计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(8):148−154 [doi: 10.15888/j.cnki.csa.007012]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 XML 配置的动态数据查询技术①

江日念, 许　锟, 林　霞

(中国石油勘探开发研究院 计算机应用技术研究所, 北京 100083)
通讯作者: 江日念, E-mail: jrn1012@petrochina.com.cn

摘　要: 针对信息系统中静态数据列表开发效率低下和不能满足用户定制化需求的问题, 提出基于 XML 配置的动态数据查询技术. 首先通过 Apache Digester 解析表单的 XML 结构化配置, 将实例化的配置信息交由动态查询引擎处理, 动态查询引擎结合配置信息通过 SQL、HQL、接口查询获取数据. 最后将配置和数据以 JSON 格式传给客户端处理, 以二次封装改进的 dhtmlxGrid 表格控件展示, 形成动态数据列表. 该动态查询技术已在多个系统中应用, 应用效果证明, 其显著提高了软件研发效率, 能够满足组合查询定制、数据列自定义、数据列渲染、数据范围控制等各种定制需求.

关键词: digester; 动态查询引擎; dhtmlxGrid; 组合查询定制; 数据范围控制

引用格式: 江日念,许锟,林霞.基于 XML 配置的动态数据查询技术.计算机系统应用,2019,28(8):148–154. http://www.c-s-a.org.cn/1003-3254/7012.html

Dynamic Data Query Technology Based on XML

JIANG Ri-Nian, XU Kun, LIN Xia

(Department of Computer Application Technology, Research Institute of Petroleum Exploration &Development, Beijing 100083, China)

Abstract: As the static data table in the information system has a low efficiency in development and cannot be changed according to customized requirements, this study proposes a dynamic query technology based on XML configurations. First of all, digester reads the instantiated configuration information from XML which will be handled by the dynamic query engine. Secondly, the dynamic query engine combined with XML configuration obtains the dataset by SQL, HQL, or interface. Finally, the configuration and dataset are transmitted to clients in the form of JSON and shown by improved dhtmlxGrid. After the above steps, the dynamic data table is formed. The technology is applied to the scientific research management system and human resource information system. The application results prove that this technology significantly improves the development efficiency and meets all kinds of customized requirements such as combination-conditions query customization, data column customization, data column rendering, and data range control.

Key words: digester; dynamic query engine; dhtmlxGrid; combination-conditions query customization; data range control

数据列表 (报表) 作为一种数据集中展示方式, 在信息系统中充当着重要的角色, 是系统重要的组成部分. 用户的需求有很大部分来源于对数据展示的需求. 传统的开发方式是逐一为数据列表编码, 为满足特定需求逐一开发界面, 开发出来的是静态数据列表. 静态数据列表存在以下 3 方面的缺点.

(1) 可重用性低, 开发效率低. 即便列表高度相似, 也必须单独实现, 代码、界面无法重用, 开发人员把大量时间浪费在相似代码的编写、调试上, 无法专注于核心业务逻辑的设计与实现, 导致开发效率不高.

① 收稿时间: 2019-01-25; 修改时间: 2019-02-27; 采用时间: 2019-03-06; csa 在线出版时间: 2019-08-08

(2) 可扩展性低, 部署能力弱. 用户在系统上线使用后, 往往都会有对列表展示、列表查询变更的需求, 此时要进行调整就比较麻烦, 必须由开发人员修改源码, 甚至重新发布系统.

(3) 开发门槛高, 参与程度低. 传统方式要求人员具备一定的开发能力, 所以在开发和后期运维调整过程中, 只能全程由开发人员参与, 将业务人员隔绝在开发之外, 不能充分参与开发与运维, 导致人力资源的浪费.

为了解决数据列表中的上述问题, 陈传波等提出一种 Web 报表模型, 通过 Crystal Reports 引擎和 ADO.NET 对报表底层接口进行控制, 实现报表自定义[1]. 金雨等人把报表分解为若干结构单元, 建立了通用报表模型, 并在 Dephi 开发环境下阐述了其设计和研发过程[2]. 为了进一步提升报表的柔性, 部分研究的工作基于 XML 来展开. 高红艳通过一种可视化的报表设计器来解决报表样式复杂的问题, 并提出一种基于 XML 的无插件方案来解决报表数据问题[3]. 卢笑天等设计了基于 XML 的可定制查询报表系统, 满足报表查询条件和查询字段的动态可定制, 同时提供了可拖动可视化的定制界面[4]. 符云清等利用 Excel 设计自定义报表, 程序读取上传的 Excel 并解析为 HTML 文件和 XML 文件, 系统根据 Excel 中配置的字段生成 SQL, 实现报表格式和数据分离[5]. 汤加等利用 XML 实现动态创建数据表, 并在此基础上配置数据源及其字段与报表单元格之间的关系, 系统将据此自动拼装出相应的 SQL, 从而快速生成各类基于单数据源的报表[6].

上述的部分文献通过 XML 实现报表的灵活定制, 但从开发者和用户的角度看, 尚不能完全满足以下四个数据列表典型需求:

(1) 组合查询定制

开发者可以很方便地增加或者修改查询条件, 而尽可能少的修改系统, 最好不要修改后台代码, 从而减少因重新部署导致系统停机风险, 同时降低开发工作量, 提升效率, 这方面已经有许多相关工作[7,8].

(2) 数据列自定义

用户可以对不关注的数据列设置隐藏, 尤其在数据列过多的情况下, 通过自定义列的显示与隐藏, 突显用户关注的信息, 屏蔽不重要信息, 提升用户体验.

(3) 数据列渲染

一个功能完备的数据列表, 不局限于数据的展示和查询, 还应该具备交互功能, 比如按钮列、选择列、超链接列、状态列等, 这些列需要通过表格组件的渲染将原始数据“翻译”并呈现在用户面前, 为用户提供交互功能. 对开发者来说, 可配置的列渲染大大节省数据列表开发工作量.

(4) 数据范围控制

对拥有不同权限的用户来说, 同一个数据列表应该有不同的数据范围, 以人员管理为例: 公司总裁可以查询公司所有人员信息, 部门经理只能查询本部门人员信息.

针对传统静态数据列表的不足, 本文提出一种动态数据查询技术, 以满足信息系统动态数据列表需求.

1 相关技术简介

1.1 XML 技术

XML 是一种数据交换格式, 它以一种开放的自我描述方式定义数据结构, 在描述数据内容的同时能突出对结构的描述, 从而体现数据之间的关系. XML 具有扩展性好、结构性好、与平台无关的特点, 它提供了统一的方法来描述和交换独立于应用程序或供应商的结构化数据.

之所以选择 XML 配置数据列表正是因为其灵活、可定制的结构性和自描述性. 其结构性可以显著降低数据列表配置的门槛, 能够使得管理人员和业务人员都可以参与到数据列表的配置开发. 同时, XML 的自描述性能够让其既作为数据源, 又可以作为一种设计文档, 它正好弥补了传统设计文档和实现之间不一致性, 因为 XML 所做的修改会及时反映到实现中, 时刻保持着与实现同步. 其作为模型存储、数据存储的文件, 已经得到了广泛的应用[9–13].

1.2 Apache Digester 解析 XML 技术

Digester 是 Apache 的一个组件, Digester 包可以配置一个从 XML 文件到 Java 对象映射[14]. 其底层采用 SAX 来解析 XML 文件. Digester 维持了一个对象栈, 可以看作对象转换平台, 用来存放转换中生成的、或是为转换临时创建的 Java 对象. 由于 Digester 屏蔽了底层实现细节, 使用者只需关注操作本身, 大大简化了转换操作. 使用 Digester 的注解模式, 通过建立与 XML 内容相互映射的 JavaBean 来解析 XML. 为了简化使用, 它通过匹配模式来定位要解析的 XML 标签. 示例如下:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<queryContext>
<query>
<column id='name' name='姓名' width="200px"/>
<column id='id' name='单位' width="200px"/>
…
</query>
</queryContext>
</xml>
```

每个标签与相应的匹配模式对应如表 1 (仅列出部分):

表 1 标签匹配模式

标签	匹配模式
<queryContext>	queryContext
<query>	queryContext/query
<column>	queryContext/query/column

如果将 XML 文件结构视为一棵树的话, 那么每个标签的匹配模式就是从根元素到这个元素的路径, 除了使用具体的标签, 还可以使用通配符.

使用匹配模式可以很方便地定位需要处理的元素, 规则在匹配模式被找到时起作用. 所有的规则都是从 org.apache.commons.digester.Rule 派生的. 可通过 @ObjectCreate、@SetProperty、@BeanProperty-Setter 等注解和 Digester 解析功能就可以将节点 queryContext, query, column 分别映射到类 Query-Context, Query, Column 上.

1.3 dhtmlxGrid 控件

dhtmlxGrid 是一个灵活、智能、容易使用的 JavaScript 表格控件, 它允许使用者以 Ajax 交互方式实现表格, 使其具有单元格编辑、固定多表头、固定多表尾、列宽可变、列可排序、列可拖动、冻结列等功能. 其丰富的功能足以满足用户对数据列表的各种展示需求.

dhtmlxGrid 控件支持 XML、JSON,官方网站提供了详细的英文 API 文档. 为了更好、更方便地在动态数据查询技术中应用 dhtmlxGrid 控件, 本文使用已改进的 dhtmlxGrid 控件, 并对其进行二次封装, 二次封装的控件以下简称 CommonGrid.

2 动态数据查询设计与实现

2.1 总体框架

动态数据查询的设计理念是: 将数据列表抽象化, 分为表级别配置与列级别配置, 并以 XML 保存该配置; 系统初始化后加 Apache Digester 读取 XML 配置并对象化缓存到内存中, 缓存时分为 Product 模式和 Debug 模型, Product 模式不会监听 XML 配置的变化, 而 Debug 模式将动态监听 XML 配置并及时更新缓存. 后端动态查询引擎根据前端发送请求, 动态获取配置和数据, 返回到前端, 前端 CommonGrid 组件对数据进行展示和渲染, 总体框架如图 1.

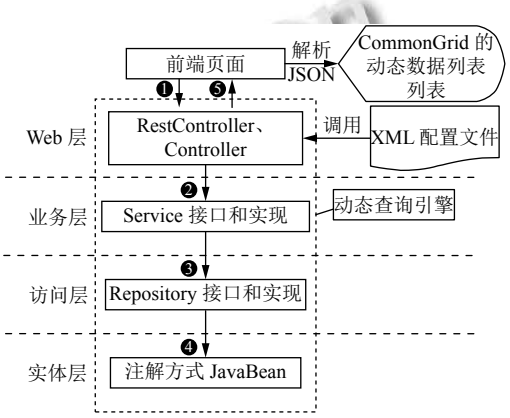

图 1 动态数据查询框架

动态数据查询框架基于 Spring Boot 实现, 分为 Web 层、业务逻辑层、数据访问层、实体层, 各层之间相互衔接, 且明确地完成了自身职责. ① 前端页面创建 CommonGrid 表格控件, 加载查询条件, 向服务器发送数据加载、查询请求, Web 获取请求参数并查询缓存的 XML 配置. ② 调用业务逻辑层的动态查询引擎处理, 进行配置解析, 将配置解析成可执行的查询操作. ③ 调用持久化方法, 执行查询操作. ④ 查询结果映射到 JavaBean 对象. ⑤ 服务器处理完请求, 将数据、分页、列等信息以 JSON 格式传送前端 CommonGrid 处理.

2.2 数据列表抽象与解析

一个基本的数据列表是由数据列、数据行组成的. 同时, 数据列表还附带着一些功能, 比如列表分页、数据查询、数据导出、合并行、合并列、多表头、多表尾、拖拽列等. 通过对数据列表的分析和抽象, 以及结合 CommonGrid 的特点, 构建数据列表模型如图 2 所示. 其模型的构建步骤如下:

(1) 列表抽象: 是对整个列表的整体抽象, 对应的是 Query 类, 用于配置列表的全局属性. 其属性包括表的 ID、表名、初始化分页信息 (是否允许分页、每页记录数)、冻结列数、初始化排序信息. 为了获取数据,

根据不同的数据获取方式, 需要配置对应的 SQL 或者映射数据指向的 className. 列表中也配置了些冗余属性, 如用于缓存数据列信息的 columnList, 用于缓存调用方法信息的 callList 等.

(2) 数据列抽象: 对数据列的抽象, 对应 Column 类, 用于配置列属性, 是数据列表中非常重要的部分. 前端组件 CommonGrid 根据数据列配置动态渲染列, 其属性配置如表 2 所示.

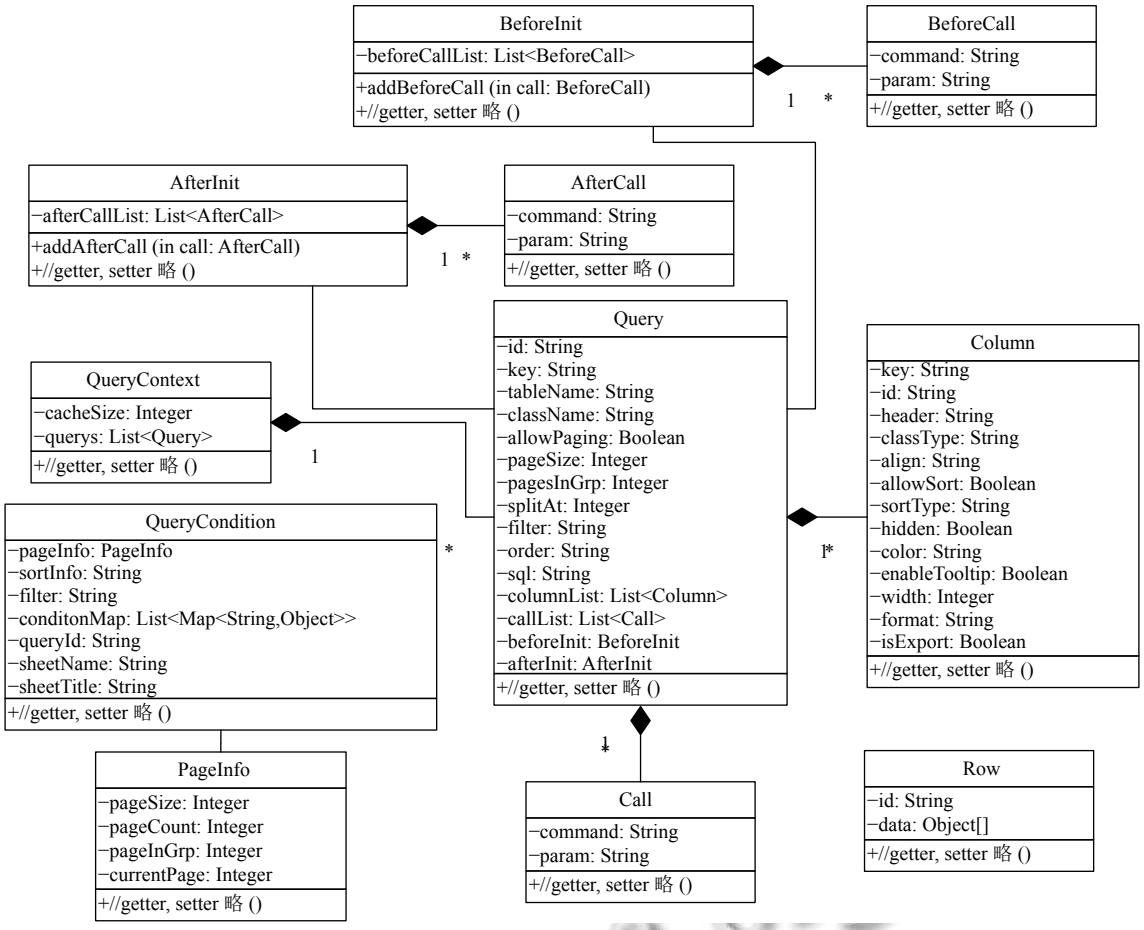

图 2 数据列表模型 (XML 结构抽象)

表 2 数据列 Column 的属性

名称	默认值	可选值	参数说明
key			数据列的数据源对应的属性 (字段), 同时作为数据列唯一标识
id			当数据列 key 一样时, 数据列唯一标识
header			数据列显示标题
classType	java.lang.String		数据类型
align	center	left,center,right	水平对齐方式
allowSort	false		是否允许列排序
hidden	false		是否隐藏数据列
enableTooltip	false		鼠标悬停是否显示文字, 主要用于长文本
maxLen			长文本截取的长度, 与 enableTooltip 一起使用
color			列颜色
operator	eq	eq,not_eq,like,in,not_in …	作为查询条件的条件操作符
width			列宽
dateFormat	yyyy-mm-dd		日期格式
isExport			是否是导出该列

(3) 数据行抽象: 从后台获取的数据行抽象, 对应 Row 类, 包括两个属性: 行 ID 和对象数组. 数据行的集合构成了数据列表的数据, 并以 JSON 格式返回给前端组件.

(4) 执行方法抽象: 为了灵活定制表格组件在初始化前、初始化中、初始化后等不同的时间节点要执行的前端方法, 定义了 BeforeCall、Call、AfterCall 类, 类提供了方法名和参数两个属性, CommonGrid 组件在对应时间节点获取这些在后台组装好 BeforeInit、CallList、AfterInit 的的钩子方法列表并执行.

(5) 查询条件抽象: 用于接收前端查询交互时的信息, 对应 QueryCondition 类, 包括的信息有分页信息 pageInfo、查询条件信息 contionMap、查询配置主键 queryId 等. 在数据列表首次加载时, 从后端缓存的 XML 配置中获取信息并组装. 非首次查询时, 由前端实时参数和前端缓存中获取, 如果两者均存在, 优先从前端实时参数中获取, 并更新到前端缓存. 数据列表的其他需要交互的实时参数, 如通用导出的 Excel 表信息也可以在 QueryCondtion 类中定义, 如表名 sheetName, 表头标题 sheetTitle.

2.3 动态查询引擎

对前端而言, 动态查询引擎就是个黑盒, 只要接收参数, 动态查询引擎就以一定的格式向客户端输出满足条件的 JSON 数据. 图 3 为其工作流程示意图.

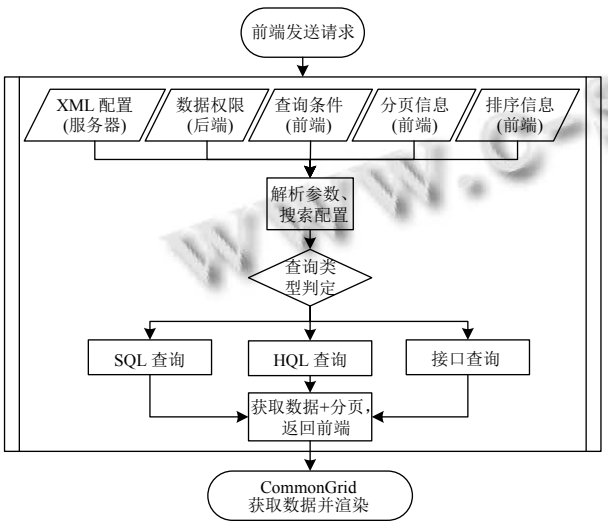

图 3 动态查询引擎工作流程

动态查询引擎支持三种查询方式: SQL、HQL、接口. SQL 查询用于对多表关联的复杂查询, HQL 常用于单表对象化查询, 接口查询是在 SQL 和 HQL 无法满足需求情况下推荐的方式, 比如条件参数的获取需要复杂的业务逻辑计算才能得到. 这三种方式都需要查询条件、分页信息、排序信息作为输入、查询数据和配置作为输出. 在界面首次加载时, 分页信息和排序信息从 XML 配置中读取, 而当用户发送查询请求、换页请求、排序请求时, 分页信息和排序信息由客户端传参而来.

下面以 SQL 查询为例, 说明动态查询引擎的具体工作过程:

(1) 组装分页信息: 如果客户端传入分页信息, 则优先读取客户端分页信息, 否则读取 XML 配置中的分页信息;

(2) 组装查询条件: 获取客户端传入的查询条件, 逐一遍历查询条件, 若传入值为空, 则跳过. 否则将获取的查询条件名、查询操作符 (between、or、like、in、eq、not_eq 等)、传入值在引擎中组装, 直至遍历结束;

(3) 组装排序信息: 如果客户端传入排序信息, 则读取客户端排序信息, 否则读取 XML 配置的排序信息, 并将排序组装到 SQL 中;

(4) 组装数据范围条件: 在权限管理中, 获取用户对列表的授权配置, 并组装到 SQL 中;

(5) 查询执行: 组装的两个查询 SQL, 一个是获取数据的 SQL, 传入的参数为查询条件, 分页信息; 一个是获取数据总记录数的 SQL, 传入参数为查询条件;

(6) 数据对象化、格式化: 通过反射将数据映射到 JavaBean 对象, 并按照 XML 配置格式化, 生成 CommonGrid 控件能够接收的 JSON 格式的数据列表;

(7) 拼接列属性: 形成 CommonGrid 要执行的方法列表. 将各列的配置属性, 按照方法名和参数拼接起来, 形成 CommonGrid 可执行的方法名和列表;

(8) 数据渲染与展示: 将分页信息、数据列表、CommonGrid 执行方法列表以 JSON 格式发送给客户端.

3 应用效果

基于该动态查询引擎实现的动态列表实现了组合查询定制、数据列自定义、数据列渲染、数据权限控制等常见用户自定义数据列表需求, 下面以人事信息

系统为例, 说明其应用效果.

3.1 组合查询定制

需求: 针对用户对数据列表查询条件频繁变更的需求, 动态查询引擎仅需要在前端页面添加或者修改相应的控件, 而无须修改后端代码, 也无须对系统进行重启.

实现: 将查询条件配置在查询框中, 也可以配置在列头, 查询引擎解析查询条件, 自动获取并展示. 如图 4 所示.

图 4 组合查询定制 (人事系统)

3.2 数据列自定义

需求: 用户可自定义数据列的显示与隐藏, 只显示自己关注的列.

实现: 如图 5 所示, 用户使用拖拽自定义列的显示与隐藏并保存到数据库中, 查询引擎将 XML 配置与用户自定义进行比对, 从而决定显示哪些数据列.

图 5 自定义数据列

3.3 数据列渲染

需求: 用户需要和数据列表进行交互, 比如按钮列、选择列等, 同时需要将部分原始数据翻译成用户可看懂的列, 比如将人员状态字段的“0”、“1”翻译成对应的“在职”、“离职”, 对长文本而言, 还需要截断, 在鼠标悬停时显示全部文本等.

实现: 在 XML 配置列通过 render 属性或者 fnRender 配置回调函数, 支持以下类型:

(1) render (type=eq) 固定值的翻译;

(2) render (type=window) 弹出窗体;

(3) render (type=link) 超链接;

(4) tooltip 鼠标悬停提示;

(5) fnRender 渲染回调函数. 前端 CommonGrid 解析配置, 按需进行数据渲染.

3.4 数据范围控制

需求: 不同权限的用户针对同一数据列表拥有不同的数据范围, 通过系统配置实现, 避免类似的代码开发多次, 这样可以提升系统可维护性.

实现: 在系统后台的权限管理中动态配置数据权限, 查询引擎解析配置, 组合到查询条件中, 形成对应权限的数据范围, 如图 6 所示.

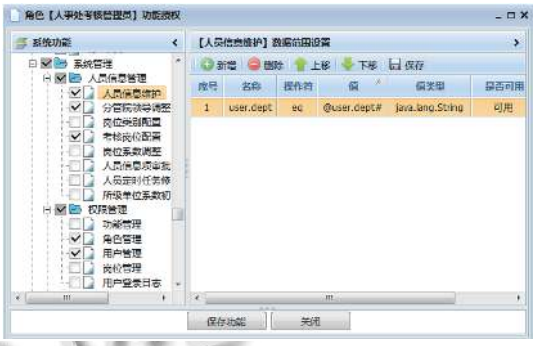

图 6 配置数据范围

除以上比较典型的功能外, 动态查询还集成了通用导出、冻结列、自定义多表头等.

4 结论与展望

本文在 Spring Boot 框架下提出一个动态数据查询技术, 从技术实现的总体框架、数据列表抽象与解析、动态查询引擎三个方面阐述了其设计和实现细节. 并从开发者和用户对动态数据列表的典型需求考虑, 选择组合查询定制、数据列自定义、数据列渲染、数据范围控制展示了系统的实现效果. 该技术已经应用于某机构的多个系统中, 在人事系统中的应用效果证明, 它能极大地提高开发效率、降低开发门槛、提高数据列表的扩展性和可维护性. 该技术具有一定的推

广性和实用性, 在企业信息系统研发中, 进一步结合代码生成器使用, 只需编写一个实体即可快速完成一个数据列表的增删改查功能, 且可以灵活扩展.

参考文献

1 陈传波, 黄刚, 刘清慧. 一种基于 ASP.NET 的自定义报表的设计与实现. 计算机工程与科学, 2006, 28(6): 112–114. [doi: 10.3969/j.issn.1007-130X.2006.06.037]

2 金雨, 张旭堂, 刘文剑. 用户自定义通用报表打印的设计与实现. 计算机应用与软件, 2008, 25(3): 132–134. [doi: 10.3969/j.issn.1000-386X.2008.03.049]

3 高红艳. Web 报表开发中的若干关键技术研究与应用[硕士学位论文]. 西安: 西安电子科技大学, 2011.

4 卢笑天, 唐慧佳. 基于 XML 的可定制查询报表系统的设计与应用. 计算机工程与设计, 2014, 35(5): 1847–1852. [doi: 10.3969/j.issn.1000-7024.2014.05.067]

5 符云清, 肖文婷, 廖希, 等. 基于 Excel 和 XML 的自定义报表设计与实现. 计算机科学, 2013, 40(7S): 147–149, 163.

6 汤加, 符云清, 万煊民. 一种基于单数据源的可视化自定义报表模型. 计算机科学, 2017, 44(5): 184–188. [doi: 10.11896/j.issn.1002-137X.2017.05.033]

7 唐伟, 施永香, 文巨峰. 基于.NET 的通用查询组件的开发. 计算机工程与设计, 2006, 27(14): 2708–2710. [doi: 10.3969/j.issn.1000-7024.2006.14.064]

8 邵雨舟. 应用系统中通用组合查询功能的实现方法. 电脑知识与技术, 2018, 14(20): 7–9.

9 信俊昌, 王国仁, 李国徽, 等. 数据模型及其发展历程. 软件学报, 2019, 30(1): 142–163.

10 陈佳铭, 王风立, 邓君湘, 等. 基于 XML 语言的导弹防御系统 HSTPN 博弈模型存储与加载. 计算机应用与软件, 2018, 35(12): 12–15.

11 王永娜, 赵奎, 王鸿亮, 等. 针对异构协议的动态解析器模型. 计算机系统应用, 2017, 26(1): 251–254. [doi: 10.15888/j.cnki.csa.005533]

12 张文宇, 许明健, 薛昱. 论 spring 的零配置与 XML 配置. 计算机系统应用, 2015, 24(2): 270–275. [doi: 10.3969/j.issn.1003-3254.2015.02.052]

13 Nassiri H, Machkour M, Hachimi M. One query to retrieve XML and Relational Data. Procedia Computer Science, 2018, 134: 340–345. [doi: 10.1016/j.procs.2018.07.201]

14 徐秀华, 汪诚波, 毕硕本. Digester 解析 XML 文档研究. 计算机系统应用, 2005, (1): 86–88. [doi: 10.3969/j.issn.1003-3254.2005.01.027]