) {
  SetRowIndex('detail', i);
  OUT(N1);
  i = i + 1;
}
```

执行结果如下：

```
1.00
2.00
3.00
4.00
5.00
6.00
0
```

循环的退出：如果需要在中途退出循环，使用break语句，比如前循环前三行，如下：

```
var count = GetRowCount('detail');
var i = 0;
while ( i < count ) {
    SetRowIndex('detail', i);
    OUT(N1);
    i = i + 1;
    if ( i == 3 ) {
        break;
    }
}
```

使用上图同样的数据，输出为：

```
1.00
2.00
3.00
```

10.5.3:对象循环

对象循环用于遍历系统中的ListView、Grid和数据对象中的表。对象循环的语法结构为loop obj (E) { SL }，其中loop表示循环函数、obj表示对象、E为条件，SL为语句序列。

在Expr6的表单中增加一个操作“对象循环”，定义如下表达式：

```
loop 'detail' (true) {
    OUT(N1);
}
```

输入同上节示例相同的数据，点击“对象循环”，执行结果如下：

```
1.00
2.00
3.00
4.00
5.00
6.00
```

注意：通过上面的结果看，同之前的while循环的输出结果相比，少了最后一行空行，默认情况下，LoopGrid不包含空行，如果需要包含空行，在对象的名称后面加上empty参数，如下：

```
Loop 'detail:empty' (true) {
    OUT(N1);
}
```

在同样的输入下，结果如下，包含了空行数据。

```
1.00
2.00
3.00
4.00
5.00
6.00
0
```

如果只输出选中行，修改条件，表达式如下：

```
Loop 'detail' (Select) {
    OUT(N1);
}
```


界面如下图所示：

表达式6

执行
对象循环

数值1

文本1

数值2

文本2

		数1	数2
1	<input type="checkbox"/>	1.00	
2	<input checked="" type="checkbox"/>	2.00	
3	<input type="checkbox"/>	3.00	
4	<input checked="" type="checkbox"/>	4.00	
5	<input checked="" type="checkbox"/>	5.00	
6	<input type="checkbox"/>	6.00	
7	<input type="checkbox"/>		

点击“对象循环”，输出如下：

```
2.00
4.00
5.00
```

如结果所示，只有选中的行的才被输出。

10.5.4:取值

在表达式中使用数据分为界面和中间层。

在表达式使用界面的数据有两种方式：

- 直接使用界面标识，界面标识分为头控件，ListView的列和Grid单元格，可以直接使用这些对象的标识来取得界面上的数据；对于ListView中列标识，取的是当前行的数据，对于Grid中的单元格取的是当前行的数据。
- 通过GetValue和GetCellValue，GetValue用于取得头控件的值，GetCellValue取得ListView和Grid中单元格的值。

在表达式中使用文档中的数据时，通过[表名.列名]的形式取得，比如HeadTable.Value，中间层取值通过[表名.列名]的形式取数或者通过GetValue(中间层)的形式；

下面我们通过例子来说明如何使用数据。源代码见EXPR应用。

10.5.4.1:客户端取数

我们需要通过POrderView中的ListView的双击事件获取行上的OID列的值，代码如下：

```
<RowDbClick>
  <![CDATA[
    Open('POrder', OID)
  ]]>
</RowDbClick>
```

这里因为ListView组件存在OID列，因此可以直接使用OID来访问当前行中OID列的值。

也可以用GetCellValue来取得OID的值，源代码如下：

```
<RowDbClick>
  <![CDATA[
    Open('POrder', GetCellValue('list', -1, 'OID'))
  ]]>
</RowDbClick>
```

取得头控件中的值的方法类似，比如我们需要取得POrder中的NO的值，在POrder中添加一个操作如下：

```
<Operation Key="GetValue" Caption="取值">
  <Action>
    <![CDATA[
      OUT(NO)
    ]]>
  </Action>
</Operation>
```

输出如下，根据运行，内容会有所不如：

```
POrder20150623000002
```

这里系统会输出打开的表单的NO的值；如果NO没有界面关联，那么必须使用POrderHead.NO来获取，修改操作如下，也一样可以获取NO的值：

```
<Operation Key="GetValue" Caption="取值">
  <Action>
    <![CDATA[
      OUT(NO);
      OUT(POrderHead.NO);
    ]]>
  </Action>
</Operation>
```

输出如下，根据运行，内容会有所不如：

```
POrder20150623000002
POrder20150623000002
```

也可以通过GetValue来获取头控的，如下：

```
<Operation Key="GetValue" Caption="取值">
  <Action>
    <![CDATA[
      OUT(NO);
      OUT(POrderHead.NO);
      OUT(GetValue('NO'));
    ]]>
  </Action>
</Operation>
```

输出如下，根据运行，内容会有所不如：

```
POrder20150623000002
POrder20150623000002
POrder20150623000002
```

10.5.4.2: 中间层取数

在POrder中保存后处理中输出NO的值，在POrder的PostSaveProcess中增加以下代码：

```
<DataObject>
```

```

...
<PostSaveProcess>
  <Process>
    <![CDATA[
OUT (POrderHead.NO)
]]>
  </Process>
</PostSaveProcess>
</DataObject>

```

增加这个处理用于在表单保存后输出表单编号，输出如下(根据运行输出不同)：

```
PORDER20150623000002
```

也可以通过中间层的GetValue来取值，修改代码如下：

```

<DataObject>
...
  <PostSaveProcess>
    <Process>
      <![CDATA[
OUT (POrderHead.NO);
OUT (GetValue('POrderHead','NO'));
]]>
    </Process>
  </PostSaveProcess>
</DataObject>

```

输出如下：

```
PORDER20150623000002
PORDER20150623000002
```

通过输出可以看到，两种方式都可以取得所需的值。

10.5.5:赋值

在表达式中赋值时包括中间层和界面。

在界面上赋值时有如下两种形式：

- 直接使用界面标识，包括头控件标识、Grid中的单元格的标识；
- 通过SetValue设置头控件的值，使用SetCellValue设置明细行的值；

修改文档中的值时，只能使用[表名.列名]的形式，中间层赋值可以使用SetValue(中间层)和[表名.列名]形式。

下面我们举例来说明，见POrder.xml配置。

10.5.5.1:界面赋值

在POrder中增加如下操作：

```

<Operation Key="SetValue" Caption="赋值">
  <Action>
    <![CDATA[
Memo='New memo'
]]>
  </Action>
</Operation>

```

执行“赋值”操作时，将“备注”的值修改为“New memo”，界面如下：

采购清单 采购订单

编辑 删除 取值 赋值

编号 PORDER2015062300 日期 2015-06-23

状态 初始 备注 New memo

也可以通过SetValue函数来修改值，将之前的“赋值”操作修改如下：

```
<Operation Key="SetValue" Caption="赋值">
  <Action>
    <![CDATA[
      Memo='New memo';
      SetValue('Memo','New memo');
    ]]>
  </Action>
</Operation>
```

以上代码也可以得到相同的效果将Memo的值修改为“New memo”。

如果要修改明细行中的值，也可以通过单元格的标识，修改“赋值”操作的内容，用于修改“数量”的值（当前行），代码如下：

```
<Operation Key="SetValue" Caption="赋值">
  <Action>
    <![CDATA[
      Memo='New memo';
      SetValue('Memo','New memo');
      Amount=100;
    ]]>
  </Action>
</Operation>
```

选中表格中的某个，点击“赋值”操作，将当前的数量修改为100，效果如下：

2	SALE 销售部	FAX 传真机	2.00
2	SALE 销售部	FAX 传真机	100.00

通过SetCellValue也可以达到同样的效果：

```
<Operation Key="SetValue" Caption="赋值">
  <Action>
    <![CDATA[
      Memo='New memo';
      SetValue('Memo','New memo');
      // Amount=100;
      SetCellValue('detail', -1, 'Amount', 100);
    ]]>
  </Action>
</Operation>
```

10.5.5.2: 中间层赋值

在POrder的中PreSaveProcess中增加处理在中间层更新Memo的值，代码如下：

```
<PreSaveProcess>
  <Process>
    <![CDATA[
      POrderHead.Memo='Mid generated memo.'
    ]]>
  </Process>
</PreSaveProcess>
```

修改表单，并清空Memo的值，然后保存，Memo的值为被修改成“Mid generated memo.”，如下图：

编号	PORDER201506230000	日期	2015-06-23
状态	初始	备注	Mid generated memo.

如果要修改明细表的值，这里将数量全部修改为100，代码如下：

```
<PreSaveProcess>
  <Process>
    <![CDATA[
POrderHead.Memo='Mid generated memo.';
Loop 'POrderDtl' (true) {
  POrderDtl.Amount=100;
}
]]>
  </Process>
</PreSaveProcess>
```

	部门	物料	数量
1	FI 财务部	COMPUTER...	100.00
2	SALE 销售部	FAX 传真机	100.00
3	TECH 技术部	PRINTER 打...	100.00
4	FI 财务部	FAX 传真机	100.00
5	SALE 销售部	PRINTER 打...	100.00
6	TECH 技术部	TELEPHONE ...	100.00
7	SALE 销售部	PRINTER 打...	100.00
8	TECH 技术部	TELEPHONE ...	100.00
9	FI 财务部	COMPUTER...	100.00

数据对象基础

目录

1. 数据对象定义	105
2. 使用表单管理数据对象	107

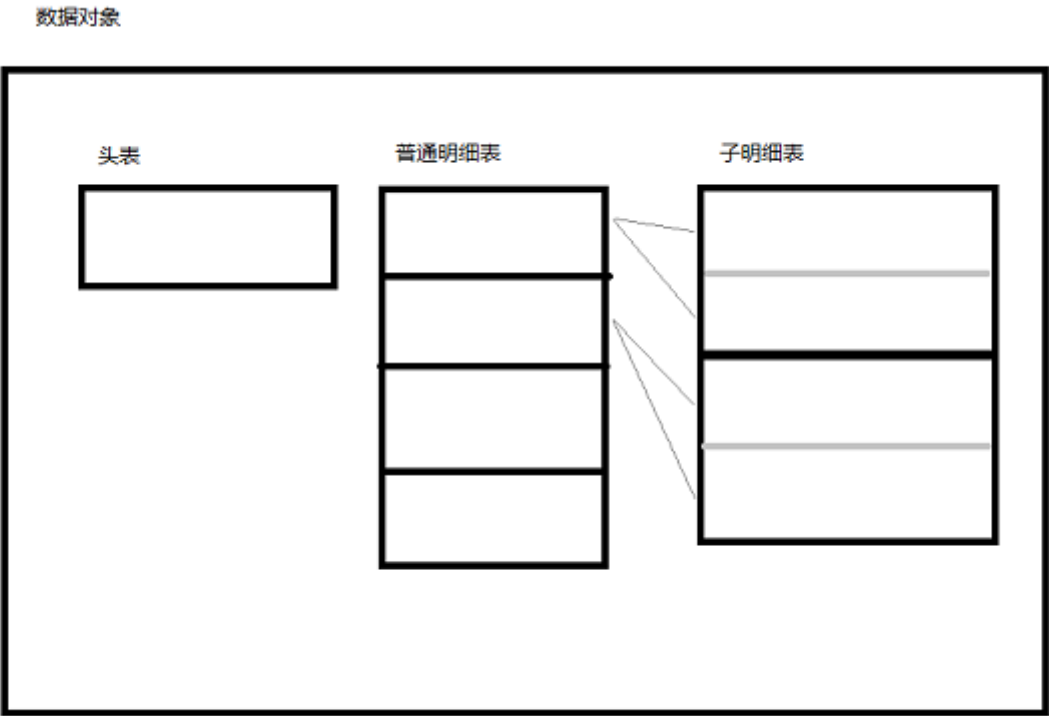
1: 数据对象定义

数据对象根据其持久化特征分为实体对象和虚拟对象，实体对象为需要持久化到数据库中的数据对象，而虚拟对象一般只用于作为数据展示。数据对象的基本属性定义如下：

```
<DataObject Key="" Caption="" PrimaryType="" SecondaryType="" PrimaryTableKey="" NoPrefix="">
</DataObject>
```

- Key 数据对象的标识，对于实体数据对象，其标识必须是全局唯一的；
- Caption 数据对象的名称；
- PrimaryType 为数据对象的主类型，只包括Entity和Virtual，Entity为实体对象，Virtual为虚拟对象；
- SecondaryType 为辅类型，在主类型为实体类型时，用于确定数据对象的业务意义，比如Normal(普通实体)、Dict(字典)、ChainDict(链式字典)，具体见参考文档；
- PrimayTableKey 为数据对象的主表标识，主表用于存储数据对象的版本信息；
- NoPrefix 为数据对象的编号前缀，定义为表达式。

每个数据对象包含一个或多个表组成，这里只说明实体数据对象的结构，虚拟对象类似，如下图所示：



实体对象具有存储要求的表，必须具有以下系统字段：

- SOID Long类型，系统OID；
- OID Long类型，对象OID；
- POID Long类型，父对象OID；
- VERID Integer类型，数据版本号；
- DVERID Integer类型，明细控制版本号；

定义的方法如下：

```
<DataSource>
  <DataObject Key="" NoPrefix="" Caption="" PrimaryTableKey="" PrimaryType="Entity" SecondaryType="Normal">
    <TableCollection>
      <Table Key="" Caption="基本信息" DBTableName="" TableMode="Head" SourceType="Table" Persist="True">
        <Column Key="OID" Caption="对象标识" Persist="True" DataType="Long" DBColumnName="OID"/>
        <Column Key="POID" Caption="父对象标识" Persist="True" DataType="Long" DBColumnName="POID"/>
        <Column Key="SOID" Caption="主对象标识" Persist="True" DataType="Long" DBColumnName="SOID"/>
        <Column Key="VERID" Caption="对象版本" Persist="True" DataType="Integer" DBColumnName="VERID"/>
        <Column Key="DVERID" Caption="对象明细版本" Persist="True" DataType="Integer" DBColumnName="DVERID"/>
      </Table>
    </TableCollection>
  </DataObject>
</DataSource>
```

对于类型为Entity的数据对象，除了以下系统字段外，其主表需要提供以下系统字段。

- CREATETIME - 创建时间；
- MODIFYTIME - 修改时间；
- HAPPTIME - 发生时间，可选定义；
- CREATOR - 创建人，可选定义；

- MODIFIER – 修改人，可选定义；
- NO – 编号字段，可选定义；
- LAYER – 数据层次标识，可选定义；
- HIDDEN – 隐藏标志标识，可选定义；同LAYER字段配合使用，该字段不存储，在定义时不要配置为Persist为True。

定义方法如下：

```
<Table Key="WM_pick" Caption="基本信息" DBTableName="WM_pick" TableMode="Head" SourceType="Table" Persist="True">
  <Column Key="OID" Caption="对象标识" Persist="True" DataType="Long" DBColumnName="OID"/>
  <Column Key="POID" Caption="父对象标识" Persist="True" DataType="Long" DBColumnName="POID"/>
  <Column Key="SOID" Caption="主对象标识" Persist="True" DataType="Long" DBColumnName="SOID"/>
  <Column Key="VERID" Caption="对象版本" Persist="True" DataType="Integer" DBColumnName="VERID"/>
  <Column Key="DVERID" Caption="对象明细版本" Persist="True" DataType="Integer" DBColumnName="DVERID"/>
  <Column Key="CREATETIME" Caption="创建日期" Persist="True" DataType="DateTime" DBColumnName="CREATETIME"/>
  <Column Key="MODIFYTIME" Caption="修改日期" Persist="True" DataType="DateTime" DBColumnName="MODIFYTIME"/>
  <Column Key="NO" Caption="单据编号" Persist="True" DataType="Varchar" Length="200" DBColumnName="NO"/>
  <Column Key="CREATOR" Caption="创建人" Persist="True" DataType="Long" DBColumnName="CREATOR"/>
  <Column Key="MODIFIER" Caption="修改人" HIDDEN="True" DataType="Long" DBColumnName="MODIFIER"/>
</Table>
```

每个实体数据对象通常需要定义其唯一的主表，通过PrimaryTable属性定义，主表必须是头表，且必须存在一行数据，实体数据对象的OID为主表中OID字段的值，其它每个表的每一行数据的SOID均取实体数据对象的OID的值；实体数据对象中数据存储表的每一行数据均具有唯一的OID。

实体数据对象的主从表通过表中的ParentKey属性定义，数据之间的从属关系通过POID来确定，POID指向父行数据OID。

2: 使用表单管理数据对象

这部分说明表及其属性，包括头表和明细表及其基本属性。

表的基本属性定义如下：

```
<Table Key="TableKey" Caption="TableCaption" TableMode="TableMode" DBTableName="TableName" SourceType="SourceType"
Persist="True"
  ParentKey="" Hidden="" UniquePrimary="">
  <Statement>
    <![CDATA[ ]]
  </Statement>
  <TableFilter>
    <![CDATA[ ]]
  </TableFilter>
  <ParameterCollection>
    <Parameter TargetTable="" TargetColumn="" SourceType="" Formula="" FieldKey="" Value="" DataType="" Description=""/>
    ...
  </ParameterCollection>
  <RuleFilterCollection>
    <RuleFilter Rule="">
      <TableFilter>
        <![CDATA[ ]]
      </TableFilter>
      <ParameterCollection>
        <Parameter TargetTable="" TargetColumn="" SourceType="" Formula="" FieldKey="" Value="" DataType=""
Description=""/>
        ...
      </ParameterCollection>
    </RuleFilter>
    ...
  </RuleFilterCollection>
</Table>
```

- Key：表的标识，在一个Object.xml范围内唯一。
- Caption：表的名称。
- DBTableName：表对应的数据库表名，如果未定义的情况下，使用Key的值。

- **TableMode** : 表的模式分为头表和明细表, 取值范围为Head和Detail, 取值为Head表示在整个Object中仅有一条数据, 取值为Detail表示可以有0到多条数据。默认值为Head。
- **SourceType** : 来源类型, 取值范围为Table、Query、Custom、Interface。默认值为Table(此项属性的含义发生了变化, 从描述数据的来源和保存模式改为仅仅描述数据的来源方式, 不再涉及保存)。
- **Persist** : 是否持久化。取值为True或False, 默认值为True。
- **Statement** : 在SourceTable为Query的情况下提供查询语句。
- **DefaultTableFilter** : 默认表过滤条件。
- **ParentKey** : 父表的标识。
- **Hidden** : 隐藏属性, 表示只作为定义使用, 不载入数据, 默认值为False。
- **UniquePrimary** : 唯一业务主键, 该属性只对实体对象的主表有效, 用于确定表中IsPrimay属性为True的那些列的取值是否全局唯一, 取值为True和False, 默认值为False。

查询及参数包含多个参数(Parameter), Parameter属性定义如下:

- **TargetTable** 参数的目标表, 当前未使用。
- **TargetColumn** : 参数的目标字段; 可以不定义, 不定义的情况下, 根据DataType定义来确定参数的类型, 如果在DataType也未定义的情况下, 根据实际值来确定参数类型。
- **DataType** : 定义参数的类型, 在定义了TargetColumn的情况下, 通过列的定义推算。
- **SourceType** : 参数的来源类型, 取值范围为Formula(来源于公式)、Field(来源于字段)、Const(常量), 默认值为Formula。
- **Formula** : 在SourceType为Formula时定义表达式。
- **FieldKey** : 在SourceType为Field时定义来源字段(组件标识或单元格标识)。
- **Value** : 在SourceType为Const时定义常量值。
- **Description** : 描述。
- **RuleFilterCollection** : 表可以包含多个按照规则定义的过滤条件, 定义在RuleFilterCollection集合中, 其中每个节点(RuleFilter)是一个过滤条件, 其属性同表中基本的同名属性。

示例: 定义拣货单的数据对象, 包括主表、明细表及子明细表。

源代码如下:

```
<DataSource>
  <DataObject Key="pick" NoPrefix="STIN" Caption="拣货单" PrimaryTableKey="WM_pick"
    PrimaryType="Entity" SecondaryType="Normal">
    <TableCollection>
      <Table Key="WM_pick" Caption="基本信息" DBTableName="WM_pick" TableMode="Head"
        SourceType="Table" Persist="True">
        <Column Key="OID" Caption="对象标识" Persist="True" DataType="Long"
          DBColumnName="OID"/>
        <Column Key="POID" Caption="父对象标识" Persist="True" DataType="Long"
          DBColumnName="POID"/>
        <Column Key="SOID" Caption="主对象标识" Persist="True" DataType="Long"
          DBColumnName="SOID"/>
        <Column Key="VERID" Caption="对象版本" Persist="True" DataType="Integer"
          DBColumnName="VERID"/>
        <Column Key="DVERID" Caption="对象明细版本" Persist="True" DataType="Integer"
          DBColumnName="DVERID"/>
        <Column Key="CREATETIME" Caption="单据日期" Persist="True" DataType="DateTime"
          DBColumnName="CREATETIME"/>
        <Column Key="NO" Caption="单据编号" Persist="True" DataType="Varchar" Length="200"
          DBColumnName="NO"/>
      </Table>
    </TableCollection>
  </DataObject>
</DataSource>
```

```
<Column Key="CREATOR" Caption="创建人" Persist="True" DataType="Long"
    DBColumnName="CREATOR"/>
</Table>
<Table Key="WM_pickDetail" Caption="拣货单明细1" DBTableName="WM_pickDetail"
    TableMode="Detail" SourceType="Table" Persist="True">
    <Column Key="OID" Caption="对象标识" Persist="True" DataType="Long"
        DBColumnName="OID"/>
    <Column Key="POID" Caption="父对象标识" Persist="True" DataType="Long"
        DBColumnName="POID"/>
    <Column Key="SOID" Caption="主对象标识" Persist="True" DataType="Long"
        DBColumnName="SOID"/>
    <Column Key="VERID" Caption="对象版本" Persist="True" DataType="Integer"
        DBColumnName="VERID"/>
    <Column Key="DVERID" Caption="对象明细版本" Persist="True" DataType="Integer"
        DBColumnName="DVERID"/>
    <Column Key="pick_Area" Caption="地区" Persist="True" DataType="Long"
        IsPrimary="True"/>
    <Column Key="pick_Material" Caption="物料" Persist="True" DataType="Varchar" Length="250"
        IsPrimary="True"/>
    <Column Key="pick_Amount" Caption="数量" Persist="True" DataType="Numeric"
        Precision="16" Scale="2"/>
</Table>
<Table Key="WM_pickDetail2" Caption="拣货单明细2" DBTableName="WM_pickDetail2"
    TableMode="Detail" SourceType="Table" Persist="True" ParentKey="WM_pickDetail">
    <Column Key="OID" Caption="对象标识" Persist="True" DataType="Long"
        DBColumnName="OID"/>
    <Column Key="POID" Caption="父对象标识" Persist="True" DataType="Long"
        DBColumnName="POID"/>
    <Column Key="SOID" Caption="主对象标识" Persist="True" DataType="Long"
        DBColumnName="SOID"/>
    <Column Key="VERID" Caption="对象版本" Persist="True" DataType="Integer"
        DBColumnName="VERID"/>
    <Column Key="DVERID" Caption="对象明细版本" Persist="True" DataType="Integer"
        DBColumnName="DVERID"/>
    <Column Key="pick_Warehouse" Caption="仓库" Persist="True" DataType="Long"
        IsPrimary="True"/>
    <Column Key="picker" Caption="拣货员" Persist="True" DataType="Varchar" Length="250"
        IsPrimary="True"/>
    <Column Key="Part_Amount" Caption="数量" Persist="True" DataType="Numeric"
        Precision="16" Scale="2"/>
</Table>
</TableCollection>
<PreSaveProcess> </PreSaveProcess>
</DataObject>
</DataSource>
```

数据对象处理

目录

1. 数据对象中的保存	110
2. 数据对象载入的处理	111
3. 中间层检查规则	112

1: 数据对象中的保存

1.1: 存储时的版本处理

数据对象保存时的版本处理规则如下:

- 所有被修改的数据行, VERID均加1;
- 如果是明细对象存储, 那么主表的DVERID加1;
- 如果是数据对象存储, 那么主表的VERIS加1, DVERID不变;

1.2: 存储时的版本冲突检测

数据存储时, 如果发生版本冲突, 系统提示数据已修改, 需要重新载入数据; 冲突在以下情况下发生:

- 数据对象存储时, 如果文档中的VERID和DVERID与数据库中VERID和DVERID有任何一个不相同, 版本冲突;
- 数据明细存储时, 如果文档中的VERID与数据库的VERID不相同, 版本冲突;
- 明细存储时, 如果文档中的VERID与数据库中的VERID相同, 但其中被修改需要存储的数据行的VERID与数据库的中相应数据的VERID不相同, 版本冲突;

1.3: 存储前处理

数据在存储到数据库之前, 按照顺序会经过以下处理阶段:

- 数据对象定义中定义的PreSaveProcess集合;
- 数据迁移, 迁移的发生规则见数据迁移部分说明;
- 系统默认的处理流, 有如下默认处理流:
 - 数据映射的反填
- 扩展的处理流, 在应用的Solution.xml文件中定义的Service为SaveData, 时间为前期的MidProcessFlow, 处理类要求实现com.bokesoft.yes.mid.service.IServiceProcess接口;
- 保存前的规则检查, 通过执行CheckRuleCollection集合中的规则, 如果有规则不满足要求, 抛出包含规则描述信息的异常, 并取消当前保存操作;

1.4: 存储后处理

数据在存储到数据库之后, 按照顺序执行以下处理阶段:

- 如果数据对象为字典，那么执行字典的维护；字典的维护规则见字典维护总分；
- 执行Solution.xml文件中定义的MidProcessFlow中定义的Service为SaveData，并且时机为后期的MidProcessFlow，处理类要求实现com.bokesoft.yes.mid.service.IServiceProcess接口；

2: 数据对象载入的处理

数据对象的载入采用逐表载入的方式，根据表的数据来源类型不同有不同的载入数据规则；

来源于Table

在数据来源于Table的情况下，系统根据指定的OID载入数据，系统会查询当前表中所有S0ID字段的值为指定OID的行数据；

来源于Query

在数据来源于Query的情况下，如果表定义了Statement，即定义了自己的语句，则使用定义的查询语句，否则根据表中的列定义生成查询语句；

来源于Custom

来源于Custom的情况下，设计是根据表达式的执行结果来获得DataTable，暂未支持；

来源于Interface

来源于Interface的情况下，设计是根据接口实例的执行结果来获得DataTable，暂未支持；

来源于LinkTable

来源于LinkTable的情况下，表示通过查询关联表来载入数据，关联表的定义必须为Hidden为True的表，否则数据会重复。

Filter过滤处理

定义的过滤处理指的是通过表的Filter属性来定义的过滤条件，为查询中必要的过滤条件，Filter中的过滤参数通过?定义，并通过FilterParameterCollection集合中的FilterParameter定义过滤参数的来源；在查询之前，客户端需要事先计算好FilterParameter中定义的过滤参数的实际值；表数据结果中不满足过滤条件的数据行不会出现在表的结果中

客户端的查询过滤处理

客户端的查询过滤处理指的是通过客户端的查询界面传递的查询过滤参数及参数值生成查询条件；

客户端查询条件通过定义过滤目标字段、目标字段的值、查询过滤符号的方式来定义分组条件，分组条件通过and条件拼接过滤条件；

查询过滤符号定义如下：

- =，等于，表示目标字段的值与指定的值相等；
- >，大于，表示目标字段的值大于指定的值；
- >=，大于或等于，表示目标字段的值大于或等于指定的值；
- <，小于，表示目标字段的值小于指定的值；
- <=，小于等于，表示目标字段的值小于或等于指定的值；
- <>，不等于，表示目标字段的值不等于指定的值；
- between，两者之间，这个时候，指定的值必须有两个，目标字段的值处于这两个值之间，只对日期类型有效；

- like, 相似, 表示目标字段的值与指定的值模式相似, 规则为全相似(参考数据库), 只对字符串类型有效;
- custom, 定制, 并接条件由二次开发指定;

数据权限处理

在数据对象载入时, 强制根据权限过滤数据对象中的数据。在需要进行权限过滤的表字段上定义NeedRights属性为True和过滤依据的字典项的ItemKey; 如果操作员角色包含内置的系统管理员角色(角色为11), 不控制权限。

数据权限根据方式方式分为两种, 一种是数据集过滤, 一种是数据过滤层;

- 数据集过滤在初始查询数据对象时即已经被处理, 在初次查询的数据表中, 不满足权限的数据已经被处理, 字段的值不在ItemKey所在的权限表中的行均不满足权限要求;
- 数据对象过滤层, 在初始查询数据对象数据后, 额外增加的一层数据过滤层, 这一层对数据进行额外的检查或删除。

数据权限根据处理要求分为剔除和否决。

- 剔除是指将不满足权限的数据从数据表中剔除;
- 否决表示在某行数据不满足要求时, 即否决整个数据对象的加载。
- 数据权限的处理分为两类, 一类是否决, 否决只用于实体数据对象的头表; 一类是移除, 除了否决的情形外的所有情况都是移除。

数据的排序规则

- 如果存在Sequence字段, 则Sequence必须加入到排序集合中(升序);
- 对于定义了Sort为Asc和Desc的列, 必须加入到排序集合中, 但只对Table的类型有效, 对于自定义的查询语句, 需要查询语句中自己提供order by子句。

3: 中间层检查规则

```
<CheckRuleCollection>
  <CheckRule>
    <![CDATA[  ]]>
  </CheckRule>
  ...
</CheckRuleCollection>
```

CheckRuleCollection包含多个CheckRule。

3.1: CheckRule属性

- Content - 定义在节点的CDATA中, 规则内容。
- ErrorInfo - 规则的出错描述。

保存前的规则检查, 通过执行CheckRuleCollection集合中的规则, 如果有规则不满足要求, 抛出包含规则描述信息的异常, 并取消当前保存操作;

数据对象高级应用

目录

1. 数据过滤	113
2. 复合字典	114
3. 表单报表	115
4. 查询条件	120

1: 数据过滤

这里介绍数据对象的OID过滤，表的默认过滤以及按规则过滤。

1.1:OIDFilter 属性描述

OIDFilter 是一个节点，用于覆盖数据对象中定义的OIDFilter定义，这样可以在不同引用的地方选择不同的OIDFilter。

```
<OIDFilter Type="" Formula="" Impl="">
  <Statement>
    <![CDATA[ ]]
  </Statement>
  <ParameterCollection>
    <Parameter TargetTable="" TargetColumn="" SourceType="" Formula="" FieldKey="" Value="" DataType="" Description="" />
    ...
  </ParameterCollection>
</OIDFilter>
```

- Type 为过滤的类型，取值为Query(查询)、Formula(表达式)、Interface(接口)，该属性在中间层解释；
- Formula Type的取值为Formula时定义表达式内容；
- Impl Type的取值为Interface时定义实现类的全路径名，这个接口后面再定义；
- Statement Type为Query时定义查询语句；
- ParameterCollection Type为Query时定义查询中参数的集合，具体定义见表中的同名定义。

1.2:DefaultTableFilter 属性描述

DefaultTableFilter - 默认表过滤条件。

```
<TableFilter>
  <![CDATA[ ]]
</TableFilter>
```

1.3:RuleFilter 属性描述

表可以包含多个按照规则定义的过滤条件，定义在RuleFilterCollection集合中，其中每个节点(RuleFilter)是一个过滤条件，其属性同表中基本的同名属性。

```
<RuleFilterCollection>
  <RuleFilter Rule="">
    <TableFilter>
      <![CDATA[ ]]
    </TableFilter>
    <ParameterCollection>
      <Parameter TargetTable="" TargetColumn="" SourceType="" Formula="" FieldKey="" Value="" DataType=""
Description="" />
      ...
    </ParameterCollection>
  </RuleFilter>
  ...
</RuleFilterCollection>
```

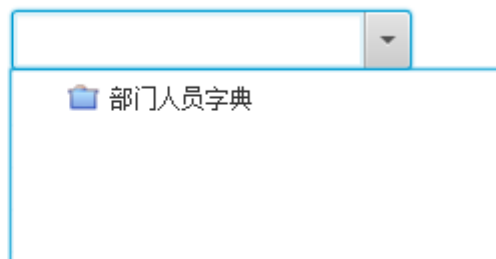
```
</RuleFilter>
...
</RuleFilterCollection>
```

2: 复合字典

复合字典为多个普通字典的数据按照从属或包含关系定义生成。这里介绍如何定义一个复合字典以及如何在复合字典控件中使用。

示例：定义一个复合字典 " 部门人员 " 字典。

复合字典: 部门-人员



1. 定义2个普通字典部门Department、Employee，（详见字典配置定义）

Department字典定义：

```
<Form Caption="部门" Key="Department" FormType="Dict">
  <DataSource>
    <DataObject Key="Department" Caption="仓库" PrimaryTableKey="dp_Department" PrimaryType="Entity" SecondaryType="Dict">
      <TableCollection>
        <Table Key="dp_Department" Caption="基本信息" DBTableName="dp_Department" TableMode="Head" SourceType="Table"
          Persist="True" SortColumns="Code desc;name">
          ...
        </Table>
      </TableCollection>
    </DataObject>
  </DataSource>
  ...
  <Body>
    <Block>
    ...
    </Block>
  </Body>
</Form>
```

Employee字典定义：

```
<Form Caption="人员" Key="Employee" FormType="Dict">
  <DataSource>
    <DataObject Key="Employee" Caption="人员" PrimaryTableKey="dp_employee" PrimaryType="Entity" SecondaryType="Dict">
      <TableCollection>
        <Table Key="dp_employee" Caption="基本信息" DBTableName="dp_employee" TableMode="Head" SourceType="Table" Persist="True"
          SortColumns="Code desc;name">
          ...
        </Table>
      </TableCollection>
    </DataObject>
  </DataSource>
  ...
  <Body>
    <Block>
    ...
    </Block>
  </Body>
</Form>
```

2. 在CompDict目录下定义数据对象Dpt-Emp.xml。

```
<DataObject Key="Dpt-Emp" Caption="部门人员字典" PrimaryType="Entity" SecondaryType="CompDict">
  <Relation Caption="部门人员字典" Description="Description">
    <Layer ItemKey="Department">
    </Layer>
    <Layer ItemKey="Employee" Relation="Belong" TableKey="dp_employee" ColumnKey="DepartmentID">
    </Layer>
  </Relation>
</DataObject>
```

- Relation 跟上层的关系，取值为Contain(包含)、Belong(从属)。

2. 再定义CompDict组件

```
<CompDict Key="dptemp" BuddyKey="L_DptEmp" ItemKey="Dpt-Emp" Caption="部门-人员">
  <DataBinding TableKey="Test_TestDict" ColumnKey="dptemp">
  </DataBinding>
</CompDict>
```

3: 表单报表

本章主要介绍如何制作表单报表，如何在界面上显示报表，以及介绍下表单的分组、汇总以及排序；Report章节主要介绍如何打印输出报表。

3.1: 如何定义报表

在表单中报表定义如下，使用Grid表格进行报表定义、格式控制。

```
<Grid Key="detail" Height="100%">
  <GridColumnCollection>
    <GridColumn Key="Dept" Caption="部门"/>
    <GridColumn Key="Mtl" Caption="物料"/>
    <GridColumn Key="Amount" Caption="数量"/>
  </GridColumnCollection>
  <GridRowCollection>
    <GridRow Key="R1" RowType="Detail" RowHeight="30" TableKey="P0OrderDtl">
      <GridCell Key="Dept" Caption="部门" CellType="Dict" ItemKey="Department">
        <DataBinding ColumnKey="Dept"/>
      </GridCell>
      <GridCell Key="Mtl" Caption="物料" CellType="Dict" ItemKey="Material">
        <DataBinding ColumnKey="Mtl"/>
      </GridCell>
      <GridCell Key="Amount" Caption="数量" CellType="NumberEditor" Precision="16" Scale="0">
        <DataBinding ColumnKey="Amount"/>
      </GridCell>
    </GridRow>
  </GridRowCollection>
</Grid>
```

详细定义可参考FORM2的Grid表格的介绍。

效果展示：

	部门	物料	数量
1	FI 财务部	COMPUTER...	12
2	SALE 销售部	FAX 传真机	2
3	TECH 技术部	PRINTER 打...	5
4	FI 财务部	FAX 传真机	20
5	SALE 销售部	PRINTER 打...	5
6	TECH 技术部	TELEPHONE ...	200
7	SALE 销售部	PRINTER 打...	54
8	TECH 技术部	TELEPHONE ...	6
9	FI 财务部	COMPUTER...	300

3.2: 定义排序

数据输入都是无序的，如果我们对输入的数据想要进行排序查看，可以通过定义数据排序，源代码如下：

```
<GridColumnCollection>
  <GridColumn Key="Dept" Caption="部门" Sortable="True"/>
  <GridColumn Key="Mtl" Caption="物料"/>
  <GridColumn Key="Amount" Caption="数量"/>
</GridColumnCollection>
```

- Sortable - 可排序。扩展列不支持排序。取值为True和False。

	部门 ▼	物料	数量
1	FI 财务部	COMPUTER...	100
2	FI 财务部	FAX 传真机	200
3	FI 财务部	PRINTER 打...	300
4	FI 财务部	TELEPHONE ...	400
5	SALE 销售部	COMPUTER...	10
6	SALE 销售部	FAX 传真机	20
7	SALE 销售部	PRINTER 打...	30
8	SALE 销售部	TELEPHONE ...	40
9	TECH 技术部	COMPUTER...	1
10	TECH 技术部	FAX 传真机	2
11	TECH 技术部	PRINTER 打...	3
12	TECH 技术部	TELEPHONE ...	4

3.3: 定义行分组和汇总

3.3.1: 分组

行分组通过明细定义行中的单元格分组类型为RowGroup的单元格定义，每个明细行中可以定义多个分组单元格。

```
<Grid Key="detail" Height="100%">
  <GridColumnCollection>
    <GridColumn Key="Dept" Caption="部门" Sortable="True"/>
    <GridColumn Key="Mtl" Caption="物料"/>
    <GridColumn Key="Amount" Caption="数量"/>
  </GridColumnCollection>
  <GridRowCollection>
    <GridRow Key="R1" RowType="Detail" RowHeight="30" TableKey="POrderDtl">
      <GridCell Key="Dept" Caption="部门" CellType="Dict" ItemKey="Department" CellGroupType="RowGroup">
        <DataBinding ColumnKey="Dept"/>
      </GridCell>
      <GridCell Key="Mtl" Caption="物料" CellType="Dict" ItemKey="Material">
        <DataBinding ColumnKey="Mtl"/>
      </GridCell>
      <GridCell Key="Amount" Caption="数量" CellType="NumberEditor" Precision="16" Scale="0">
        <DataBinding ColumnKey="Amount"/>
      </GridCell>
    </GridRow>
    <GridRow Key="R2" RowType="Group" RowHeight="30" TableKey="POrderDtl" GroupKey="Dept">
      <GridCell Caption="小计"/>
      <GridCell />
      <GridCell Key="DeptSumHead" >
        <DataBinding DefaultFormulaValue="Sum(' Amount')"/>
      </GridCell />
    </GridRow>
  </GridRowCollection>
</Grid>
```

效果显示:

	部门	物料	数量
1	TECH 技术部	COMPUTER...	1
2	TECH 技术部	FAX 传真机	2
3	TECH 技术部	PRINTER 打...	3
4	TECH 技术部	TELEPHONE ...	4
5	小计		10.0
6	SALE 销售部	COMPUTER...	10
7	SALE 销售部	FAX 传真机	20
8	SALE 销售部	PRINTER 打...	30
9	SALE 销售部	TELEPHONE ...	40
10	小计		100.0
11	FI 财务部	COMPUTER...	100
12	FI 财务部	FAX 传真机	200
13	FI 财务部	PRINTER 打...	300
14	FI 财务部	TELEPHONE ...	400
15	小计		1000.0

3.3.2: 汇总

进一步扩充此例子，增加总计行，配置如下：

```
<Grid Key="detail" Height="100%">
  <GridColumnCollection>
    <GridColumn Key="Dept" Caption="部门" Sortable="True"/>
    <GridColumn Key="Mtl" Caption="物料"/>
    <GridColumn Key="Amount" Caption="数量"/>
  </GridColumnCollection>
  <GridRowCollection>
    <GridRow Key="R1" RowType="Detail" RowHeight="30" TableKey="P0OrderDtl">
      <GridCell Key="Dept" Caption="部门" CellType="Dict" ItemKey="Department" CellGroupType="RowGroup">
        <DataBinding ColumnKey="Dept"/>
      </GridCell>
      <GridCell Key="Mtl" Caption="物料" CellType="Dict" ItemKey="Material">
        <DataBinding ColumnKey="Mtl"/>
      </GridCell>
      <GridCell Key="Amount" Caption="数量" CellType="NumberEditor" Precision="16" Scale="0">
        <DataBinding ColumnKey="Amount"/>
      </GridCell>
    </GridRow>
    <GridRow Key="R2" RowType="Group" RowHeight="30" TableKey="P0OrderDtl" GroupKey="Dept">
      <GridCell Caption="小计"/>
      <GridCell />
      <GridCell Key="DeptSumHead" >
        <DataBinding DefaultFormulaValue="Sum(' Amount')"/>
      </GridCell />
    </GridRow>
    <GridRow Key="R3" RowType="Total" RowHeight="30" TableKey="P0OrderDtl" GroupKey="Dept">
      <GridCell Caption="总计"/>
      <GridCell />
      <GridCell Key="SumHead" >
        <DataBinding DefaultFormulaValue="Sum(' DeptSumHead')"/>
      </GridCell>
    </GridRow>
  </GridRowCollection>
</Grid>
```

```
</GridRowCollection>
</Grid>
```

效果显示：

	部门	物料	数量
1	TECH 技术部	COMPUTER...	1
2	TECH 技术部	FAX 传真机	2
3	TECH 技术部	PRINTER 打...	3
4	TECH 技术部	TELEPHONE ...	4
5	小计		10.0
6	SALE 销售部	COMPUTER...	10
7	SALE 销售部	FAX 传真机	20
8	SALE 销售部	PRINTER 打...	30
9	SALE 销售部	TELEPHONE ...	40
10	小计		100.0
11	FI 财务部	COMPUTER...	100
12	FI 财务部	FAX 传真机	200
13	FI 财务部	PRINTER 打...	300
14	FI 财务部	TELEPHONE ...	400
15	小计		1000.0
16	总计		1110.0

3.4: 定义列扩展

列扩展的目标是把多行数据，根据选定的关键字转换成一行表格行以展现和编辑数据。

以“用途”为例，进行列扩展，源代码如下：

```
<Grid Key="detail" Height="100%">
  <GridColumnCollection>
    <GridColumn Key="Mtl" Caption="物料" Sortable="True"/>
    <GridColumn Key="Amount" Caption="数量" ColumnExpand="true" ColumnType="Detail" >
      <ColumnExpand ExpandType="title"/>
      <GridColumnCollection>
        <GridColumn Key="UseType" Caption="用途" Width="100px" ColumnExpand="true">
          <ColumnExpand ExpandType="data" ExpandSourceType="custom" ColumnKey="UseType" >
            <![CDATA["0, 自用;1, 转租;2, 储备"]]>
          </ColumnExpand>
        </GridColumn>
      </GridColumnCollection>
    </GridColumn>
  </GridColumnCollection>
</Grid>
<GridRowCollection>
  <GridRow Key="R1" RowType="Detail" RowHeight="30" TableKey="MtlReqDtl">
    <GridCell Key="Mtl" Caption="物料" CellType="Dict" ItemKey="Material">
      <DataBinding ColumnKey="Mtl"/>
    </GridCell>
    <GridCell Key="Amount" Caption="数量" CellType="NumberEditor" Precision="16" Scale="0">
```

```

        <DataBinding ColumnKey="Amount"/>
    </GridCell>
</GridRow>
</GridRowCollection>
</Grid>

```

效果显示:

	物料	数量		
		自用	转租	储备
1	COMPUTER...	1	2	3
2	FAX 传真机	10	20	30
3	PRINTER 打...	100	200	300
4	TELEPHONE ...	1,000	2,000	3,000

4: 查询条件

定义查询界面的界面组成，查询界面的定义同普通的表单定义相同，只在不同的组件中定义条件相关的属性，同一个字段可以为对个数据源表构造查询条件 结构如下：

```

<Component>
    <Condition CondSign="" Group="" GroupHead="" GroupTail="" TableKey="" ColumnKey="" TargetTableKey="" OpValue="" Impl="">
        <ConditionTarget TableKey="" ColumnKey=""/>
    </Condition>
</Component>

```

示例：对单张表单定义一组查询过滤条件

```

<FluidTableLayoutPanel Key="query" Height="100%" RepeatCount="4" RepeatGap="10" RowGap="5" ColumnGap="5" Padding="5px"
    RowHeight="25">
    <TableColumnCollection ColumnGap="10">
        <TableColumn Width="110px"/>
        <TableColumn Width="100%"/>
    </TableColumnCollection>
    <Label Key="L_LOH_LOGISTICS_NO" Caption="物流订单号"/>
    <TextEditor Key="LOH_LOGISTICS_NO" Caption="物流订单号" BuddyKey="L_LOH_LOGISTICS_NO">
        <Condition CondSign="" TableKey="WM_LogisticsOrderHead" ColumnKey="LOH_LOGISTICS_NO"/>
    </TextEditor>
    <Label Key="L_Status" Caption="状态"/>
    <ComboBox Key="Status" Caption="状态" BuddyKey="L_Status" SourceType="Status">
        <Condition CondSign="" TableKey="WM_LogisticsOrderHead" ColumnKey="Status" />
    </ComboBox>
    <Label Key="L_LOH_ORG_ID" Caption="接单机构"/>
    <Dict Key="LOH_ORG_ID" Caption="接单机构" ItemKey="Organization" BuddyKey="L_LOH_ORG_ID" AllowMultiSelection="true">
        <Condition CondSign="in" TableKey="WM_LogisticsOrderHead" ColumnKey="LOH_ORG_ID"/>
    </Dict>
    <Label Key="L_LOH_TYPE" Caption="订单类型"/>
    <ComboBox Key="LOH_TYPE" Caption="订单类型" BuddyKey="L_LOH_TYPE">
        <Condition CondSign="" TableKey="WM_LogisticsOrderHead" ColumnKey="LOH_TYPE"/>
    </ComboBox>
    <Item Key="outbound_deliver" Caption="出库配送" Value="outbound_deliver"/>
    <Item Key="collect_deliver" Caption="提货配送" Value="collect_deliver"/>
    <Item Key="collect_inbound" Caption="提货入库" Value="collect_inbound"/>
    <Item Key="outbound_inbound" Caption="调拨（无运输）" Value="outbound_inbound"/>
    <Item Key="outbound_transport_inbound" Caption="调拨（含运输）" Value="outbound_transport_inbound"/>
    <Item Key="return_outbound_deliver" Caption="出库退厂" Value="return_outbound_deliver"/>
    <Item Key="return_collect_deliver" Caption="提货退厂" Value="return_collect_deliver"/>
    <Item Key="return_collect_inbound" Caption="提货退库" Value="return_collect_inbound"/>
    </ComboBox>
    <Label Key="L_LOH_OWNER_ID" Caption="货主/托运商"/>
    <Dict Key="LOH_OWNER_ID" Caption="货主/托运商" ItemKey="Owner" BuddyKey="L_LOH_OWNER_ID" AllowMultiSelection="true">
        <Condition CondSign="in" TableKey="WM_LogisticsOrderHead" ColumnKey="LOH_OWNER_ID"/>
    </Dict>
    <Label Key="L_LOH_OWNER_NO" Caption="货主/托运商单号"/>
    <TextEditor Key="LOH_OWNER_NO" Caption="货主/托运商单号" BuddyKey="L_LOH_OWNER_NO">
        <Condition CondSign="" TableKey="WM_LogisticsOrderHead" ColumnKey="LOH_OWNER_NO"/>
    </TextEditor>

```

```

</TextEditor>
<Label Key="L_LOH_VENDOR_ID" Caption="供应商"/>
<Dict Key="LOH_VENDOR_ID" Caption="供应商" ItemKey="VendorCustomer" BuddyKey="L_LOH_VENDOR_ID" AllowMultiSelection="true">
  <Condition CondSign="in" TableKey="WM_LogisticsOrderHead" ColumnKey="LOH_VENDOR_ID"/>
</Dict>
<Label Key="L_LOH_CUSTOMER_ID" Caption="客户"/>
<Dict Key="LOH_CUSTOMER_ID" Caption="客户" ItemKey="VendorCustomer" BuddyKey="L_LOH_CUSTOMER_ID" AllowMultiSelection="true">
  <Condition CondSign="in" TableKey="WM_LogisticsOrderHead" ColumnKey="LOH_CUSTOMER_ID"/>
</Dict>
<Label Key="L_LOH_EMERGENCY_DEGREE_ID" Caption="紧急程度"/>
<Dict Key="LOH_EMERGENCY_DEGREE_ID" Caption="紧急程度" ItemKey="EmergencyDegree" BuddyKey="L_LOH_EMERGENCY_DEGREE_ID"
AllowMultiSelection="true">
  <Condition CondSign="in" TableKey="WM_LogisticsOrderHead" ColumnKey="LOH_EMERGENCY_DEGREE_ID"/>
</Dict>
<Label Key="L_CREATETIME" Caption="托运时间"/>
<DatePicker Key="CREATETIME" Caption="托运时间" BuddyKey="L_CREATETIME">
  <Condition CondSign="between" Group="ORDER_DATETIME" GroupHead="true" TableKey="WM_LogisticsOrderHead"
ColumnKey="CREATETIME"/>
</DatePicker>
<Label Key="L_CREATETIME2" Caption="到"/>
<DatePicker Key="CREATETIME2" Caption="到" BuddyKey="L_CREATETIME2">
  <Condition CondSign="in" Group="ORDER_DATETIME" GroupTail="true" TableKey="WM_LogisticsOrderHead" ColumnKey="CREATETIME"/>
</DatePicker>
<Label Key="L1" Caption=""/>
<Button Key="Query" Caption="查询" Enable="true">
  <OnClick>
    <![CDATA[DealCondition();LoadData();ShowData();]]>
  </OnClick>
</Button>
<Button Key="cancel" Caption="重置" Enable="true">
  <OnClick>
    <![CDATA[ResetCondition();]]>
  </OnClick>
</Button>
</FluidTableLayoutPanel>
<ListView Key="list" TableKey="WM_LogisticsOrderHead">
  <RowDbClick>
    <![CDATA[Open('ConsignOrder2', OID_LV)]]>
  </RowDbClick>
  <ListViewColumnCollection>
    <ListViewColumn Key="OID_LV" Width="30px" Caption="对象标识" ColumnType="NumberEditor" Visible="false" DataColumnKey="OID">
    </ListViewColumn>
    <ListViewColumn Key="LOH_LOGISTICS_NO_LV" Width="200px" Caption="物流订单号" ColumnType="HyperLink"
DataColumnKey="LOH_LOGISTICS_NO">
      <OnClick>
        <![CDATA[ Open('ConsignOrder2', OID_LV) ]]>
      </OnClick>
    </ListViewColumn>
    <ListViewColumn Key="Status_LV" Width="70px" Caption="状态" SourceType="Formula" ColumnType="ComboBox"
DataColumnKey="Status">
      <FormulaItems>
        <![CDATA[GetStatusItems()]]>
      </FormulaItems>
    </ListViewColumn>
    <ListViewColumn Key="LOH_ORG_ID_LV" Width="100px" Caption="机构" ColumnType="Dict" ItemKey="Organization"
DataColumnKey="LOH_ORG_ID"/>
    <ListViewColumn Key="LOH_TYPE_LV" Width="70px" Caption="订单类型" ColumnType="ComboBox" DataColumnKey="LOH_TYPE">
      <Item Key="outbound_deliver" Caption="出库配送" Value="outbound_deliver"/>
      <Item Key="collect_deliver" Caption="提货配送" Value="collect_deliver"/>
      <Item Key="collect_inbound" Caption="提货入库" Value="collect_inbound"/>
      <Item Key="outbound_inbound" Caption="调拨（无运输）" Value="outbound_inbound"/>
      <Item Key="return_transport_inbound" Caption="调拨（含运输）" Value="outbound_transport_inbound"/>
      <Item Key="return_outbound_deliver" Caption="出库退厂" Value="return_outbound_deliver"/>
      <Item Key="return_collect_deliver" Caption="提货退厂" Value="return_collect_deliver"/>
      <Item Key="return_collect_inbound" Caption="提货退库" Value="return_collect_inbound"/>
    </ListViewColumn>
    <ListViewColumn Key="CREATETIME_LV" Width="100px" Caption="托运时间" ColumnType="DatePicker" DataColumnKey="CREATETIME"/>
    <ListViewColumn Key="LOH_OWNER_ID_LV" Width="100px" Caption="货主" ColumnType="Dict" ItemKey="Owner"
DataColumnKey="LOH_OWNER_ID"/>
    <ListViewColumn Key="LOH_EMERGENCY_DEGREE_ID_LV" Width="70px" Caption="紧急程度" ColumnType="Dict" ItemKey="EmergencyDegree"
DataColumnKey="LOH_EMERGENCY_DEGREE_ID"/>
    <ListViewColumn Key="LOH_VENDOR_ID_LV" Width="150px" Caption="供应商" ColumnType="Dict" ItemKey="VendorCustomer"
DataColumnKey="LOH_VENDOR_ID"/>
    <ListViewColumn Key="LOH_CUSTOMER_ID_LV" Width="150px" Caption="客户" ColumnType="Dict" ItemKey="VendorCustomer"
DataColumnKey="LOH_CUSTOMER_ID"/>
  </ListViewColumnCollection>
</ListView>

```

Condition 属性说明

- CondSign 条件符号，取值为=、>、>=、<、<=、<>、between、like、custom、none；

- Group 组标识，如果未定义表示为独立的条件字段；
- GroupHead 在Group不空的情况下，定义是否是组头；
- GroupTail 在Group不空的情况下，定义是否是组尾；
- TableKey 关联的表标识；
- ColumnKey 关联的列标识；
- TargetTableKey 查询的目标表标识；
- OpValue 条件的初始值；
- Impl 定制条件时，条件的实现类全路径名。

效果显示：

物流订单号

状态

接单机构

订单类型

货主/托运商

货主/托运商单号

供应商

客户

紧急程度

托运时间

到

查询

重置

物流订单号	状态	机构	订单类型	托运时间	货主	紧急程度	供应商
-------	----	----	------	------	----	------	-----

如需要对多张表单进行数据查询，可使用参数ConditionTarget

```
<Dict Key="Material_C" Caption="物料" X="2" Y="0" ItemKey="Material" Enable="true">
  <Condition CondSign="=">
    <ConditionTarget TableKey="InventoryQuery2" ColumnKey="Material"/>
    <ConditionTarget TableKey="StockInDetail" ColumnKey="Material"/>
    <ConditionTarget TableKey="StockOutDetail" ColumnKey="Material"/>
  </Condition>
</Dict>
```

物料

数量

到

查询

库存查询

入库明细查询

出库明细查询

物料	地区	数量
----	----	----

第 11 章 数据迁移

目录

11.1. 基本介绍	123
11.2. 入库实例	125
11.3. 出库实例	130
11.4. 退单实例	134
11.5. 期间实例	135

11.1: 基本介绍

数据迁移定义数据对象之间的归集关系；迁移关系通过定义源数据对象中的表集合和迁移表对象之间的表集合，并通过在表字段之间设置迁移关系来定义源字段值的迁移目标字段；迁移关系的目标数据对象中的表只能有一个。



源表出库表：每个仓库中每种物料的出库数据。有两行[上海-电脑]、两行[上海-手机]、三行[北京-电脑]的数据信息。

目标表库存表：每个仓库中每种物料的储量数据，在出库后，需要按照仓库和物料扣减其储量；因此对于源数据中的每对仓库-物料，需要单独扣减其储量值；源表中有两行[上海-电脑]的出库量，合计为1+2=3，在迁移后目标表中[上海-电脑]的数量减去3为97。如下图所示。

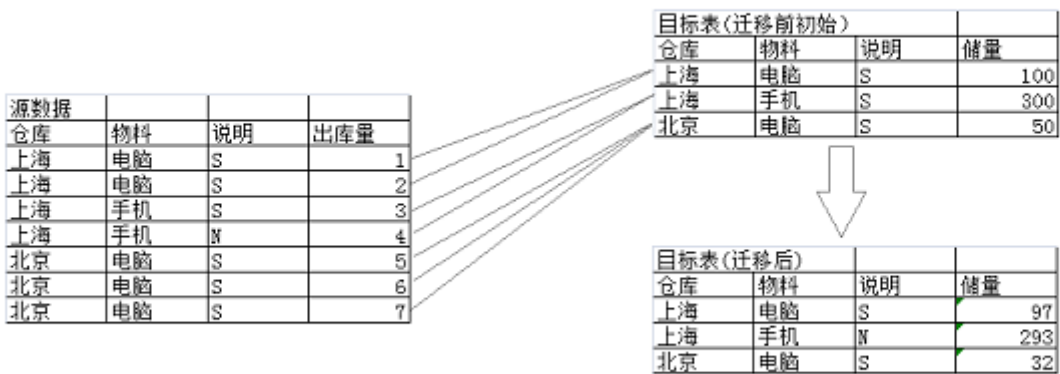


图3-1

11.1.1: 分组策略

数据迁移主要的功能就是数据的归集，通过DataObject中的IsGroup属性定义数据源中的分组字段，数据源表中相同分组值的数据行归集到目标表的同一行，如[图3-1]所示，相同仓库和物料的数据在目标表中也被归集到相同的仓库和物料行；

源数据中的字段值向分组关键字转换有以下两种方法：

- 直接量。取值为Discrete的字段，源值直接转换成分组关键字，当前直接量的分组策略方法无定义；
- 期间分组。定义不同期间数据之间的数据的期初和期末数的处理方式。取值为Peroid的字段，需要通过分组策略将源值转换为分组关键字，分组策略有以下几种(通过GroupingPolicyDef定义)：
 - 无(None)，源值已经是分组关键字，不需要进行任何转换；
 - 年(Year)，源值向其表示的日期所在的年份转换；
 - 月(Month)，源值向其表示的 年-月有转换；
 - 日(Day)，源值向其所表示的年-月-日转换；
 - 值映射(Map)，源值通过映射函数转换为分组关键字(系统扩展保留，暂时未实现)

11.1.2: 值迁移关系

值迁移关系用于定义源数据的值如何向目标数据行进行归集，有以下处理规则：

- 分组字段值的处理方式，分组字段直接将分组关键字的值赋于目标字段，如果目标表中不存在分组关键字集合所定义的行，需要新增一新的数据行，并填充分组关键字集合数据；
- 对于普通的数据字段，其数据的来源有几种形式，分别是：来源于字段(通过Type=Field定义)，即数据来源于源数据的字段域；来源于公式(Type=Formula定义，Definition定义表达式内容)，数据来源于表达式的执行结果；来源于常量(Type=Const定义，Definition定义常量值)；
- 在计算出数据字段的值之后，在将数据字段的值赋于目标的过程中有以下几种方式：
 - 加变化值(OpSign=AddDelta)，在源多次迁移的情况下，将源数据的值的变化量加(或减)到目标数据行的目标字段上；源数据值的变化量加或减到目标字段通过是否负迁移(IsNegative属性定义)来决定，在负迁移的情况下，变化量减到目标字段值上；在这种处理方式下，源值和目标字段均只能是数值类型；
 - 加直接量(OpSign=AddValue)，将源数据值直接加(或)减到目标数据行的目标字段上；同样的，加或减通过是否负迁移属性来定义；在这种处理方式下，源值和目标字段均只能是数值类型；
 - 赋直接量(OpSign=Assign)，将源数据值直接赋予目标数据行的目标字段，目标字段的原值被覆盖，在这种情况下，是否负迁移无意义；


11.1.3: 迁移条件

迁移的条件分为状态条件和数据条件，其中状态条件通过StatusFieldKey和StatusValue来定义，数据条件通过Condition来定义。

状态条件：定义迁移对象满足某个状态条件的情况下，会对数据进行迁移，满足状态是指数据对象的状态小于或者等于设定的状态条件。在状态条件发生逆转的时候需要撤消迁移。

在状态达到满足迁移状态条件的情况下，将数据迁移至目标

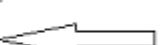
	已审批		
	审批中		
	初始状态		
物料	数量		
电脑	10		



	迁移前	迁移后
物料	数量	
电脑	100	110

在状态从满足条件到不满足条件的情况下，撤销之前的数据迁移

	已审批		
	审批中		
	初始状态		
物料	数量		
电脑	10		



	撤销前	撤销后
物料	数量	
电脑	110	100

数据条件：定义某个迁移对象满足数据条件的情况下，源数据向目标数据迁移；数据撤销迁移的时候同样也是满足条件的数据行的数据被撤销，因此要求在迁移/撤销过程中，数据条件值必须保持不变，否则会导致迁移的撤销错误。

条件：标记='Y'

源数据			
仓库	物料	标记	出库量
上海	电脑	Y	1
上海	电脑	Y	2
上海	手机	Y	3
上海	手机	N	4
北京	电脑	N	5
北京	电脑	N	6
北京	电脑	Y	7

目标表(迁移前初始)

仓库	物料	储量
上海	电脑	100
上海	手机	300
北京	电脑	50

↓

目标表(迁移后)

仓库	物料	储量
上海	电脑	97
上海	手机	297
北京	电脑	43

11.2: 入库实例

一家电子商城新开张，主要销售产品有电脑、手机、平板三种产品，并且在A、B、C三个城市都有仓库。现在三个城市的仓库都是空的，需要分别给三个仓库进货。

11.2.1: 主要信息：

销售产品：电脑、手机、平板

仓库地址：A仓库、B仓库、C仓库

电脑入库量：5000/仓库

手机入库量：10000/仓库

平板入库量：8000/仓库

涉及表单：

入库单序时簿[StockInView]：此表单为序时簿。罗列入库单列表集合，是打开入库单、新建入库单的入口；

入库单[StockIn]：此表单为单据。新建、保存入库信息；

入库库存迁移[StockIn_InventoryList]：此表单为数据迁移。描述入库迁移的详细信息；

库存余额表[InventoryListView]：此表单为序时簿。罗列库存余额表中的所有数据信息；

库存余额表[InventoryList]：此表单为数据对象。是迁移表的数据源。

其他表单：物料[Material] (字典)、仓库[Warehouse] (字典)；

入库成功的条件：当入库单的单据状态变成“审批通过”时，入库成功。

（注意：此次迁移主要描述的是正向迁移）

11.2.2:前期准备：

第一步：新建两个字典，Key和Caption分别为：Material（物料）和Warehouse（仓库）。并把销售产品和仓库地址输入数据。（此处不做字典创建的具体介绍，详情请参考字典介绍）

如图：



第二步：添加状态字段。在CommonDef.xml文件中加入所有状态字段，如下配置：

```
<CommonDef>
  <StatusCollection>
    <Status Key="Init" Caption="初始" Value="0" />
    <Status Key="Auditing" Caption="审批中" Value="1" />
    <Status Key="Audited" Caption="审批通过" Value="2" />
    <Status Key="Denied" Caption="否决" Value="-1" />
  </StatusCollection>
</CommonDef>
```

解析：状态分别为初始、审批中、审批通过、否决。新单据保存后的状态默认为：初始，当单据的状态字段变为“审批通过”时，迁移发生；当单据状态字段变为“否决”时，迁移撤销。

第三步：新建入库单（StockIn），FormType为Entity。入库单包含两个表，入库单头表（WM_StockIn）和入库单明细表（WM_StockInDetail），此表中分别以下字段：


注意：每个数据对象的表中是都必须含有OID[对象标识]、POID[父对象标识]、SOID[主对象标识]、VERID[对象版本]、DVERID[对象明细版本]五个字段

字段名	含义
NO	单据编号
BillDate	单据日期
Warehouse	仓库
Status	状态
Memo	备注
字段名	含义

Material 物料
Amount 数量

入库单的数据对象定义如下：（具体的界面配置在此不作具体介绍，详情请参考控件介绍）

```
<DataObject Key="StockIn" NoPrefix="STIN" Caption="入库单" PrimaryTableKey="WM_StockIn" PrimaryType="Entity"
SecondaryType="Normal">
  <TableCollection>
    <Table Key="WM_StockIn" Caption="基本信息" DBTableName="WM_StockIn" TableMode="Head" SourceType="Table" Persist="True">
      <Column Key="OID" Caption="对象标识" Persist="True" DataType="Long" DBColumnName="OID" />
      <Column Key="POID" Caption="父对象标识" Persist="True" DataType="Long" DBColumnName="POID"/>
      <Column Key="SOID" Caption="主对象标识" Persist="True" DataType="Long" DBColumnName="SOID"/>
      <Column Key="VERID" Caption="对象版本" Persist="True" DataType="Integer" DBColumnName="VERID"/>
      <Column Key="DVERID" Caption="对象明细版本" Persist="True" DataType="Integer" DBColumnName="DVERID"/>
      <Column Key="BillDate" Caption="单据日期" Persist="True" DataType="DateTime" DBColumnName="CREATETIME"/>
      <Column Key="NO" Caption="单据编号" Persist="True" DataType="Varchar" Length="200" DBColumnName="NO"/>
      <Column Key="Warehouse" Caption="仓库" Persist="True" DataType="Long" DBColumnName="WAREHOUSE"/>
      <Column Key="Status" Persist="True" Caption="状态" DataType="Integer" DBColumnName="STATUS"/>
      <Column Key="Memo" Caption="备注" Persist="True" DataType="Varchar" Length="250" DBColumnName="MEMO"/>
    </Table>
    <Table Key="WM_StockInDetail" Caption="入库明细" DBTableName="WM_StockInDetail" TableMode="Detail" SourceType="Table"
Persist="True">
      <Column Key="OID" Caption="对象标识" Persist="True" DataType="Long" DBColumnName="OID" />
      <Column Key="POID" Caption="父对象标识" Persist="True" DataType="Long" DBColumnName="POID"/>
      <Column Key="SOID" Caption="主对象标识" Persist="True" DataType="Long" DBColumnName="SOID"/>
      <Column Key="VERID" Caption="对象版本" Persist="True" DataType="Integer" DBColumnName="VERID"/>
      <Column Key="DVERID" Caption="对象明细版本" Persist="True" DataType="Integer" DBColumnName="DVERID"/>
      <Column Key="Material" Caption="物料" Persist="True" DataType="Long" IsPrimary="True"/>
      <Column Key="Amount" Caption="数量" Persist="True" DataType="Numeric" Precision="16" Scale="2"/>
    </Table>
  </TableCollection>
</DataObject>
```

第四步：创建迁移表，也是库存余额表。在DataObject目录下新建一张数据对象，如： **InventoryList1.xml**。

其中库存余额表的字段信息为：

字段名	含义
Warehouse	仓库
Material	物料
Amount	库存数量

根据上表信息定义InventoryList.xml文件中的配置如下：

```
<DataObject Key="InventoryList" Caption="库存余额表" PrimaryTableKey="WM_InventoryList" PrimaryType="Entity"
SecondaryType="Migration">
  <TableCollection>
    <Table Key="WM_InventoryList" Caption="库存余额表" DBTableName="WM_InventoryList" TableMode="Detail" SourceType="Table"
Persist="True">
      <Column Key="OID" SysKey="OID" Caption="对象标识" Persist="True" DataType="Long" DBColumnName="OID" />
      <Column Key="POID" Caption="父对象标识" Persist="True" DataType="Long" DBColumnName="POID"/>
      <Column Key="SOID" Caption="主对象标识" Persist="True" DataType="Long" DBColumnName="SOID"/>
      <Column Key="VERID" Caption="对象版本" Persist="True" DataType="Integer" DBColumnName="VERID"/>
      <Column Key="DVERID" Caption="对象明细版本" Persist="True" DataType="Integer" DBColumnName="DVERID"/>
      <Column Key="BillDate" Caption="日期" Persist="True" DataType="Date" IsGroup="True" GroupType="Period"/>
      <Column Key="Warehouse" Caption="仓库" Persist="True" DataType="Integer" IsGroup="True" GroupType="Discrete"/>
      <Column Key="Material" Caption="物料" Persist="True" DataType="Integer" IsGroup="True" GroupType="Discrete"/>
      <Column Key="Amount" Caption="库存数量" Persist="True" DataType="Numeric" Precision="16" Scale="0" IsGroup="False"
SplitType="Period"/>
    </Table>
  </TableCollection>
</DataObject>
```

注意：

1. SecondaryType为数据对象的辅助类型，这里需要设置为Migration(迁移表)。
2. Warehouse是分组字段，分组策略类型为直接量。

3. Material是分组字段，分组策略类型为直接量。

4. Amount不是分组字段，无分组策略类型。

第五步：创建迁移余额表[InventoryListView]。此迁移余额表为序时簿：FormType为View。用来展示余额表中的数据。（InventoryListView.xml文件的配置为序时簿的配置方法一致，操作在此不作具体介绍）

11.2.3: 入库迁移配置

新建数据迁移（StockIn_InventoryList.xml）。StockIn_InventoryList.xml文件中的配置如下：

```
<DataMigration Key="StockIn_InventoryList" SrcDataObjectKey="StockIn" TgtDataObjectKey="InventoryList" Caption="入库存存迁移"
Description="" StatusFieldKey="Status" StatusValue="2" Condition="">
  <SourceTableCollection>
    <SourceTable TableKey="WM_StockIn">
      <SourceField Type="Field" Definition="BillDate" OpSign="Assign" IsNegative="False" TargetTableKey="WM_InventoryList"
PeriodGranularity="Day" TargetFieldKey="BillDate"/>
      <SourceField Type="Field" Definition="Warehouse" OpSign="Assign" IsNegative="False" TargetTableKey="WM_InventoryList"
TargetFieldKey="Warehouse"/>
    </SourceTable>
    <SourceTable TableKey="WM_StockInDetail" IsPrimary="True">
      <SourceField Type="Field" Definition="Material" OpSign="Assign" IsNegative="False" TargetTableKey="WM_InventoryList"
TargetFieldKey="Material"/>
      <SourceField Type="Field" Definition="Amount" OpSign="AddDelta" IsNegative="False" TargetTableKey="WM_InventoryList"
TargetFieldKey="Amount"/>
    </SourceTable>
  </SourceTableCollection>
  <TargetTableCollection>
    <TargetTable TableKey="WM_InventoryList">
      <TargetField FieldKey="BillDate"/>
      <TargetField FieldKey="Warehouse"/>
      <TargetField FieldKey="Material"/>
      <TargetField FieldKey="Amount"/>
    </TargetTable>
  </TargetTableCollection>
</DataMigration>
```

配置分析：

- 源数据对象：StockIn。其配置信息：SrcDataObjectKey="StockIn"。（StockIn为入库单的数据对象Key）
- 目标数据对象：InventoryList。其配置信息：TgtDataObjectKey="InventoryList"。（InventoryList为库存余额表的数据对象Key）
- 状态字段：Status。其配置信息：StatusFieldKey="Status"。（Status为WM_StockIn1中的“Status”字段）
- 状态字段值：Audited。其配置信息：StatusValue="2"。（InventoryList为库存余额表的数据对象Key）
- 源表1：WM_StockIn。WM_StockIn为入库单数据对象中的头表，其配置信息：

```
<SourceTable TableKey="WM_StockIn">
  <SourceField Type="Field" Definition="Warehouse" OpSign="Assign" IsNegative="False" TargetTableKey="WM_InventoryList"
TargetFieldKey="Warehouse"/>
</SourceTable>
```

注意：

1. 入库单据是正向迁移，Warehouse为正向迁移，所以IsNegative="False"；

- 源表2： WM_StockInDetail。WM_StockInDetail为入库单数据对象中的明细表，其配置信息为：

```
<SourceTable TableKey="WM_StockInDetail" IsPrimary="True">
```

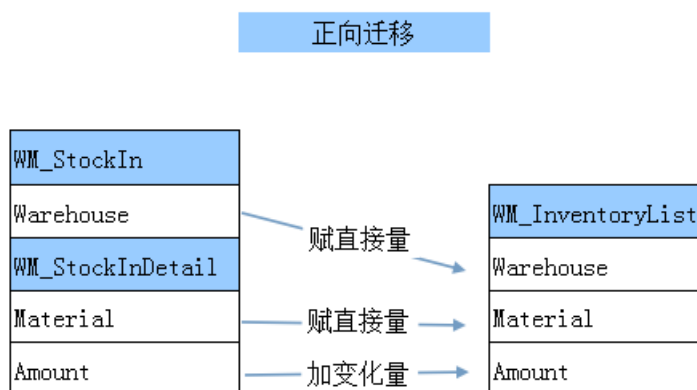
```
<SourceField Type="Field" Definition="Material" OpSign="Assign" IsNegative="False" TargetTableKey="WM_InventoryList"
TargetFieldKey="Material"/>
<SourceField Type="Field" Definition="Amount" OpSign="AddDelta" IsNegative="False" TargetTableKey="WM_InventoryList"
TargetFieldKey="Amount"/>
</SourceTable>
```

注意：

1. 入库单据是正向迁移，Material和Amount都为正向迁移，所以IsNegative="False"；
 2. Material的值处理方式是：直接把WM_StockInDetail中的Material字段值“赋给”目标WM_InventoryList的Material字段，所以Material的 OpSign="Assign"。
 3. Amount的值处理方式是：把WM_StockInDetail中的Amount字段值“加到”目标WM_InventoryList的Amount字段中原有值上，所以Amount的 OpSign="AddDelta"。
- 目标表： WM_InventoryList。WM_InventoryList为InventoryList数据对象中的表，其配置信息为：

```
<TargetTable TableKey="WM_InventoryList">
<TargetField FieldKey="Warehouse"/>
<TargetField FieldKey="Material"/>
<TargetField FieldKey="Amount"/>
</TargetTable>
```

迁移配置展现效果如下图：



11.2.4: 入库迁移效果展示

步骤一：填写“入库单”。此处没有做流程操作，为展示效果，直接把状态（Status）字段设置为“审批通过（Audited）”。

步骤二：打开迁移表。这时就能够发现“库存余额表”中的数据已经发生改变。A仓库中已经有电脑5000台，手机10000部，平板8000，如图：

入库单1

获取时间

编辑

删除

下推

预览

审批记录

单据编号

STIN20150427000002

单据日期

2015-04-27

仓库

C001 A仓库

状态

审批通过

备注

	物料	数量
1	w001 电脑	5,000.00
2	w002 手机	10,000.00
3	w003 平板	8,000.00

库存余额表

序号	OID	仓库	物料	库存量
1	12205	C001 A仓库	w001 电脑	5,000.00
2	12207	C001 A仓库	w002 手机	10,000.00
3	12209	C001 A仓库	w003 平板	8,000.00

首页

上页

1

下页

尾页

刷新

1

页

确定

步骤三：照上步骤，依次给B，C两个仓库补充货源，最后库存余额表的数据如图：

库存余额表				
序号	OID	仓库	物料	库存量
1	12205	C001 A仓库	w001 电脑	5,000.00
2	12207	C001 A仓库	w002 手机	10,000.00
3	12209	C001 A仓库	w003 平板	8,000.00
4	12305	C002 B仓库	w001 电脑	5,000.00
5	12307	C002 B仓库	w002 手机	10,000.00
6	12309	C002 B仓库	w003 平板	8,000.00
7	12315	C003 C仓库	w001 电脑	5,000.00
8	12317	C003 C仓库	w002 手机	10,000.00
9	12319	C003 C仓库	w003 平板	8,000.00

1/1

首页

上页

1

下页

尾页

刷新

1

页

确定

11.3: 出库实例

新店开张了，生意红火，当天A城市和B城市分别先后签订一个大订单。A城市的订单签给了A客户，B城市的订单签给了B客户。

11.3.1: 主要信息

订单A：电脑（1000）、手机（2000）、平板（800），从A仓库出库。

订单B：电脑（800）、手机（1200）、平板（1000），从B仓库出库。

涉及表单：

出库单序时簿[StockOutView]：此表单为序时簿。罗列出库单列表集合，是打开出库单、新建出库单的入口；

出库单[StockOut]：此表单为单据。新建、保存出库信息；

出库库存迁移[StockOut_InventoryList]：此表单为数据迁移。描述出库迁移的详细信息；

库存余额表[InventoryListView]：此表单为序时簿。罗列库存余额表中的所有数据信息；

库存余额表1[InventoryList]：此表单为数据对象。是迁移表的数据源。

其他表单：物料[Material]（字典）、仓库[Warehouse]（字典）；

出库条件：当出库单的单据状态变成“审批通过”时，出库成功。

出库撤销：当出库单的单据状态变成“否决”时，出库撤销。

（注意：此次迁移主要描述的是负向迁移，并且对于出库单序时簿 和库存余额表序时簿的配置省略）

11.3.2: 前期准备

新建出库单（StockOut），FormType为Entity。出库单包含两个表，出库单头表（WM_StockOut）和出库单明细表（WM_StockOutDetail），此表中分别以下字段：

注意：每个数据对象的表中是都必须含有OID[对象标识]、POID[父对象标识]、SOID[主对象标识]、VERID[对象版本]、DVERID[对象明细版本]五个字段

字段名	含义
NO	单据编号
BillDate	单据日期
Warehouse	仓库
Status	状态
Memo	备注

字段名	含义
Material	物料
Amount	数量

出库单的数据对象定义如下：

```
<DataObject Key="StockOut" NoPrefix="STIN" Caption="出库单" PrimaryTableKey="WM_StockOut" PrimaryType="Entity"
SecondaryType="Normal">
  <TableCollection>
    <Table Key="WM_StockOut" Caption="基本信息" DBTableName="WM_StockOut" TableMode="Head" SourceType="Table" Persist="True">
      <Column Key="OID" Caption="对象标识" Persist="True" DataType="Long" DBColumnName="OID" DefaultFormulaValue="SOID+BillNO"/>
      <Column Key="POID" Caption="父对象标识" Persist="True" DataType="Long" DBColumnName="POID"/>
      <Column Key="SOID" Caption="主对象标识" Persist="True" DataType="Long" DBColumnName="SOID"/>
      <Column Key="VERID" Caption="对象版本" Persist="True" DataType="Integer" DBColumnName="VERID"/>
      <Column Key="DVERID" Caption="对象明细版本" Persist="True" DataType="Integer" DBColumnName="DVERID"/>
      <Column Key="BillDate" Caption="单据日期" Persist="True" DataType="DateTime" DBColumnName="CREATETIME"/>
      <Column Key="NO" Caption="单据编号" Persist="True" DataType="Varchar" Length="200" DBColumnName="NO"/>
      <Column Key="Warehouse" Caption="仓库" Persist="True" DataType="Long"/>
      <Column Key="Status" Persist="True" Caption="状态" DataType="Integer" />
      <Column Key="Memo" Caption="备注" Persist="True" DataType="Varchar" Length="250"/>
    </Table>
    <Table Key="WM_StockOutDetail" Caption="出库明细" DBTableName="WM_StockOutDetail" TableMode="Detail" SourceType="Table"
Persist="True">
      <Column Key="OID" Caption="对象标识" Persist="True" DataType="Long" DBColumnName="OID" />
      <Column Key="POID" Caption="父对象标识" Persist="True" DataType="Long" DBColumnName="POID"/>
      <Column Key="SOID" Caption="主对象标识" Persist="True" DataType="Long" DBColumnName="SOID"/>
      <Column Key="VERID" Caption="对象版本" Persist="True" DataType="Integer" DBColumnName="VERID"/>
      <Column Key="DVERID" Caption="对象明细版本" Persist="True" DataType="Integer" DBColumnName="DVERID"/>
      <Column Key="OID" Caption="对象标识" Persist="True" DataType="Long" DBColumnName="OID" />
      <Column Key="Seq" Caption="序号" Persist="True" DBColumnName="Sequence" DataType="Integer" IsSequence="true"/>
      <Column Key="Material" Caption="物料" Persist="True" DataType="Long" IsPrimary="True"/>
      <Column Key="Amount" Caption="数量" Persist="True" DataType="Numeric" Precision="16" Scale="2"/>
    </Table>
  </TableCollection>
</DataObject>
```

11.3.3: 出库迁移

新建数据迁移（StockOut_InventoryList.xml）。StockOut_InventoryList.xml文件中的配置如下：

```
<DataMigration Key="StockOut_InventoryList" SrcDataObjectKey="StockOut" TgtDataObjectKey="InventoryList" Caption="出库库存迁
移" Description="" StatusFieldKey="Status" StatusValue="2" Condition="">
```



```

<SourceTableCollection>
  <SourceTable TableKey="WM_StockOut">
    <SourceField Type="Field" Definition="Warehouse" OpSign="Assign" IsNegative="True" TargetTableKey="WM_InventoryList"
    TargetFieldKey="Warehouse"/>
  </SourceTable>
  <SourceTable TableKey="WM_StockOutDetail" IsPrimary="True">
    <SourceField Type="Field" Definition="Material" OpSign="Assign" IsNegative="True" TargetTableKey="WM_InventoryList"
    TargetFieldKey="Material"/>
    <SourceField Type="Field" Definition="Amount" OpSign="AddDelta" IsNegative="True" TargetTableKey="WM_InventoryList"
    TargetFieldKey="Amount"/>
  </SourceTable>
</SourceTableCollection>
<TargetTableCollection>
  <TargetTable TableKey="WM_InventoryList">
    <TargetField FieldKey="Warehouse"/>
    <TargetField FieldKey="Material"/>
    <TargetField FieldKey="Amount"/>
  </TargetTable>
</TargetTableCollection>
</DataMigration>

```

配置分析：

- 源数据对象：StockOut1。其配置信息：SrcDataObjectKey="StockOut"。（StockOut为出库单的数据对象的Key）
- 目标数据对象：InventoryList1。其配置信息：TgtDataObjectKey="InventoryList"。（InventoryList为库存余额表的数据对象的Key）
- 状态字段：Status。其配置信息：StatusFieldKey="Status"。（Status为WM_StockOut中的“Status”字段）
- 状态字段值：Audited。其配置信息：StatusValue="2"。（InventoryList为库存余额表的数据对象Key）
- 源表1：WM_StockOut。WM_StockOut为出库单数据对象中的头表，其配置信息：

```

<SourceTable TableKey="WM_StockOut">
  <SourceField Type="Field" Definition="Warehouse" OpSign="Assign" IsNegative="True" TargetTableKey="WM_InventoryList"
  TargetFieldKey="Warehouse"/>
</SourceTable>

```

注意：

1. 出库单据为负向迁移，Warehouse为负向迁移，所以IsNegative="True"；

- 源表2： WM_StockOutDetail。WM_StockOutDetail为出库单数据对象中的明细表，其配置信息为：

```

<SourceTable TableKey="WM_StockOutDetail" IsPrimary="True">
  <SourceField Type="Field" Definition="Material" OpSign="Assign" IsNegative="True" TargetTableKey="WM_InventoryList"
  TargetFieldKey="Material"/>
  <SourceField Type="Field" Definition="Amount" OpSign="AddDelta" IsNegative="True" TargetTableKey="WM_InventoryList"
  TargetFieldKey="Amount"/>
</SourceTable>

```

注意：

1. 出库单据是负向迁移，Material和Amount都为负向迁移，所以IsNegative="True"；

2. Material的值处理方式是：直接把WM_StockInDetail中的Material字段值“赋给”目标WM_InventoryList的Material字段，所以Material的 OpSign="Assign"。

3. Amount的值处理方式是：把WM_StockInDetail中的Amount字段值“加到”目标WM_InventoryList的Amount字段中原有值上，所以Amount的 OpSign="AddDelta"。

- 目标表： WM_InventoryList。WM_InventoryList为InventoryList数据对象中的表，其配置信息为：

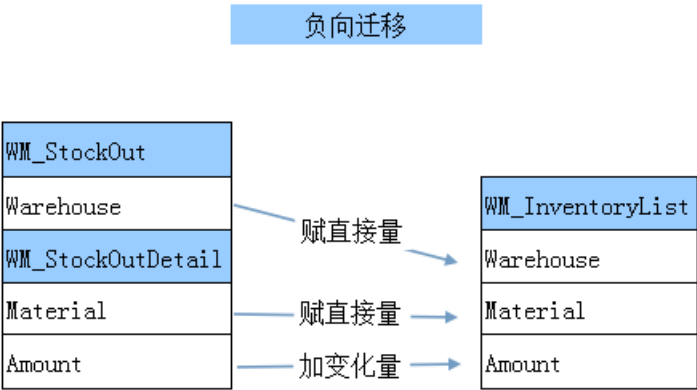
```

<TargetTable TableKey="WM_InventoryList">

```

```
<TargetField FieldKey="Warehouse"/>
<TargetField FieldKey="Material"/>
<TargetField FieldKey="Amount"/>
</TargetTable>
```

迁移配置展现效果如下图：



11.3.4: 出库迁移效果展示

步骤一：填写“出库单”，完成订单A的操作。此处没有做流程图，为展示效果，直接把状态（Status）字段设置为“审批通过（Audited）”。

步骤二：打开迁移表。这时就能够发现“库存余额表”中的数据已经发生改变。A仓库中已经有电脑5000台，手机10000部，平板8000，如图：

订单A是从A仓库中出库了电脑（1000）、手机（2000）、平板800。在状态为“审批通过”时，数据发生迁移，库存余额表发生变化。图中可以看出：
A仓库：电脑由5000→4000，减少1000；
手机由10000→8000，减少2000；
平板由8000→7200，减少800；

出库单1

编辑 下推 删除

单据编号: STIN20150428000001

单据日期: 2015-04-28

仓库: C001 A仓库

状态: 审批通过

备注: 订单A出库

物料	数量
1 w001 电脑	1,000.00
2 w002 手机	2,000.00
3 w003 平板	800.00

库存余额表

序号	OID	仓库	物料	库存量
1	12205	C001 A仓库	w001 电脑	5,000.00
2	12207	C001 A仓库	w002 手机	10,000.00
3	12209	C001 A仓库	w003 平板	8,000.00
4	12305	C002 B仓库	w001 电脑	5,000.00
5	12307	C002 B仓库	w002 手机	10,000.00
6	12309	C002 B仓库	w003 平板	8,000.00
7	12315	C003 C仓库	w001 电脑	5,000.00
8	12317	C003 C仓库	w002 手机	10,000.00
9	12319	C003 C仓库	w003 平板	8,000.00

出库前

库存余额表

序号	OID	仓库	物料	库存量
1	12205	C001 A仓库	w001 电脑	4,000.00
2	12207	C001 A仓库	w002 手机	8,000.00
3	12209	C001 A仓库	w003 平板	7,200.00
4	12305	C002 B仓库	w001 电脑	5,000.00
5	12307	C002 B仓库	w002 手机	10,000.00
6	12309	C002 B仓库	w003 平板	8,000.00
7	12315	C003 C仓库	w001 电脑	5,000.00
8	12317	C003 C仓库	w002 手机	10,000.00
9	12319	C003 C仓库	w003 平板	8,000.00

出库后

步骤三：填写“出库单”，完成订单B的操作，操作步骤同订单A的操作一样。订单B的出库量为：电脑（800）、手机（1200）、平板（1000），效果如图：

订单B是从B仓库中出库了电脑（800）、手机（122000）、平板（1000）。在状态为“审批通过”时，数据发生迁移，库存余额表发生变化。图中可以看出：
B仓库：电脑由5000→4200，减少800；
手机由10000→8800，减少1200；
平板由8000→7000，减少1000；

出库单1

编辑 下推 删除

单据编号: STIN20150428000002

单据日期: 2015-04-28

仓库: C002 B仓库

状态: 审批通过

备注: 订单B出库

	物料	数量
1	w001 电脑	800.00
2	w002 手机	1,200.00
3	w003 平板	1,000.00

库存余额表

序号	OID	仓库	物料	库存量
1	12205	C001 A仓库	w001 电脑	4,000.00
2	12207	C001 A仓库	w002 手机	8,000.00
3	12209	C001 A仓库	w003 平板	7,200.00
4	12305	C002 B仓库	w001 电脑	4,200.00
5	12307	C002 B仓库	w002 手机	8,800.00
6	12309	C002 B仓库	w003 平板	7,000.00
7	12315	C003 C仓库	w001 电脑	5,000.00
8	12317	C003 C仓库	w002 手机	10,000.00
9	12319	C003 C仓库	w003 平板	8,000.00

出库后

11.4: 退单实例

B客户在签订成功后，因为一些特殊原因，B客户希望能够退货。撤销B订单的操作。

此时：客户B如果要完成退单步骤，订单B要撤销，相应的库存也应该发生改变（还原出库前的数量）。而在撤销B订单时，订单B的状态由“审批通过”到“否决”状态。

注意：因为此处没有做流程处理，这里我们直接把订单B的状态改成“否决”，最终效果如下图：

撤销订单B后，B订单的状态变为“否决”时，B订单迁移撤销，库存余额表发生变化。图中可以看出：
B仓库：电脑由4200→5000，还原800；
手机由88000→1000，还原1200；
平板由7000→8000，还原1000；

出库单1

编辑 下推 删除

单据编号: STIN20150428000002

单据日期: 2015-04-28

仓库: C002 B仓库

状态: 否决

备注: 订单B出库(退货处理)

物料	数量
1 w001 电脑	800.00
2 w002 手机	1,200.00
3 w003 平板	1,000.00

库存余额表 (退货前)

序号	OID	仓库	物料	库存量
1	12205	C001 A仓库	w001 电脑	4,000.00
2	12207	C001 A仓库	w002 手机	8,000.00
3	12209	C001 A仓库	w003 平板	7,200.00
4	12305	C002 B仓库	w001 电脑	4,200.00
5	12307	C002 B仓库	w002 手机	8,800.00
6	12309	C002 B仓库	w003 平板	7,000.00
7	12315	C003 C仓库	w001 电脑	5,000.00
8	12317	C003 C仓库	w002 手机	10,000.00
9	12319	C003 C仓库	w003 平板	8,000.00

库存余额表 (退货后)

序号	OID	仓库	物料	库存量
1	12205	C001 A仓库	w001 电脑	4,000.00
2	12207	C001 A仓库	w002 手机	8,000.00
3	12209	C001 A仓库	w003 平板	7,200.00
4	12305	C002 B仓库	w001 电脑	5,000.00
5	12307	C002 B仓库	w002 手机	10,000.00
6	12309	C002 B仓库	w003 平板	8,000.00
7	12315	C003 C仓库	w001 电脑	5,000.00
8	12317	C003 C仓库	w002 手机	10,000.00
9	12319	C003 C仓库	w003 平板	8,000.00

11.5: 期间实例

经过一段时间后，商场的入库量和出库量逐步稳定。商场每月都会对每个仓库商品的入库和出库量做一个总结和对比。为达到此目的，商场需要做一个期间粒度为月的分组迁移。

修改InventoryList1.xml文件：添加月份期间分组字段，库存数量设置为余额字段。

修改后的InventoryList.xml文件配置如下：

```
<DataObject Key="InventoryList" Caption="库存余额表" PrimaryTableKey="WM_InventoryList" PrimaryType="Entity"
SecondaryType="Normal">
<TableCollection>
<Table Key="WM_InventoryList" Caption="库存余额表" DBTableName="WM_InventoryList" TableMode="Detail" SourceType="Table"
Persist="True">
<Column Key="OID" SysKey="OID" Caption="对象标识" Persist="True" DataType="Long" DBColumnName="OID" />
<Column Key="POID" Caption="父对象标识" Persist="True" DataType="Long" DBColumnName="POID" />
<Column Key="SOID" Caption="主对象标识" Persist="True" DataType="Long" DBColumnName="SOID" />
<Column Key="VERID" Caption="对象版本" Persist="True" DataType="Integer" DBColumnName="VERID" />
<Column Key="DVERID" Caption="对象明细版本" Persist="True" DataType="Integer" DBColumnName="DVERID" />

<Column Key="MonthPeriod" Caption="月份" Persist="True" DataType="Integer" IsGroup="True" GroupType="Peroid" />
<Column Key="Warehouse" Caption="仓库" Persist="True" DataType="Integer" IsGroup="True" GroupType="Discrete" />
<Column Key="Material" Caption="物料" Persist="True" DataType="Integer" IsGroup="True" GroupType="Discrete" />
<Column Key="Amount" Caption="库存数量" Persist="True" DataType="Numeric" Precision="16" Scale="0" IsGroup="False"
SplitType="Period" />
</Table>
</TableCollection>
</DataObject>
```

注意：

1. 期间粒度：取值为Year(年份)、Month(月份)、Day(天)、FiscalMonth(会计月份)。当期间粒度为Year、Month、Day、FiscalMonth时，其对于的期间分组字段的数据类型为“Integer”，而不是“DateTime”；所以MonthPeriod的DataType = “Integer”；
2. 同时MonthPeriod也是属于分组字段，分组类型为期间，所以IsGroup="True"和GroupType="Peroid"；

3. Amount字段需要设置成余额字段，即SplitType="Period"。注意：如果Amount没有设置成余额字段，期间分组字段的设置等于无意义。而如果设置了Amount为余额字段时，期间分组字段是必须设置的。
4. InventoryList配置成功后，启动Tomcat，在数据库中会自动生成以“_beginning”和“_ending”结尾的“期初”和“期末”字段，所以“期初”和“期末”字段的列明分别为“MonthPeriod_beginning”和“MonthPeriod_ending”。
5. 在数据结转计算中，在数据库中会自动生成一张以“_lastpoint”为后缀名的表。此表主要是来保存在期间的最后结转节点的数据。所以此迁移会自动生成：InventoryList_lastpoint。

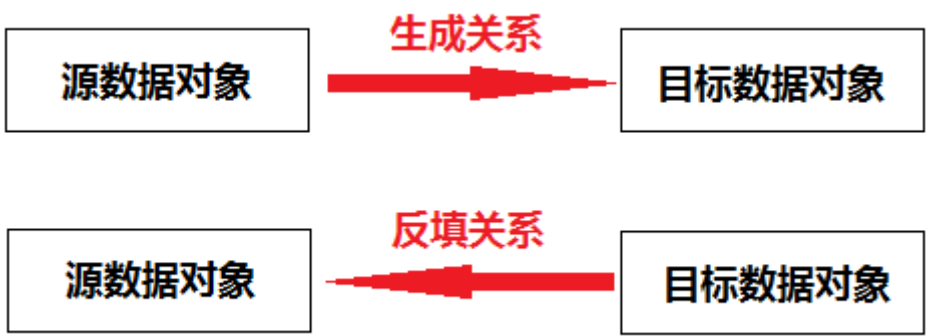
第 12 章 数据映射

目录

12.1. 基本介绍	137
12.2. 示例1(普通映射)	141
12.3. 示例2(最大可映射值&多种值映射)	144
12.4. 示例3(可用映射量计算)	145
12.5. 示例4(数据反填&反填条件)	147
12.6. 示例5(跨级反填)	148
12.7. 示例6(上引下推)	150

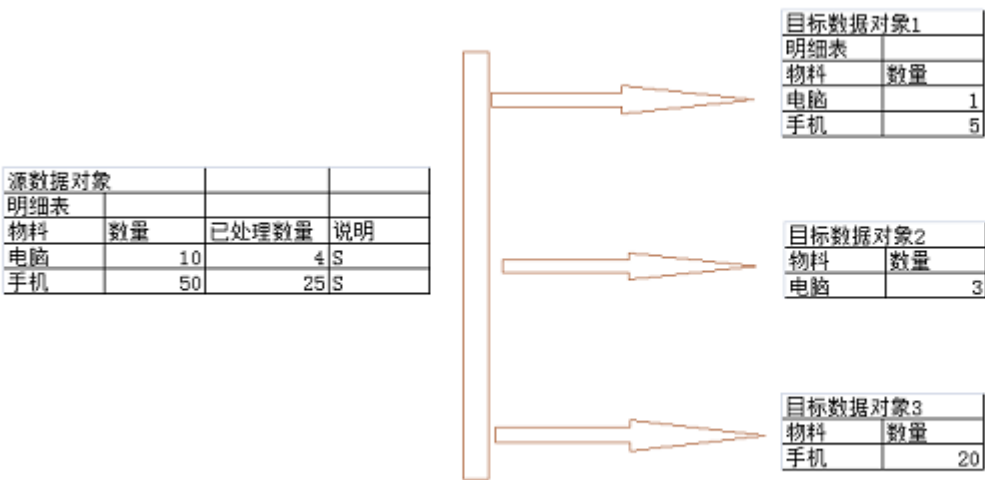
12.1:基本介绍

数据映射就是源数据对象与目标数据对象之间的数据生成关系；以及目标数据对象与源数据对象(或其它数据对象)之间的反向填值关系。如下图所示：



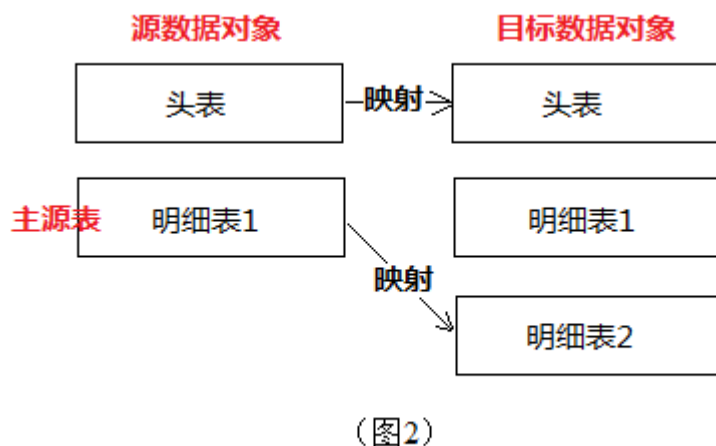
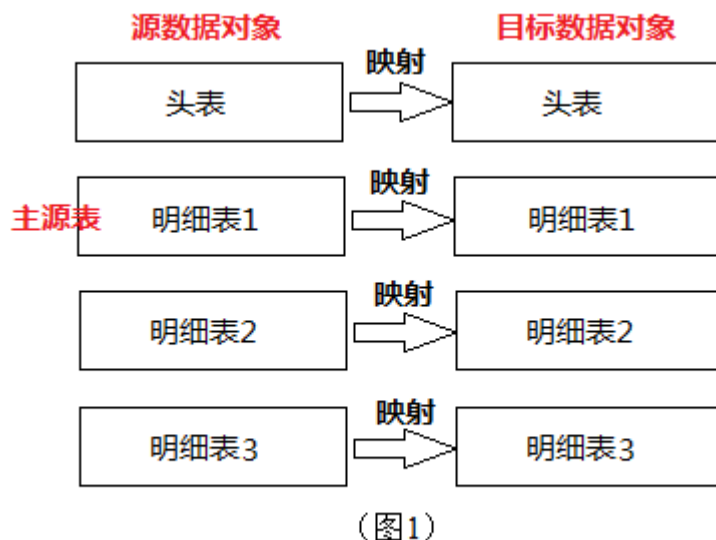
12.1.1:数据生成

说明如何从源数据对象生成目标数据对象，每个源数据对象可以生成多个目标数据对象。如下图所示：源数据对象生成了目标数据对象1、目标数据对象2、目标数据对象3。



12.1.2: 数据生成的规则

数据对象由多张表的数据按层次结构组成，所有的头表数据均属于第一层次，明细表1属于第二层次，明细表2分别属于自己的父层次明细的下层；数据映射的目的就是将在主源表中数据行定义的逻辑数据单元映射到目标数据对象中生成相关的逻辑数据单元。



注意

映射形式有2种：

如图1所示，在字段映射关系的定义中，根据主源表—主目标表配对，源值映射相同层级的数据上。源数据头表对应目标数据头表，源数据明细1对应目标数据明细1，源数据明细2对应目标数据明细2，源数据明细3对应目标数据明细3。

如图2所示，源值只能映射到较低层级的数据上，而不能将低层级的数据映射高层级的数据上。可以将源表[子明细1]中的数据向目标表[子明细1]、[子明细2]进行映射，但不能将源表[子明细2]的数据向目标表[子明细1]及以上层次的表进行映射；

12.1.3: 值映射：

映射过程中，源字段值的来源Definition有以下方式：

- 来源于字段 (Type=Field定义)，即数据来源于源表中的字段值；
- 来源于公式 (Type=Formula定义)，即数据来源于Definition定义的公式的计算值；
- 来源于常量 (Type=Const定义)，即数据来源于Definition定义的常量值；

值的映射关系类型EdgeType如下：

- 普通数据映射 (EdgeType=Normal)，数据的映射过程为一般的数据处理方式，不做任何任何逻辑上的判断；
- 关注字段映射 (EdgeType=Focus)，数据的映射过程为需要关注数量的处理方式，计算过程中需要做可用量计算，在满足关注字段规则的情况下，生成新的目标数据；
- 关系映射 (EdgeType=Relation)，数据的映射为关系树关联字段的处理方式，只能将源表中的OID字段的值映射到目标，同时此属必也标识了源—目标表配对关系；

说明，在值的映射关系类型定义中，关系映射和关注字段映射只能是定义在主源表—主目标表配对上，事实上，主源表和主目标表配对是通过关系映射所在的表来确定；

12.1.4: 可用映射量计算：

可用映射量计算针对主源表中需要下推的每行数据进行计算，在一个单行的数据中，可用映射量计算考虑在当前行、指定映射关系的情况下，还有多少可以映射的关注字段值。

例如：一行数据针对某个映射的已映射量定义为AMV (MapKey)；那么SrcObj1中的行的针对S1_Obj1已映射量，计算方法为AMV (S1_Obj1)=10+20=30；SrcObj1中的行针对S1_Obj2的已映射量，计算方法为AMV (S1_Obj2)=50+50=100；

源数据对象 (SrcObj1)	
主源表	
OID	FocusValue
1000	300

目标数据对象1 (TgtObj1)			
主目标表			
OID	SrcOID	MapKey	FocusValue
1001	1000	S1_Obj1	10
1002	1000	S1_Obj1	20

目标数据对象2 (TgtObj2)			
主目标表			
OID	SrcOID	MapKey	FocusValue
1003	1000	S1_Obj2	50
1004	1000	S1_Obj2	50

第一种，TgtObj1扣减可用映射量，TgtObj2扣减可用映射量，那么SrcObj1中的总的可用映射量为 $300 - \text{AMV}(\text{S1_Obj1}) - \text{AMV}(\text{S1_Obj2}) = 300 - 30 - 100 = 170$ ；

第二种，TgtObj1增加可用映射量，TgtObj2扣减可用映射量，那么SrcObj1中的总的可用映射量为 $300 + \text{AMV}(\text{S1_Obj1}) - \text{AMV}(\text{S1_Obj2}) = 300 + 30 - 100 = 230$ ；

12.1.5: 关注字段规则

这部分定义一个数据字段所在的数据行如何生成目标数据，这里只描述关注字段的值生成规则，规则如下：

- 在允许超量的情况下，始终根据最大映射量 (m) 计算当前行的关注字段值 (f)，如果这个值比最小映射量大，则生成关注字段值为 f 的目标数据行，否则略过当前行；

- 在不允许超量的情况下，计算最大映射量(m)和可用映射量(a)，取其中的最小值取得关注字段值(f)，如果这个值比最小映射量大，则生成关注字段值为f的目标数据行，否则略过当前行。

12.1.6: 分数据映射:

分数据映射用于中间层自动通过映射产生后续数据的情况，分数据通过接口实现对同一个数据对象分段映射产生一个或多个目标数据对象。分数据策略包括三种：无拆分、分组和自定义，无拆分将整单表单映射为一个目标对象，这里不再介绍，后面主要说明分组和自定义。

分组通过将源数据分组的方式生成多个目标对象，在分组的情况下Type取值为Group。

自定义方式的分数据根据二次开发的代码来生成分数据并生成式个目标，Type的取值为Custom。

12.1.7: 数据反填:

数据反填：指的是在映射的目标数据对象保存时，当数据满足反填条件后，将数据回填至源数据对象的过程。

跨级反填：将数据回填至映射的源数据对象的上级映射，称为跨级反填。例如：A映射至B映射至C，C反填至A，这个过程称之为跨级反填。

反填条件包括状态条件和数据条件。

第一级条件是反填源对象的状态条件：没有定义的情况下不控制状态条件，如果定义了状态条件，那么反填源对象的状态大于或等于该状态值时反填；

第二级条件是数据的条件，满足条件的数据行需要反填。

反填对于值的处理方式有两种，一种是加变化量，另外一种是赋直接量。

例如：

源数据对象 (SrcObj1)		
主源表		
OID	FocusValue	ProcessedValue
1000	300	30

目标数据对象1 (TgtObj1)			
主目标表			
OID	SrcOID	MapKey	FocusValue
1001	1000	S1_Obj1	10
1002	1000	S1_Obj1	20

- 加变化量，如上图所示，TgtObj1中的目标数据行，分别将自己的变化量10和20加到源数据行的ProcessedValue中，ProcessedValue的值为30；
- 赋直接量，如上图所示，在处理TgtObj1的第一行数据时，源数据行的ProcessedValue的值为10，在处理TgtObj1的第二行数据时，源数据行的ProcessedValue的值为20；

注意：反填的源数据行通过数据映射关系树逐级向上查询，如果映射关系未定义关系映射，那么不能支持反填操作；

12.1.8: 上引下推

将不同来源的多张单据下推到一张表单中，称为上引下推。比如说有些用户需要将一张或多张表单中的数据，通过查询后下推到一张表单中。如下图所示。从三张表单中查找出上海仓库的所有数据，并下推到一张新表单中去。具体配置参见示例。



12.2: 示例1(普通映射)

公司接到一笔订单，根据订单进行统一采购。订单主要包括电脑、手机、平板。将订单中的物料数量映射到采购单中去。

源数据对象：订单		目标数据对象：采购单	
物料	数量	物料	数量
电脑	50	电脑	50
手机	100	手机	100
平板	80	平板	80

涉及表单：

订单采购单映射表[Order2Purchase]：；此表单为数据映射。描述订单映射到采购单的详细信息；

其他表单：订单、订单序时簿、采购单、采购序时簿；

涉及字典：物料字典；

操作步骤：

第一步：新建一个物料字典Material。并把物料产品输入数据（字典创建详情请参考字典介绍）

第二步：新建订单（Order）、订单序时簿（OrderView）、采购单（Purchase）、采购单序时簿（PurchaseView）。具体参见代码定义。

订单包含2个表，订单头表（WM_Order）、订单明细表（WM_OrderDetail），此表中分别以下字段：

字段	含义
BillDate	订单日期
No	订单序号

字段	含义
Material	物料

Amount 数量

采购单包含2个表，采购单头表（WM_Purchase）、采购单明细表（WM_PurchaseDetail），此表中分别以下字段：

字段 含义
BillDate 单据日期
No 单据序号

字段 含义
Material 物料
Amount 数量

每个数据对象的表中注意：是都必须含有OID[对象标识]、POID[父对象标识]、SOID[主对象标识]、VERID[对象版本]、DVERID[对象明细版本]五个字段。

在Purchase.xml 的配置文件中，必须包含MAPKEY字段。

```
<DataObject Key="Purchase" NoPrefix="STIN" Caption="采购单" PrimaryTableKey="WM_Purchase" PrimaryType="Entity"
SecondaryType="Normal">
<TableCollection>
<Table Key="WM_Purchase" Caption="基本信息1" DBTableName="WM_Purchase" TableMode="Head" SourceType="Table" Persist="True">
<Column Key="OID" Caption="对象标识" Persist="True" DataType="Long" DBColumnName="OID" DefaultFormulaValue="SOID+BillNO"/>
<Column Key="POID" Caption="父对象标识" Persist="True" DataType="Long" DBColumnName="POID"/>
<Column Key="SOID" Caption="主对象标识" Persist="True" DataType="Long" DBColumnName="SOID"/>
<Column Key="VERID" Caption="对象版本" Persist="True" DataType="Integer" DBColumnName="VERID"/>
<Column Key="DVERID" Caption="对象明细版本" Persist="True" DataType="Integer" DBColumnName="DVERID"/>
<Column Key="Purchase_BillDate" Caption="单据日期" Persist="True" DataType="DateTime" DBColumnName="CREATETIME"/>
<Column Key="Purchase_NO" Caption="单据编号" Persist="True" DataType="Varchar" Length="200" DBColumnName="NO"/>
<Column Key="Memo" Caption="备注" Persist="True" DataType="Varchar" Length="250"/>
</Table>
<Table Key="WM_PurchaseDetail" Caption="采购明细" DBTableName="WM_PurchaseDetail" TableMode="Detail" SourceType="Table"
Persist="True">
<Column Key="OID" Caption="对象标识" Persist="True" DataType="Long" DBColumnName="OID" />
<Column Key="POID" Caption="父对象标识" Persist="True" DataType="Long" DBColumnName="POID"/>
<Column Key="SOID" Caption="主对象标识" Persist="True" DataType="Long" DBColumnName="SOID"/>
<Column Key="VERID" Caption="对象版本" Persist="True" DataType="Integer" DBColumnName="VERID"/>
<Column Key="DVERID" Caption="对象明细版本" Persist="True" DataType="Integer" DBColumnName="DVERID"/>
<Column Key="Purchase_Material" Caption="物料" Persist="True" DataType="Long" IsPrimary="True"/>
<Column Key="Purchase_Amount" Caption="数量" Persist="True" DataType="Numeric" Precision="16" Scale="2"/>
<Column Key="SrcOID" Caption="" Persist="True" DataType="Long" Length="250"/>
<Column Key="MAPKEY" Caption="" Persist="True" DataType="Varchar" Precision="100" Scale="2"/>
</Table>
</TableCollection>
<PreSaveProcess>
</PreSaveProcess>
</DataObject>
```

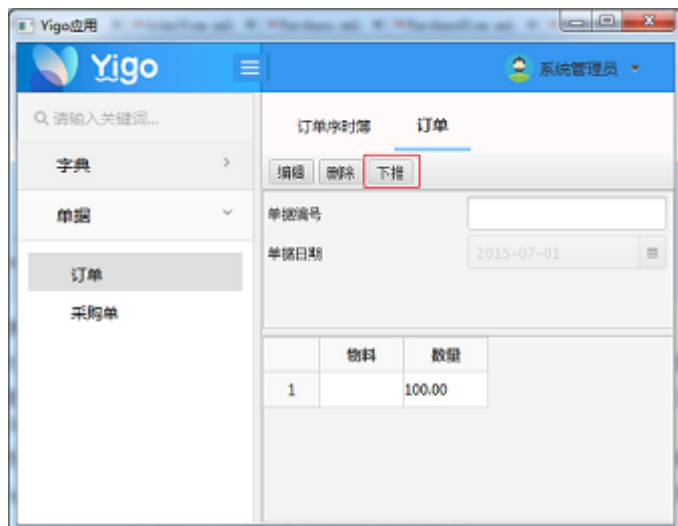
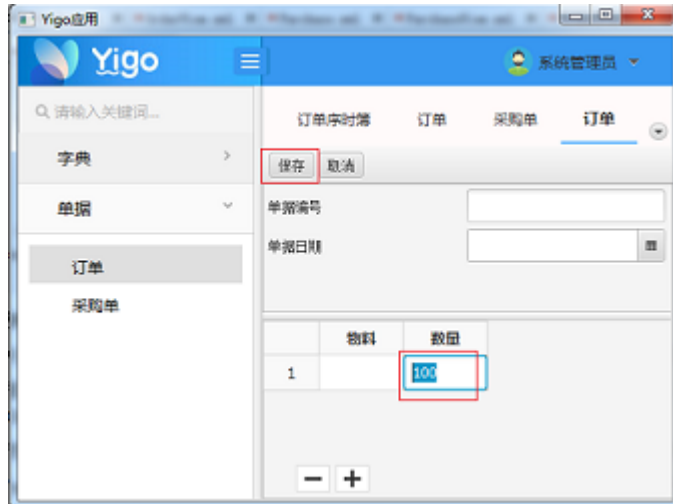
第三步：新建数据映射表（Order2Purchase.xml）。Order2Purchase.xml文件中的配置如下：

```
<Map Caption="订单下推采购单" Key="Order2Purchase" SrcDataObjectKey="Order" TgtDataObjectKey="Purchase">
<SourceTableCollection Height="660" Width="297" X="30" Y="30">
<SourceTable Key="WM_Order" TargetTableKey="WM_Purchase">
<SourceField Definition="Order_BillDate" TargetFieldKey="Purchase_BillDate" TargetTableKey="WM_Purchase"/>
<SourceField Definition="Order_No" TargetFieldKey="Purchase_No" TargetTableKey="WM_Purchase"/>
</SourceTable>
<SourceTable IsPrimary="true" Key="WM_OrderDetail" TargetTableKey="WM_PurchaseDetail">
<SourceField Definition="Order_Material" TargetFieldKey="Purchase_Material" TargetTableKey="WM_PurchaseDetail"/>
<SourceField Definition="Order_Amount" TargetFieldKey="Purchase_Amount" EdgeType="Focus"
TargetTableKey="WM_PurchaseDetail"/>
<SourceField Definition="OID" TargetFieldKey="SrcOID" EdgeType="Relation" TargetTableKey="WM_PurchaseDetail"/>
</SourceTable>
</SourceTableCollection>
<TargetTableCollection Height="660" Width="297" X="419" Y="30">
<TargetTable Key="WM_Purchase">
<TargetField Definition="Purchase_BillDate"/>
<TargetField Definition="Purchase_No"/>
</TargetTable>
<TargetTable IsPrimary="true" Key="WM_PurchaseDetail">
<TargetField Definition="Purchase_Material"/>
<TargetField Definition="Purchase_Amount"/>
<TargetField Definition="SrcOID"/>
</TargetTable>
</TargetTableCollection>
</Map>
```

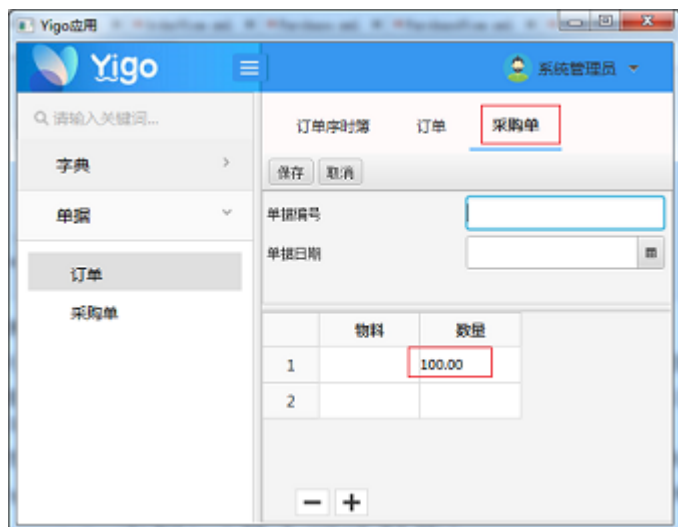
```
</TargetTable>
</TargetTableCollection>
</Map>
```

界面展示：

1. 新增一张订单，输入数据数量为100，保存单据。点击下推，数据将被映射至采购单。



2. 采购单中存在映射后的数据。



配置说明：

1. 映射关系通过<Map>中的SrcObjectKey和TgtObjectKey属性定义；映射关系线为从源字段到目标字段的对应指向线。
2. 源数据对象：Order。其配置信息：SrcDataObjectKey="Order"。SourceTableCollection集合定义数据源表集合，在一个源表集合中必须存在唯一的主表，主表通过IsPrimary属性定义 IsPrimary="true"，WM_OrderDetail为主表。
3. 目标数据对象：Purchase。其配置信息：TgtDataObjectKey="Purchase"。TargetTableCollection集合定义数据目标表集合，TargetField定义目标字段，其Type只能取Field；
4. EdgeType - 映射关系的类型，取值为Normal(普通数据映射)、Focus(关注字段映射)、Relation(关系映射)；

12.3: 示例2（最大可映射值 & 多种值映射）

某些物料可能会因为一些不可控因素如运输损耗，质检不合格等因素，因此我们可以对物料的映射值进行修改。而且还有一个最大映射/最小映射的概念，就是不允许超过的映射范围。

例如：公司订购一批易碎品玻璃杯100只，（在运输过程中可能有损耗。或者质检不过等因素），我们在映射到采购单后将玻璃杯数量改为120只，设置最大映射值为150。

源数据对象		目标数据对象	
物料	数量	物料	数量
玻璃杯	100	玻璃杯	100 120

配置文件如下：

```
<Map Caption="订单下推采购单" Key="Order2Purchase" SrcDataObjectKey="Order" TgtDataObjectKey="Purchase" MaxPushValue="150">
  <SourceTableCollection Height="660" Width="297" X="30" Y="30">
    <SourceTable Key="WM_Order" TargetTableKey="WM_Purchase">
      <SourceField Type="Const" Definition="2018-06-12" TargetFieldKey="Purchase_BillDate" TargetTableKey="WM_Purchase"/>
    </SourceTable>
    <SourceTable Key="WM_OrderDetail" TargetTableKey="WM_PurchaseDetail">
      <SourceField Definition="Order_No" TargetFieldKey="Purchase_No" TargetTableKey="WM_Purchase"/> //来源于字段映射
      <SourceField Definition="Order_Material" TargetFieldKey="Purchase_Material" TargetTableKey="WM_PurchaseDetail"/>
    </SourceTable>
  </SourceTableCollection>
</Map>
```

```

        <SourceField Type="Formula" Definition="(Order_Amount1+Order_Amount2)" TargetFieldKey="Total_Purchase_Amount"
TargetTableKey="WM_PurchaseDetail"/> //来源于表达式映射
        <SourceField Definition="OID" TargetFieldKey="SrcOID" EdgeType="Relation" TargetTableKey="WM_PurchaseDetail"/>
    </SourceTable>
</SourceTableCollection>
<TargetTableCollection Height="660" Width="297" X="419" Y="30">
    <TargetTable Key="WM_Purchase">
        <TargetField Definition="Purchase_BillDate"/>
        <TargetField Definition="Purchase_No"/>
    </TargetTable>
    <TargetTable IsPrimary="true" Key="WM_PurchaseDetail">
        <TargetField Definition="Purchase_Material"/>
        <TargetField Definition="Total_Purchase_Amount"/>
        <TargetField Definition="SrcOID"/>
    </TargetTable>
</TargetTableCollection>
</Map>

```

配置说明：

1. 实例允许玻璃杯因为损耗而将数量改为超量值，并设置超量范围150只。这里需要用到以下2个参数：

MaxPushValue最大映射值：指关注字段最多可以映射到目标对象的值。这里对应的是关注字段Order_Amount不允许超过150。

2. 本实例列举了多种类型的值映射：

单据编号下推使用字段映射，SourceField 属性Type="Field"。

字段映射是默认映射，可不写Type="Field"。Type取值为Field时Definition为字段的标识。

```

<SourceField Definition="Order_No" TargetFieldKey="Purchase_No" TargetTableKey="WM_Purchase"/> //来源于字段映射

```

单据日期下推使用常量映射，Type="Const"。将单据日期常量2018-06-12进行下推。

Type取值为Const时Definition为常量的字面量。

```

<SourceField Type="Const" Definition="2018-06-12" TargetFieldKey="Purchase_BillDate" TargetTableKey="WM_Purchase"/> //来源于常量映射

```

物料数量下推使用表达式映射，物料数量1和数量2的相加的总和下推到新单据中，Type="Formula"为表达式映射，Definition为表达式的内容：

```

<SourceField Type="Formula" Definition="(Order_Amount1+Order_Amount2)" TargetFieldKey="Total_Purchase_Amount"
TargetTableKey="WM_PurchaseDetail"/> //来源于表达式映射

```

12.4: 示例3(可用映射量计算)

配置文件如下：

```

<Map Caption="拣货单下推发货单" Key="Pick2Ship" SrcDataObjectKey="Pick" TgtDataObjectKey="Ship">
    <SourceTableCollection Height="660" Width="297" X="30" Y="30">
        <SourceTable Key="WM_Pick" TargetTableKey="WM_Ship">
            <SourceField Definition="Pick_BillDate" TargetFieldKey="Ship_BillDate" TargetTableKey="WM_Ship"/>
            <SourceField Definition="Pick_No" TargetFieldKey="Ship_No" TargetTableKey="WM_Ship"/>
        </SourceTable>
        <SourceTable IsPrimary="true" Key="WM_PickDetail" TargetTableKey="WM_ShipDetail">
            <SourceField Definition="Pick_Material" TargetFieldKey="Ship_Material" TargetTableKey="WM_ShipDetail"/>
            <SourceField Definition="Pick_Amount" TargetFieldKey="Ship_Amount" EdgeType="Focus"
TargetTableKey="WM_ShipDetail"/>
            <SourceField Definition="OID" TargetFieldKey="SrcOID" EdgeType="Relation" TargetTableKey="WM_ShipDetail"/>
        </SourceTable>
    </SourceTableCollection>
    <TargetTableCollection Height="660" Width="297" X="419" Y="30">
        <TargetTable Key="WM_Ship">
            <TargetField Definition="Ship_BillDate"/>
            <TargetField Definition="Ship_No"/>
        </TargetTable>
    </TargetTableCollection>
</Map>

```

```
<TargetTable IsPrimary="true" Key="WM_ShipDetail">
  <TargetField Definition="Ship_Material"/>
  <TargetField Definition="Ship_Amount"/>
  <TargetField Definition="SrcOID"/>
</TargetTable>
</TargetTableCollection>
</Map>
```

效果演示：

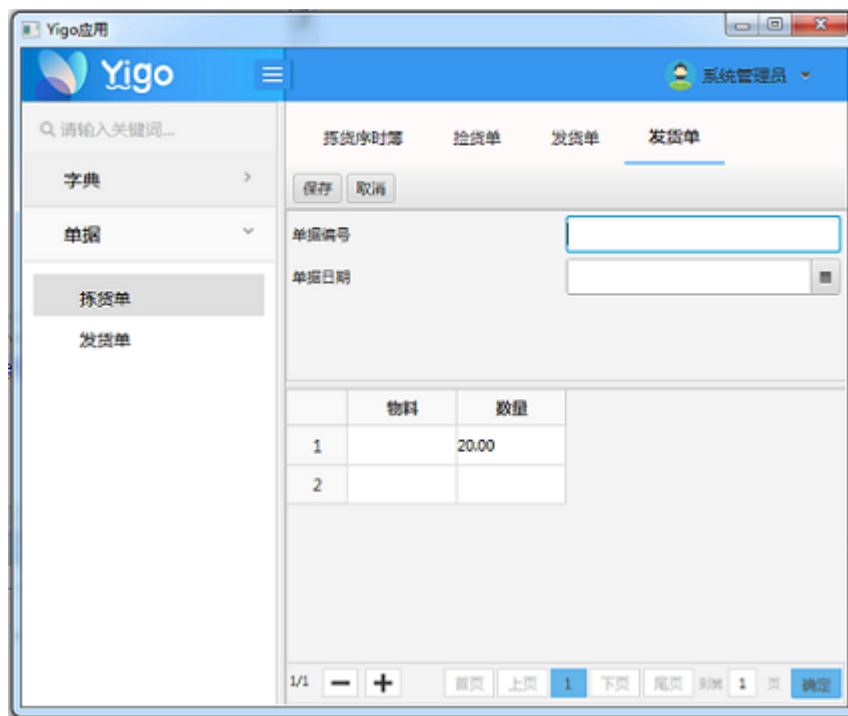
首先，在拣货单输入物料数量50后，下推。

物料	数量
1	50.00

数据被下推到发货单，保存后，将物料数量修改为30。

物料	数量
1	30.00
2	

返回拣货单，再次下推50的物料数量到收货单，这时，能被下推的物料数量只能为20了。

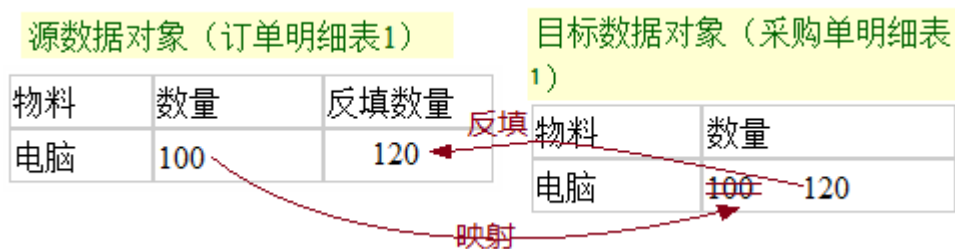


配置说明：

1. 在下推每个仓库的值时，都需要进行计算，看还有多少可以映射的值。

12.5: 示例4(数据反填&反填条件)

公司订单为电脑100台，映射到采购单后，对电脑数量有更改，将采购电脑台数改为120台，并将数据反填回去。



其他文件配置同实例1基本相同，请参照实例1进行配置。

Order2Purchase.xml文件中的配置要改为如下：

```
<Map Caption="订单下推采购单" Key="Order2Purchase" SrcDataObjectKey="Order" TgtDataObjectKey="Purchase">
  <SourceTableCollection Height="660" Width="297" X="30" Y="30">
    <SourceTable Key="WM_Order" TargetTableKey="WM_Purchase">
      <SourceField Definition="Order_BillDate" TargetFieldKey="Purchase_BillDate" TargetTableKey="WM_Purchase"/>
      <SourceField Definition="Order_No" TargetFieldKey="Purchase_No" TargetTableKey="WM_Purchase"/>
    </SourceTable>
    <SourceTable IsPrimary="true" Key="WM_OrderDetail" TargetTableKey="WM_PurchaseDetail">
      <SourceField Definition="Order_Material" TargetFieldKey="Purchase_Material" TargetTableKey="WM_PurchaseDetail"/>
      <SourceField Definition="Order_Amount" TargetFieldKey="Purchase_Amount" EdgeType="Focus" TargetTableKey="WM_PurchaseDetail"/>
      <SourceField Definition="OID" TargetFieldKey="SrcOID" EdgeType="Relation" TargetTableKey="WM_PurchaseDetail"/>
    </SourceTable>
  </SourceTableCollection>
  <TargetTableCollection Height="660" Width="297" X="419" Y="30">
    <SourceTable IsPrimary="true" Key="WM_OrderDetail" TargetTableKey="WM_PurchaseDetail">
      <SourceField Definition="Order_Material" TargetFieldKey="Purchase_Material" TargetTableKey="WM_PurchaseDetail"/>
    </SourceTable>
  </TargetTableCollection>
</Map>
```



```

    <SourceField Definition="Order_Amount" TargetFieldKey="Purchase_Amount" EdgeType="Focus"
    TargetTableKey="WM_PurchaseDetail"/>
    <SourceField Definition="OID" TargetFieldKey="SrcOID" EdgeType="Relation" TargetTableKey="WM_PurchaseDetail"/>
  </SourceTable>
  <TargetTable IsPrimary="true" Key="WM_PurchaseDetail">
    <TargetField Definition="Purchase_Material"/>
    <TargetField Definition="Purchase_Amount">
      <Feedback DataObjectKey="Order" TableKey="WM_OrderDetail" FieldKey="Order_FB_Amount" OpSign="Assign" >
    </Feedback>
    </TargetField>
    <TargetField Definition="SrcOID"/>
  </TargetTable>
</TargetTableCollection>
<FeedbackCollection >
  <FeedbackObject ObjectKey="Order">
    <FeedbackTable TableKey="WM_OrderDetail">
      <FeedbackField FieldKey="Order_FB_Amount"/>
    </FeedbackTable>
  </FeedbackObject>
</FeedbackCollection>
<RelateDataMapCollection>
</RelateDataMapCollection>
</Map>

```

配置说明：

- TargetField通过反填目标属性(Feedback)来定义反填的目标；
- FeedbackCollection为目标数据对象的反对数据对象集合，FeedbackCollection不包含源数据对象；
- 反填的源数据行通过数据映射关系树逐级向上查询，如果映射关系未定义关系映射，那么不能支持反填操作；
- 反填支持向更高的关系树层级进行数据反填，规则同相邻级的反填一致；

如需要添加反填条件，配置如下更改

1. 状态条件：当状态为“审批中”时，进行数据反填。

```
<FeedbackCollection StatusFieldKey="Status" StatusFieldValue="Audited" >
```

2. 数据条件：当仓库为“上海”时，进行数据反填。（暂未实现）

```
<FeedbackCollection Condition="Getdictvalue('Warehouse','Purchase_Material=10008','name')">
```

12.6: 示例5(跨级反填)

公司有一批拣货单，映射到目标数据发货单明细表后，再将发货单明细表的物料数量映射到收货单明细(具体数据如下图所示)。最后，将收货单明细中的物料数量改为100台，反填回拣货单明细中的反填数量中去。

源数据对象(拣货单头表)

订单日期	类型
2014-10-12	IT 设备

目标数据对象(发货单头表)

单据日期	类型
2014-10-12	IT 设备

目标数据对象(收货单头表)

单据日期	类型
2014-10-12	IT 设备

源数据对象(拣货单明细表1)

物料	数量	反填数量
电脑	50	50 (100)
手机	100	100
平板	100	100

目标数据对象(发货单明细表1)

物料	数量
电脑	50
手机	100
平板	100

目标数据对象(收货单明细表1)

物料	数量
电脑	50 (100)
手机	100
平板	100



第一步：新建3张表单Pick和PickView、Ship和ShipView、Delivery和DeliveryView。（具体定义可以查看详细配置文件）

第二步：新建两个数据映射配置文件：Pick2Ship.xml和Ship2Delivery.xml

Pick2Ship.xml映射配置如下：

```
<Map Caption="拣货单下推发货单" Key="Pick2Ship" SrcDataObjectKey="Pick" TgtDataObjectKey="Ship">
  <SourceTableCollection Height="660" Width="297" X="30" Y="30">
    <SourceTable Key="WM_Pick" TargetTableKey="WM_Ship">
      <SourceField Definition="Pick_BillDate" TargetFieldKey="Ship_BillDate" TargetTableKey="WM_Ship"/>
      <SourceField Definition="Pick_No" TargetFieldKey="Ship_No" TargetTableKey="WM_Ship"/>
    </SourceTable>
    <SourceTable IsPrimary="true" Key="WM_PickDetail" TargetTableKey="WM_ShipDetail">
      <SourceField Definition="Pick_Material" TargetFieldKey="Ship_Material" TargetTableKey="WM_ShipDetail"/>
      <SourceField Definition="Pick_Amount" TargetFieldKey="Ship_Amount" EdgeType="Focus"
TargetTableKey="WM_ShipDetail"/>
      <SourceField Definition="OID" TargetFieldKey="SrcOID" EdgeType="Relation" TargetTableKey="WM_ShipDetail"/>
    </SourceTable>
  </SourceTableCollection>
  <TargetTableCollection Height="660" Width="297" X="419" Y="30">
    <TargetTable Key="WM_Ship">
      <TargetField Definition="Ship_BillDate"/>
      <TargetField Definition="Ship_No"/>
    </TargetTable>
    <TargetTable IsPrimary="true" Key="WM_ShipDetail">
      <TargetField Definition="Ship_Material"/>
      <TargetField Definition="Ship_Amount"/>
      <TargetField Definition="SrcOID"/>
    </TargetTable>
  </TargetTableCollection>
  <RelateDataMapCollection>
  </RelateDataMapCollection>
</Map>
```

Ship2Delivery.xml映射配置如下：

```
<Map Caption="发货单下推收货单" Key="Ship2Delivery" SrcDataObjectKey="Ship" TgtDataObjectKey="Delivery">
  <SourceTableCollection Height="660" Width="297" X="30" Y="30">
    <SourceTable Key="WM_Ship" TargetTableKey="WM_Delivery">
      <SourceField Definition="Ship_BillDate" TargetFieldKey="Delivery_BillDate" TargetTableKey="WM_Delivery"/>
      <SourceField Definition="Ship_No" TargetFieldKey="Delivery_No" TargetTableKey="WM_Delivery"/>
    </SourceTable>
    <SourceTable IsPrimary="true" Key="WM_ShipDetail" TargetTableKey="WM_DeliveryDetail">
      <SourceField Definition="Ship_Material" TargetFieldKey="Delivery_Material" TargetTableKey="WM_DeliveryDetail"/>
      <SourceField Definition="Ship_Amount" TargetFieldKey="Delivery_Amount" EdgeType="Focus"
TargetTableKey="WM_DeliveryDetail"/>
      <SourceField Definition="OID" TargetFieldKey="SrcOID" EdgeType="Relation" TargetTableKey="WM_DeliveryDetail"/>
    </SourceTable>
  </SourceTableCollection>
  <TargetTableCollection Height="660" Width="297" X="419" Y="30">
    <TargetTable Key="WM_Delivery">
      <TargetField Definition="Delivery_BillDate"/>
      <TargetField Definition="Delivery_No"/>
    </TargetTable>
    <TargetTable IsPrimary="true" Key="WM_DeliveryDetail">
      <TargetField Definition="Delivery_Material"/>
      <TargetField Definition="Delivery_Amount">
        <Feedback DataObjectKey="Pick" TableKey="WM_PickDetail" FieldKey="Pick_FB_Amount" OpSign="AddDelta" >
          </Feedback>
        </TargetField>
      <TargetField Definition="SrcOID"/>
    </TargetTable>
  </TargetTableCollection>
  <FeedbackCollection>
    <FeedbackObject ObjectKey="Pick">
      <FeedbackTable TableKey="WM_PickDetail">
        <FeedbackField FieldKey="Pick_Amount"/>
      </FeedbackTable>
    </FeedbackObject>
  </FeedbackCollection>
  <RelateDataMapCollection>
  </RelateDataMapCollection>
</Map>
```

12.7: 示例6(上引下推)

在实际业务中，经常会有多张单据，通过查询关键字，将查询出的新数据下推到新的一张表单中去。我们以一张单据上引下推为例，如下图所示，将仓库=上海作为过滤条件，查询出所有上海仓的数据并下推到一张新表单中去。

表单1

序号	日期	物料	仓库	数量
1	2015-6-12	电脑	上海仓	20
2	2015-7-20	ipad	上海仓	15
1	2015-6-12	手机	北京仓	20
2	2015-7-20	电脑	上海仓	15
1	2015-6-12	电脑	北京仓	50
2	2015-7-20	ipad	广州仓	45
2	2015-7-20	手机	上海仓	30

下推的新表单

序号	日期	物料	仓库	数量
1	2015-6-12	电脑	上海仓	20
2	2015-7-20	ipad	上海仓	15
3	2015-7-20	电脑	上海仓	15
4	2015-7-20	手机	上海仓	30

配置思路：

第一步：新建一个单据A（订单）下推到单据B（采购单）配置。（此步骤就不赘述了，参见普通映射实例1）

第二步：在单据B（采购单）中上引出一张新单据Query（实际为订单明细序时簿），通过查询过滤条件后，下推到单据B（采购单）。

上引这个动作我们通过ShowModal来打开一张新单据。

```
<Operation Key="Show" Caption="上引" Visible="ReadOnly()">
  <Action>
    <![CDATA[ShowModal('Query');]]>
  </Action>
</Operation>
```

定义3个查询过滤条件

物料 仓库： 日期：

```
<Label Key="L_Material" Caption="物料" X="0" Y="0" XSpan="1" YSpan="1"/>
<Dict Key="Order_Material" Caption="物料" ItemKey="Material" BuddyKey="L_Material" X="0" Y="1" XSpan="1" YSpan="1">
  <Condition CondSign="in" TableKey="WM_OrderDetail" ColumnKey="Order_Material"/>
</Dict>

<Label Key="L_Warehouse" Caption="仓库：" X="1" Y="0" XSpan="1" YSpan="1"/>
<Dict Key="Order_Warehouse" Caption="仓库" ItemKey="Warehouse" BuddyKey="L_Warehouse" X="1" Y="1" XSpan="1" YSpan="1">
  <Condition CondSign="in" TableKey="WM_OrderDetail" ColumnKey="Order_Warehouse"/>
</Dict>

<Label Key="L_Billdate" Caption="日期：" X="2" Y="0" XSpan="1" YSpan="1"/>
<DatePicker Key="Order_BillDate" Caption="日期时间" BuddyKey="L_Billdate" X="2" Y="1" XSpan="1" YSpan="1">
  <Condition CondSign="between" Group="Order_BillDate" GroupHead="true" TableKey="WM_OrderDetail" ColumnKey="Order_BillDate"/>
</DatePicker>

<Button Key="button1" Caption="查询" X="0" Y="2" XSpan="1" YSpan="1">
  <OnClick>
    <![CDATA[
      DealCondition();LoadData();ShowData();
    ]]>
  </OnClick>
```

```

</Button>
<Button Key="cancel" Caption="重置" Enable="true" X="1" Y="2" XSpan="1" YSpan="1">
  <OnClick>
    <![CDATA[
      ResetCondition();
    ]]>
  </OnClick>
</Button>

```

当上引的时候序时簿Query类型为Virtual时，需要在DataObject中添加一张单据，描述Query的数据对象。如下定义：

```

<DataObject Key="Query" Caption="上引" PrimaryTableKey="WM_OrderDetail" PrimaryType="Virtual" SecondaryType="Normal">
  <TableCollection>
    <Table Key="WM_OrderDetail" Caption="订单明细" DBTableName="WM_OrderDetail" TableMode="Detail" SourceType="Query"
      Persist="True">
      <Statement>
        <![CDATA[select
WM_Order.Order_BillDate,WM_Order.Order_NO,WM_Order.Memo,WM_OrderDetail.OID,WM_OrderDetail.POID,WM_OrderDetail.SOID,WM_OrderDetail.VERID,WM_OrderDetail.DVERID,WM_OrderDetail.Order_Material,WM_OrderDetail.Order_Warehouse,WM_OrderDetail.Order_Amount
from WM_Order join WM_OrderDetail on WM_Order.OID=WM_OrderDetail.SOID]]>
      </Statement>
      <Column Key="Order_BillDate" Caption="单据日期" Persist="True" DataType="DateTime" DBColumnName="Order_BillDate"/>
      <Column Key="Order_NO" Caption="单据编号" Persist="True" DataType="Varchar" Length="200" DBColumnName="Order_NO"/>
      <Column Key="Memo" Caption="备注" Persist="True" DataType="Varchar" Length="250"/>
      <Column Key="OID" Caption="对象标识" Persist="True" DataType="Long" DBColumnName="OID" />
      <Column Key="POID" Caption="父对象标识" Persist="True" DataType="Long" DBColumnName="POID"/>
      <Column Key="SOID" Caption="主对象标识" Persist="True" DataType="Long" DBColumnName="SOID"/>
      <Column Key="VERID" Caption="对象版本" Persist="True" DataType="Integer" DBColumnName="VERID"/>
      <Column Key="DVERID" Caption="对象明细版本" Persist="True" DataType="Integer" DBColumnName="DVERID"/>
      <Column Key="Order_Material" Caption="物料" Persist="True" DataType="Long" IsPrimary="True" DBColumnName="Order_Material"/>
      <Column Key="Order_Warehouse" Caption="仓库" Persist="True" DataType="Long" DBColumnName="Order_Warehouse"/>
      <Column Key="Order_Amount" Caption="数量" Persist="True" DataType="Numeric" Precision="16" Scale="2"
        DBColumnName="Order_Amount"/>
    </Table>
  </TableCollection>
</DataObject>

```

在Form中的Query单据中，就不需要再定义了，只需要定义RefObjectKey就可以了。

```

<DataSource RefObjectKey="Query">
</DataSource>

```

新单据Query下推到单据Purchase 的映射配置如下：

```

<Map Caption="订单下推采购单" Key="Query2Purchase" SrcDataObjectKey="Query" TgtDataObjectKey="Purchase">
  <SourceTableCollection Height="660" Width="297" X="30" Y="30">
    <SourceTable IsPrimary="true" Key="WM_OrderDetail" TargetTableKey="WM_PurchaseDetail">
      <SourceField Definition="Order_BillDate" TargetFieldKey="Purchase_BillDate" TargetTableKey="WM_Purchase"/>
      <SourceField Definition="Order_No" TargetFieldKey="Purchase_No" TargetTableKey="WM_Purchase"/>
      <SourceField Definition="Order_Material" TargetFieldKey="Purchase_Material" TargetTableKey="WM_PurchaseDetail"/>
      <SourceField Definition="Order_Amount" TargetFieldKey="Purchase_Amount" EdgeType="Focus"
        TargetTableKey="WM_PurchaseDetail"/>
      <SourceField Definition="OID" TargetFieldKey="SrcOID" EdgeType="Relation" TargetTableKey="WM_PurchaseDetail"/>
    </SourceTable>
  </SourceTableCollection>
  <TargetTableCollection Height="660" Width="297" X="419" Y="30">
    <TargetTable Key="WM_Purchase">
      <TargetField Definition="Purchase_BillDate"/>
      <TargetField Definition="Purchase_No"/>
    </TargetTable>
    <TargetTable IsPrimary="true" Key="WM_PurchaseDetail">
      <TargetField Definition="Purchase_Material"/>
      <TargetField Definition="Purchase_Warehouse"/>
      <TargetField Definition="Purchase_Amount"/>
      <TargetField Definition="SrcOID"/>
    </TargetTable>
  </TargetTableCollection>
</Map>

```

效果演示：

第一步，订单下推到采购单的过程就不截图演示了。

第二步，在采购单中点击上引按钮，打开上引出的序时簿。 打开的序时簿：

Yigo应用

系统管理员

请输入关键词...

字典

单据

订单

采购单

采购单

单据编号

单据日期

	物料	仓库	数量
1	001 phone	bj 北京	72.00

上引单

下推

物料

仓库

日期

查询

重置

	物料	仓库	数量	备注
1	001 phone	sh 上海	20.00	
2	002 computer	bj 北京	85.00	
3	001 phone	bj 北京	95.00	
4	003 ipad	gz 广州	16.00	
5	001 phone	bj 北京	72.00	
6	001 phone	gz 广州	25.00	

1/1

首页 上一页 1 下一页 末页 刷新 打印 退出

第三步，选择“北京”仓库。ListView过滤出所有的北京仓库的数据。点击下推，下推到采购单。

第 13 章 业务流程基础

目录

13.1. 基本概念	154
13.2. 业务流程元素	156
13.3. 工作台	159

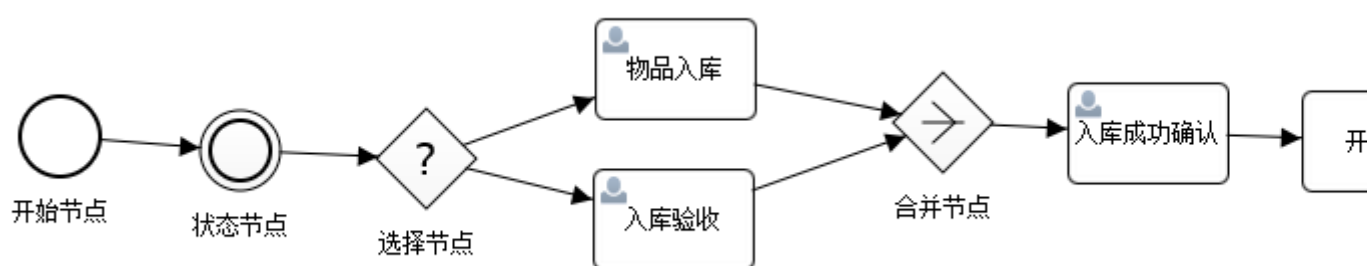
业务流程定义组织的业务处理过程，用于控制组织的业务处理、监控及优化。Yigo语言为业务流程的建模提供了整套模型框架用于描述业务过程、案例、任务、执行资源、工作项系统。下面首先介绍一下Yigo业务流程的基本概念。

本章主要介绍业务流程的基本知识，后面一章通过实例说明流程的简单应用，更多的内容请见业务流程专题教程。

13.1: 基本概念

13.1.1: 过程

生活中有许多不同的工作需要处理，比如订机票、采购办公用品等，这些工作需要一系列的任务按照一定的规则和次序进行，这些工作的处理步骤及逻辑被称为“过程”，在Yigo中通过过程(Process)定义描述。过程定义了哪些任务需要被执行，以及按照什么样的次序被执行。一个过程描述了一类工作的执行，根据具体工作的不同的属性，过程会采取不同的措施，由于工作的属性不同，工作的任务执行的次序也会发生变化，本质上过程由任务和条件组成。过程定义了工作的生命周期，因此每个过程定义都有开始和结束，在Yigo中通过“开始”和“结束”节点定义。在Yigo系统中过程通过过程定义文件来描述，一个过程可以是一个完整的业务流程的定义，也可以是一个更大的业务流程中的一部分。一个具体的过程定义如下图所示：



每个过程必须有开始和结束，中间包括一系列任务。

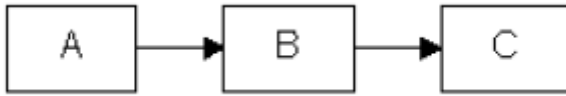
13.1.2: 任务

任务是过程中一个具体的处理步骤，任务是组成过程的基本元素，每个过程均由一系列任务组成，任务定义了一个处理步骤中具体需要做什么、如何做、谁来做，比如评估报价、经理审批、财务复核都是任务。任务由手工完成的也有系统自动完成的。在过程中，一个任务具有不可分割性，即要么全部完成，要么什么都不做。任务的详细信息在流程节点中介绍。

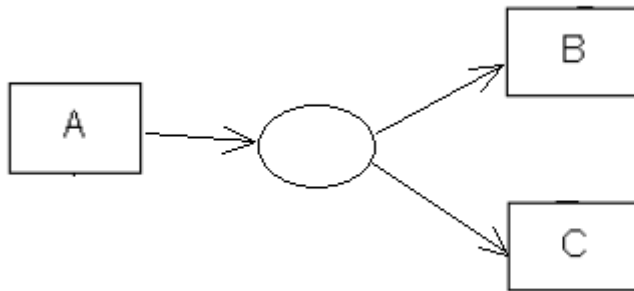
13.1.3: 路由

路由定义了过程中的任务是否执行以及以何种顺序执行，通常路由具有以下类型：

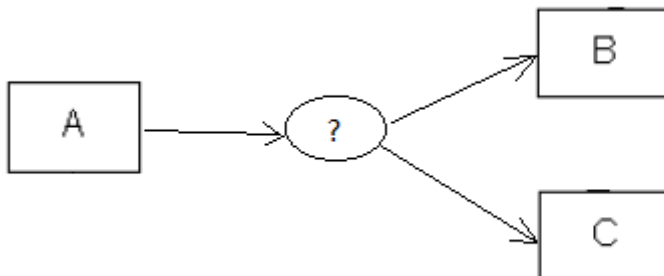
- 顺序：最简单的路由形式是任务的顺序执行，它们按顺序一个接一个的执行；通常情况下，它们之间存在清晰的依赖关系，例如前一个任务的结果是另一个任务的输出，或者仅仅是时间上的顺序；如下图所示：



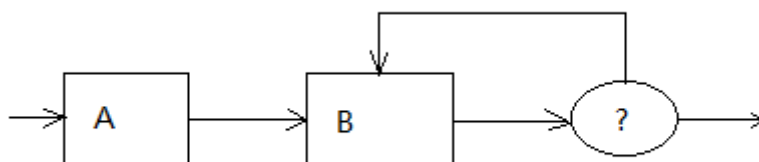
- 并行：如果两个任务可以同时以任何顺序执行，称为并行路由；这种情况下，两个任务都需要被执行，且相互不影响；任务开始于拆分，同步于合并；如下图所示：



- 选择：当在两个或更多的任务间选择时，称为选择路由；该选择依赖于案例的相关属性；如下图所示：



- 循环：在理想情况下，案例中的任务的执行都不超过一次，然后有时需要多次执行某些任务，这种称为循环路由；如下图所示：



13.1.4: 流程实例

流程实例定义了过程的一次执行，也称为“过程实例”，业务流程的基本任务就是处理流程实例，在Yigo中每个流程实例有一个唯一64位整型标识，流程实例记录在BPM_Instance表中。每个流程实例都有其生命周期，通过实例的状态来定义，一个实例会处于四种状态，分别如下：

- 已注册，表示实例已产生，但未绑定过程，值为0；
- 运行中，表示实例正在处理中，值为1；
- 已结束，表示实例正常结束，值为2；
- 强制关闭，表示实例被强制结束，值为3；

13.1.5: 工作项

当一个具体的流程实例需要执行某个任务时，会生成一个工作项，代表了一个具体需要执行的任务，工作项同流程实例中的一个具体任务相关联，并且会指定有谁来执行。

每个工作项都会处理一个状态，如下：

- 尚未处理，值为1
- 已经完成，值为2
- 尚未处理但暂时隐藏，值为3
- 已禁用，值为4

13.2: 业务流程元素

13.2.1: 过程定义

Yigo的过程定义通过一个唯一的字符串标识确定，称为过程标识，过程定义存储在工程的BPM及其目录下，每个过程标识具有多个定义，每个定义称为过程的一个版本，因此一个过程标识可以具有多个版本，在任何一个时刻只能有一个活动版本，通过过程的布署信息描述。

13.2.2: 过程节点

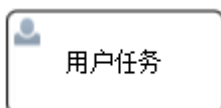
13.2.2.1: 开始节点



开始节点

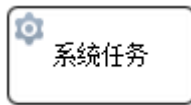
表示一个过程的开始：

13.2.2.2: 用户任务



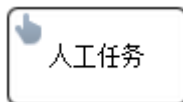
过程中需要用户处理的任务。

13.2.2.3: 系统任务



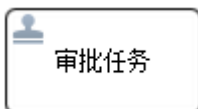
过程中系统自动处理的任务。

13.2.2.4: 人工任务



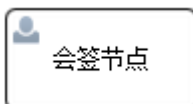
系统外任务，不需要用户通过系统来处理，也不需要系统自动处理。

13.2.2.5: 审批



过程中的审批任务，通常审批任务用来确定“同意”或“不同意”，也有更多的处理选项的审批任务。

13.2.2.6: 会签



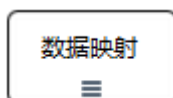
过程中的会签任务，需要多个人来处理的任务，用于处理多个人对某件事情进行投票以最终确定是否同意。

13.2.2.7: 子流程



过程中的用于产生新的流程实例的节点，当前流程可以等待新的流程实例结束或者继续后续处理。

13.2.2.8: 数据映射



过程中用于产生其它表单数据的任务，新产生的表单数据可以开启自己的流程，当前流程可以选择是等待还是继续后续处理。

13.2.2.9: 选择节点



选择节点

选择节点用于判断某个条件是否成功，成立或不成立可以分别走向不同的后续任务。

13.2.2.10: 分支节点



分支节点

分支节点可以连接多个后续的任务；

13.2.2.11: 互斥分支



互斥分支

互斥分支可以连接多个后续的任务，但只有其中某个任务完成，则其它任务均不能再继续处理。

13.2.2.12: 合并



合并节点

合并节点用于同步多个任务，只有这些任务完成后，才可以继续后续的处理，合并节点可以对需要同步的任务的数量进行控制。

13.2.2.13: 复杂合并



复杂合并

复杂合并用于同步多个任务，这些任务只有在满足指定的条件才可以继续后续的处理，同合并不同，复杂合并的条件需要定制。

13.2.2.14: 结束节点



结束节点

过程的结束，进入结束节点后，流程实例生命周期结束。

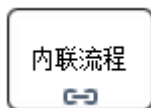
13.2.2.15: 分支结束节点



分支结束

分支结束表示某个处理分支完成。

13.2.2.16: 嵌入节点



嵌入节点用于在当前流程实例中引用一个子处理过程，但不会产生新实例，流程实例必须等当前子处理过程完成才能继续。

13.2.2.17: 状态



状态节点

状态节点用于表示流程实例关联表单的状态。

13.3: 工作台

工作台的目的是用于检索分发给当前登录用户的工作任务，其工作项数据来源于WF_Workitem表，当前用户的过滤通过WF_Participator表定义，后面先介绍WF_Workitem和WF_Participator的表结构，再介绍如何定制一个工作台。

13.3.1: 工作项 (WF_Workitem)

定义工作项系统同 workflow 集成时必须具有的字段。

表 13.1. WF_WorkItem

字段标识	字段意义	类型
WorkitemID	工作项的唯一标识	Long
WorkitemName	工作项的名称	Varchar (512)
WorkitemState	工作项的状态	Integer
CreateTime	工作项的创建时间	DateTime
FinishTime	工作项的处理结束时间	DateTime
OperatorID	操作员的标识	Long
SrcOperatorID	原操作员的标识	Long
UserInfo	用户信息	Varchar (512)
ResultInfo	处理结果描述信息	Varchar (32)
InstanceID	关联的流程实例的标识	Long
NodeID	关联的过程节点的标识	Integer
NodeType	关联的过程节点的类型	Integer
DelegateID	代理或授权的标识	Long

13.3.2: 参与者 (WF_Participant)

记录一个工作项的所有候选处理人。

表 13.2. WF_Participant

字段标识	字段意义	类型
WorkitemID	工作项的唯一标识	Long
OperatorID	参与者的操作员标识	Long
SrcOperatorID	参与者的授权者或代理者操作员标识	Long
DelegateID	代理或授权的标识	Long

13.3.3: 定制工作台

以下列出一个基本的工作台定义表单。这里通过一个具体的示例来说明工作台的处理流程。

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Form Caption="我的待办事宜" Key="ToDoList" FormType="View">
  <DataSource>
    <DataObject Key="ToDoList" Caption="我的待办事宜" PrimaryType="Virtual" SecondaryType="Normal" MainTableKey="ToDoList">
      <TableCollection>
        <Table Key="ToDoList" Caption="待办信息" DBTableName="WF_workitem" TableMode="Detail" SourceType="Query" Persist="False">
          <Statement>
            <![CDATA[select * from WF_Workitem where WF_Workitem.WorkitemState=1 and WF_Workitem.WorkitemID in (select WorkitemID
from WF_Participant where OperatorID=?)]]>
          </Statement>
          <Column Key="workItemID" Caption="工作项标识" Persist="True" DataType="Long" DBColumnName="workItemID" />
          <Column Key="workItemName" Caption="工作项名称" Persist="True" DataType="Varchar" DBColumnName="workItemName"/>
          <Column Key="createTime" Caption="创建时间" Persist="True" DataType="DateTime" DBColumnName="createTime"/>
          <QueryParameterCollection>
            <QueryParameter Key="OperatorID" Formula="GetOperator()" />
          </QueryParameterCollection>
        </Table>
      </TableCollection>
    </DataObject>
  </DataSource>
</Form>
```

```

    </QueryParameterCollection>
  </Table>
</TableCollection>
</DataObject>
</DataSource>
<OperationCollection/>
<OnLoad type="Formula">
<![CDATA[LoadData();]]>
</OnLoad>
<Body>
  <Block>
    <FlexFlowLayoutPanel Key="main">
      <ToolBar Key="main_toolbar" Height="pref" IsDefault="True"/>
      <ListView Key="list" TableKey="ToDoList" Height="100%">
        <RowDbClick>
          <![CDATA[
OpenWorkitem(workItemID)
]]>
        </RowDbClick>
        <ListViewColumnCollection>
          <ListViewColumn Key="workItemID" Width="100px" Caption="工作项标识" ColumnType="Label" DataColumnKey="workItemID">
        </ListViewColumn>
          <ListViewColumn Key="workItemName" Width="350px" Caption="工作项名称" ColumnType="Label" DataColumnKey="workItemName">
        </ListViewColumn>
          <ListViewColumn Key="createTime" Width="350px" Caption="创建时间" ColumnType="Label" DataColumnKey="createTime">
        </ListViewColumn>
        </ListViewColumnCollection>
      </ListView>
    </FlexFlowLayoutPanel>
  </Block>
</Body>
</Form>

```

定制工作台的目的为了检索分发给登录操作员的工作项，其数据来源于WF_Workitem表。在WF_Participator表中存在当前操作员的工作项，并且其状态1(未完成)的工作项。在后续的章节中的定制工作台都使用这个简单的工作台定义。界面如下图所示：

我的待办事宜			
工作项标识	工作项名称	创建时间	
101	部门主管审批	2015- 04- 15	

检索出需要处理的工作项后，后面需要定义如何对这些工作项进行处理，这里我们通过OpenWorkitem函数对工作项进行处理，这个函数的目的是通过工作项打开关联的表单，并在表单界面上创建工作项的操作。OpenWorkitem接收一个长整型的参数作为工作项的标识。在当前的界面上，我们通过定义ListView的RowDbClick事件来打开工作项。

```

<RowDbClick>
  <![CDATA[
OpenWorkitem(workItemID)
]]>
</RowDbClick>

```

我们以示例中的请假流程为例，通过打开“部门主管审批”这个工作项，会打开相应的请假单以供审批。

我的待办事宜

请假单

审批

审批记录

工号

001

姓名

张三

开始时间

2015-04-15

结束时间

2015-04-17

事由

类型

事假

总计

3

如上图所示，在请假单界面上会生成一个“审批”按钮。点击“审批”按钮弹出审批窗口。

审批对话框

意见

同意！

通过

驳回

取消

这里的“审批”界面是系统为审批任务生成的默认处理界面，后面会介绍具体的任务操作。

第 14 章 业务流程任务处理

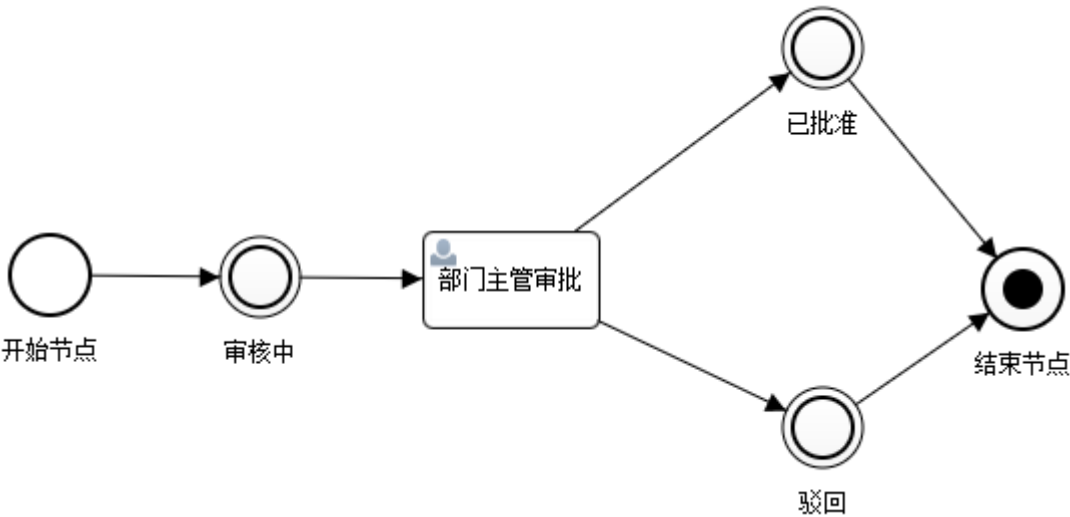
目录

14.1. 审批	163
14.2. 状态	170
14.3. 用户任务	171
14.4. 系统任务	173
14.5. 选择	174
14.6. 分支与合并	175
14.7. 会签	176
14.8. 子流程	180
14.9. 数据映射	181
14.10. 加签	182
14.11. 流程重启/重提交	184
14.12. 状态机操作	188

本章通过一些具体的实例来说明业务流程的任务处理，对于每个过程定义来说，都必须有开始和结束，后面不再一一说明。本章的实例以请假流程为例说明流程的基本使用方法，首先说明一下本章所需要的一些基本数据准备。本章的1.1—1.6的示例，需要四种参与者类别，分别是职员、部门主管、经理和人事，为了简便起见，对应系统中的四个操作员，分别是Staff、Leader、Manager和HR，为方便定义，这4个用户的角色均为管理员。

14.1: 审批

一个简单的审批过程如下图所示：



原代码如下：

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Process Caption="请假" Key="VacationRequestAudit">
  <Begin Caption="开始节点" ID="1" Key="Begin">
    <TransitionCollection>
      <SequenceFlow Condition="" ID="9" Key="BeginToState" TargetNodeKey="State">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic X="14" Y="204"/>
  </Begin>
  <State Caption="审核中" ID="2" Key="State" Status="Auditing">
```



```

        <TransitionCollection>
            <SequenceFlow Condition="" ID="10" Key="StatetoDecision" TargetNodeKey="UserTask3">
                <TransitionGraphic LineStyle="StraightLine"/>
            </SequenceFlow>
        </TransitionCollection>
        <NodeGraphic X="119" Y="205"/>
    </State>
    <Audit Caption="部门主管审批" Key="UserTask3" ID="105">
        <TransitionCollection>
            <SequenceFlow Condition="True" Key="UserTask3_State1" ID="108" TargetNodeKey="State1">
                <TransitionGraphic LineStyle="StraightLine"/>
            </SequenceFlow>
            <SequenceFlow Condition="False" Key="UserTask3_State2" ID="109" TargetNodeKey="State2">
                <TransitionGraphic LineStyle="StraightLine"/>
            </SequenceFlow>
        </TransitionCollection>
        <OperationCollection>
            <Operation Action="ShowModal(&quot;AuditDialog&quot;)" Caption="审批" Key="op1"/>
        </OperationCollection>
        <AssistanceCollection/>
        <ParticipatorCollection>
            <Dictionary ItemKey="Operator" ItemID="20000"/>
        </ParticipatorCollection>
        <NodeGraphic Height="48" Width="88" X="220" Y="202"/>
    </Audit>
    <State Caption="已批准" Key="State1" ID="104" Status="Audited">
        <TransitionCollection>
            <SequenceFlow Condition="" ID="21" TargetNodeKey="End">
                <TransitionGraphic LineStyle="StraightLine"/>
            </SequenceFlow>
        </TransitionCollection>
        <NodeGraphic Height="34" Width="48" X="400" Y="91"/>
    </State>
    <State Caption="驳回" Key="State2" ID="106" Status="Denied">
        <TransitionCollection>
            <SequenceFlow Condition="" ID="23" Key="EE" TargetNodeKey="End">
                <TransitionGraphic LineStyle="StraightLine"/>
            </SequenceFlow>
        </TransitionCollection>
        <NodeGraphic Width="78" X="400" Y="280"/>
    </State>
    <End Caption="结束节点" ID="8" Key="End">
        <TransitionCollection/>
        <NodeGraphic X="500" Y="211"/>
    </End>
</Process>

```

这里我们主要讲述审批任务的定义，其源代码如下：

```

<Audit Caption="部门主管审批" Key="UserTask3" ID="105">
    <TransitionCollection>
        <SequenceFlow Condition="True" Key="UserTask3_State1" ID="108" TargetNodeKey="State1">
            <TransitionGraphic LineStyle="StraightLine"/>
        </SequenceFlow>
        <SequenceFlow Condition="False" Key="UserTask3_State2" ID="109" TargetNodeKey="State2">
            <TransitionGraphic LineStyle="StraightLine"/>
        </SequenceFlow>
    </TransitionCollection>
    <OperationCollection>
        <Operation Action="ShowModal(&quot;AuditDialog&quot;)" Caption="审批" Key="op1"/>
    </OperationCollection>
    <AssistanceCollection/>
    <ParticipatorCollection>
        <Dictionary ItemKey="Operator" ItemID="20000"/>
    </ParticipatorCollection>
    <NodeGraphic Height="48" Width="88" X="220" Y="202"/>
</Audit>

```

这里“部门经理审批”有两个后续流转(通过SequenceFlow定义)，其中Condition为True的那个连线表示审批通过的情况下后续流转到State1，Condition为False的那个连线表示审批不通过的情况下后续流转到State2；

审批任务默认取值为任务中AuditResult属性(WF Workitem中的AuditResult)，即审批结果，审批结果的取值为整型，默认情况下，其值取0时表示不通过，取1时表示通过，程序也可以通过AuditResult定义更多的审批结果，相应的后续的连线上的Condition也需要取相应的取值，假如审批结果包含三个选项，分别是同意(1)，不同意(0)，同意并报请复核(2)，那么后续的连线上的Condition的取值分别取0、1、2，而不是上面例子中的True和False。

审批任务的处理通过任务的操作集合来定义，这里定义了一个处理操作，即弹出一个审批对话框 AuditDialog来处理审批任务；这里AuditDialog对话框通过AuditWorkitem来完成审批任务，该函数的原型为void AuditWorkitem(workitemID, result, userInfo)，其中workitemID为工作项的标识，result为审批结果，即AuditResult的最终取值，userInfo为用户的审批意见，例如：

```
AuditWorkitem(-1, 1, "同意，请酌情办理！")
```

AuditDialog对话框的定义可以参见BPM_C2源程序，这里列出其代码，后面的审批除非特别说明均以AuditDialog为审批窗口：

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Form Caption="审批对话框" FormType="" Key="AuditDialog">
  <DataSource>
    <DataObject>
      <TableCollection/>
    </DataObject>
  </DataSource>
  <Body HAlign="Left" OverflowX="Visible" OverflowY="Visible">
    <Block>
      <GridLayoutPanel Caption="GridLayoutPanel0" HasBorder="false" Key="GridLayoutPanel0" Padding="5px">
        <TextArea Area="" BottomMargin="" BottomPadding="" Caption="意见" Enable="True" HasBorder="false"
Key="Opinion" LeftMargin="" LeftPadding="" LeftPadding="" Padding="" RightMargin="" RightPadding="" TopMargin="" TopPadding="" X="0"
XSpan="4" Y="1" HAlign="Left" VAlign="Top">
          <DataBinding/>
        </TextArea>
        <Button Area="" BottomMargin="" BottomPadding="" Caption="通过" Enable="True" HasBorder="false" Key="APPROVE"
LeftMargin="" LeftPadding="" Padding="" RightMargin="" RightPadding="" ShowText="false" TopMargin="" TopPadding="" X="1"
Y="3">
          <DataBinding/>
        </Button>
        <OnClick>
          <![CDATA[
parent.CommitWorkitem(-1, 1, getvalue("Opinion"));Close();
]]>
        </OnClick>
      </Block>
      <Block>
        <Button Area="" BottomMargin="" BottomPadding="" Caption="驳回" Enable="True" HasBorder="false" Key="OPPOSE"
LeftMargin="" LeftPadding="" Padding="" RightMargin="" RightPadding="" ShowText="false" TopMargin="" TopPadding="" X="2"
Y="3">
          <DataBinding/>
        </Button>
        <OnClick>
          <![CDATA[
parent.CommitWorkitem(-1, 0, getvalue("Opinion"));Close();
]]>
        </OnClick>
      </Block>
      <Block>
        <Button Area="" BottomMargin="" BottomPadding="" Caption="取消" Enable="True" Key="cancel" LeftMargin=""
LeftPadding="" Padding="" RightMargin="" RightPadding="" TopMargin="" TopPadding="" X="3" Y="3">
          <DataBinding/>
        </Button>
        <OnClick>
          <![CDATA[
Close();
]]>
        </OnClick>
      </Block>
      <Block>
        <Button Area="" BottomMargin="" BottomPadding="" Caption="意见" HasBorder="false" Key="T_Opinion" LeftMargin=""
LeftPadding="" Padding="" RightMargin="" RightPadding="" TopMargin="" TopPadding="" X="0" XSpan="4" Y="0">
          <DataBinding/>
        </Button>
        <Label>
          <RowDefCollection RowGap="8">
            <RowDef Height="30px"/>
            <RowDef Height="174px"/>
            <RowDef Height="10px"/>
            <RowDef Height="20px"/>
            <RowDef Height="10px"/>
          </RowDefCollection>
          <ColumnDefCollection ColumnGap="8">
            <ColumnDef Width="100%"/>
            <ColumnDef Width="100px"/>
            <ColumnDef Width="100px"/>
            <ColumnDef Width="100px"/>
          </ColumnDefCollection>
        </Label>
      </Block>
    </Body>
  </Form>
```

以Staff用户登录系统，创建一张请假单，填写必要的信息，保存后会有“启动流程”按钮，如下图：

请假单列表
请假单

流程启动
审批记录

工号	001
姓名	张三
开始时间	2015-04-20
结束时间	2015-04-20
事由	
类型	事假
总计	1

为了有“流程启动”按钮，必须在请假单的表单配置的操作集合中增加一个占位操作，同时需要增加表单和流程关联信息，如下：

提交流程的占位(在表单的OperationCollection中)：

```
<Operation Caption="BPM" Key="BPM" Tag="BPM"/>
```

表单流程关联信息(在BPM.xml文件中)：

```
<ProcessMapCollection>
  <ProcessMap InitDate="2014.06.12 17:39:41 GMT+08:00" Key="VacationRequest" ProcessKey="VacationRequestAudit" StartCaption="流程启动"/>
</ProcessMapCollection>
```

其中StartCaption为提交流程用的按钮的名称，提交流程后，流程正式启动运行，在当前的流程中，会产生一个审批工作项分发给部门经理。

以Leader用户登录系统，在工作台上，会产生一个分发给Leader的工作项，如下图：

我的待办事宜			
工作项标识	工作项名称	创建时间	
301	部门主管审批	2015-04-20	

双击该待办事项，打开Staff填写的请假单。

我的待办事宜

请假单

审批

审批记录

工号

001

姓名

张三

开始时间

2015-04-20

结束时间

2015-04-20

事由

类型

事假

总计

1

在打开请假单后，请假单上会显示“审批”按钮，为了显示“审批”这样的按钮，需要在请假单的表单配置操作集合中增加下面的内容：

```
<Operation Caption="WORKITEM" Key="WORKITEM" Tag="WORKITEM"/>
```

在打开工作项后，这个占位会被任务的操作集合替换成按钮集合，操作集合的定义在任务的OperationCollection中，当前“审批”中摘录如下：

```
<OperationCollection>
  <Operation Action="ShowModal (&quot;AuditDialog&quot;)" Caption="审批" Key="op1"/>
</OperationCollection>
```

点击“审批”，如Operation中定义的一样，弹出AuditDialog对话框，如下图：

审批对话框

意见

通过

驳回

取消

部门经理有两个选择，一个是“通过”，那么流程转向“已批准”并结束；一个是“驳回”，流程转向“驳回”并结束。审批的结果反应在审批日志中，审批日志位于BPM_Log表中。通过“审批记录”按钮可以显示审批日志，“审批记录”的操作定义如下：

```
<Operation Key="ShowAuditDetil" Caption="审批记录" Visible="ReadOnly()">
  <Action>
    <![CDATA[
ShowModal('WFLog')
]]>
  </Action>
</Operation>
```

这个操作通过WFLog表单显示审批记录，WFLog的源代码如下所示：

```
<Form Caption="审批记录" FormType="Normal" Key="WFLog" PreferSize="1000px, 600px">
  <DataSource>
    <DataObject Key="BPM_Log" NoPrefix="STIN" Caption="审批记录" MainTableKey="BPM_Log" PrimaryType="Entity"
SecondaryType="Normal">
      <TableCollection>
        <Table Key="BPM_Log" Caption="基本信息" DBTableName="BPM_Log" TableMode="Head" SourceType="Query" Persist="True">
          <TableFilter>
            <![CDATA[
workItemID in (select workitemID from BPM_WorkitemInfo where instanceID=(select instanceID from BPM_Instance where OID=?))
]]>
          </TableFilter>
          <ParameterCollection>
            <Parameter Key="OID" Caption="标识" TargetColumn="workItemID" Formula="Para('POID')"/>
          </ParameterCollection>
          <Statement>
            <![CDATA[select workItemID,workItemName,creatTime,finishTime,operatorID,auditResult,userInfo from BPM_Log]]>
          </Statement>
          <Column Key="workItemID" Caption="工作项标识" Persist="True" DataType="Long" DBColumnName="workItemID"/>
          <Column Key="workItemName" Caption="父对象标识" Persist="True" DataType="Varchar" Length="200"
DBColumnName="workItemName"/>
          <Column Key="creatTime" Caption="创建时间" Persist="True" DataType="DateTime" DBColumnName="creatTime"/>
          <Column Key="finishTime" Caption="提交时间" Persist="True" DataType="DateTime" DBColumnName="finishTime"/>
          <Column Key="operatorID" Caption="人员" Persist="True" DataType="Integer" DBColumnName="operatorID"/>
          <Column Key="userInfo" Caption="审批意见" Persist="True" DataType="Varchar" Length="200" DBColumnName="userInfo"/>
          <Column Key="auditResult" Caption="审批结果" Persist="True" DataType="Integer" DBColumnName="auditResult"/>
        </Table>
      </TableCollection>
    </DataObject>
  </DataSource>
  <OperationCollection>
  </OperationCollection>
  <OnLoad type="Formula">
    <![CDATA[LoadData();
]]>
  </OnLoad>
  <FormParaCollection>
    <FormPara Key="POID" Type="Formula" Formula="GetParentOID()"/>
  </FormParaCollection>
  <Body HAlign="Left" OverflowX="Visible" OverflowY="Visible" PopWidth="700px" PopHeight="500px">
    <Block>
      <GridLayoutPanel Key="main">
        <RowDefCollection RowHeight="30" RowGap="3">
          <RowDef Height="30%"/>
          <RowDef Height="70%"/>
        </RowDefCollection>
        <ColumnDefCollection>
          <ColumnDef Width="100%"/>
        </ColumnDefCollection>
        <Grid Key="detail_grid" Caption="入库单明细" UsePage="false" PageRowCount="6" ShowRowHead="true" Enable="False" X="0"
XSpan="1" Y="0" YSpan="1">
          <GridColumnCollection>
            <GridColumn Key="workItemID" Caption="工作项标识" Width="79px" ColumnExpand="false" ColumnType="Detail"/>
            <GridColumn Key="workItemName" Caption="父对象标识" Width="79px" ColumnExpand="false" ColumnType="Detail"/>
            <GridColumn Key="creatTime" Caption="创建时间" Width="160px" ColumnExpand="false" ColumnType="Detail"/>
            <GridColumn Key="finishTime" Caption="提交时间" Width="160px" ColumnExpand="false" ColumnType="Detail"/>
            <GridColumn Key="operatorID" Caption="人员" Width="50px" ColumnExpand="false" ColumnType="Detail"/>
            <GridColumn Key="auditResult" Caption="审批结果" Width="65px" ColumnExpand="false" ColumnType="Detail"/>
            <GridColumn Key="userInfo" Caption="审批意见" Width="79px" ColumnExpand="false" ColumnType="Detail"/>
          </GridColumnCollection>
          <GridRowCollection>
            <GridRow Key="R1" RowType="Detail" TableKey="BPM_Log" GroupKey="Area">
              <GridCell Key="workItemID" Caption="工作项标识" CellType="NumberEditor">
                <DataBinding ColumnKey="workItemID"/>
              </GridCell>
              <GridCell Key="workItemName" Caption="父对象标识" CellType="LABEL">
                <DataBinding ColumnKey="workItemName"/>
              </GridCell>
              <GridCell Key="creatTime" Caption="创建时间" CellType="DatePicker">
                <DataBinding ColumnKey="creatTime"/>
              </GridCell>
              <GridCell Key="finishTime" Caption="提交时间" CellType="DatePicker">

```

```

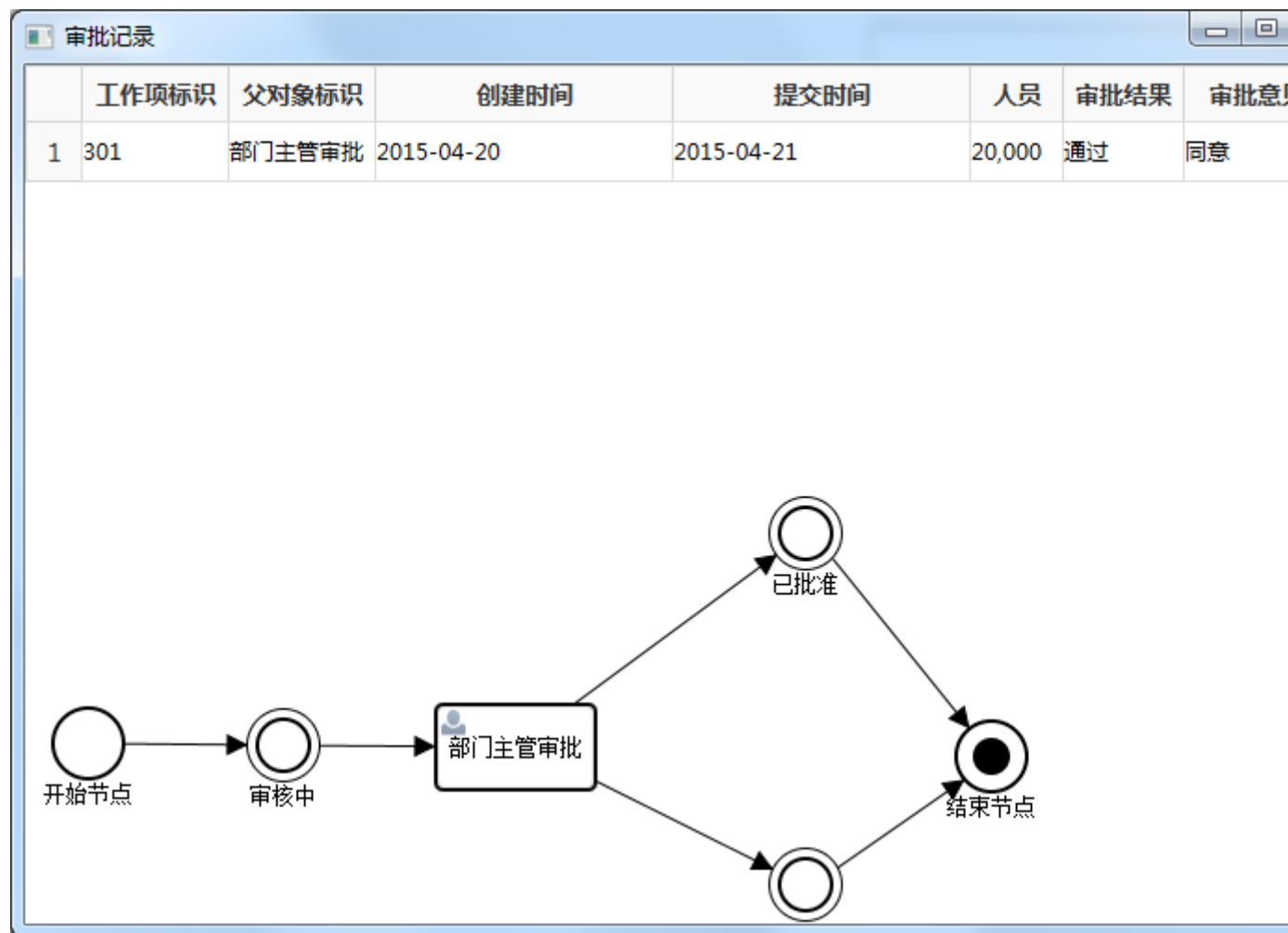
        <DataBinding ColumnKey="finishTime"/>
    </GridCell>
    <GridCell Key="operatorID" Caption="人员" CellType="NumberEditor">
        <DataBinding ColumnKey="operatorID"/>
    </GridCell>
    <GridCell Key="auditResult" Caption="审批结果" CellType="ComboBox" >
        <DataBinding ColumnKey="auditResult"/>
        <Item Caption="通过" Key="1" Value="1"/>
        <Item Caption="驳回" Key="0" Value="0"/>
    </GridCell>
    <GridCell Key="userInfo" Caption="审批意见" CellType="LABEL">
        <DataBinding ColumnKey="userInfo"/>
    </GridCell>
</GridRow>
</GridRowCollection>
</Grid>
<BPMGraph Key="graph" Caption="流程图" ProcessKey="parent.GetProcessKey()" ProcessVer="parent.GetProcessVer()" X="0"
XSpan="1" Y="1" YSpan="1">
    </BPMGraph>
</GridLayoutPanel>
</Block>
</Body>
</Form>

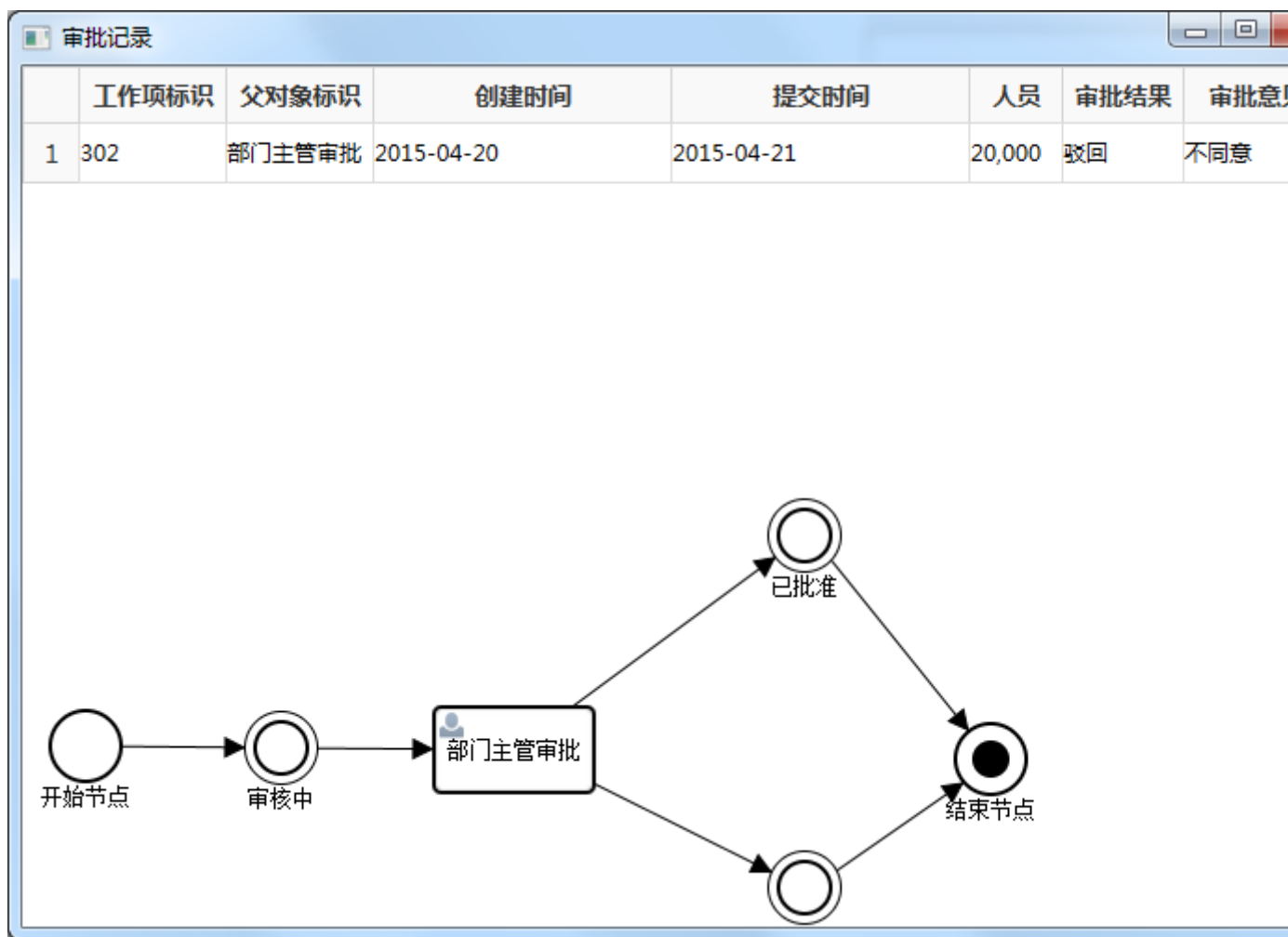
```

WFLog表单通过从BPM_Log表中抽取当前请假单的日志信息来获取审批日志，其通过

```
workItemID in(select workitemID from BPM_WorkitemInfo where instanceID=(select instanceID from BPM_Instance where OID=?))
```

来检索属于当前请假单的审批日志。具体参见流程数据表参考。下面我们以两个请假为例来看一下审批结果，其中一个是通过的审批，另外一个驳回的审批。





经过审批后，根据流程，请假的状态如下：

对象...	父对...	对象...	对象...	工号	姓名	开始...	结束...	事由	状态
40001	-1	-1	0	001	张三	2015...	2015...		审批通过
40002	-1	-1	0	002	李四	2015...	2015...		否决

14. 2: 状态

状态节点用于标识流程关联表单的状态或者仅定义流程的运行状态。

对于状态节点，我们不再另外定义流程，在审批中已经用到了状态，其配置代码如下：

```

<State Status="" UseStateTask="" CreateTrigger="" FinishTrigger="">
  <ParticipatorCollection>
    <Dictionary ItemKey="" ItemID=""/>
    <Query>
      <![CDATA[SQL]]>
      <QueryParameterCollection>
        <QueryParameter Formula="" DataType=""/>
      </QueryParameterCollection>
    </Query>
    <MidFormula Formula=""/>
    ...
  </ParticipatorCollection>
</State>
  
```

这里Status为Audited，定义了表单的关联状态，一旦流转进入该节点时，流程将表单的状态改为“Audited”。

状态节点除具有节点的属性外，还包含以下属性：

- Status 状态的标识；
- UseStateTask 是否在状态节点产生工作项，若后续有StateAction节点参与，则该属性为True，否则为False，默认模式为False。
- ParticipatorCollection 状态节点的参与者，说明状态节点将产生状态机专用工作项，此处没有Operation，他需要的工作项是从后续的StateAction节点中获取的；

14.3: 用户任务

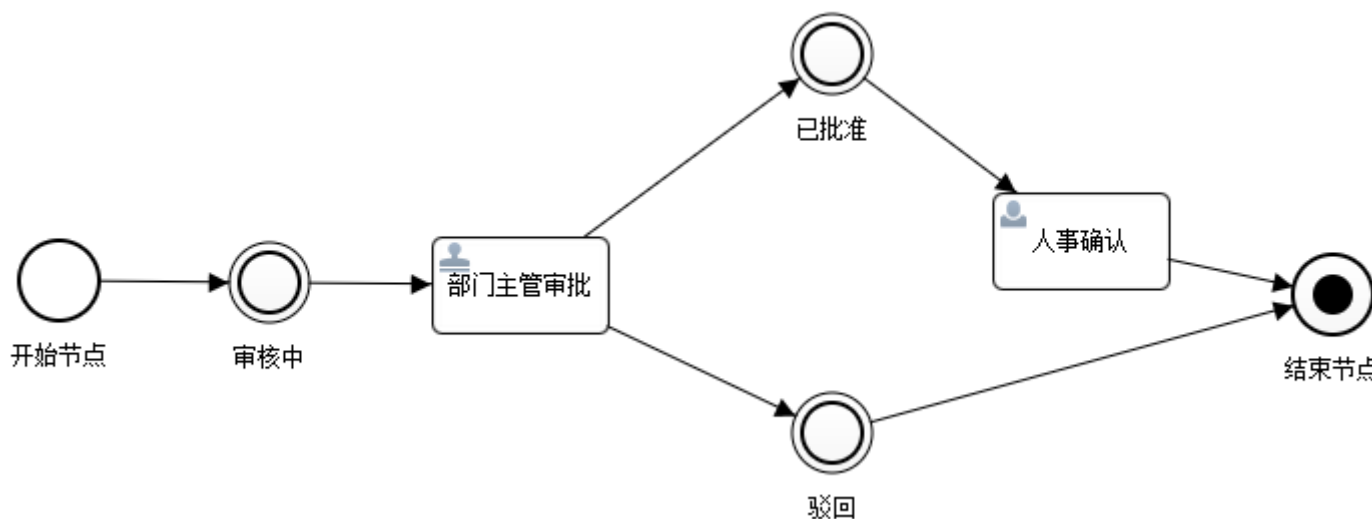
用户任务是一个需要用户通过系统来完成的工作，这里我们仍然以请假流程为例，在审批通过之后需要人事确认有效，在VacationRequest表单增加Valid属性。我们将请假流程更新到新版本(VacationRequestAudit_V2.xml)所示。流程的版本通过复制一个新的文件并将新文件的流程的Version属性改为更高的版本，为规范起见，我们建议流程的文件名采用ProcessKey+“_V”+版本号来定。比如VacationRequestAudit流程的初始版本为VacationRequestAudit_V1，新生成的版本为VacationRequestAudit_V2，升级后的流程的信息如下：

```
<Process Caption="请假" Key="VacationRequestAudit" Version="2">
...
</Process>
```

将请假单的流程部署信息改为：

```
<DeployInfo InitDate="2014.06.12 17:39:41 GMT+08:00" Key="VacationRequestAudit" Version="2"/>
```

这里我们在新版本的流程中增加一个用户任务“人事确认”，如下图所示：



“人事确认”的源代码如下：

```
<UserTask Key="Confirm" Caption="人事确认" ID="200">
  <TransitionCollection>
    <SequenceFlow Key="End" ID="108" TargetNodeKey="End">
      <TransitionGraphic LineStyle="StraightLine"/>
    </SequenceFlow>
  </TransitionCollection>
  <OperationCollection>
    <Operation Caption="确认" Key="op1">
      <Action>
        <![CDATA[Valid=True;SaveData();CommitWorkitem(-1, 1, "已确认");]]>
      </Action>
    </Operation>
  </OperationCollection>
</UserTask>
```



```
</Operation>
</OperationCollection>
<ParticipatorCollection>
  <Dictionary ItemKey="Operator" ItemID="20002"/>
</ParticipatorCollection>
<NodeGraphic Height="48" Width="88" X="500" Y="202"/>
</UserTask>
```

在这任务中我们定义任务的处理人为HR (OpertorID的OID为20002的操作员)；确认操作的定义为修改请假单的有效性标志为True，保存数据后提交完成当前工作项。

新建一个请假单，经过提交和审批后，进入“人事确认”任务，打开任务，界面如下：

我的待办事宜

请假单

确认

审批记录

工号	001
姓名	张三
开始时间	2015-04-22
结束时间	2015-04-22
事由	
类型	事假
总计	1
状态	审批通过
有效性	<input type="checkbox"/> 有效性

点击确认后，“有效性”被打勾，如下图：

我的待办事宜

请假单

确认

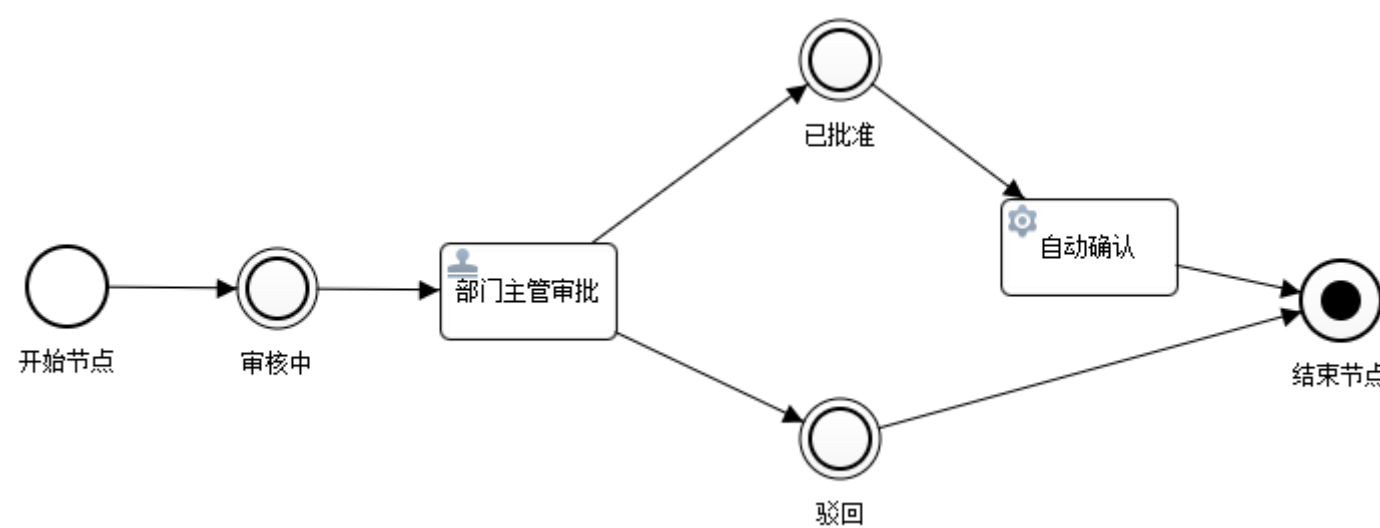
审批记录

工号	<div>001</div>
姓名	<div>张三</div>
开始时间	<div>2015-04-22</div>
结束时间	<div>2015-04-22</div>
事由	<div></div>
类型	<div>事假</div>
总计	<div>1</div>
状态	<div>审批通过</div>
有效性	<div><input checked="" type="checkbox"/> 有效性</div>

而此时，流程向后流转到结束。

14.4: 系统任务

系统任务是系统自动处理的任务，在上面的例子中，我们将手工确认有效性改为系统在审批通过后自动确认为有效，将流程版本升为3，全部代码请见VacationRequestAudit_V3.xml；我们将版本2中的用户任务改为系统任务，流程如下图所示：



系统任务的源代码如下：

```
<ServiceTask Key="Confirm" Caption="自动确认" ID="200">
  <TransitionCollection>
    <SequenceFlow Key="End" ID="108" TargetNodeKey="End">
      <TransitionGraphic LineStyle="StraightLine"/>
    </SequenceFlow>
  </TransitionCollection>
  <NodeGraphic Height="48" Width="88" X="500" Y="180"/>
</ServiceTask>
```

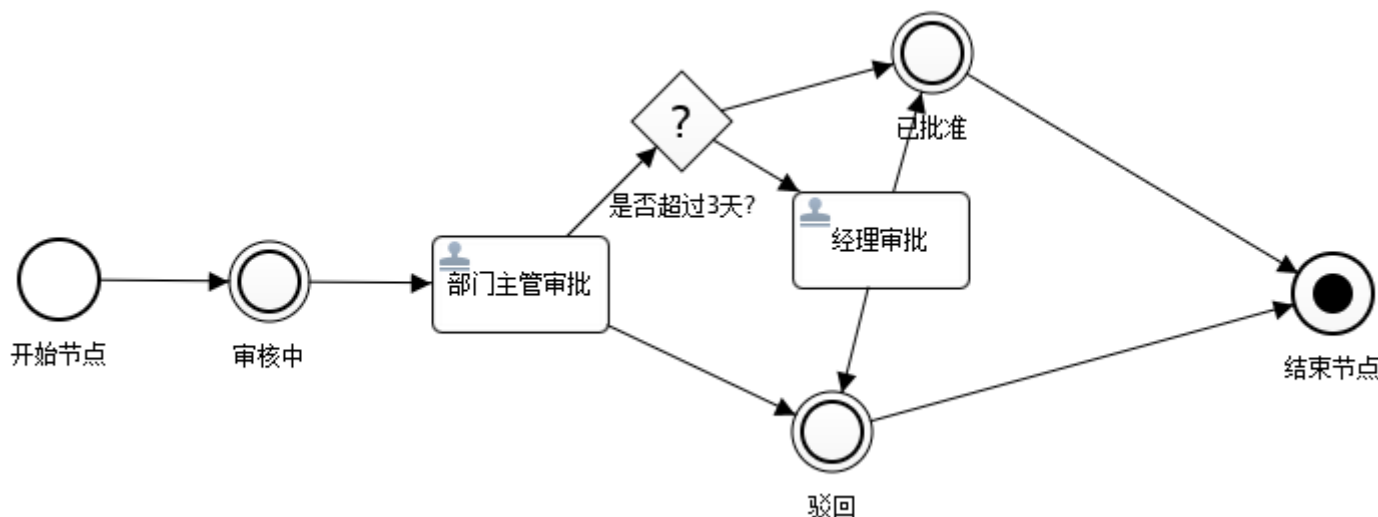
修改请假单的流程部署信息为：

```
<DeployInfo InitDate="2014.06.12 17:39:41 GMT+08:00" Key="VacationRequestAudit" Version="3"/>
```

在“部门主管审批”之后，在批准的情况下，“自动确认”被自动执行，将表单的“有效性”标志置为1。

14.5: 选择

选择用于根据特定的条件决定后续的执行路径，在请假流程中，我们以“部门主管审批”之后，如果请假天数超过3天需要增加经理审批，请假流程增加到版本4，将流程改成如下图所示：



这里增加了一个判断请假天数的节点，选择节点的源代码如下：

```
<Decision Key="CheckDays" Caption="是否超过3天?" ID="201"
Condition="GetValue(&quot;VacationRequest&quot;;&quot;DayCount&quot;)>3">
  <TransitionCollection>
    <SequenceFlow Condition="True" ID="202" TargetNodeKey="UserTask4">
      <TransitionGraphic LineStyle="StraightLine"/>
    </SequenceFlow>
    <SequenceFlow Condition="False" ID="203" TargetNodeKey="State1">
      <TransitionGraphic LineStyle="StraightLine"/>
    </SequenceFlow>
  </TransitionCollection>
  <NodeGraphic Height="48" Width="88" X="320" Y="120"/>
</Decision>
```

这里，Condition定义了判断的条件为请假天数大于3天，如果大于3天，流转到UserTask4(经理审批)，否则流转到State1(已批准)。

用Staff操作员分别生成两张请假单，请假天数分别是1天和4天，经常审批之后我们观察其审批历史。

请假天数为1天的审批历史如下：

	工作项标识	父对象标识	创建时间	提交时间	人员	审批结果	审批意见
1	301	部门主管审批	2015-04-23	2015-04-23	20,000	通过	同意

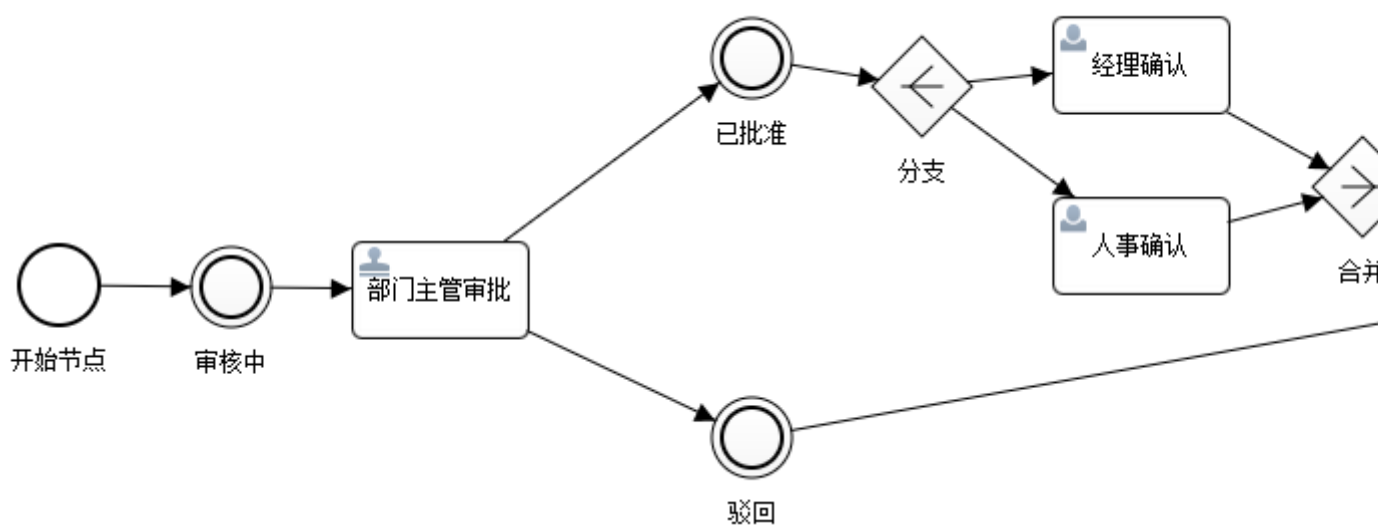
请假天数为4天的审批历史如下：

	工作项标识	父对象标识	创建时间	提交时间	人员	审批结果	审批意见
1	302	部门主管审批	2015-04-23	2015-04-23	20,000	通过	同意
2	303	经理审批	2015-04-23	2015-04-23	20,001	通过	同意

可以看到在请假天数为4天的情况下，流程流转到了“经理审批”。

14.6: 分支与合并

分支节点用于连接后续多个可以并行处理的任务，假定在“部门主管”审批之后，需要告知经理和人事，经理和人事需要确认知晓，请假流程才算正式结束。这里我们使用分支来描述后续的“经理确认”和“人事确认”两个任务，我们将流程升级到版本5，如下：



分支的源代码如下：

```

<Fork Key="Fork" Caption="分支" ID="202">
  <TransitionCollection>
    <SequenceFlow TargetNodeKey="ManagerConfirm">
      <TransitionGraphic LineStyle="StraightLine"/>
    </SequenceFlow>
    <SequenceFlow TargetNodeKey="HRConfirm">
      <TransitionGraphic LineStyle="StraightLine"/>
    </SequenceFlow>
  </TransitionCollection>
  <NodeGraphic Height="48" Width="88" X="440" Y="100"/>
</Fork>

```

合并的源代码如下：

```

<Join Key="Join" Caption="合并" ID="203">
  <TransitionCollection>
    <SequenceFlow ID="108" TargetNodeKey="End">
      <TransitionGraphic LineStyle="StraightLine"/>
    </SequenceFlow>
  </TransitionCollection>
  <NodeGraphic Height="48" Width="88" X="660" Y="150"/>
</Join>

```

在部门经理审批通过后，分别使用经理 (Manager) 和人事 (HR) 登录系统查询工作台可以看到，分别都有一个确认任务。

经理工作台：

我的待办事宜

工作项标识	工作项名称	创建时间
402	经理确认	2015-04-23

人事工作台：

我的待办事宜

工作项标识	工作项名称	创建时间
403	人事确认	2015-04-23

审批日志如下图所示：

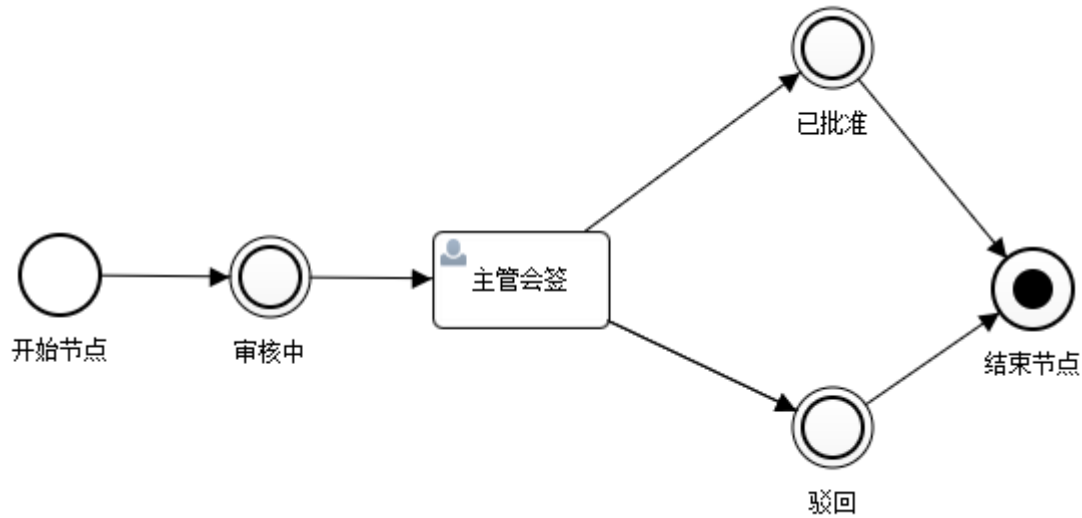
	工作项标识	父对象标识	创建时间	提交时间	人员	审批结果	审批意见
1	401	部门主管审批	2015-04-23	2015-04-23	20,000	通过	同意
2	402	经理确认	2015-04-23		-1		
3	403	人事确认	2015-04-23		-1		

在经理和人事都确认之前，流程实例并不结束，只有在人事和经理都确认后，请假流程才算结束。可以通过先让经理确认，然后在人事的工作台中仍然应该可以看到“人事确认”的任务。确认完成后，日志如下：

	工作项标识	父对象标识	创建时间	提交时间	人员	审批结果	审批意见
1	401	部门主管审批	2015-04-23	2015-04-23	20,000	通过	同意
2	402	经理确认	2015-04-23	2015-04-24	20,001	通过	已确认
3	403	人事确认	2015-04-23	2015-04-24	20,002	通过	已确认

14.7: 会签

在流程业务管理中，我们可以把一个人到另一个人的签字审批的每个环节都定义为任务，但若是这样，这个流程业务有一点固定模式，就是说签批人是固定的。而多个人同时处理一个任务，这种任务我们称之为会签任务。这种业务需求也很常见，如一个请款单，领导审批环节中，就需要多个部门领导签字。流程图如下所示：



流转进入会签时，系统创建根据工作项的参与者，为每个工作项创建一个工作项；工作项由以下三个条件，分别是：

- 会签的完成条件，默认情况下必须所有参考者都进行会签，可以通过自定义函数改变会签的完成条件，表达式的内容返回True时即可以完成会签；
- 会签的成立条件，只有满足会签的成立条件时，会签的结果才有效，否则会签的结果为2(不成立)；会签的成立条件有三种定义方式：
 - 固定数量，在会签数量满足固定数值的会签数目时，会签即成立；
 - 比例，在会签数量满足所有参与者的比例时，会签即成立；
 - 自定义，会签的成立条件通过表达式定义，表达式返回True时会签成立，否则不成立；
- 会签的通过条件，在会签成立的情况下，需要计算会签的结果，返回值为1(通过)和0(不通过)，有三种通过条件：
 - 固定数量，在通过会签的数目大于等于固定数量时即表示会签通过；
 - 比例，在通过会签的数目达到所有参与者的某个比例时表示会签通过；
 - 自定义，通过表达式的返回值来判断，True为通过，False为不通过。

会签节点的源代码如下：

```

<Countersign Caption="会签" Key="Countersign" ID="" PassType="" PassCondition="" ValidType="" ValidCondition=""
FinishCondition="">
  <TransitionCollection>
    <SequenceFlow Condition="0" ID="" TargetNodeKey="">
      <TransitionGraphic LineStyle="StraightLine"/>
    </SequenceFlow>
    <SequenceFlow Condition="1" ID="" TargetNodeKey="">
      <TransitionGraphic LineStyle="StraightLine"/>
    </SequenceFlow>
  </TransitionCollection>
  <OperationCollection>
    <Operation Caption="审批" Key="op1">
      <Action>
        <![CDATA[ShowModal("AuditDialog")]]>
      </Action>
    </Operation>
  </OperationCollection>
  <AssistanceCollection/>
  <ParticipatorCollection>
    <Dictionary ItemKey="Operator" ItemID=""/> //获取会签用户的集合，是在后台会签节点上配置的人员；
  </ParticipatorCollection>
</Countersign>
  
```

```
<TimerCollection>
  <TimerAutoPass UserInfo=""/>
  <TimerAutoDeny UserInfo=""/>
  <TimerAutoAbstain UserInfo=""/>
</TimerCollection>
<NodeGraphic Height="48" Width="88" X="180" Y="202"/>
</Countersign>
```

当然也可以节点内部发起会签，只需在辅助节点集合中定义节点内部流转所需要的节点任务会签。

```
<AssistanceCollection>
  <Countersign/>
</AssistanceCollection>
```

主要负责会签节点的几个属性：PassType=""（通过类型）、PassCondition=""（通过条件）、ValidType=""（成立条件类型）、ValidCondition=""（成立条件）、FinishCondition=""（完成会签的条件）

实例分析：员工唐山海提交请假单，主管会签参与者为：选择角色为部门经理、人事和总经理的操作员（本数据库共四人），会签通过的条件为其中2人同意会签就会通过，会签成立的条件为其中50%人数参与审批。会签完成的条件为默认条件。

步骤一：本示例设定组织架构如下表：

	销售部（部门经理：张若昀）	
总经理室（郑京浩 2015）		
	行政部（部门经理：陈深）	人事部（萧雅）

步骤二：会签流程定义

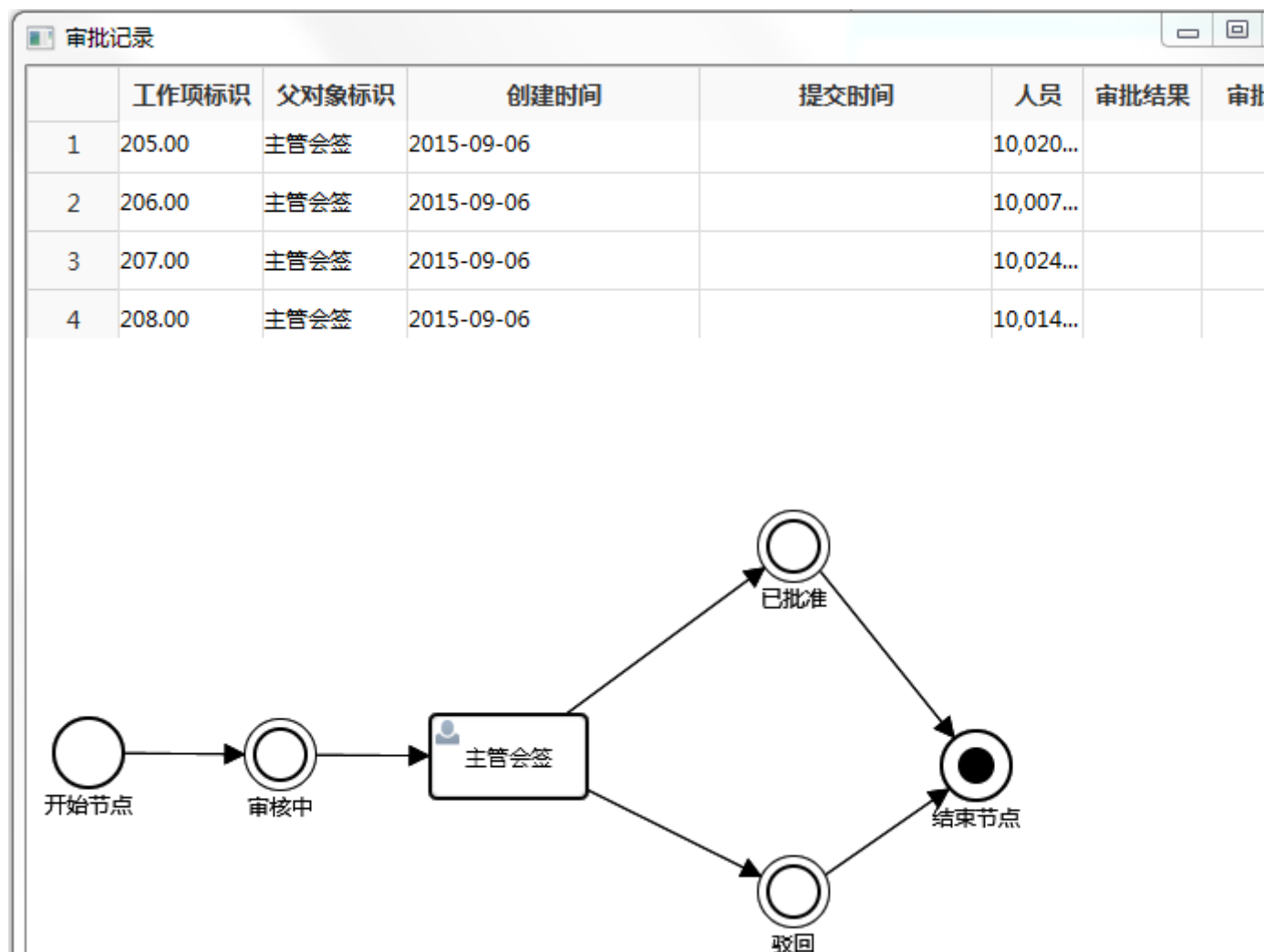
流程源代码如下：

```
<Process Caption="请假" Key="VacationRequestAudit">
  <Begin Caption="开始节点" ID="1" Key="Begin">
    <TransitionCollection>
      <SequenceFlow Condition="" ID="9" Key="BegintoState" TargetNodeKey="State">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic X="14" Y="204"/>
  </Begin>
  <State Caption="审核中" ID="2" Key="State" Status="Auditing">
    <TransitionCollection>
      <SequenceFlow Condition="" ID="10" Key="StatetoUserTask3" TargetNodeKey="UserTask3">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic X="119" Y="205"/>
  </State>
  <Countersign Caption="主管会签" Key="UserTask3" ID="105" PassType="Number" PassCondition="2" ValidType="Proportion" ValidCondition="50%" FinishCondition="">
    <TransitionCollection>
      <SequenceFlow Condition="1" Key="UserTask3_State1" ID="108" TargetNodeKey="State1">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
      <SequenceFlow Condition="0" Key="UserTask3_State2" ID="109" TargetNodeKey="State2">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
    </TransitionCollection>
    <OperationCollection>
      <Operation Caption="审批" Key="op1">
        <Action>
          <![CDATA[ShowModal("AuditDialog")]]>
        </Action>
      </Operation>
    </OperationCollection>
    <AssistanceCollection/>
    <ParticipatorCollection>
      <Query>
        <![CDATA[SELECT OID FROM SYS_Operator WHERE SOID IN (SELECT SOID FROM SYS_OperatorRole WHERE Role IN (10008,10009, 10010))]]>
      </Query>
    </ParticipatorCollection>
```

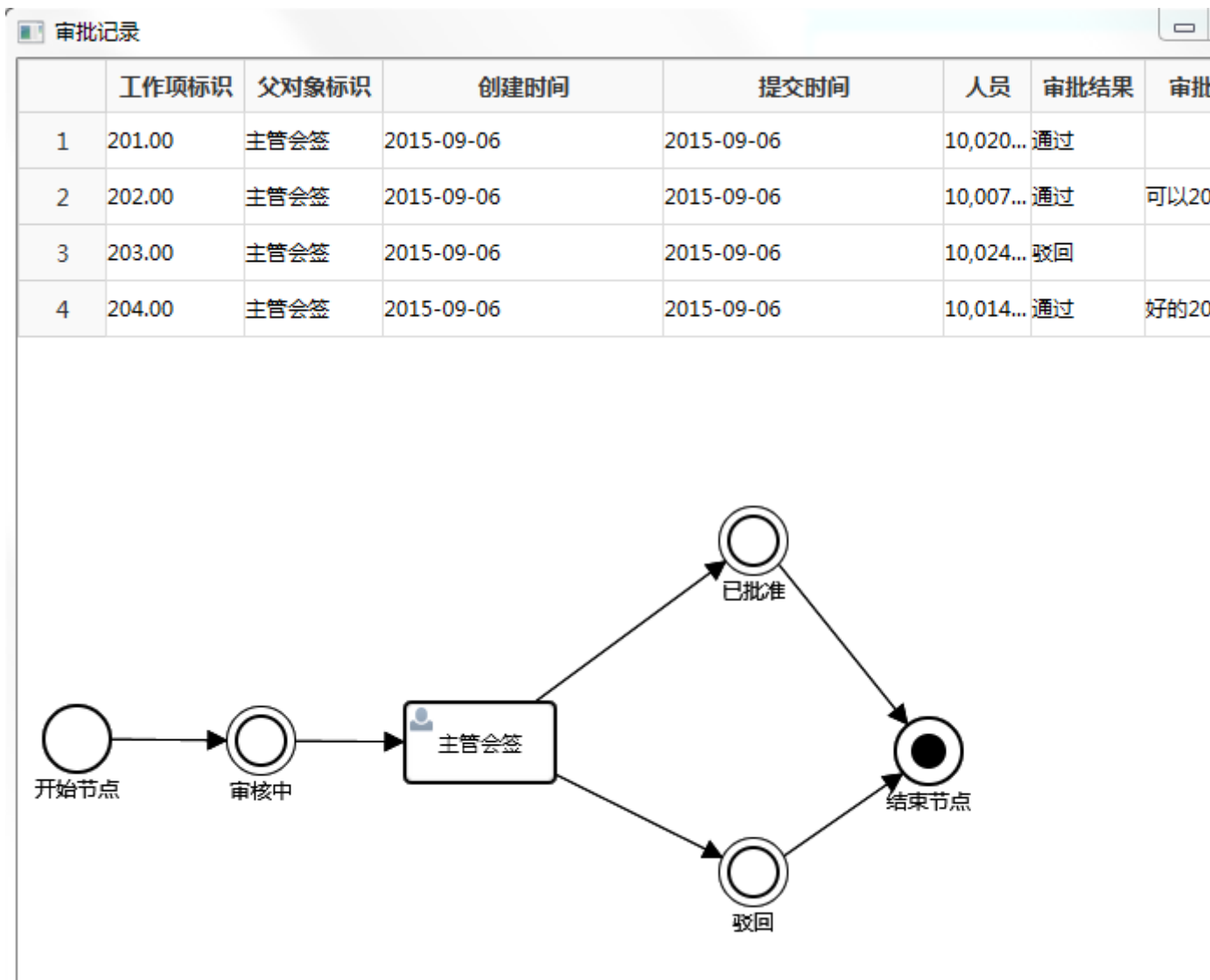
```

        <NodeGraphic Height="48" Width="88" X="220" Y="202"/>
    </Countersign>
    <State Caption="已批准" Key="State1" ID="104" Status="Audited">
        <TransitionCollection>
            <SequenceFlow Condition="" ID="21" TargetNodeKey="End">
                <TransitionGraphic LineStyle="StraightLine"/>
            </SequenceFlow>
        </TransitionCollection>
        <NodeGraphic Height="34" Width="48" X="400" Y="91"/>
    </State>
    <State Caption="驳回" Key="State2" ID="106" Status="Denied">
        <TransitionCollection>
            <SequenceFlow Condition="" ID="23" Key="EE" TargetNodeKey="End">
                <TransitionGraphic LineStyle="StraightLine"/>
            </SequenceFlow>
        </TransitionCollection>
        <NodeGraphic Width="78" X="400" Y="280"/>
    </State>
    <End Caption="结束节点" ID="8" Key="End">
        <TransitionCollection/>
        <NodeGraphic X="500" Y="211"/>
    </End>
</Process>
    
```

步骤三：员工唐山海提交一张请假单，流程启动后，根据会签条件，会流转到角色OID为10008（总经理）、10009（部门经理）和10010（人事）的工作台。流程成功启动，系统在BPM_log中创建4条会签，且状态为1流程成功启动。



步骤四：而该会签成立的条件为两位操作员审批通过，则为通过该任务节点。批准请假，流程结束；若不满足以上会签条件，驳回申请，该流程亦结束。



审批成功，BPM_Instance流程状态为2。 BPM_Node为中对应记录的Result为1。详细配置代码请参照BPM_C2.7。

14. 8:子流程

当一个业务流程办理的节点很多，或者说业务流程实例启动后持续办理的周期很长，甚至几个月的时候，那么这种类型的流程节点数量一定很多。用流程建模的工具来查看或者编辑，会显得很笨拙，节点数量太多，一个界面放不下，这种情况，我们可以选择子流程的方式来分解，这样会使界面很简洁。流转进入子流程时，系统创建子流程任务；如果中间层创建子流程，那么执行创建子流程事件，否则等客户端操作完成子流程创建；在创建子流程后，系统通过同步模式来决定处理方式：在同步的情况下，必须等待所创建的子流程完成才可以继续向下流转，在异步的情况下，子流程创建后，当前流程实例继续向下流转。我们只需要给SubProcess流程节点添加transition，并且唯一命名每个transition。

```
<SubProcess Caption="子流程" ID="4" Key="SubProcess" SubProcessKey="SubProcess1" SyncMode="Sync" AutoSubInstanceTrigger="">
  <TransitionCollection>
    <SequenceFlow ID="11" TargetNodeKey="Task3">
      <TransitionGraphic/>
    </SequenceFlow>
  </TransitionCollection>
  <NodeGraphic Height="75" Width="130" X="345" Y="203"/>
  <OperationCollection/>
  <AssistanceCollection/>
  <ParticipatorCollection/>
  <TimerItemCollection/>
</SubProcess>
```

```
<Perm/>
</SubProcess>
```

子流程节点，除了具有用户任务的所有属性外，同时包括以下属性：

- SyncMode 同步模式，取值为Sync(同步)、Async(异步)，默认值为Sync；
- AutoSubInstanceTrigger 中间层自动创建子流程触发器，此属性表明中间层公式自动创建子流程实例；
- SubProcessKey 子流程标识，说明了节点会使用的流程。

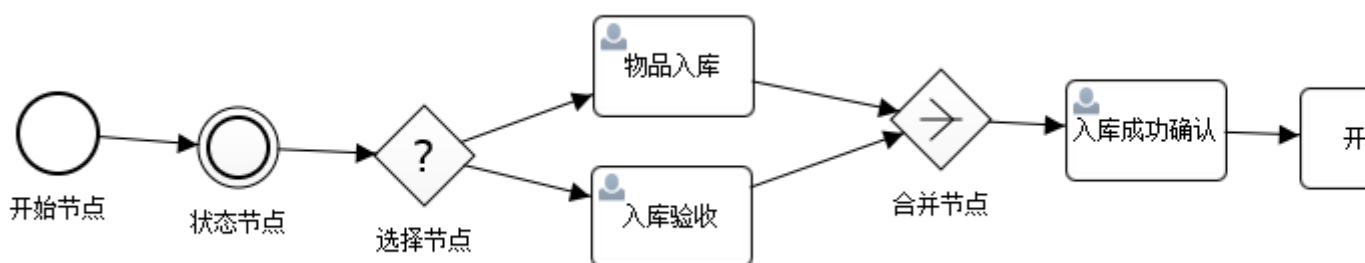


注意

本节点涉及到二次开发内容，具体实例参考专题教程。

14.9: 数据映射

在如下流程图中，开发票即为数据映射节点。



该流程中数据映射节点的源代码如下：

```

<DataMap Caption="开发票" ID="8" Key="DataMap01" SyncTriggerType="InstanceStart">
  <TransitionCollection>
    <SequenceFlow Caption="顺序流" ID="9" Key="SequenceFlow4" TargetNodeKey="State1"/>
  </TransitionCollection>
  <NodeGraphic X="917" Y="153"/>
  <OperationCollection>
    <Operation Caption="数据映射" Key="op3">
      <Action><![CDATA[
        BPMMMap("StockIn2StockIn1", "StockIn1")
      ]]></Action>
    </Operation>
    <Operation Caption="提交" Key="op1">
      <Action><![CDATA[
        CommitWorkitem(-1, 1, "")
      ]]></Action>
    </Operation>
  </OperationCollection>
  <AssistanceCollection/>
  <ParticipatorCollection>
    <Query><![CDATA[
      SELECT OID FROM SYS_Operator
    ]]></Query>
  </ParticipatorCollection>
  <TimerItemCollection/>
  <Perm/>
  <TransitionCollection>
    <SequenceFlow Caption="顺序流" ID="9" Key="SequenceFlow4" TargetNodeKey="State1"/>
  </TransitionCollection>
  <OperationCollection>
    <Operation Caption="数据映射" Key="op3">
      <Action><![CDATA[
        BPMMMap("StockIn2StockIn1", "StockIn1")
      ]]></Action>
    </Operation>
    <Operation Caption="提交" Key="op1">
      <Action><![CDATA[

```

```

        CommitWorkitem(-1,1,"")
    ]]></Action>
</Operation>
</OperationCollection>
<AssistanceCollection/>
<ParticipatorCollection>
    <Query><![CDATA[
        SELECT  OID FROM SYS_Operator
    ]]></Query>
</ParticipatorCollection>
<BillDataMapInfoCollection>
    <BillDataMapInfo DataMapKey="StockIn2StockIn1"/>
</BillDataMapInfoCollection>
<NodeGraphic X="917" Y="153"/>
</DataMap>

```

数据映射节点除了具有用户任务的所有属性外，还具有以下属性：

- MidDataMap 是否中间层映射，取值为True和False，默认为False；
- SyncTriggerType 同步触发时机，取值为DataSave(数据保存时)、InstanceStart(实例启动时)、InstanceEnd(实例结束时)，默认值为空；
- DataMapFinishCondition 数据映射完结条件；
- DataMapInfoCollection 数据映射集合；包含多个映射定义，每个映射的属性如下：
 - DataMapKey 数据映射关系标识；
 - Caption 数据映射关系的名称(这里只是一个流程中的代称)；
 - AutoStartMidMapInstance 是否中间层自动映射，取值为True和False，默认为False；

详细代码参照文件夹BPM_C2.9.

14. 10: 加签

EndorseTask ：在指定的工作项上发起加签动作，会产生一个新的加签工作项，只有这个加签工作项被提交了，当前流程才会继续流转。

参数一： workitemID (Long) 工作项ID，值为-1时取当前界面的工作项

参数二： operatorID (Integer) 加签目标操作员的ID

参数三： launchInfo (String) 发起加签的备注(可以不提供，则为默认值false)

加签任务的处理通过任务的操作集合来定义，这里定义了一个处理操作，即弹出一个加签对话框EndorseDialog来处理加签任务；这里EndorseDialog对话框通过endorsetask来完成加签任务，该函数的原型为void Endorse Task(workitemID, operatorID , launchInfo)，其中workitemID为工作项的标识，operatorID 为加签目标操作员的ID，launchInfo为 发起加签的备注，例如：

```

//当前界面的工作项发起加签
endorsetask(-1, operatorID, "请指示!")

```

配置举例分析：

EndorseDialog对话框的定义这里列出其代码：

```

<Form Caption="加签" Key="EndorseDialog">
    <DataSource/>
    <Body PopHeight="300px" PopWidth="800px">
        <Block>
            <GridLayoutPanel Caption="GridLayoutPanel0" Key="GridLayoutPanel0">
                <Button Caption="取消" Key="Button1" X="5" Y="4">
                    <OnClick RunType="Client"><![CDATA[close();]]></OnClick>
                </Button>
                <CheckBox Caption="始终有效" Key="alwaysValid" X="4" Y="3"/>
                <Button Caption="确定" Key="Button0" X="4" Y="4">

```

```

        <!-- <OnClick RunType="Client">
        <![CDATA[endorsetask(-1, getvalue("OperatorID"));Close();]]>
        </OnClick-->
        </Button>
        <Label Caption="加签人员" Key="Label1" X="1" Y="1"/>
        <Dict AllowMultiSelection="true" Caption="OperatorID" ItemKey="Operator" Key="OperatorID" X="2" XSpan="2"
Y="1"/>
        <TextArea Caption="意见" Enable="True" HAlign="Left" HasBorder="false" Key="Opinion" X="1" XSpan="3" Y="2"
YSpan="2">
            <DataBinding/>
        </TextArea>
        <RowDefCollection RowGap="8">
            <RowDef Height="20px"/>
            <RowDef Height="30px"/>
            <RowDef Height="30px"/>
            <RowDef Height="30px"/>
            <RowDef Height="30px"/>
        </RowDefCollection>
        <ColumnDefCollection ColumnGap="8">
            <ColumnDef Width="19px"/>
            <ColumnDef Width="70px"/>
            <ColumnDef Width="196px"/>
            <ColumnDef Width="13px"/>
            <ColumnDef Width="70px"/>
            <ColumnDef Width="74px"/>
            <ColumnDef Width="70px"/>
        </ColumnDefCollection>
        </GridLayoutPanel>
    </Block>
    <LayoutCollection/>
</Body>
</Form>

```

在流程文件主管审批节点中加一操作“加签”。源代码如下：

```

<Audit Caption="部门主管审批" Key="UserTask3" ID="105">
    <TransitionCollection>
        <SequenceFlow Condition="True" Key="UserTask3_State1" ID="108" TargetNodeKey="State1">
            <TransitionGraphic LineStyle="StraightLine"/>
        </SequenceFlow>
        <SequenceFlow Condition="False" Key="UserTask3_State2" ID="109" TargetNodeKey="State2">
            <TransitionGraphic LineStyle="StraightLine"/>
        </SequenceFlow>
    </TransitionCollection>
    <OperationCollection>
        <Operation Caption="审批" Key="op1">
            <Action>
                <![CDATA[ShowModal("AuditDialog")]]>
            </Action>
        </Operation>
        <Operation Caption="加签" Key="op2">
            <Action>
                <![CDATA[ShowModal("EndorseDialog")]]>
            </Action>
        </Operation>
    </OperationCollection>
    <AssistanceCollection/>
    <ParticipatorCollection>
        <Dictionary ItemKey="Operator" ItemID="10904"/>
    </ParticipatorCollection>
    <NodeGraphic Height="48" Width="88" X="220" Y="202"/>
</Audit>

```

界面展示：

请假单

审批

加签

审批记录

工号

姓名

开始时间

结束时间

事由

类型

总计

状态

有效性

加签

加签人员

2015071001 郑京浩

意见

请指示！

☐ 始终有效

确定

取消

详细代码参照文件夹BPM_C2.10。

14.11: 流程重启/重提交

14.11.1: 流程重启/重提交

(Begin) 开始结点：

```
<Begin ResetInstance="" State="">
</Begin>
```

开始节点具有节点的属性，同时具有以下属性：

- ResetInstance 流转进入开始节点时是否重置流程，取值为True和False，默认值为True；
- State 重置流程的情况下，数据对象的状态，即将数据对象的状态重置为该属性定义的状态。

示例：

```
<Begin Caption="开始节点" Key="Begin" ResetInstance="True" State="Init">
  <TransitionCollection>
    <SequenceFlow Condition="" Key="BegintoState" TargetNodeKey="State">
      <TransitionGraphic LineStyle="StraightLine"/>
    </SequenceFlow>
  </TransitionCollection>
  <NodeGraphic X="14" Y="204"/>
</Begin>
```

RestartInstance：重启一个流程实例 instanceID 为流程实例的编号，若值为-1，代表从当前界面环境中获取instanceID。

```
// 重启请假流程
RestartInstance("~VacationRequest")
```

配置举例分析：

请假流程中添加重启流程的操作：

```

<Process Caption="请假" Key="VacationRequestAudit">
  <Begin Caption="开始节点" ID="1" Key="Begin">
    <TransitionCollection>
      <SequenceFlow Condition="" ID="9" Key="BegintoState" TargetNodeKey="State">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic X="14" Y="204"/>
  </Begin>
  <State Caption="审核中" ID="2" Key="State" Status="Auditing">
    <TransitionCollection>
      <SequenceFlow Condition="" ID="10" TargetNodeKey="UserTask3">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic X="100" Y="205"/>
  </State>
  <Audit Caption="部门主管审批" Key="UserTask3" ID="105">
    <TransitionCollection>
      <SequenceFlow Condition="True" ID="108" TargetNodeKey="State1">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
      <SequenceFlow Condition="False" ID="109" TargetNodeKey="State2">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
    </TransitionCollection>
    <OperationCollection>
      <Operation Caption="审批" Key="op1">
        <Action>
          <![CDATA[ShowModal("AuditDialog")]]>
        </Action>
      </Operation>
      <Operation Caption="重启流程" Key="op2">
        <Action>
          <![CDATA[RestartInstance(-1);]]>
        </Action>
      </Operation>
    </OperationCollection>
    <AssistanceCollection/>
    <ParticipatorCollection>
      <Dictionary ItemKey="Operator" ItemID="10004"/>
    </ParticipatorCollection>
    <NodeGraphic Height="48" Width="88" X="180" Y="202"/>
  </Audit>
  <State Caption="已批准" Key="State1" ID="104" Status="Audited">
    <TransitionCollection>
      <SequenceFlow Condition="" ID="21" TargetNodeKey="Fork">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic Height="34" Width="48" X="360" Y="91"/>
  </State>
  <State Caption="驳回" Key="State2" ID="106" Status="Denied">
    <TransitionCollection>
      <SequenceFlow Condition="" ID="23" TargetNodeKey="End">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic Width="78" X="360" Y="280"/>
  </State>
  <Fork Key="Fork" Caption="分支" ID="202">
    <TransitionCollection>
      <SequenceFlow TargetNodeKey="ManagerConfirm">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
      <SequenceFlow TargetNodeKey="HRConfirm">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic Height="48" Width="88" X="440" Y="100"/>
  </Fork>
  <UserTask Key="ManagerConfirm" Caption="经理确认" ID="201">
    <TransitionCollection>
      <SequenceFlow ID="108" TargetNodeKey="Join">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
    </TransitionCollection>
    <OperationCollection>
      <Operation Caption="确认" Key="op1">
        <Action>
          <![CDATA[CommitWorkitem(-1, 1, "已确认");]]>
        </Action>
      </Operation>
    </OperationCollection>
  </UserTask>
  <Join Key="Join" Caption="汇合" ID="203">
    <TransitionCollection>
      <SequenceFlow TargetNodeKey="End">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic Height="34" Width="48" X="520" Y="91"/>
  </Join>
  <End Caption="结束节点" ID="3" Key="End">
    <NodeGraphic X="554" Y="204"/>
  </End>
</Process>

```

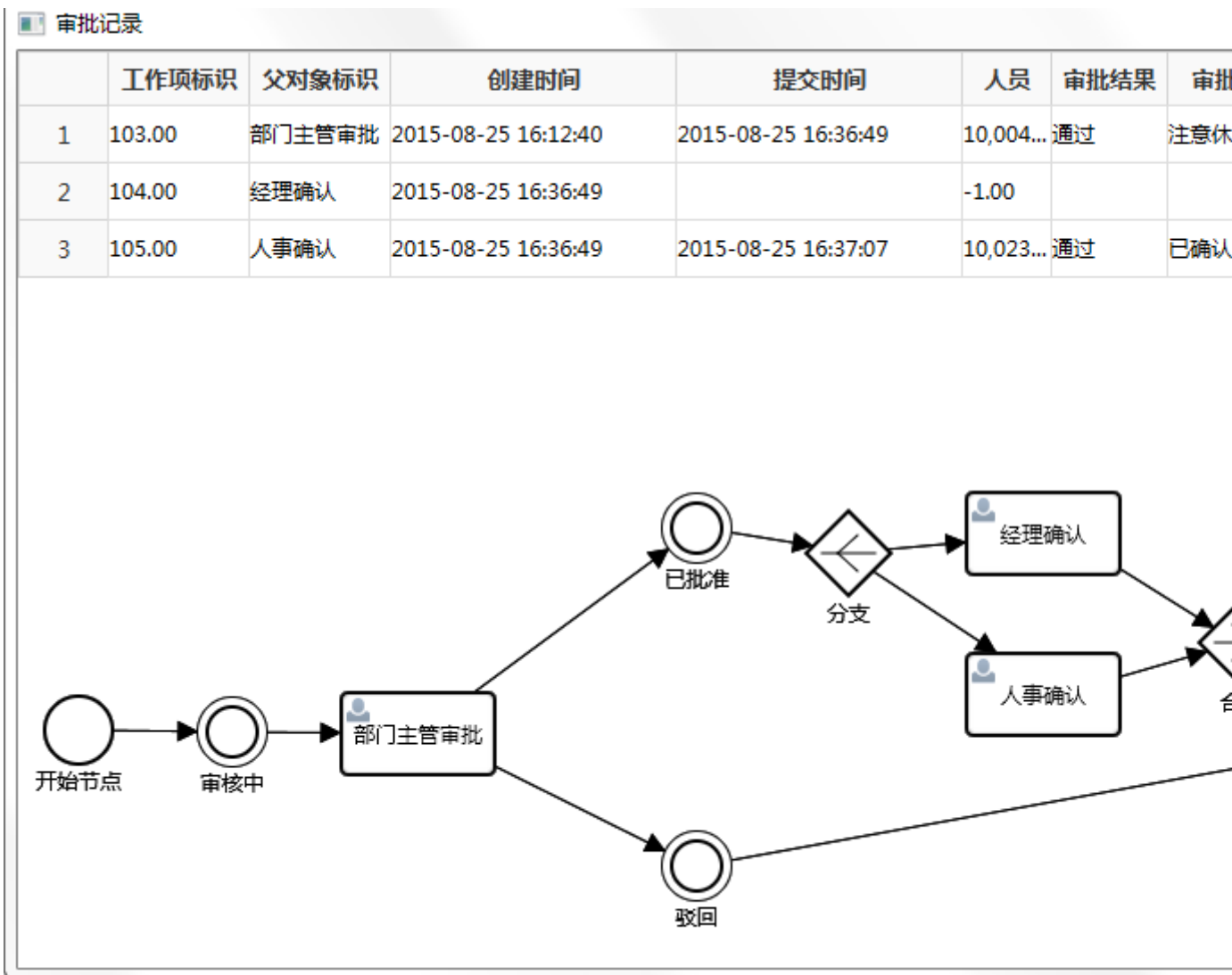
```

        </Operation>
        <Operation Caption="重启流程" Key="op2">
            <Action>
                <![CDATA[RestartInstance(-1);]]>
            </Action>
        </Operation>
    </OperationCollection>
    <AssistanceCollection/>
    <ParticipatorCollection>
        <Dictionary ItemKey="Operator" ItemID="10026"/>
    </ParticipatorCollection>
    <NodeGraphic Height="48" Width="88" X="530" Y="90"/>
</UserTask>
<UserTask Key="HRConfirm" Caption="人事确认" ID="200">
    <TransitionCollection>
        <SequenceFlow ID="108" TargetNodeKey="Join">
            <TransitionGraphic LineStyle="StraightLine"/>
        </SequenceFlow>
    </TransitionCollection>
    <OperationCollection>
        <Operation Caption="确认" Key="op1">
            <Action>
                <![CDATA[CommitWorkitem(-1, 1, "已确认");]]>
            </Action>
        </Operation>
        <Operation Caption="重启流程" Key="op2">
            <Action>
                <![CDATA[RestartInstance(-1);]]>
            </Action>
        </Operation>
    </OperationCollection>
    <AssistanceCollection/>
    <ParticipatorCollection>
        <Dictionary ItemKey="Operator" ItemID="10023"/>
    </ParticipatorCollection>
    <NodeGraphic Height="48" Width="88" X="530" Y="180"/>
</UserTask>
<Join Key="Join" Caption="合并" ID="203">
    <TransitionCollection>
        <SequenceFlow ID="108" TargetNodeKey="End">
            <TransitionGraphic LineStyle="StraightLine"/>
        </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic Height="48" Width="88" X="660" Y="150"/>
</Join>
<End Caption="结束节点" ID="8" Key="End">
    <TransitionCollection/>
    <NodeGraphic X="740" Y="211"/>
</End>
</Process>

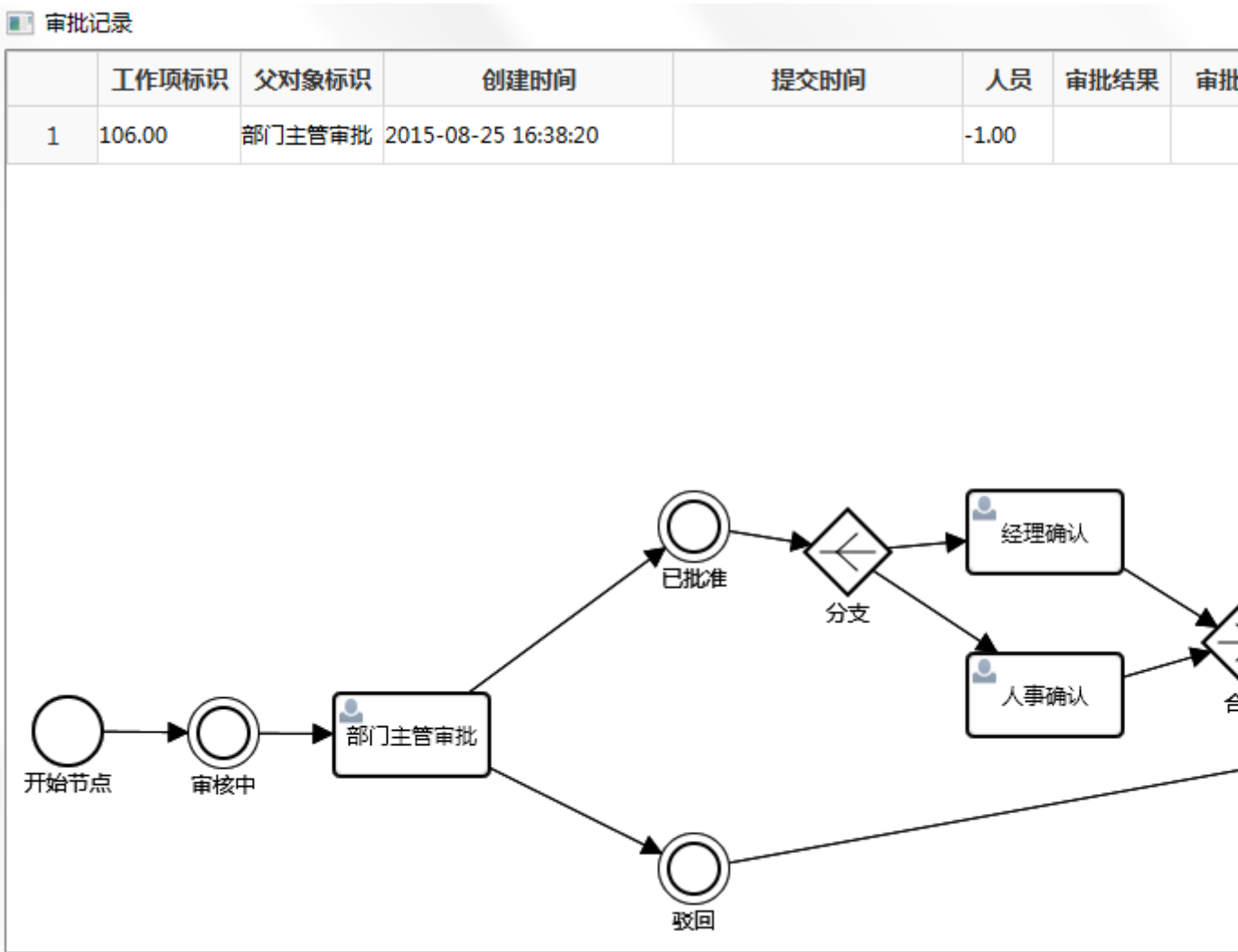
```

界面展示:

步骤一：销售部员工唐山海新建一张请假单，按照请假流程启动，流转 to 部门主管张若昀名下，然后分支到人事萧雅和总经理郑京浩名下。

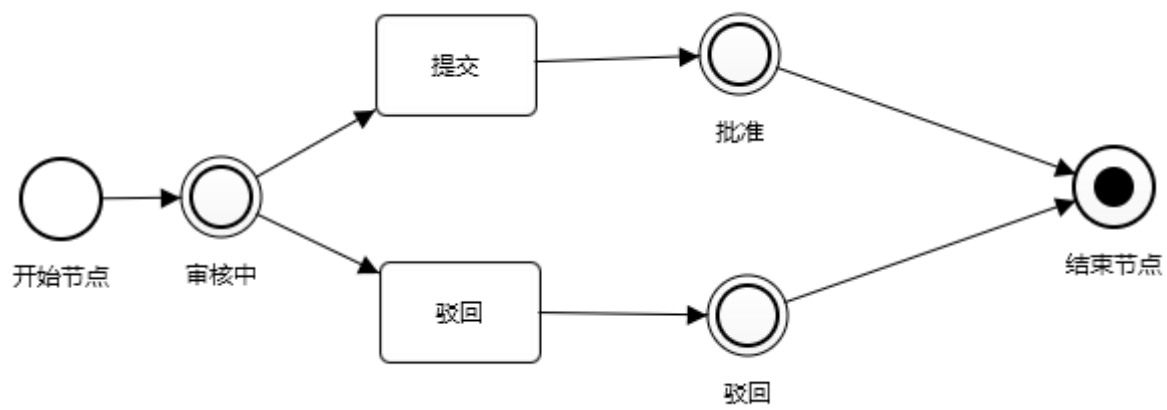


步骤二：总经理郑京浩查看审批记录，判断需要流程重启，本单据要重新提交，故点击流程重启按钮。则审批记录如下。



14. 12: 状态机操作

状态机工作流由一组状态组成。每个状态都可以接收一组特定事件。视事件而定，可以转换到另一个状态。StateAction结点不仅具有结点的基本属性还具有Operation属性。



```
<StateAction>  
  <Operation Key="" Caption="" Enable="" Visible="" Icon="">  
    <Action>
```

```

        <![CDATA[ ]]
    </Action>
</Operation>
</StateAction>

```

实例分析：

以请假流程为例，下面为状态机模式的请假流程，详细代码参照文件夹BPM_C2.12。

```

<Process Caption="请假" Key="VacationRequestAudit">
  <Begin Caption="开始节点" ID="1" Key="Begin">
    <TransitionCollection>
      <SequenceFlow ID="2" Key="BeginToState1" TargetNodeKey="State1"/>
    </TransitionCollection>
    <NodeGraphic X="14" Y="204"/>
  </Begin>
  <State Caption="审核中" ID="3" Key="State1" Status="Auditing" UseStateTask="true">
    <TransitionCollection>
      <SequenceFlow ID="4" TargetNodeKey="StateAction1"/>
      <SequenceFlow ID="5" TargetNodeKey="StateAction2"/>
    </TransitionCollection>
    <NodeGraphic X="94" Y="203"/>
    <ParticipatorCollection>
      <Dictionary ItemID="10504" ItemKey="Operator"/>
    </ParticipatorCollection>
    <Perm/>
  </State>
  <StateAction Caption="提交" ID="6" Key="StateAction1">
    <TransitionCollection>
      <SequenceFlow ID="7" Key="StateAction1_State2" TargetNodeKey="State2"/>
    </TransitionCollection>
    <NodeGraphic X="191" Y="132"/>
    <ParticipatorCollection/>
    <Operation Caption="提交" Key="op1">
      <Action><![CDATA[
        CommitWorkitem(-1,1,"");Close();
      ]]></Action>
    </Operation>
  </StateAction>
  <State Caption="批准" ID="8" Key="State2" Status="Audited">
    <TransitionCollection>
      <SequenceFlow ID="9" TargetNodeKey="End"/>
    </TransitionCollection>
    <NodeGraphic Height="34" Width="48" X="353" Y="132"/>
    <ParticipatorCollection/>
    <Perm/>
  </State>
  <StateAction Caption="驳回" ID="10" Key="StateAction2">
    <TransitionCollection>
      <SequenceFlow ID="11" Key="StateAction2_State3" TargetNodeKey="State3"/>
    </TransitionCollection>
    <NodeGraphic X="193" Y="255"/>
    <ParticipatorCollection/>
    <Operation Caption="否决" Key="op2">
      <Action><![CDATA[
        CommitWorkitem(-1,0,"");Close();
      ]]></Action>
    </Operation>
  </StateAction>
  <State Caption="驳回" ID="12" Key="State3" Status="Denied">
    <TransitionCollection>
      <SequenceFlow ID="13" TargetNodeKey="End"/>
    </TransitionCollection>
    <NodeGraphic Width="78" X="357" Y="262"/>
    <ParticipatorCollection/>
    <Perm/>
  </State>
  <End Caption="结束节点" ID="14" Key="End">
    <NodeGraphic X="540" Y="198"/>
  </End>
  <PermCollection>
    <Perm>
      <OptPerm>
        <OptPermItem Key="ShowAuditDetail"/>
      </OptPerm>
    </Perm>
  </PermCollection>
</Process>

```



注意

StateAction节点前一定是State节点，且该State节点是带有参与者ParticipatorCollection的。说明该状态节点将产生状态机专用工作项，State此处没有Operation，他需要的工作项是从后续的StateAction节点中获取的。

详细代码参照文件夹BPM_C2.12。

第 15 章 自由流程

目录

15.1. 背景	191
15.2. 案例分析	191

随着信息化建设的发展，对于工作办理审批、工作流转、异地协同办公各有实现方式。

举一个简单的例子，一位员工申请一个报销审批流程，经过部门经理审批后，转到财务部门审批，接着转到副总经理审批，这是企业规范的办理情况，后面有可能会根据不同领导的心理临时决定是否需要其他人员协助办理工作，这个就体现了工作流的灵活性、随意性。当领导需要其他部门经理为其提供决策的意见和建议（由领导临时决定需要哪些人员），或者经过领导审批以后临时选择另外领导查看审批信息等。这就体现了工作流办理的灵活性，体现了工作流的精髓。

15.1: 背景

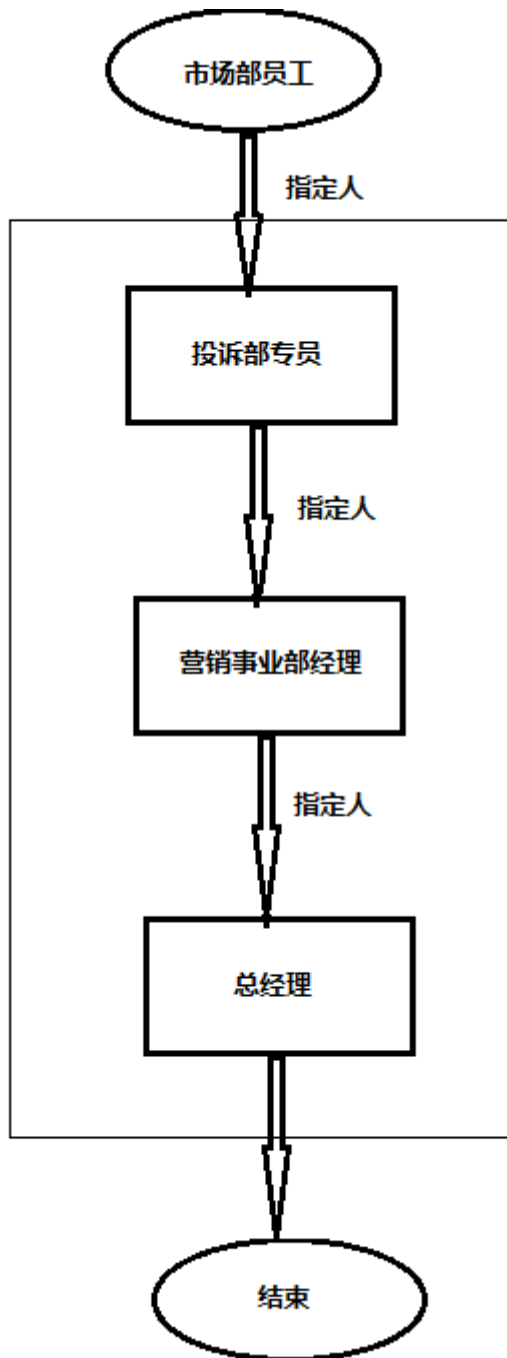
自由流程是审批人可以自行设置审批流转步骤，由上一步骤指定下一步骤的审批者。例如某个事件处理相关临时性审批时候，在不确定审批人的情况下，申请人可以根据业务需求通过自由流程选择下一步的审批者。

一般情况下，管理规范的规范性企业流程上都是相对固定的。换言之，申请人在审批流程流转过程中，无权更改审批流程。但是对于一些需要办理临时性特殊业务的企业来说，固定流程不能够满足他们的审批需要，所以就有了自由流程这一解决方案。

15.2: 案例分析

业务场景：公司的市场部人员接到一份投诉处理单，投诉对象可能是因为产品问题，也可能由于服务态度问题。种种情况下，我们需要根据不同的问题来选择不一样的审批处理人员。如果是产品问题，我们需要向质量部提交申请。如果是态度问题，我们应该向服务部门提交申请。由于事件是突发的、临时的。需要根据投诉问题不同性质来选择下一位审批人去处理审批，在这种情况下，我们选择自由流程来完成这一操作。

操作举例：当关于产品质量的投诉，从市场部人员发起申请开始，根据时间需求选择下一位审批对象，这里指定为投诉部门专员审批，审批完后，投诉部门专员可以对于下一步审批人进行审批，这里指定为营销事业部经理，审批完后，营销事业部经理开始对下一步审批人进行指定，这里指定为总经理，审批完毕后，总经理决定下一步不要指定其他人了，此时结束，那么整个流程就结束。如下图：



自由流程指定下一位审批人的表单界面：

指定操作员	<input type="text"/>		起始日期	<input type="text"/>
结束日期	<input type="text"/>	<input type="button" value="日历"/>	<input type="checkbox"/> 始终有效	
				<input type="button" value="确定"/>

界面配置如下：

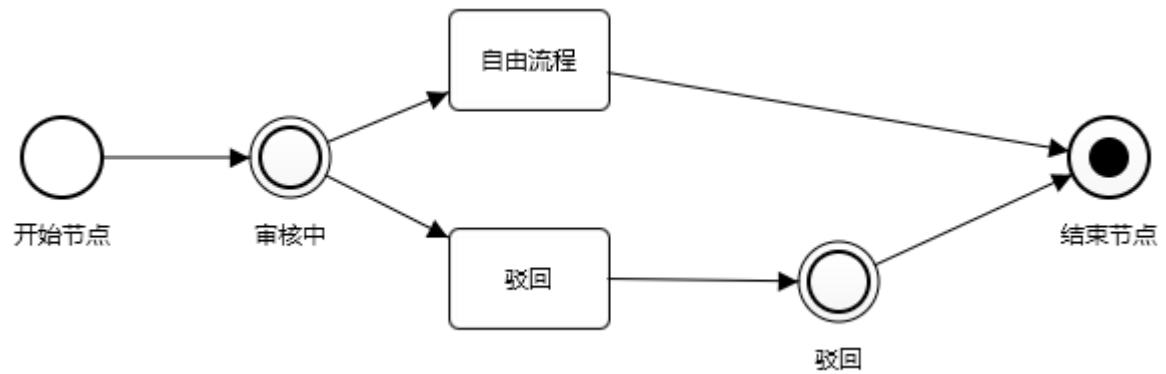
```

<Process Caption="投诉" Key="ComplainAudit">
  <Begin Caption="开始节点" ID="1" Key="Begin">
    <TransitionCollection>
      <SequenceFlow ID="11" Key="BeginToState1" TargetNodeKey="State1"/>
    </TransitionCollection>
    <NodeGraphic X="14" Y="204"/>
  </Begin>
  <State Caption="审核中" ID="2" Key="State1" Status="Auditing" UseStateTask="true">
    <TransitionCollection>
      <SequenceFlow ID="12" TargetNodeKey="StateAction1"/>
      <SequenceFlow ID="13" TargetNodeKey="StateAction2"/>
    </TransitionCollection>
    <NodeGraphic X="128" Y="204"/>
    <ParticipatorCollection>
      <Dictionary ItemID="10302" ItemKey="Operator"/>
    </ParticipatorCollection>
    <Perm/>
  </State>
  <StateAction Caption="自由流程" ID="3" Key="StateAction1">
    <TransitionCollection>
      <SequenceFlow ID="14" Key="StateAction1_End" TargetNodeKey="End"/>
    </TransitionCollection>
    <NodeGraphic X="227" Y="150"/>
    <ParticipatorCollection/>
    <Operation Caption="自由流程" Key="op1">
      <Action><![CDATA[
        ShowModal("AddDelegate")
      ]]></Action>
    </Operation>
  </StateAction>
  <StateAction Caption="驳回" ID="4" Key="StateAction2">
    <TransitionCollection>
      <SequenceFlow ID="15" Key="StateAction2_State3" TargetNodeKey="State3"/>
    </TransitionCollection>
    <NodeGraphic X="227" Y="259"/>
    <ParticipatorCollection/>
    <Operation Caption="否决" Key="op2">
      <Action><![CDATA[
        CommitWorkitem(-1,0,"");Close();
      ]]></Action>
    </Operation>
  </StateAction>
  <State Caption="驳回" ID="5" Key="State3" Status="Denied">
    <TransitionCollection>
      <SequenceFlow ID="16" TargetNodeKey="End"/>
    </TransitionCollection>
    <NodeGraphic Width="78" X="402" Y="266"/>
    <ParticipatorCollection/>
    <Perm/>
  </State>
  <End Caption="结束节点" ID="7" Key="End">
    <NodeGraphic X="537" Y="204"/>
  </End>
</PermCollection>

```

```
<Perm>
  <OptPerm>
    <OptPermItem Key="ShowAuditDetail"/>
  </OptPerm>
</Perm>
</PermCollection>
</Process>
```

流程图如下：



总结：自由流程的优势在于中间部分可以在后台自定义，针对一些小的、临时的事件处理的审批流程尤为适用，方便。如下图：



注意

本章所讲述的自由流程涉及到二次开发内容，具体案例在后续专题教程里将会详解分析。

16. 2. 2: 参与者实例

公司A，有采购部门、销售部门、仓存部门三个部门，每个部门都有各自的部门经理，另还有人事和总经理直属于公司A。

16. 2. 2. 1: 主要信息：

涉及表单：

请假单列表[VacationRequestView]：此表单为序时簿。罗列请假单列表集合，是打开请假单、新建请假单的入口；

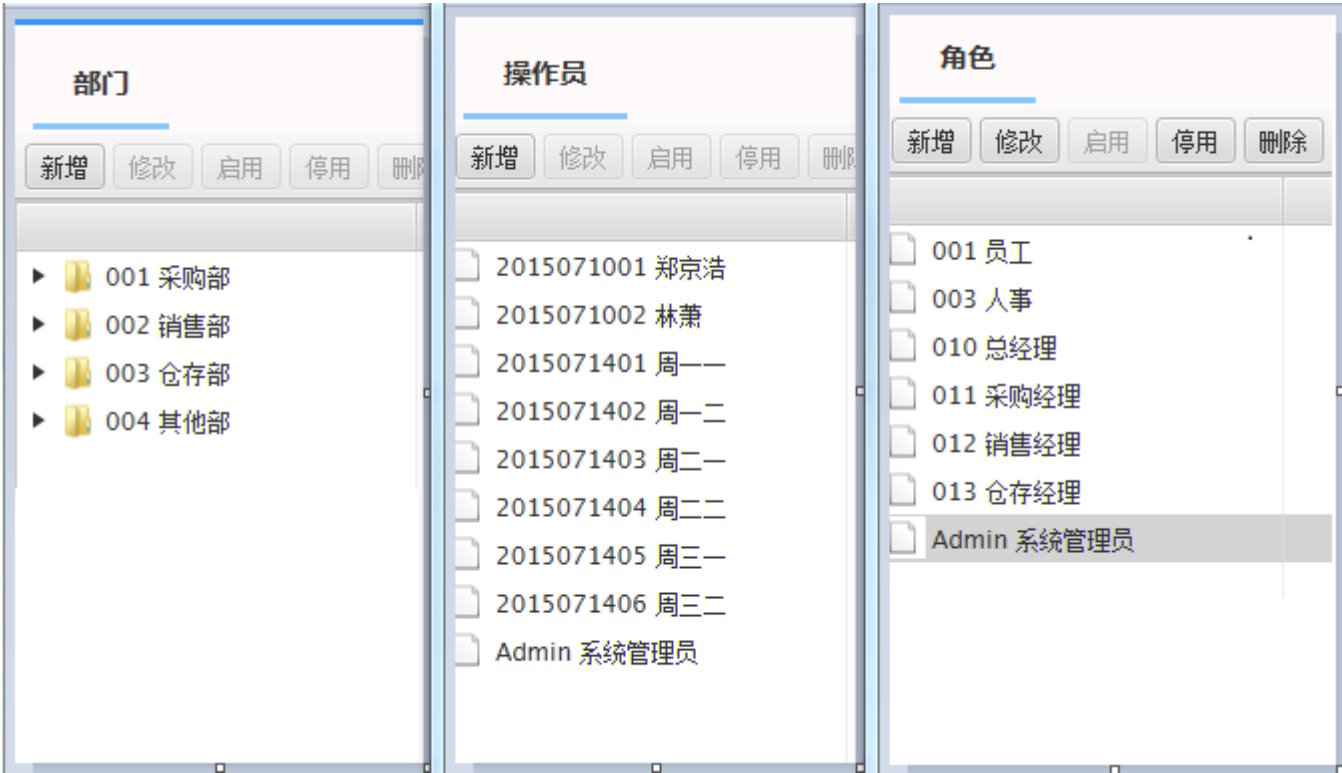
请假单[VacationRequest]：此表单为单据。新建、保存请假信息；

其他表单：操作员[Operator] (字典)、角色[Role] (字典)、部门[Department] (字典)、

16. 2. 2. 2: 前期准备：

第一步：新建三个字典，Key和Caption分别为：Operator（操作员）、Role（角色）、Department（部门）。并输入数据。（此处不做字典创建的具体介绍，详情请参考字典介绍）

如图：



第二步：三个表中对应的字段分别是

操作员： 角色： 部门：

DEV007-PC\SQLXP... dbo.SYS_Operator		列名	数据类型	列名	数据类型
列名	数据类型	列名	数据类型	列名	数据类型
OID	bigint	OID	bigint	OID	bigint
POID	bigint	POID	bigint	POID	bigint
SOID	bigint	SOID	bigint	SOID	bigint
VERID	int	VERID	int	VERID	int
DVERID	int	DVERID	int	DVERID	int
Status	int	Status	int	Status	int
Enable	int	Enable	int	Enable	int
NodeType	int	NodeType	int	NodeType	int
ParentID	bigint	ParentID	bigint	ParentID	bigint
TLeft	int	TLeft	int	TLeft	int
TRight	int	TRight	int	TRight	int
Code	varchar(50)	Code	varchar(50)	Code	varchar(50)
Name	varchar(50)	Name	varchar(50)	Name	varchar(50)
Password	varchar(512)	IsAdmin	int	SLOCK	int
SLOCK	int	SLOCK	int	MgrID	bigint

第三步：新建请假单（VacationRequest），FormType为Entity。此表中分别以下字段：

注意：每个数据对象的表中是都必须含有OID[对象标识]、POID[父对象标识]、SOID[主对象标识]、VERID[对象版本]、DVERID[对象明细版本]五个字段

字段名	含义
IDNumber	工号
Name	姓名
ToTime	开始时间
ToTime	结束时间
Reason	事由
Status	状态
Type	类型
DayCount	总计

16.2.2.3: 流程配置

新建请假流程审批（VacationRequestAudit1）

```
<Process Caption="请假" Key="VacationRequestAudit">
  <Begin Caption="开始节点" ID="1" Key="Begin">
    <TransitionCollection>
      <SequenceFlow Condition="" ID="9" Key="BegintoState" TargetNodeKey="State">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic X="14" Y="204"/>
  </Begin>
  <State Caption="审核中" ID="2" Key="State" Status="Auditing">
    <TransitionCollection>
      <SequenceFlow Condition="" ID="10" TargetNodeKey="UserTask3">
        <TransitionGraphic LineStyle="StraightLine"/>
      </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic X="100" Y="205"/>
  </State>
  <Audit Caption="部门主管审批" Key="UserTask3" ID="105">
    <TransitionCollection>
      <SequenceFlow Condition="True" ID="108" TargetNodeKey="State1">

```

```

        <TransitionGraphic LineStyle="StraightLine"/>
    </SequenceFlow>
    <SequenceFlow Condition="False" ID="109" TargetNodeKey="State2">
        <TransitionGraphic LineStyle="StraightLine"/>
    </SequenceFlow>
</TransitionCollection>
<OperationCollection>
    <Operation Caption="审批" Key="opl">
        <Action>
            <![CDATA[ShowModal("AuditDialog")]]>
        </Action>
    </Operation>
</OperationCollection>
<AssistanceCollection/>
<ParticipatorCollection>
    <Query>
        <![CDATA[ select MgrID from dbo.dp_Department where OID in (SELECT DeptID from SYS_Operator where OID=?)]]>
        <QueryParameterCollection>
            <QueryParameter Formula="GetOperator()" DataType="Long"/>
        </QueryParameterCollection>
    </Query>
</ParticipatorCollection>
<NodeGraphic Height="48" Width="88" X="180" Y="202"/>
</Audit>
<State Caption="已批准" Key="State1" ID="104" Status="Audited">
    <TransitionCollection>
        <SequenceFlow Condition="" ID="21" TargetNodeKey="Fork">
            <TransitionGraphic LineStyle="StraightLine"/>
        </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic Height="34" Width="48" X="360" Y="91"/>
</State>
<State Caption="驳回" Key="State2" ID="106" Status="Denied">
    <TransitionCollection>
        <SequenceFlow Condition="" ID="23" TargetNodeKey="End">
            <TransitionGraphic LineStyle="StraightLine"/>
        </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic Width="78" X="360" Y="280"/>
</State>
<Fork Key="Fork" Caption="分支" ID="202">
    <TransitionCollection>
        <SequenceFlow TargetNodeKey="ManagerConfirm">
            <TransitionGraphic LineStyle="StraightLine"/>
        </SequenceFlow>
        <SequenceFlow TargetNodeKey="HRConfirm">
            <TransitionGraphic LineStyle="StraightLine"/>
        </SequenceFlow>
    </TransitionCollection>
    <NodeGraphic Height="48" Width="88" X="440" Y="100"/>
</Fork>
<UserTask Key="ManagerConfirm" Caption="经理确认" ID="201">
    <TransitionCollection>
        <SequenceFlow ID="108" TargetNodeKey="Join">
            <TransitionGraphic LineStyle="StraightLine"/>
        </SequenceFlow>
    </TransitionCollection>
    <OperationCollection>
        <Operation Caption="确认" Key="opl">
            <Action>
                <![CDATA[CommitWorkitem(-1, 1, "已确认");]]>
            </Action>
        </Operation>
    </OperationCollection>
    <AssistanceCollection/>
    <ParticipatorCollection>
        <Dictionary ItemKey="Operator" ItemID="10307"/> //select oid from sys_operator where soid in (select SOID from
SYS_OperatorRole where role=10903)
    </ParticipatorCollection>
    <NodeGraphic Height="48" Width="88" X="530" Y="90"/>
</UserTask>
<UserTask Key="HRConfirm" Caption="人事确认" ID="200">
    <TransitionCollection>
        <SequenceFlow ID="108" TargetNodeKey="Join">
            <TransitionGraphic LineStyle="StraightLine"/>
        </SequenceFlow>
    </TransitionCollection>
    <OperationCollection>
        <Operation Caption="确认" Key="opl">
            <Action>
                <![CDATA[CommitWorkitem(-1, 1, "已确认");]]>
            </Action>
        </Operation>
    </OperationCollection>

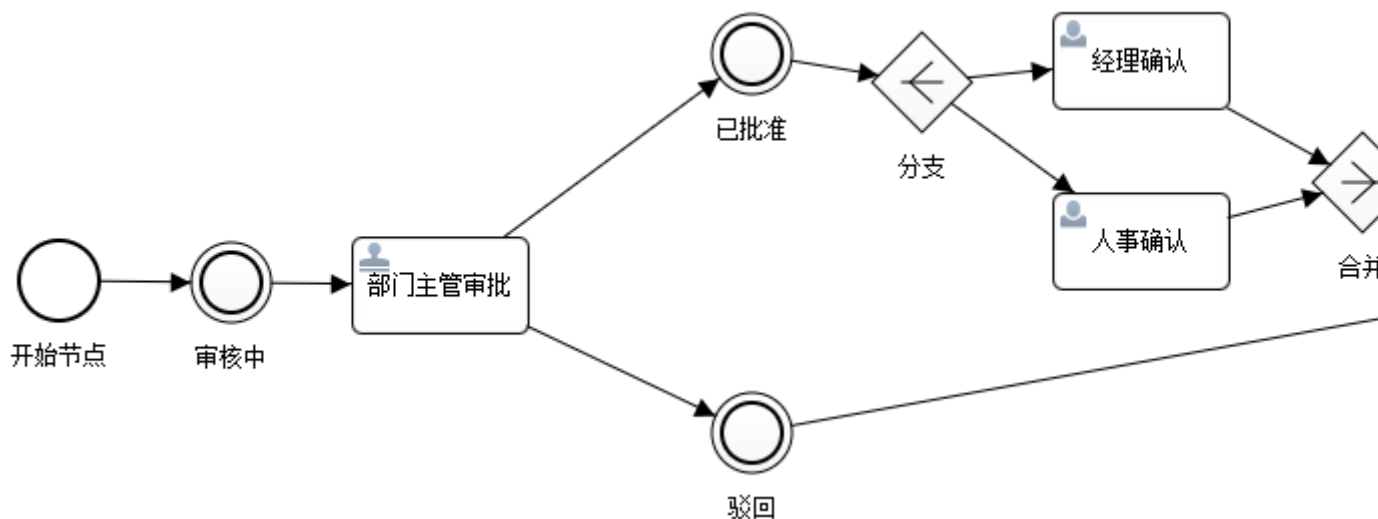
```

```

</OperationCollection>
<AssistanceCollection/>
<ParticipatorCollection>
  <Dictionary ItemKey="Operator" ItemID="10310"/><<select oid from sys_operator where soid in (select SOID from
SYS_OperatorRole where role=10305)
</ParticipatorCollection>
  <NodeGraphic Height="48" Width="88" X="530" Y="180"/>
</UserTask>
<Join Key="Join" Caption="合并" ID="203">
  <TransitionCollection>
    <SequenceFlow ID="108" TargetNodeKey="End">
      <TransitionGraphic LineStyle="StraightLine"/>
    </SequenceFlow>
  </TransitionCollection>
  <NodeGraphic Height="48" Width="88" X="660" Y="150"/>
</Join>
<End Caption="结束节点" ID="8" Key="End">
  <TransitionCollection/>
  <NodeGraphic X="740" Y="211"/>
</End>
<PermCollection>
  <Perm>
    <OptPerm>
      <OptPermItem Key="ShowAuditDetil"/>
    </OptPerm>
  </Perm>
</PermCollection>
</Process>

```

该流程显示流程图如下：



配置分析：

部门主管审批任务的参与者：定义参与者来源于查询结果集“部门主管”需要从当前操作者的OID开始找出该人所属的部门，根据部门信息找到部门主管的OID。

```

<ParticipatorCollection>
  <Query>
    <![CDATA[ select MgrID from dbo.dp_Department where OID in (SELECT DeptID from SYS_Operator where OID=?)]]>
    <QueryParameterCollection>
      <QueryParameter Formula="GetOperator()" DataType="Long"/>
    </QueryParameterCollection>
  </Query>
</ParticipatorCollection>

```

经理确认任务的参与者：定义参与者来源于字典项的计算值 该任务的参与者只有一人，所以来源于字典项和查询结果集皆可以，本流程参与者来源于字典标识为10307的操作员（角色标识为10903）。

```

<ParticipatorCollection>
  <Dictionary ItemKey="Operator" ItemID="10307"/> //select oid from sys_operator where soid in (select SOID from
SYS_OperatorRole where role=10903)

```

```
</ParticipatorCollection>
```

人事确认任务的参与者：定义参与者来源于字典项的计算值该任务参照经理确认任务。

```
<ParticipatorCollection>
  <Dictionary ItemKey="Operator" ItemID="10310"/> //select oid from sys_operator where soid in (select SOID from
  SYS_OperatorRole where role=10305)
</ParticipatorCollection>
```

16.2.2.4: 请假流程效果展示

步骤一：填写“请假单”流程启动。当前操作员为销售部门员工周二二，该部门经理为周二一。

请假单列表								
新增								
对象...	父对...	对象...	对象...	工号	姓名	开始时间	结束时间	事由
12401			0	2015071404	周二二	2015-07-21 09:19:40	2015-07-22 09:19:43	

步骤二：登录部门经理的账号，查看待办事宜下的请假单。

我的待办事宜		
工作项标识	工作项名称	创建时间
901	部门主管审批	2015-07-20 09:19:57

步骤三：照上步骤，操作审批同意，按照流程流转至人事，经理，只有在人事和经理都确认后，请假流程才算结束。同BPM_C2.

我的待办事宜

请假单

审批

审批记录

工号

姓名

开始时间

结束时间

事由

类型

总计

状态

有效性

审批对话框

意见

同意!

通过

驳回

取消

第 17 章 任务的操作实现

目录

17.1. 任务转移	202
17.2. 任务的撤消(回滚模式)	206

17.1: 任务转移

转移关系对象是由关系定义描述的，每一个转移(Transition)关系由起始活动、到达活动、转移条件、发生转移的状态信息构成。描述任务执行之间的各种关系(串行、并行、选择执行等)。这里我们分任务的代理和授权两部分先来举例说明。

17.1.1: 任务的代理

代理是指将代理人拥有的对某个任务或整个过程的处理指定给某些选定的操作者，代理人本人不再处理这些任务，代理具有以下特征：

- 代理具有时效性，代理需要指定代理的有效期间，在有效期内，被代理的任务被指定给被代理人，超出时效后，新生成任务仍然由代理人处理；可以指定无限期的时效。
- 代理的范围，代理需要指定被代理任务的范围，包括几个不同的级别：所有代理人任务、某个过程的任务、某个表单、指定的任务；
- 代理可以被撤销，撤销后的新任务由代理人处理；对已经生成的被代理人任务的处理由下面的规则说明。

代理任务的收回

- 如果代理指定了自动收回已分配任务，那么在代理失效后立刻收回被代理任务返回给代理人；
- 代理人可以手工收回被代理任务。

17.1.2: 任务的授权

授权是指授权人将拥有的任务的处理授权给某些选定的操作者，代理人本人可以继续处理这些任务，授权具有以下特征：

- 授权具有时效性，授权需要指定授权的有效期间，在有效期内，被授权的任务同时也会分配至被授权人，超出时效后，新生成的任务不再分配至被授权人；可以指定无限期的时效；
- 授权的范围，授权需要指定被授权任务的范围，包括几个不同的级别：所有授权人任务、某个过程的任务、指定的任务；
- 授权可以被撤销，撤销后的新任务不再发给被授权人；对已经分配给被授权人的任务的处理由下面的规则说明。

授权任务的收回

- 如果授权指定了自动收回已分配任务，那么在授权失效后立刻收回被授权人对任务的处理；
- 授权人可以手工收回被授权任务。

授权，访问控制，是管理资源访问的过程，换言之，也是控制一个任务中谁有权利访问什么。授权的例子：是否允许操作员查看这个页面、编辑数据、看到按钮、或者从这台打印机打印？这些决定是一个操作员可以访问什么的决断。权限只描述行为（和资源相关的动作），并不关心谁有能力执行这个

动作。定义谁（操作员）被允许做什么（权限）需要将权限赋给操作员，这通常取决于数据模型而且经常改变。

17.1.3: 任务转移实例

假定A公司销售部门主管周一由于休假一段时间决定将请假流程中部门主管审批的任务代理给该部门员工林一一。详见SRC配置文件。

代理授权界面源代码

```
<Form Caption="我的代理授权" FormType="View" Key="Delegate">
  <DataSource>
    <DataObject Caption="我的代理授权" Key="Delegate">
      <TableCollection>
        <Table Caption="代理授权" DBTableName="WF_Delegate" Key="Delegate" Persist="false" SourceType="Query"
TableMode="Detail">
          <Statement>
            <![CDATA[select DelegateID, DelegateType, TgtOperatorID, ObjectType, ObjectKey, NodeKey, StartTime, EndTime, AlwaysValid, '删除'
as[delete] from WF_Delegate where SrcOperatorID=?]]>
          </Statement>
          <Column Caption="代理编号" DataType="Long" Key="DelegateID"/>
          <Column Caption="代理类型" DataType="Integer" Key="DelegateType"/>
          <Column Caption="目标操作员" DataType="Long" Key="TgtOperatorID"/>
          <Column Caption="代理类型" DataType="Integer" Key="ObjectType"/>
          <Column Caption="标识" DataType="Varchar" Key="ObjectKey" Length="255"/>
          <Column Caption="节点" DataType="Varchar" Key="NodeKey" Length="255"/>
          <Column Caption="起始时间" DataType="Date" Key="StartTime"/>
          <Column Caption="结束时间" DataType="Date" Key="EndTime"/>
          <Column Caption="永久有效" DataType="Boolean" Key="AlwaysValid"/>
          <Column Caption="删除" DataType="Varchar" Key="delete"/>
          <ParameterCollection>
            <Parameter Formula="GetOperator()" />
          </ParameterCollection>
        </Table>
      </TableCollection>
    </DataObject>
  </DataSource>
  <OperationCollection>
    <Operation Key="Save" Caption="添加代理信息" Visible="!ReadOnly()">
      <Action>
        <![CDATA[ShowModal('AddDelegate');]]>
      </Action>
    </Operation>
  </OperationCollection>
  <OnLoad type="Formula">
    <![CDATA[LoadData();]]>
  </OnLoad>
  <Body>
    <Block>
      <FlowLayoutPanel Key="main">
        <ToolBar Height="pref" IsDefault="true" Key="main_toolbar"/>
        <ListView Height="100%" Key="list" TableKey="Delegate">
          <ListViewColumnCollection>
            <ListViewColumn Caption="代理编号" ColumnType="Label" DataColumnKey="DelegateID" Key="DelegateID"
Width="100px"/>
            <ListViewColumn Caption="代理类型" ColumnType="ComboBox" DataColumnKey="DelegateType"
Key="delegateType">
              <Item Caption="授权" Key="AUTHORIZE" Value="1"/>
              <Item Caption="代理" Key="DELEGATE" Value="2"/>
            </ListViewColumn>
            <ListViewColumn Caption="对象类型" ColumnType="ComboBox" DataColumnKey="ObjectType" Key="ObjectType">
              <Item Caption="操作员" Key="OPERATOR" Value="1"/>
              <Item Caption="流程" Key="PROCESS" Value="2"/>
              <Item Caption="流程节点" Key="PROCESS_NODE" Value="3"/>
              <Item Caption="表单" Key="OBJECT" Value="4"/>
            </ListViewColumn>
            <ListViewColumn Caption="目标操作员" ColumnType="Dict" DataColumnKey="TgtOperatorID"
ItemKey="Operator" Key="TgtOperatorID" Width="100px"/>
            <ListViewColumn Caption="标识" ColumnType="Label" DataColumnKey="ObjectKey" Key="ObjectKey"
Width="100px"/>
            <ListViewColumn Caption="节点" ColumnType="Label" DataColumnKey="NodeKey" Key="NodeKey" Width="350px"/>
          </ListViewColumnCollection>
          <ListViewColumn Caption="起始时间" ColumnType="DatePicker" DataColumnKey="StartTime" Key="StartTime"
Width="350px"/>
          <ListViewColumn Caption="结束时间" ColumnType="DatePicker" DataColumnKey="EndTime" Key="EndTime"
Width="350px"/>
          <ListViewColumn Caption="永久有效" ColumnType="CheckBox" DataColumnKey="AlwaysValid" Enable="false"
Key="AlwaysValid" Width="100px"/>
        </ListView>
      </FlowLayoutPanel>
    </Block>
  </Body>
</Form>
```



```
Key="delete">
    <ListViewColumn Caption="删除" ColumnType="Button" DataColumnKey="delete" DefaultValue="删除"
    <OnClick RunType="Client"><![CDATA[DeleteDelegateData(DelegateID);loadData();refreshControl('list');]]></OnClick>
    </ListViewColumn>
</ListViewColumnCollection>
</ListView>
</FlexFlowLayoutPanel>
</Block>
</Body>
</Form>
```

界面如下图所示:

我的代理授权

添加代理信息

代理编号	代理类型	对象类型	目标操作员	标识	节点
1	授权	操作员	001 test01	AAA	
3	代理	流程	002 test02	某流程的KEY	

代理授权信息的添加选择对话框源代码:

```
<Form Caption="代理添加对话框" FormType="Normal" Key="AddDelegate">
    <DataSource/>
    <Body PopHeight="300px" PopWidth="800px">
        <Block>
            <GridLayoutPanel Caption="GridLayoutPanel0" Key="GridLayoutPanel0">
                <Button Caption="取消" Key="Button1" X="8" Y="6">
                    <OnClick RunType="Client"><![CDATA[close();]]></OnClick>
                </Button>
                <CheckBox Caption="始终有效" Key="alwaysValid" X="4" XSpan="5" Y="4"/>
                <Button Caption="确定" Key="Button0" X="6" Y="6">
                    <OnClick RunType="Client"><![CDATA[AddDelegateData(delegateType, GetOperator(), tgtOperatorID, objectType,
                    objectKey, nodeKey, startTime, alwaysValid, endTime);parent.loadData();parent.RefreshControl('list');close();]]></OnClick>
                </Button>
                <Label Caption="结束日期" Key="Label5" X="1" Y="4"/>
                <DatePicker Caption="endTime" Key="endTime" X="2" Y="4"/>
                <DatePicker Caption="startTime" Key="startTime" X="5" XSpan="4" Y="3"/>
                <Label Caption="起始日期" Key="Label4" X="4" Y="3"/>
                <Label Caption="节点" Key="Label3" X="1" Y="3"/>
                <TextEditor Caption="nodeKey" Key="nodeKey" X="2" Y="3"/>
                <Label Caption="标识" Key="Label2" X="4" Y="2"/>
                <TextEditor Caption="objectKey" Key="objectKey" X="5" XSpan="4" Y="2"/>
                <Label Caption="代理操作员" Key="Label1" X="1" Y="2"/>
                <Label Caption="对象类型" Key="Label0" X="4" Y="1"/>
                <ComboBox Caption="objectType" Editable="false" ItemsDependency="" Key="objectType" X="5" XSpan="4" Y="1">
                    <Item Caption="操作员" Key="OPERATOR" Value="1"/>
                    <Item Caption="流程" Key="PROCESS" Value="2"/>
                    <Item Caption="流程节点" Key="PROCESS_NODE" Value="3"/>
                    <Item Caption="表单" Key="OBJECT" Value="4"/>
                </ComboBox>
                <Label Caption="代理类型" Key="Label6" X="1" Y="1"/>
                <ComboBox Caption="delegateType" Editable="false" ItemsDependency="" Key="delegateType" X="2" Y="1">
                    <Item Caption="授权" Key="AUTHORIZE" Value="1"/>
                    <Item Caption="代理" Key="DELEGATE" Value="2"/>
                </ComboBox>
                <Dict Caption="tgtOperatorID" ItemKey="Operator" Key="tgtOperatorID" X="2" Y="2"/>
                <RowDefCollection RowGap="8">
                    <RowDef Height="20px"/>
                    <RowDef Height="30px"/>
                    <RowDef Height="30px"/>
                    <RowDef Height="30px"/>
                    <RowDef Height="30px"/>
                    <RowDef Height="30px"/>
                    <RowDef Height="20px"/>
                    <RowDef Height="20px"/>
                </RowDefCollection>
                <ColumnDefCollection ColumnGap="8">
                    <ColumnDef Width="20px"/>
                    <ColumnDef Width="70px"/>
```

```
<ColumnDef Width="250px"/>
<ColumnDef Width="13px"/>
<ColumnDef Width="70px"/>
<ColumnDef Width="117px"/>
<ColumnDef Width="70px"/>
<ColumnDef Width="20px"/>
<ColumnDef Width="70px"/>
<ColumnDef Width="20px"/>
</ColumnDefCollection>
</GridLayoutPanel>
</Block>
</Body>
</Form>
```

界面如下图所示：

代理添加对话框

代理类型

代理

代理操作员

001 test01

节点

结束日期

对象类型

标识

起始日期

☐ 始终有效

操作员

流程

流程节点

表单

确定

代理添加对话框

代理类型

授权

代理操作员

001 test01

节点

结束日期

对象类型

标识

起始日期

☐ 始终有效

操作员

流程

流程节点

表单

确定

补充说明：任务转移的代理和授权的对象类型有：操作员、流程、流程结点、表单。

当对象类型为操作员时，标识和结点无；

当对象类型为流程时，标识为所对应流程的KEY, 结点无；

当对象类型为流程结点时，标识为所对应流程的KEY, 结点为指定的任务的KEY；

当对象类型为表单时，标识为当前表单的KEY。

本案例中部门经理将授权操作员林一一请假流程中部门主管审批指定任务结点，有效时间为11天。下图为界面展示：

我的代理授权					
添加代理信息					
代理编号	代理类型	对象类型	目标操作员	标识	节点
501	代理	流程节点	2015070101 林一一	VacationRequ...	UserTask3

登录代理人林一一我的待办事宜操作界面，查看授权信息。

我的待办事宜		
工作项标识	工作项名称	创建时间
801	部门主管审批	2015-07-17 17:46:58

代理类型如果是代理，则源代理人将该结点操作完全给予目标代理人林一一，在11天内无法进行部门主管审批该任务；若代理类型为授权，则源代理人和目标代理人皆可操作部门主管审批的任务。

17.2: 任务的撤消(回滚模式)

RollbackToWorkitem 将流程数据回滚到制定工作项提交之前的状态，实现之前的数据回滚模式

参数：workitemID 为工作项的编号

```
//任务的撤销
RollbackToWorkitem(workItemID)
```

实例分析：请假审批记录中添加“撤销”按钮，源代码如下：

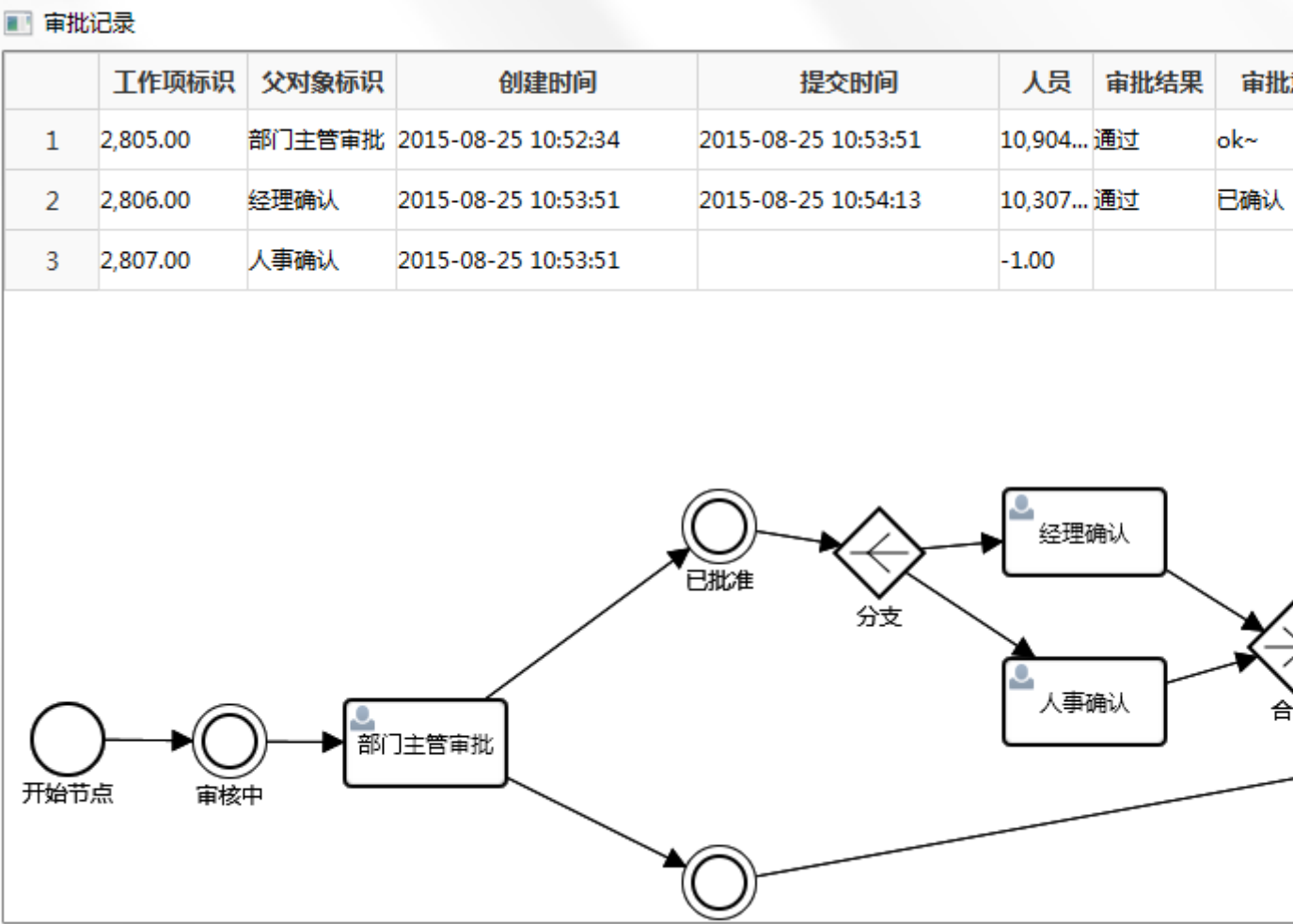
```
<Form Caption="审批记录" FormType="Normal" Key="WFLog" PreferSize="1000px,600px">
<DataSource>
  <DataObject Key="BPM_Log" NoPrefix="STIN" Caption="审批记录" MainTableKey="BPM_Log" PrimaryType="Entity"
  SecondaryType="Normal">
    <TableCollection>
      <Table Key="BPM_Log" Caption="基本信息" DBTableName="BPM_Log" TableMode="Head" SourceType="Query" Persist="True"
      Filter="workItemID in(select workitemID from BPM_WorkitemInfo where instanceID=(select instanceID from BPM_Instance where
      OID=?))">
        <ParameterCollection>
          <Parameter TargetColumn="workItemID" Formula="GetParentOID()" MidParameter="False" />
        </ParameterCollection>
        <Statement>
          <![CDATA[select workItemID,workItemName,createTime,finishTime,operatorID,auditResult,userInfo from BPM_Log]]>
        </Statement>
        <Column Key="workItemID" Caption="工作项标识" Persist="True" DataType="Long" DBColumnName="workItemID" />
        <Column Key="workItemName" Caption="父对象标识" Persist="True" DataType="Varchar" Length="200"
        DBColumnName="workItemName"/>
        <Column Key="createTime" Caption="创建时间" Persist="True" DataType="DateTime" DBColumnName="createTime"/>
        <Column Key="finishTime" Caption="提交时间" Persist="True" DataType="DateTime" DBColumnName="finishTime"/>
      </Table>
    </TableCollection>
  </DataObject>
</DataSource>
```

```

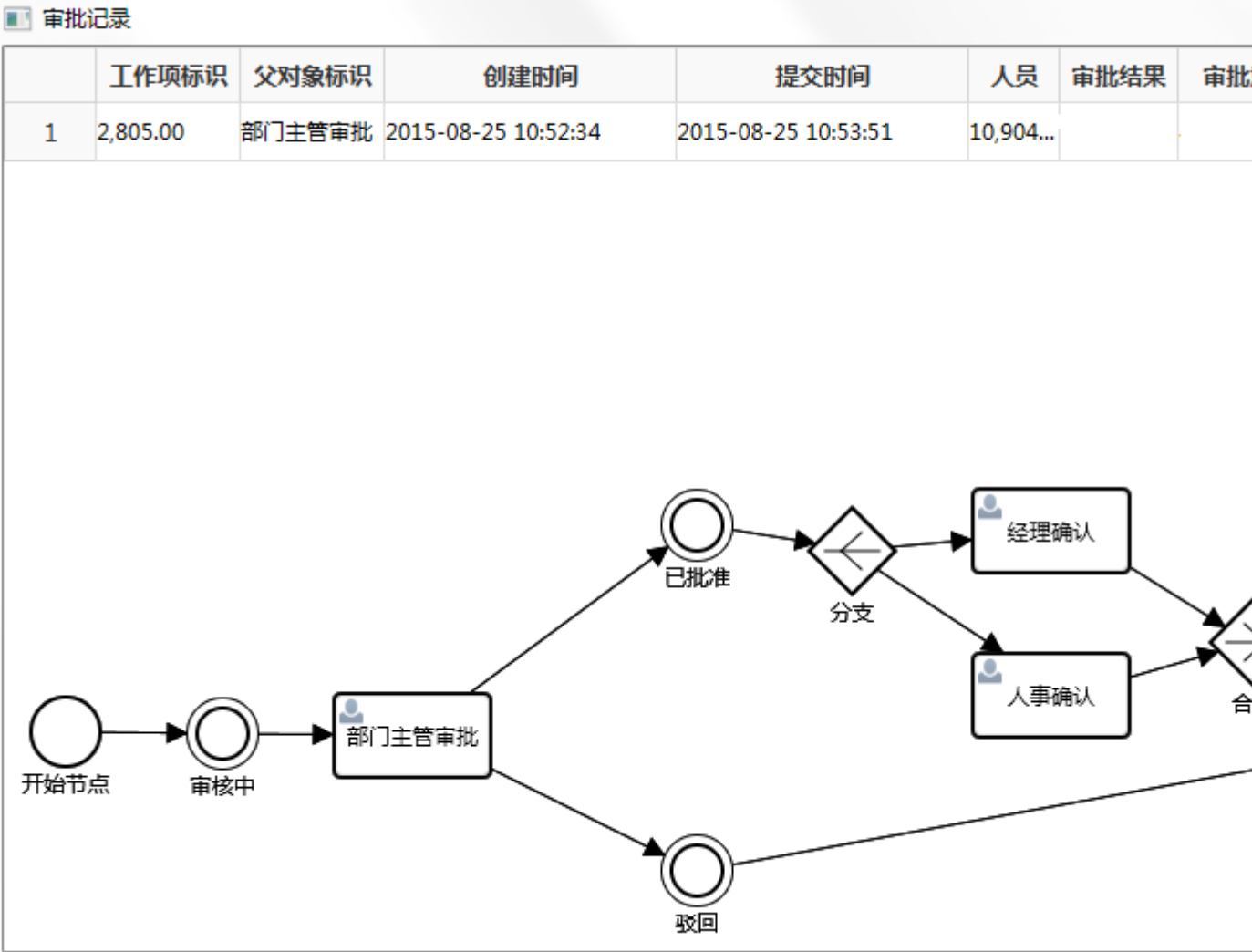
        <Column Key="operatorID" Caption="人员" Persist="True" DataType="Integer" DBColumnName="operatorID"/>
        <Column Key="userInfo" Caption="审批意见" Persist="True" DataType="Varchar" Length="200" DBColumnName="userInfo"/>
        <Column Key="auditResult" Caption="审批结果" Persist="True" DataType="Integer" DBColumnName="auditResult"/>
    </Table>
</TableCollection>
</DataObject>
</DataSource>
<OperationCollection>
</OperationCollection>
<OnLoad type="Formula">
    <![CDATA[LoadData();
]]>
</OnLoad>
<Body HAlign="Left" OverflowX="Visible" OverflowY="Visible" PopWidth="700px" PopHeight="500px">
    <Block>
        <GridLayoutPanel Key="main">
            <RowDefCollection RowHeight="30" RowGap="3">
                <RowDef Height="30"/>
                <RowDef Height="70"/>
            </RowDefCollection>
            <ColumnDefCollection>
                <ColumnDef Width="100"/>
            </ColumnDefCollection>
            <Grid Key="detail_grid" Caption="入库单明细" UsePage="false" PageRowCount="6" ShowRowHead="true" Enable="False" X="0"
XSpan="1" Y="0" YSpan="1">
                <GridColumnCollection>
                    <GridColumn Key="workItemID" Caption="工作项标识" Width="79px" ColumnExpand="false" ColumnType="Detail"/>
                    <GridColumn Key="workItemName" Caption="父对象标识" Width="79px" ColumnExpand="false" ColumnType="Detail"/>
                    <GridColumn Key="createTime" Caption="创建时间" Width="160px" ColumnExpand="false" ColumnType="Detail"/>
                    <GridColumn Key="finishTime" Caption="提交时间" Width="160px" ColumnExpand="false" ColumnType="Detail"/>
                    <GridColumn Key="operatorID" Caption="人员" Width="50px" ColumnExpand="false" ColumnType="Detail"/>
                    <GridColumn Key="auditResult" Caption="审批结果" Width="65px" ColumnExpand="false" ColumnType="Detail"/>
                    <GridColumn Key="userInfo" Caption="审批意见" Width="79px" ColumnExpand="false" ColumnType="Detail"/>
                    <GridColumn Key="Rollback" Caption="撤销" Width="79px" ColumnExpand="false" ColumnType="Detail" Enable="true"/>
                </GridColumnCollection>
                <GridRowCollection>
                    <GridRow Key="R1" RowType="Detail" TableKey="BPM_Log" GroupKey="Area">
                        <GridCell Key="workItemID" Caption="工作项标识" CellType="NumberEditor">
                            <DataBinding ColumnKey="workItemID"/>
                        </GridCell>
                        <GridCell Key="workItemName" Caption="父对象标识" CellType="LABEL">
                            <DataBinding ColumnKey="workItemName"/>
                        </GridCell>
                        <GridCell Key="createTime" Caption="创建时间" CellType="DatePicker">
                            <DataBinding ColumnKey="createTime"/>
                        </GridCell>
                        <GridCell Key="finishTime" Caption="提交时间" CellType="DatePicker">
                            <DataBinding ColumnKey="finishTime"/>
                        </GridCell>
                        <GridCell Key="operatorID" Caption="人员" CellType="NumberEditor">
                            <DataBinding ColumnKey="operatorID"/>
                        </GridCell>
                        <GridCell Key="auditResult" Caption="审批结果" CellType="ComboBox">
                            <DataBinding ColumnKey="auditResult"/>
                            <Item Caption="通过" Key="1" Value="1"/>
                            <Item Caption="驳回" Key="0" Value="0"/>
                        </GridCell>
                        <GridCell Key="userInfo" Caption="审批意见" CellType="LABEL">
                            <DataBinding ColumnKey="userInfo"/>
                        </GridCell>
                        <GridCell Key="Rollback" Caption="撤销" CellType="Button" ShowText="撤销" Enable="true">
                            <OnClick>
                                <![CDATA[RollbackToWorkitem(workItemID);]]>
                            </OnClick>
                        </GridCell>
                    </GridRow>
                </GridRowCollection>
            </Grid>
            <BPMGraph Key="graph" Caption="流程图" ProcessKey="parent.GetProcessKey()" ProcessVer="parent.GetProcessVer()" X="0"
XSpan="1" Y="1" YSpan="1">
                <BPMGraph>
            </BPMGraph>
        </GridLayoutPanel>
    </Block>
</Body>
</Form>

```

该界面显示：



撤销之后的审批记录如下：



第 18 章 打印报表基础

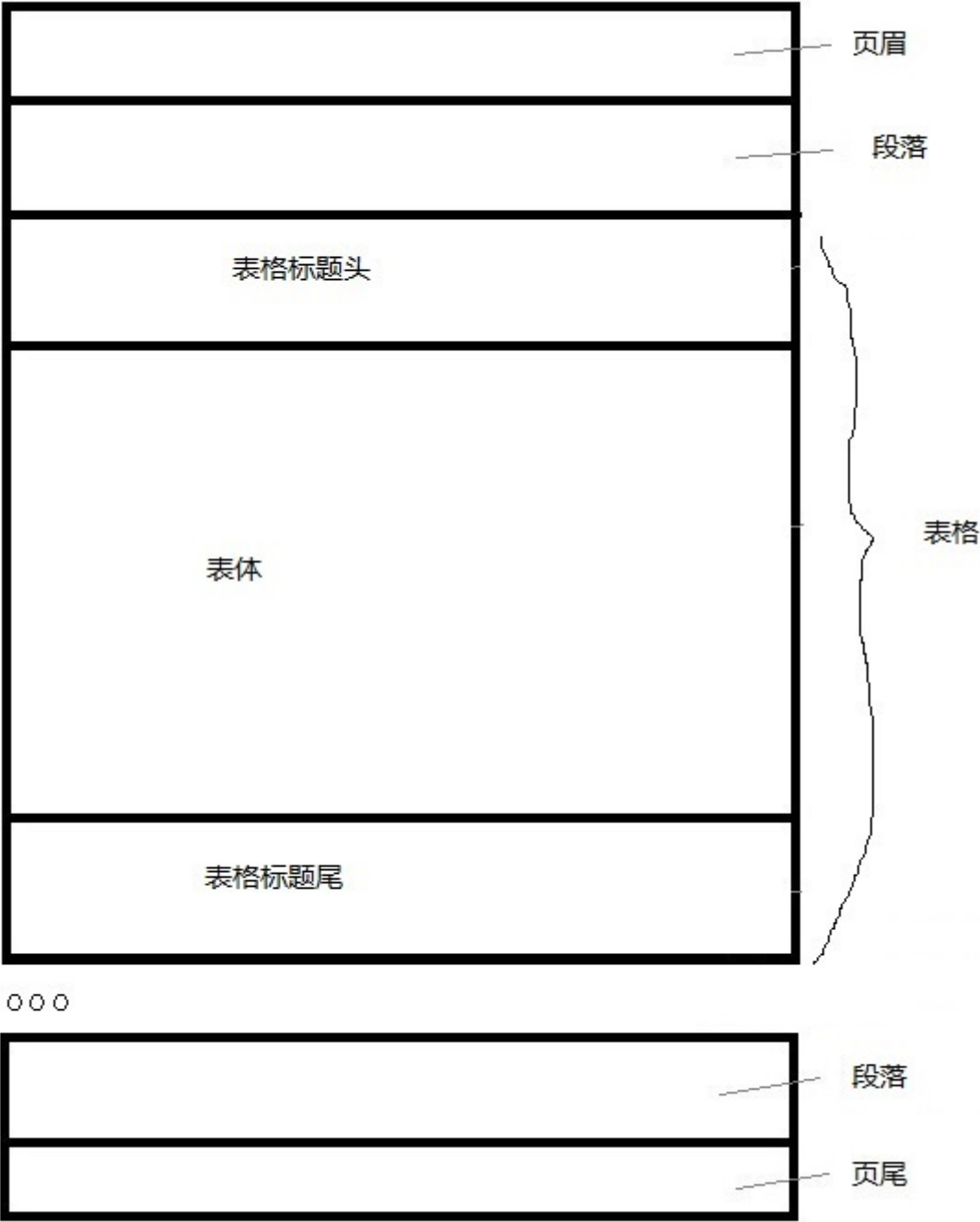
目录

18.1. 报表的结构	210
18.2. 一般明细报表	211
18.3. 分组	214
18.4. 列扩展示例-单层扩展	218
18.5. 列扩展示例-嵌套扩展	220
18.6. 列扩展示例-列扩展合计	220
18.7. 内容的溢出处理	221
18.8. 页眉和页尾	224
18.9. 自动拆行	225

打印报表为系统中的纸张输出报表，本章中的源代码见REPORT_C1。

18.1: 报表的结构

报表按照结构通常分为页眉、段落、表格和页尾。如下图所示：



- 页眉用于打印纸张页眉部分的内容，每一页均会出现的内容；
- 段落出现在纸张除了页眉和页尾部分的任何位置，但打印过程中只会出现一次；
- 表格用于描述多行明细数据，在报表中可以有多个表格部分；
- 页尾用于打印纸张页尾部分的内容，每一页均会出现的内容。

18. 2:一般明细报表

下面以一个简单的采购订单为例说明一般明细报表的制作，以采购订单为例。基础数据如下图所示：

	部门	物料	数量
1	FI 财务部	COMPUTER 电脑	12.00
2	SALE 销售部	FAX 传真机	2.00
3	TECH 技术部	PRINTER 打印机	5.00
4	FI 财务部	FAX 传真机	20.00
5	SALE 销售部	PRINTER 打印机	5.00
6	TECH 技术部	TELEPHONE 电话机	200.00
7	FI 财务部	PRINTER 打印机	54.00
8	SALE 销售部	TELEPHONE 电话机	6.00
9	TECH 技术部	COMPUTER 电脑	45.00

定义报表的过程一般分为如下几个步骤：

定义报表的属性，源代码如下：

```
<Report Caption="采购订单" FormKey="POrder" Key="POrder" LeftMargin="5" PageHeight="841" PageOrientation="Landscape"
PageWidth="595" PaperOrientation="Landscape" TopMargin="5">
  <DataSource>
    ...
  </DataSource>
  <Grid>
    ...
  </Grid>
  ...
</Report>
```

主要的属性如下：

- Key 报表的标识，要求全局唯一；
- Caption 表示报表的名称，显示用；
- FormKey 为关联表单的标识，PrintPreview这类的函数通过该属性查询相应表单的报表输出模板；
- PageOrientation 为纸张的方向；
- PageWidth和PageHeight为页面的宽高，单位为1/72英寸；

定义数据源，源代码如下：

```
<DataSource>
  <Table Caption="基本信息" Key="POrderHead" SourceType="Table">
    <Field DBFieldKey="BillDate" Key="BillDate"/>
    <Field DBFieldKey="NO" Key="NO"/>
  </Table>
  <Table Caption="明细表" Key="POrderDtl" SourceType="Table">
    <Field Caption="部门" DBFieldKey="Dept" Key="Dept"/>
    <Field Caption="物料" DBFieldKey="Mtl" Key="Mtl"/>
    <Field Caption="数量" DBFieldKey="Amount" Key="Amount"/>
  </Table>
</DataSource>
```

这里取的是采购订单中的部分数据。

定义报表体，源代码如下：

```

<Grid>
  <Section Type="Table">
    <Columns>
      <Column Width="85"/>
      <Column Width="85"/>
      <Column Width="85"/>
    </Columns>
    <Rows>
      <Row Height="22" Type="DetailHead">
        <Cell Caption="部门"/>
        <Cell Caption="物料"/>
        <Cell Caption="数量"/>
      </Row>
      <Row Height="22" TableKey="POrderDt1" Type="Detail">
        <Cell Key="Dept" Caption="部门" FieldKey="Dept" SourceType="Field" FillEmptyContent="True">
          <Display>
            <Format DataType="Dict" ItemKey="Department" FieldKeys="Code, Name"/>
          </Display>
        </Cell>
        <Cell Key="Mtl" Caption="物料" FieldKey="Mtl" SourceType="Field" FillEmptyContent="True">
          <Display>
            <Format DataType="Dict" ItemKey="Material" FieldKeys="Code, Name"/>
          </Display>
        </Cell>
        <Cell Caption="数量" FieldKey="Amount" Key="Amount" SourceType="Field" FillEmptyContent="True"/>
      </Row>
    </Rows>
  </Section>
</Grid>

```

Grid用于确定报表的格式定义，每个报表的主体部分都是由网格组成，每个网格由多个区段(Section)组成；Yigo中的报表格式由多区段的网格组成。

在当前例子中，我们定义一个3列的区段用于显示“部门”、“物料”、“数量”的明细数据。

输出报表，通过在表单中定义“打印”操作输出关联表单的报表，源代码如下：

```

<Operation Key="Print" Caption="打印" Visible="True" Enable="True">
  <Action>
    <![CDATA[
PrintPreview()
]]>
  </Action>
</Operation>

```

最终输出的报表如下图所示：

采购清单 ×

采购订单 ×

打印

前一页

下一页

关闭

导出

部门	物料	数量
FI 财务部	COMPUTER 电脑	12
SALE 销售部	FAX 传真机	2
TECH 技术部	PRINTER 打印机	5
FI 财务部	FAX 传真机	20
SALE 销售部	PRINTER 打印机	5
TECH 技术部	TELEPHONE 电话机	200
SALE 销售部	PRINTER 打印机	54
TECH 技术部	TELEPHONE 电话机	6
FI 财务部	COMPUTER 电脑	300

18. 3: 分组

在前一部分的明细报表输出中我们发现，输出的数据是无序的，如果希望对数据进行分组排列，通过在明细单元格上定义分组属性可以将数据分组排列，假如按照“部门”分组排列，源代码如下：

```
<Cell Key="Dept" Caption="部门" FieldKey="Dept" SourceType="Field" GroupType="RowGroup">
  ...
</Cell>
```

以上代码定义了Dept单元格的分组类型为RowGroup，输出报表如下图所示：

采购清单 ×

采购订单 ×

打印

前一页

下一页

关闭

导出

部门	物料	数量
FI 财务部	COMPUTER 电脑	12
FI 财务部	FAX 传真机	20
FI 财务部	COMPUTER 电脑	300
SALE 销售部	FAX 传真机	2
SALE 销售部	PRINTER 打印机	5
SALE 销售部	PRINTER 打印机	54
TECH 技术部	PRINTER 打印机	5
TECH 技术部	TELEPHONE 电话机	200
TECH 技术部	TELEPHONE 电话机	6

进一步扩充此例子，为每个分组增加分组汇总行，因此在明细行下方增加一如下行，如下：

```
<Row Height="22" TableKey="POrderDtl" Type="Detail">
  ...
</Row>
<Row GroupKey="Dept" Height="22" Type="Group" BackColor="#eeeeee">
  <Cell Caption="小计"/>
  <Cell/>
  <Cell Formula="sum(' Amount' )" SourceType="Formula"/>
</Row>
```

这里定义了一行Type的取值为Group的行，即分组行，GroupKey定义关联的明细上的分组单元格。运行结果如下图所示：

采购清单 ×

采购订单 ×

打印

前一页

下一页

关闭

导出

部门	物料	数量
FI 财务部	COMPUTER 电脑	12
FI 财务部	FAX 传真机	20
FI 财务部	COMPUTER 电脑	300
小计		332
SALE 销售部	FAX 传真机	2
SALE 销售部	PRINTER 打印机	5
SALE 销售部	PRINTER 打印机	54
小计		61
TECH 技术部	PRINTER 打印机	5
TECH 技术部	TELEPHONE 电话机	200
TECH 技术部	TELEPHONE 电话机	6
小计		211

增加总计行，代码如下：

```
<Row Height="22" Type="DetailTail" BackColor="LightGray">
  <Cell Caption="总计"/>
  <Cell />
  <Cell SourceType="Formula" Formula="sum(' amount')"/>
</Row>
```

报表的结果如下：

采购清单 ×

采购订单 ×

打印

前一页

下一页

关闭

导出

部门	物料	数量
FI 财务部	COMPUTER 电脑	12
FI 财务部	FAX 传真机	20
FI 财务部	COMPUTER 电脑	300
小计		332
SALE 销售部	FAX 传真机	2
SALE 销售部	PRINTER 打印机	5
SALE 销售部	PRINTER 打印机	54
小计		61
TECH 技术部	PRINTER 打印机	5
TECH 技术部	TELEPHONE 电话机	200
TECH 技术部	TELEPHONE 电话机	6
小计		211
总计		604

如果需要给单元格加上边框，定义单元格的边框属性：

```
<Cell>
  <Display>
    <Border BottomColor="0x000000ff" BottomStyle="1" LeftColor="0x000000ff" LeftStyle="1" RightColor="0x000000ff"
      RightStyle="1" TopColor="0x000000ff" TopStyle="1"/>
  </Display>
</Cell>
```

这里我们为表格设置全边框，运行结果如下：

采购清单 ×

采购订单 ×

打印

前一页

下一页

关闭

导出

部门	物料	数量
FI 财务部	COMPUTER 电脑	12
FI 财务部	FAX 传真机	20
FI 财务部	COMPUTER 电脑	300
小计		332
SALE 销售部	FAX 传真机	2
SALE 销售部	PRINTER 打印机	5
SALE 销售部	PRINTER 打印机	54
小计		61
TECH 技术部	PRINTER 打印机	5
TECH 技术部	TELEPHONE 电话机	200
TECH 技术部	TELEPHONE 电话机	6
小计		211
总计		604

18. 4: 列扩展示例-单层扩展

列扩展是指在列向根据数据或预先定义的规则将多行数据显示在同一行中，将对列进行扩展以显示这些数据。我们以领料单为例来说明列扩展报表。有如下数据：

	物料	用途	数量
1	COMPUTER 电脑	自用	1.00
2	COMPUTER 电脑	转租	2.00
3	COMPUTER 电脑	储备	3.00
4	FAX 传真机	自用	10.00
5	FAX 传真机	转租	20.00
6	FAX 传真机	储备	30.00
7	PRINTER 打印机	自用	100.00
8	PRINTER 打印机	转租	200.00
9	PRINTER 打印机	储备	300.00
10	TELEPHONE 电话机	自用	1,000.00
11	TELEPHONE 电话机	转租	2,000.00
12	TELEPHONE 电话机	储备	3,000.00

我们需要在列向根据用途显示数据，列扩展的源代码如下：

```
<Cell Key="AmountTitle" Caption="数量" ColumnExpand="True">
  <ColumnExpand ExpandType="Title"/>
</Cell>

<Cell Key="UseType" Caption="用途" ColumnExpand="True">
  <ColumnExpand ExpandType="Data" SourceType="Data" ColumnKey="UseType">
    </ColumnExpand>
    <Display>
      <Format DataType="List">
        <ListItem Value="0" Text="自用"/>
        <ListItem Value="1" Text="转租"/>
        <ListItem Value="2" Text="储备"/>
      </Format>
    </Display>
  </Cell>
```

注意：这里的两个单元格的定义从两行的定义抽取，完整的定义请见源泉代码，见REPORT_C1中的MtlReq.xml，本例的数据请使用REPORT_C1.bak备份。

代码中说明了单元格ColumnExpand为True，表示单元格为列扩展定义，ExpandType定义扩展的依据来源于数据，具体的属性请参见Yigo语言参考，ColumnKey定义扩展依据的数据列标识，Display为显示定义，这里不讨论。报表的运行结果如下：

领料单 ×

领料单 ×

打印

前一页

下一页

关闭

导出

物料	数量		
	自用	转租	储备
COMPUTER 电脑	1	2	3
FAX 传真机	10	20	30
PRINTER 打印机	100	200	300
TELEPHONE 电话机	1000	2000	3000

18. 5: 列扩展示例-嵌套扩展

这是我们考虑有两个扩展字段的情况。源代码见REPORT_C2中的Mt1Req.xml，本例中的数据库请使用REPORT_C2.bak备份。结果示例如下：

领料单 ×

领料单 ×

打印

前一页

下一页

关闭

导出

物料	数量					
	自用		转租		储备	
	自提	委托	自提	委托	自提	委托
TELEPHONE 电话机	1000	2000	3000	4000	5000	6000
FAX 传真机	10	20	30	40	50	60
PRINTER 打印机	100	200	300	400	500	600
COMPUTER 电脑	1	2	3	4	5	6

18. 6: 列扩展示例-列扩展合计

这里我们仍然使用上例的列扩展，但在使用类型中增加列向合计。源代码见REPORT_C3中的Mt1Req.xml，本例中的数据库请使用REPORT_C3.bak。结果示例如下：

领料单 ×

领料单 ×

打印

前一页

下一页

关闭

导出

物料	数量								小计
	自用		小计	转租		小计	储备		
	自提	委托		自提	委托		自提	委托	
TELEPHONE 电话机	1000	2000	3000	3000	4000	7000	5000	6000	11000
FAX 传真机	10	20	30	30	40	70	50	60	110
PRINTER 打印机	100	200	300	300	400	700	500	600	1100
COMPUTER 电脑	1	2	3	3	4	7	5	6	11

增加列向总计，增加列并添加相应的单元格，列的源代码如下：

```
<Column Width="150" ExpandKey="AmountTitle" Type="Total"/>
```

程序的运行结果如下：

领料单 ×

领料单 ×

打印

前一页

下一页

关闭

导出

物料	数量								小计	总计
	自用		小计	转租		小计	储备			
	自提	委托		自提	委托		自提	委托		
TELEPHONE 电话机	1000	2000	3000	3000	4000	7000	5000	6000	11000	21000
FAX 传真机	10	20	30	30	40	70	50	60	110	210
PRINTER 打印机	100	200	300	300	400	700	500	600	1100	2100
COMPUTER 电脑	1	2	3	3	4	7	5	6	11	21

18. 7: 内容的溢出处理

这里我们以采购订单为例来说明。源代码见REPORT_C4中的P0Order.xml。

内容的溢出处理方式定义通过单元格的Overflow属性来定义：

```
<Cell Overflow="" />
```

溢出的处理方式以如下两种：

- 适应单元格大小，即调整内容以适应单元格的大小，Overflow的取值为1；
- 适应内容，即调整单元格的高度以适应内容，Overflow的取值为2。

在POrder表单的明细中增加Memo字段，同时在报表中的明细中也增加Memo字段，假定有如下原始数据：

	部门	物料	数量	备注
1	FI 财务部	COMPUTER 电脑	12	1:a string
2	SALE 销售部	FAX 传真机	2	2:a string 2
3	TECH 技术部	PRINTER 打印机	5	3:This is a string,This is a string,This is a string,
4	FI 财务部	FAX 传真机	20	4:a string
5	SALE 销售部	PRINTER 打印机	5	5:This is a string,This is a string,This is a string,
6	TECH 技术部	TELEPHONE 电话机	200	6:a string
7	SALE 销售部	PRINTER 打印机	54	7:empty
8	TECH 技术部	TELEPHONE 电话机	6	8:This is a string,This is a string,This is a string,
9	FI 财务部	COMPUTER 电脑	300	9:string

我们设定报表中Memo这一列的输出为适应大小(Overflow取值为1)，代码如下：

```
<Cell Caption="备注" FieldKey="Memo" Key="Memo" SourceType="Field" FillEmptyContent="True" Overflow="1">
</Cell>
```

则输出结果如下：

采购清单 ×

采购订单 ×

打印

前一页

下一页

关闭

导出

部门	物料	数量	备注
FI 财务部	COMPUTER 电脑	12	1:a string
FI 财务部	FAX 传真机	20	4:a string
FI 财务部	COMPUTER 电脑	300	9:string
小计		332	
SALE 销售部	FAX 传真机	2	2:a string 2
SALE 销售部	PRINTER 打印机	5	3 THIS IS A SETTING,THIS IS A SETTING,THIS IS A SETTING,
SALE 销售部	PRINTER 打印机	54	7:empty
小计		61	
TECH 技术部	PRINTER 打印机	5	3 THIS IS A SETTING,THIS IS A SETTING,THIS IS A SETTING,
TECH 技术部	TELEPHONE 电话机	200	5:a string
TECH 技术部	TELEPHONE 电话机	6	3 THIS IS A SETTING,THIS IS A SETTING,THIS IS A SETTING,
小计		211	
总计		604	

如果将Memo这一列的输出改为适应内容 (Overflow的取值为2)，则输出为：

采购清单 ×

采购订单 ×

打印

前一页

下一页

关闭

导出

部门	物料	数量	备注
FI 财务部	COMPUTER 电脑	12	1:a string
FI 财务部	FAX 传真机	20	4:a string
FI 财务部	COMPUTER 电脑	300	9:string
小计		332	
SALE 销售部	FAX 传真机	2	2:a string 2
SALE 销售部	PRINTER 打印机	5	5:This is a string,This is a string,This is a string,
SALE 销售部	PRINTER 打印机	54	7:empty
小计		61	
TECH 技术部	PRINTER 打印机	5	3:This is a string,This is a string,This is a string,
TECH 技术部	TELEPHONE 电话机	200	5:a string
TECH 技术部	TELEPHONE 电话机	6	3:This is a string,This is a string,This is a string,
小计		211	
总计		604	

18. 8: 页眉和页尾

通过Section的类型可以定义某个区段为页眉和页尾，如下：

```
<!-- 页眉 -->
<Section Type="PageHead">
...
</Section>

<!-- 页尾 -->
<Section Type="PageTail">
...
</Section>
```

Type为PageHead时为布眉，为PageTail时为页尾。通常页眉或页尾部分用于显示页号或标题，可以通过单元格的内容来定义，如下：

```
<Cell SourceType="PageNo" PageNo="" />
```

其中SourceType为PageNo时定义内容来源于页号，此时PageNo定义页号的形式，有两种取值，分别是Page(页序号)、PageOfCount(在总页数中的页序号)；在当前例子中我们假定需要在页眉部分打印公司标题和总页数中的页序号，在页尾部分只打印序号。源代码如下，见REPORT_C3中的P0rder.xml打印报表。

```
<!-- 页眉部分打印公司名称和页号 -->
<Section Type="PageHead">
  <Columns>
    <Column Width="170"/>
    <Column Width="85"/>
  </Columns>
  <Rows>
    <Row Height="22">
      <Cell Caption="博科资讯股份有限公司"/>
      <Cell SourceType="PageNo" PageNo="PageOfCount"/>
    </Row>
  </Rows>
</Section>

<!-- 页尾部分打印页号 -->
<Section Type="PageTail">
  <Columns>
    <Column Width="85"/>
    <Column Width="85"/>
    <Column Width="85"/>
  </Columns>
  <Rows>
    <Row Height="22">
      <Cell />
      <Cell />
      <Cell SourceType="PageNo" PageNo="Page"/>
    </Row>
  </Rows>
</Section>
```

效果如下：

博科资讯股份有限公司		第1页, 共1页
部门	物料	数量
FI 财务部	COMPUTER 电脑	12
FI 财务部	FAX 传真机	20

第1页

18.9: 自动拆行

自动拆行是在超长文本的情况下，用于将行拆到多个页上打印，自动拆分通过行的LinkBreak属性定义，在允许分拆的行中的单元格如果内容超出当前页面允许拆分到下一页。

在P0rder表单中增加一个协议备注，将此文本打印成合同中的条款。完整的代码见REPORT_C4中的P0rder.xml报表。这里只列出行的源代码。

```
<Row LineBreak="True">
  <Cell Overflow="2">
    </Cell>
    ...
  </Row>
```

具体效果由于图片过大，这里不再列出。