基于 RESTful Web 技术的资源管理系统设计与实现

章 武 媚
(浙江同济科技职业学院 浙江 杭州 311231)

摘 要 随着 Web 技术的进步,对资源管理平台的功能需求不仅仅局限于管理实体资源对象,还包括将异种异构的资源封装成统一资源进行描述并加以管理的能力。Web 资源管理平台接入的资源描述信息往往具有海量、异构和可变的特性。提出一种应用 RESTful(Representational State Transfer)风格 Web 服务架构与 NoSQL(非关系型数据库)技术的资源管理三层架构系统,并将其实现为基于.NET 的 WCF RESTful Web 服务,实现资源的操作、配置、状态监控以及数据管理功能。

关键词 资源管理 RESTful 风格 WCF

中图分类号 TP393 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2014.05.006

DESIGN AND IMPLEMENTATION OF RESOURCE MANAGEMENT SYSTEM BASED ON RESTful WEB TECHNOLOGY

Zhang Wumei
(*Zhejiang Tongji Vocational College of Science and Technology, Hangzhou 311231, Zhejiang, China*)

Abstract With the development of Web technology, the requirements on the functions of resource management platform are not only limited to the management of physical devices objects, but also include the capability of encapsulating the heterogeneous resources into general resource for effective description and management. The description information for the data source registered into resource management platform usually has the characteristics of massive, heterogeneous and changeable. Based on this, we propose a three-layered architecture system for resource management on the basis of applying RESTful (Representational State Transfer) Web service style and NoSQL technology, and implement it as the .NET-based WCF RESTful Web service, which realises the functions of resource operation, configuration, state monitoring and data management.

Keywords Resource management RESTful style WCF

0 引 言

Web 资源是互联网资源的一部分,主要指采用超文本传输协议(HTTP)和用户进行交互的资源[1,2]。目前有很多系统都采用 Web 技术实现了对 Web 资源的有效管理。Web 资源管理主要就是指对资源进行有效地抽象、描述、分析、组织和存储,从而让用户有效准确地从海量数据中获取感兴趣的信息。Web 资源管理的方法具有实践价值,已广泛应用于商务管理,网站设计和搜索引擎研发等领域[3]。Web 资源管理技术早已成为研究的热点之一,在 Web 资源管理过程中最重要的步骤就是将 Web 页面的信息进行提取,统一表述,然后转化为具备良好数据结构的形式进行存储。因此国内外研究的 Web 资源管理相关技术成果主要集中在以下几个方面:

1) Web 信息表示:Web 资源大多为结构化或者半结构化的数据,因此需要建立模型对 Web 信息资源做有效地映射。国外较为经典的研究成果是斯坦福和 IBM 提出的结合了有向图与标示符的 OEM 模型以及国内由东南大学提出的用于描述异构数据的 OIM 模型[4]。

2) Web 信息提取:指的是从 Web 页面包含的结构化或者半结构化的资源中识别、转化出结构化、语义清晰的上下文,这也是一种"包装器"形式的处理手段。主要分为领域知识手工构建、归纳学习法、统计分析法、结构学习法等。典型的代表系统有 W4F 系统[5]、SRV 和 WHISK[6]、联合正则法、树距离法以及频率聚类法等[7,8]。

3) Web 资源检索:目前较为成熟的技术是全文检索技术,也就是搜索引擎技术,其速度快,使用简单,并且信息查全率比较高。经典的搜索引擎技术包括了基于内容的检索技术和协同过滤技术等[9]。当然也有研究者着重于 Web 资源检索语言的研究,如基于传统数据库的查询语言 SQL,OQL 以及基于规则的语言 RuleML 等。其中 W3C 制定的 SPARQL 为 Web 查询语言提供了统一的语法和语义标准。

4) Web 资源更新技术:根据更新的比例我们可以将 Web 资源的更新技术划分为统一更新法和个体更新法。前者根据同样的频率访问所有的 Web 资源,简单而易于实现;后者会根据资源个体改变的频率来重新访问各个页面,精确率较高但是复杂度也有所提升。

收稿日期:2013-04-08。浙江教育规划重点课题(SB141)。**章武媚**,副教授,主研领域:计算机应用及教学。

其它相关的技术还包括了链接分析、XPath、数据冗余与一致性维护等,此处不再赘述。

由于不同的 Web 资源管理系统以及平台的架构、技术千差万别,各个系统 Web 资源信息之间的数据无法共享。同时 Web 资源数据由于表述框架和监控机制存在严重不足,存储数据存在大量的冗余度和不完整性。Web 环境下的资源管理面临了更多的挑战。由此,本文针对 Web 资源管理的共性问题提出了一种 Web 资源管理系统的架构,通过建立完整统一的 Web 资源管理中心来为用户提供统一的资源访问和监测开放接口,更加全面、方便地访问 Web 资源。

1 资源管理平台需求分析

1.1 Web 资源特性

Web 2.0 模式作为用户从信息接受者的"被动模式"转换为信息组织、创建和生成来源的新型互联网"主动模式"新理念[10],其中的信息资源也呈现出独特的特性。

1) 异构性:在 Web 2.0 模式中,会有更多的信息资源接入到应用和平台中来。不同的资源对象通常会有不同的描述和表征方式[11]。

2) 动态性:互联网接入的资源往往是动态变化的。在获取 Web 资源的同时,资源对象都有可能在动态变化,表征资源特征的数据也会不断地积累[12]。

3) 无序化:互联网上资源具有高度的互操作性、跨平台性和松耦合的特点,随之带来的冗余很容易造成信息的无序化,用户无法充分有效地利用资源,寻找资源。

4) 内容和结构复杂:Web 资源内容包括了众多文本、多媒体、表单、文件等;Web 数据很可能通过不同的数据格式如 HTML,XML,JSON 等来进行表征[13]。

这些 Web 数据特性给我们的 Web 资源管理系统的设计与实现提出了三点需求:

1) Web 资源获取与抽象:系统需要针对管理的领域,根据指定的 URL 批量获取目标资源,同时将半结构化的无序资源转化为结构化的开放资源供用户理解与使用。

2) Web 资源统一描述:由于 Web 资源具有的动态性和无序性,需要通过通用的 Web 建模方式来对资源的结构和含义进行统一描述,从而促进信息融合、推理和决策[14]。

3) Web 资源数据监控与管理:系统需要记录 Web 资源的状态信息、访问方式与路径、更改情况,并保存记录;并提供接入系统资源数据、状态信息数据以及其他业务数据的接口。

1.2 RESTful Web Service 风格

根据上节中的需求分析,我们采用 RESTful Web Service 对 Web 资源管理系统进行描述和设计。REST 表示的意思是表征状态转移,指的是一组架构约束条件和原则。满足这些约束条件和原则的应用程序或设计就是 RESTful。

选择该架构风格的原因在于:RESTful 针对的就是围绕 HTTP协议上数据的传输,采用的 http 谓词与 Web 资源管理一致;并且该架构风格支持传输 XML、JavaScript Object Notation (JSON),这也有利于 Web 资源的统一描述文档(XML 或 JSON)管理;依照 RESTful 的设计原则,一切 Web 资源都以 URI 的方式进行统一的描述和访问,这也方便用户和开发者通过系统检索、获取和管理系统中的 Web 资源数据、状态信息数据以及其他业务数据。

RESTful 风格可以说是一种轻量级的方法。在 RESTful 系统中,服务器利用 URI(Universal Resource Identifier)暴露资源,客户端使用四个 http 谓词来访问资源[15]。因此需要显式地使用 CRUD 方法,即——创建、读取、更新和删除 CRUD(create, read, update, and delete),从而与 HTTP 服务建立四种映射:

1) 若要在服务器上创建资源,应该使用 POST 方法。

2) 若要检索某个资源,应该使用 GET 方法。

3) 若要更改资源状态或对其进行更新,应该使用 PUT 方法。

4) 若要删除某个资源,应该使用 DELETE 方法。

总的来说,通过 RESTful 风格设计 Web 资源管理系统能够赋予其高伸缩性和高灵活性[16]。因为被指向和操作的资源是客户端通过 http 谓词来访问的;而每个 Web 资源都通过唯一的 URI 进行开放,并且通过标准的语法进行统一的表述,从而简化了整个系统架构,改进了子系统之间交互的可见性,也简化了客户端和服务器的实现。这非常适合异构、动态和分布特性并存的 Web 资源环境。

2 Web 资源管理系统功能模块

从第一节的研究可以看出,Web 资源管理系统的对象,其实就是"Web 信息资源"。Web 资源在 Web 2.0 时代呈现出来的新特性,也使得 Web 资源管理系统能够运用多维度的技术手段来研究 Web 2.0 信息资源的分布情况和利用规律。通过对信息资源的有效组织和存储,开发与利用,监控与配置,从而充分挖掘 Web 信息资源的内在价值,实现知识的创新需求并且为用户所共享、调用。基于此我们设计了 Web 资源管理系统,其功能模块包括四个部分:

- Web 资源抽象模块
- Web 信息管理模块
- 资源状态监控模块
- 资源数据管理模块

Web 资源管理系统通过抓取 Web 资源,进行统一的建模,描述和存储,同时提供资源的监控和管理功能,将资源封装成统一的 Web 接口供用户进行监控和调用。下面分别对四个模块进行详细描述:

1) Web 资源抽象模块:Web 资源抽象的功能即在资源接入 Internet 能力的基础上,按照统一的资源描述方式,将资源的内容和对象抽象成 RESTful 风格的 Web 资源。用户可以通过统一资源标志符(URI)对这些资源进行访问。这是由于大量 Web 资源是通过不同的管理系统分布式接入的,相互之间的同步和通信较为复杂;通过引入统一标识 URI 能够实现这些资源的"全局统一"调用,从而满足用户的开放性需求。

基于此,在本模块中"统一标识 URI 分配"的功能设计上,该模块引入了 GUID 技术。它是一种全局 ID 分配技术,主要应用在具有多个节点、多台机器的网络系统中,其总数达到了 2 的 128 次幂,随机生成两个相同 GUID 的可能性非常之小,在理想情况下任何计算机集群都不会生成两个相同的 GUID。因此注册到我们的资源管理系统中的所有 Web 资源都会分配到唯一

的 URI 标识。

光分配 URI 标识资源是不够的,实际上由于资源的复杂性和异构性,Web 资源抽象模块还需要描述注册资源本身的类型和功能,将功能标签作为资源的简要特性记录入系统,并结合 URI 分配机制生成 URL;用户进而可以用 Web 的方式(如通过 REST 风格的 Web 服务)获取资源数据、监控资源状态,有效地管理资源以及操作资源。简单来说就是"URI + 功能标签 = URL"的模式。比如说要表征一个"查看状态数值"能力的资源,Web 资源抽象模块生成的 URL 为 551abc76 - 7b3f - 433c - b234 - 571fce25yt33/value/Status. xml,其中 551abc76 - 7b3f - 433c - b234 - 571fce25yt33 是 GUID 形式的资源 URI,后面的 value/Status 是该资源功能的标签,两者结合成 URL,用户即可以通过此访问获取资源。

2) Web 信息管理模块:Web 资源的信息管理指的是资源数据和信息接入系统后,即资源数据接入到系统数据库以后,将其接入资源自身的信息上报给系统的资源信息管理模块;从而根据用户不同的请求,资源管理系统能够自动实现资源信息的注册、查询、配置和删除。

这部分的实现方式与大多数信息管理系统相似(如前文提到的 W4F),资源的注册、查询、配置和删除分别对应到数据层次的增、查、改、删的过程。本系统也以与传统管理系统相似的设计结构为基础,采用视图、模型、控件相分离(借鉴 MVC 模式),以及数据操作和业务逻辑相分离(采用三层架构)[17,18]的基本设计模式,进行设计和实现上的解耦。

3) 资源状态监控模块:资源的状态监控指的是管理系统对于接入的 Web 资源提供方,能够实时获取其运行状态信息、访问方式与路径、故障情况,并保存记录。通过监测相互连接网关的运行状态,管理系统可对网关的正常或故障情况进行进一步管理。通过获取 Web 资源实时地访问方式与访问路径(如 URL),可以提供网关设备的资源,供经认证的用户或第三方调用。

对于接入管理平台的资源,不仅仅需要资源标识符,还需要完整的 http URL 才能供用户直接访问,这包括了 URL 的主机地址。由于 Web 资源接入并非通过一个统一的平台,其接入主机地址(IP 和端口)可能是变化的,就引入了心跳机制应对变化的资源主机地址。心跳是指每隔一段时间,由接入的资源定时上传的一个数据包,包信息只含有接入的 IP 和端口。心跳监控的目的有两个:1. 确认资源处于活动状态;2. 当由于路由改变、资源接入网关的 IP 地址改变等原因产生时,平台设备管理系统能尽快得知。

通过心跳功能,平台能够获取网关的实时 IP 地址,进而方便地将网关设备封装成 Web 资源,供用户通过 URI 等方式访问。

4) 资源数据管理模块:资源数据管理指的是提供接入系统网关的资源数据、状态信息数据以及其他业务数据的接口;管理系统对上述资源数据实现分类、存储和索引的机制;向经认证的用户或第三方开放便捷易用、功能丰富的资源数据查询接口。

通过以上的管理系统架构,用户能够充分实现对 Web 资源的协同与集成化管理,保证资源的有序性和有效性;同时能够方便快捷地查询、获取信息,强化对 Web 信息资源的充分利用;并且通过开放接口访问资源,快速生成服务和应用,更加多元化地满足了用户的深层次需求。

3　Web 资源管理系统设计

3.1　系统整体架构设计

Web 资源管理系统的整体架构采用三层架构,由表示层、业务逻辑层(BLL)、数据访问层(DAL)组成。系统中还包括 DAL 调用的数据库(DB)。整体如图 1 所示。

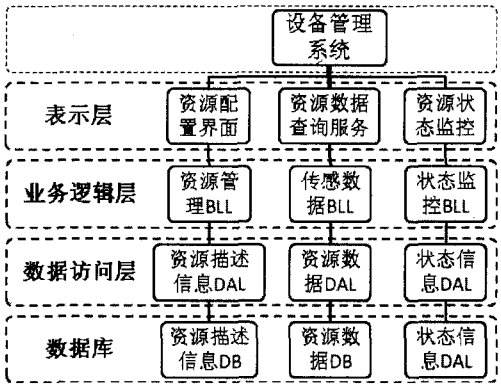

图 1　系统平台三层结构

图 1 中各层的分工如下:

1) 数据访问层:负责连接数据库,为业务逻辑层提供各类数据的添加、删除、查询、修改接口,封装对数据层的访问。为上层(即管理系统的业务流程和逻辑层)提供数据库操作接口,这些接口封装了数据库操作的细节(如查询方式等),但可以将数据库操作异常信息呈现,以供调试等方便。

2) 业务逻辑层:对于资源管理系统的各种操作制定详细的业务流程和逻辑,并调用数据访问层实现数据库的读取。主要的业务流程包括:资源注册、资源搜索、资源配置、资源移除、状态监控、历史数据存储等等。本层调用 DAL 层接入数据,然后对上层封装了逻辑实现的细节。比如:资源的注册逻辑,流程上的步骤有:(1)接入注册请求;(2)对请求是否合法进行认证;(3)验证注册信息格式的合法性;(4)进行资源-能力映射;(5)分配资源统一标识符;(6)进行数据库记录操作;(7)返回结果。综上所述,业务逻辑层把复杂的逻辑做了封装,对上层实现更好的服务。

3) 表示层:提供用户界面或其他形式的输入输出方式,调用业务逻辑层实现业务流程,并将结果反馈到界面。与上述层次等级规则相似,表示层调用业务逻辑层的接口,实现用户与系统的交互。表示层形式不单包括 UI(用户界面),还包括 Web Service 形式的 API 接口,供第三方应用调用。

3.2　NoSql 数据库方案的选用与设计

在诸多管理系统中,以 Sql 类数据库为代表的关系型数据库得到广泛应用。然而关系型数据库应用在本系统中却存在如下弊端:二维表用于映射复杂树状结构数据映射关系复杂、结构冗余;在资源描述格式发生变化时,可能引起大量二维表的修改和调整。基于此,本系统选用了非关系型数据库(NoSQL)中目前较为易用和流行的数据库:MongoDB 作为解决方案。其存储结构是树形的文档结构(类似 JSON 文档),这样文档的提供一种富数据模型,可与编程语言中的变量类型无缝连接,能够更方便地封装复杂的数据模型。值得一提的是,目前主流的社交网

站如人人,微博都用引进了 MongoDB 的解决方案。

在具体的设计上,在 MongoDB 数据库中我们使用数据集(Collection)代替了传统的关系型的数据表(Data Table),因为其数据集可以存储树状结构,存储能力相当于多个传统数据表,一个数据集就可以存储一种特定粒度的数据。比如在资源数据 Resource 中存在了很多的映射关系,使用二维数据表必须通过 table 来表述这些复杂的映射,但是采用 Collection 结构只需一个树状的数据集去实现,因此单个粒度的资源就可以对应一个数据集,这样的设计方式同样起到了明晰资源层次粒度的作用。

本系统共设计了 6 种数据集,依照 MongoDB 数据库树形存储的结构,数据集采用 JSON 结构存储:

WebResourceInfo	web 资源的描述信息
WebResourceStatus	web 资源所处的状态
HistoryData	web 资源上存储的历史数据
WebResourceConfigData	web 资源配置数据
WebResourceScenario	web 资源场景的信息
WebResourceIndexData	web 资源索引数据

以"资源描述实体"(WebResourceInfo)的数据集为例。其下属的字节点包括:资源 ID、数据类型(复合节点,包括数据单位、数据类型)、资源描述(复合节点,包括名称、标签、备注信息)、资源相关事件(复合节点,包括事件 ID、事件名称、事件参数列表)、资源相关操作(复合节点,包括操作 ID、操作名称、操作参数列表)。字节点信息都采用 JSON 文档格式存储在 WebResourceInfo 这一数据集中,如下所示:

```
{"ResourceID":"dfdfa-2342d145c787-fet6234-545ea",
"DataTypes":[
{"Unit":"%",
"datatype":"crowdity"}
],
"Description":{
"Name":"509NetworkSystemLoad",
"Tag":"net ability",
"Remark":""
},
"Events":[
{"eventID":"alert",
"eventName":"crowdity alert",
"eventParameters":[]
}
],
"Operations":[
{"opID":"show",
"opName":"show crowdity",
"opParameters":[]
}
]
}
```

3.3 数据接入层的设计

在使用 NoSQL 设计了资源管理数据库以后,数据处理层 DAL 需要能够对资源数据库中的数据集单元素进行选择、读取、写入、删除等操作,同时对上提供 BLL 层调用的功能接口供其调用。DAL 层的基类以及派生类的示意如图 2 所示。

我们设计的 DAL 层主要包括两类:

通用 DAL 类 PlatformDaoBase:提供一个通用的 DAL 层的基础框架,提供通用的增删改查基础架构,实体类基础架构以及 ORMapping 基础架构,该类面向所有的数据集。此层以通用为目标,面向数据库所有的数据表;作为核心框架只实现一次,通常不捕获任何异常。

专用 DAL 类 ResourceDAL,StatusDAL,ServiceDataDAL:基于通用 DAL 类的基础架构,这三个类为数据库中描述 Resource,Status 和 Service 三个主要的表都提供实体类和 DAL 类,实现一些专用的操作。此层以数据集为中心,为针对单数据集的各种可能的访问提供专用方法,可以作为单数据集的入口。专用 DAL 类的实体和成员定义完全向数据表对齐,函数定义中的参数和存储过程的参数完全匹配。

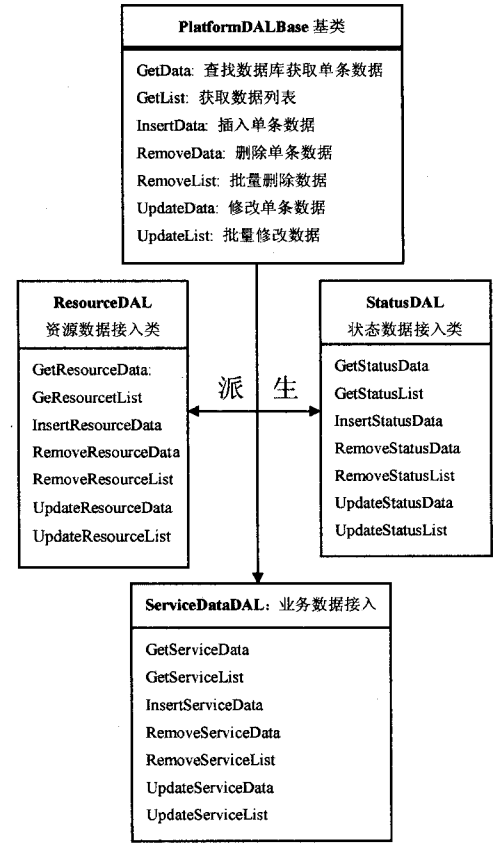

图 2 DAL 层的基类和派生类示意图

为了增强系统架构的鲁棒性,专用 DAL 类都使用了异常处理机制。具体的机制是:数据操作方法(函数)的返回值类型是字符串,表征数据操作的结果;在方法内部,使用 try...catch 捕获数据库操作过程中的异常,若捕捉到异常则返回异常信息,否则返回"数据操作成功"的信息。由于所有的数据接入类都采用相同的机制,异常信息放在基类中供其他类继承。

3.4 业务逻辑层的设计

业务逻辑,指资源管理系统执行各种资源管理业务时的流程和逻辑。业务逻辑层程序按照资源管理行为的流程,向上通过 RESTful Web 服务和表示层与系统中的其他实体进行通信,向下调用数据接入层的接口执行相关数据操作,是承上启下的一环。

业务逻辑层主要实现的业务流程包括:资源的注册、配置、

查询、状态监控以及 Web 资源历史数据的接入和查询等等。下面简要概述资源信息管理各业务流程的设计:

1) 资源注册:当有新的资源加入管理系统时,通过此业务流程在平台上注册资源,并将资源抽象成服务。在注册流程中,由管理系统为所有的服务分配全局唯一的业务 ID(基于 GUID 技术)。在此之后,用户可通过管理系统获取服务的业务 ID,进而调用相应的资源。

2) 资源配置:当已注册的资源变更了其描述信息时,通过此接口修改管理系统上原注册的资源配置信息。在配置过程中,管理系统将通过基于 XML Schema Description(XML 语义描述语言)形式的标准检查上传的配置信息是否合法,防止非法改动。

3) 资源查询:该业务提供对资源信息的查询索引接口。索引方式分为两种:(1)精确匹配,指通过对资源的业务 ID 进行索引;(2)模糊搜索,指对注册信息中的特定字段做简单的关键字搜索。

4) 资源删除:管理系统将资源的注册信息进行注销。业务流程涉及到相关数据的删除和相关索引删除的逻辑。

5) Web 资源的状态监控,是指系统能够实时获取资源的状态信息、访问方式与路径。通过获取资源实时的访问方式与访问路径(如 URL),可以暴露 Web 资源,供经认证的用户或第三方调用。

6) Web 资源历史数据的接入和查询:每隔一段时间,系统会将频繁被调用的资源数据存入管理系统服务器端的"历史缓存"数据库。这样,每当用户通过表示层调用业务逻辑层接口,发出查询请求时,业务逻辑层调用数据接入层查询数据库,并返回历史数据结果。

3.5 表示层的设计

根据资源管理系统的的需求,表示层采用 REST 风格的 Web Service 封装了资源管理和数据管理的业务流程接口。主要功能是将业务逻辑层提供的接口按照 REST 风格封装成 Web 服务,进而可以通过 HTTP 直接调用。

资源管理服务分为两种:资源服务和数据服务。

资源服务的 RESTful Web Service 接口如表 1 所示。

表 1　资源服务的 RESTful Web Service 设计

Uri	http 方法	描述
resource/registry	POST	新资源注册
resource/{svcid}	PUT	配置指定 ID 的资源
	DELETE	删除指定 ID 的资源
resource/{svcid}.xml	GET	查询指定 ID 的资源
resource/{svcid}/config	POST	配置指定 ID 的资源
resource/{svcid}/devices	POST	更新指定 ID 的资源列表
resource/{svcid}/devices/{DID}	DELETE	删除指定 ID 的资源下指定 ID 的资源
resource/all	GET	返回平台上所有已注册资源的列表及信息(XML 格式)
resource/all.json	GET	返回平台上所有已注册资源的列表及信息(JSON 格式)
resource/index	GET	返回平台记录的 ID_Index 表

在表 1 涉及的接口设计中,考虑到了两个因素:一是通过不同方式对资源进行索引,即通过两种资源标识:服务 ID(SvcID)和原有 ID(OID)索引;二是针对不同的资源操作定义不同的 HTTP 方法,包括:增添、修改操作定义为 POST(由于有些场景不支持 PUT 方法),获取操作定义为 GET,删除操作定义为 DELETE方法。

数据服务的 RESTful Web Service 接口如表 2 所示。

表 2　数据服务的 RESTful Web Service 设计

Uri	http 方法	描述
{id}/status	POST	资源上传状态数据
{id}/status.json	GET	返回指定 ID 的资源上传的状态数据(JSON 格式)
Uri	http 方法	描述
{id}/status.xml	GET	返回指定 ID 的资源上传的状态数据(XML 格式)
{id}/history	POST	资源上传历史数据
{id}/history.xml	GET	返回指定 ID 资源上传的最新 N 条数据
all/history.xml	GET	返回所有资源上传的所有历史数据
all_history	DELETE	删除所有资源上传的所有历史数据

4 Web 资源管理系统原型实现

基于资源管理系统设计,我们采用多种技术实现了 Web 资源管理平台。其中三层结构各个层次使用到的技术实现如下。

4.1 数据访问层的实现技术

资源管理系统的数据库使用 MongoDB 数据库,数据访问层调用 MongoDB 在.NET 环境中的驱动 MongoDB.dll 进行连接操作。类似 JDBC,该驱动定义了便捷的连接方式,指定数据库连接字符串和数据库名即可连接到数据库。

MongoDB 采用了类似面向对象语言的查询语言,而数据访问层使用的 MongoDB.dll 驱动将其进一步封装使得其支持基于.NET 的 LINQ 查询方式,与.NET 中常用集合类(如继承了 ICollection 接口的集合类)使用的查询方式无缝吻合。下面用一段代码展示增删查改操作的实例:

```
//建立 MongoDB 临时连接
using (MongoDBLinker DBL = new MongoDBLinker(_connectString,_dbName))
{
.....
//通过 LINQ 查询搜索记录
var linqQuery = DBL.GetCollection<Entity>(_collectionName).Linq().
Where(x.Number > 10 && x.Number<30);
List<Entity> entityList = linqQuery.ToList();
.....
}
```

数据接入层的增删查改操作与.NET 中常用的集合操作非常类似,使用了面向对象语言式的查询方式,查询性能和易用性都较好。

4.2 业务流程层的实现技术

在设备信息管理的所有业务流程中,都需要对上传的资源

信息进行格式验证。通信协议的格式是 XML[19]，故使用 XSD (XML Schema Definition，XML 语义定义)语言，对 XML 文件进行格式规范。.NET 平台提供工具 xsd.exe，可根据 XML 文件快速生成 XSD 文件，然后再手动编辑 XSD 文件，使其语义格式符合限定的要求。为使用程序代码实现验证功能，在业务逻辑层程序中创建了工具类 XmlValidator，下面用伪代码流程的形式描述验证 XML 的主要流程：

1) 在程序初始化时，导入作为语义验证规则的 XSD 文件。

2) 将作为入口参数传递进来的实体类序列化为 XML 文本。

3) 检查 XML 文本是否符合指定 XSD 文本所规定的语义。

4) 返回检查结果。

5) 其中，实现上的重点是 XML 文本的语义检查，步骤如下：

6) 由 XML 和 XSD 的字符串文本，创建 StringReader 实例。

7) 由 XSD 的 StringReader 实例创建含有 XML 语义的 XML 解析类 XmlReader 的实例。

8) 将含有 XML 语义的 XmlReader 转换成语义集合 XmlSchemaSet 的对象。

9) 建立 XML 文件的 XmlReader，载入 XML 文本的 StringReader 以及语义集合对象。

10) 用该 XmlReader 对象调用读取 XML 的方法，并通过 C# 的事件委托机制接收读取过程中的语义错误信息。

11) 若读取中发现语义错误则事件 handler 抛出异常原因，若未抛出异常则 XML 的语义合法。

4.3 表示层的实现技术

表示层需要实现 Web Service 的 RESTful 风格，因此采用.NET 中 WCF RESTful Web Service 4.0 的模板架构实现。这个模板是基于微软 WCF(Windows Communication Foundation)技术的，能够帮助程序员轻松创建和使用 RESTful service 的工具。在 WCF 4.0 中，WCF 的核心已经融入了 REST Starter Kit 中的 URL 引擎，在 WebGetAttribute 与 WebInvokeAttribute 已经可以支持 REST 的功能。

该模板的具体实现主要有以下两个关键点：

1) HTTP 方法的和交互格式的指定

在各个接口的代码实现中，使用 C#预编译头 UriTemplate 指定服务的路径和 HTTP 方法。使用预编译头 DataContract 指定交互格式(XML 或 JSON)，如下：

```
/// <summary>
/// 根据业务 ID 获取 Web 资源
/// </summary>
/// <param name = "Svc_ID" >资源的业务 ID </param >
/// <returns > </returns >
[WebGet(UriTemplate = "resource.xml? Svc_ID = {Svc_ID}", RequestFormat = WebMessageFormat.Xml)]
//UriTemplate:该接口调用的 URI
//RequestFormat:接口请求数据格式,此处是 XML
publicWebResourceGetResourceBySvcID(string Svc_ID)
{……}
```

2) 不同名称 REST 服务的注册

该 WCF REST 模板的实现中需要在 Global.asax 中进行服务注册。并且将结果发布在 IIS 上作为 WCF REST 服务供用户获取。

5 结 语

针对现今 Web 资源的异构、动态等特性对 Web 资源管理系统提出的新需求，本文设计了基于 REST 风格的 Web 资源管理系统，完成了资源描述定义、资源抽象、资源及其数据管理服务的研究和设计，将类型各异、实现功能不同的资源有效地抽象封装成服务，并实现对资源的注册管理、状态管理、管理数据历史数据的查询，以及为应用提供统一的 Web 调用接口。最后，用 C#实现了三层架构的资源管理系统程序。

参 考 文 献

[1] 彭巍，肖青. 物联网业务体系架构演进研究[J]. 移动通信，2010，34(15)：15-20.

[2] 王伟军，孙晶. Web 2.0 的研究与应用综述[J]. 情报科学，2007，25(12)：1907-1913.

[3] 崔慧超. Web 资源质量信息提取与管理技术的研究与实现[D]. 西南交通大学，2010.

[4] 王秋玲. 基于 RDF 的 Web 资源管理关键技术研究与应用[D]. 中国人民解放军信息工程大学，2006.

[5] Arnaud Sahugue, Fabien AZavan. Building intelligent Web applications using lightweight wrappers[J]. Data Knowledge Engineering, 2001，36(3)：283-316.

[6] 宋晖. 基于 Ontofogy 的 Web 信息抽取和信息集成的研究[D]. 上海交通大学，2004.

[7] Reis D D C, Golgher P B, Silva A S D, et al. Automatic Web news extraction using tree edit distance[C]//Proceedings of the 13th international conference on World Wide Web. 2004：502-511.

[8] Beil F, Ester M, Xu X W. Frequent Term-Based Text Clustering[C]. ACM，2002：436-442.

[9] 刘丹. 基于语义网的 Web 资源管理研究[D]. 南京信息工程大学，2011.

[10] 唐琳. 战略新兴产业——物联网的产生及发展[J]. 赤峰学院学报：自然科学版，2011，27(3)：30-32.

[11] 沈苏彬，毛燕琴，范曲立，等. 物联网概念模型与体系结构[J]. 南京邮电大学学报：自然科学版，2010，30(4)：1-8.

[12] 刘赛赛. 基于 Web Service 的智能数据中心研究与设计[J]. 科技传播，2010(19)：225.

[13] Dominique Guinard. VladTrifa Institute for Pervasive Computing[C]// ETHZurichand SAP Research CEC Zurich 8092 Zurich, Switzerland, Erik Wilde School of Information, UC Berkeley Berkeley CA 94720, USA, Architecting a MashableOpen World Wide Web of Things，2010.

[14] 王若梦，刘云，张振江，等. 基于事件驱动 SOA 的物联网管理平台研究[J]. 电信科学，2010，26(11)：80-84.

[15] Guinard D, Mueller M, Pasquier-Rocha J. Giving RFID a REST: Building a Web-Enabled EPCIS[C]. Interbet of Things，2010，1-8.

[16] Leonard Richardson, Sam Ruby. RESTful Web Services 中文版[M]. 徐涵，译. 北京：电子工业出版社，2008.

[17] Christian Nagel. C#入门经典[M]. 齐立波，等译. 5 版. 北京：清华大学出版社，2009.

[18] 夏晖，董平，苏力萍. 基于. Net 框架的设备管理系统的设计与实现[J]. 微计算机信息，2006，22(8-3)：110-111，57.

[19] 李禹生. XML 技术教程[M]. 北京：清华大学出版社，2009.