第16卷 第11期
2006年11月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol.16 No.11
Nov. 2006

XML在Web Services中的应用与研究

杨振宇，武 波
（西安电子科技大学 软件工程研究所，陕西 西安 710071）

摘 要：针对传统的Web应用开发所面临的问题，通过对比与分析传统的Web开发方式和基于XML技术的Web应用开发方式，提出了一种新颖的基于XML技术的Web开发方式。该基于XML技术的Web开发方式与传统的Web开发方式相比，其具有的最大优点是：能够解决Web Services中业务系统抽象、统一多种业务受理流程、分隔显示逻辑与业务逻辑、构建灵活的数据交换、提供具有方便快捷的二次开发、提高Web应用开发的效率，从而以适应越来越复杂的Web应用系统的开发需求。

关键词：XML；Web Services；XSL；数据交换；二次开发
中图分类号：TP311.52 **文献标识码**：A **文章编号**：1673－629X(2006)11－0250－04

Application and Research of XML on Web Services

YANG Zhen-yu，WU Bo
(Software Engineering Institute，Xidian University，Xi'an 710071，China)

Abstract：Focusing on existing drawbacks in the course of application and development，a novel Web development method based on XML technology is presented in this paper，resting on a comparison and an analysis of the classical Web development method with one based on XML technology. Most remarkable advantages of this development method suggested in this paper over the classical one，are to abstract and unify many kinds of business disposing process，separate display logic and operation logic，quickly deal with data exchange，supply the ability of secondary development and improve the development efficiency etc. Thereby，it can meet the requirements of the more and more complicated Web application.

Key words：XML；Web services；XSL；data exchange；secondary development

0 引 言

Web Services[1]是一种新的Web应用程序的分支，它们是自包含、自描述、模块化的应用，可以在网络（通常是Web）中被描述、发布、查找以及通过Web来调用。它是基于网络的、分布式的模块化组件，它执行特定的任务，遵守具体的技术规范，这些规范使得Web服务能与其他兼容的组件进行互操作。它可以使用标准的互联网协议，将功能体现在互联网和企业内部网上。

Web Services能够统一地封装信息、行为、数据及商务流程，把应用程序改变成可重用的和柔性的组件。普通的Web Services技术在使用的过程中也暴露出了自身的某些不适应。比如，当面对一个具有二次开发工具的业务系统时，各业务如何方便、快捷地提供给用户？如何让开发出来的系统能够同时拥有软件的可移植性、可扩展性、灵活性？如果使用通常的Web Services技术，人们会发现它不适应不断动态开发业务的业务系统，因为使用通常的Web Services技术编程的工作量会很大，而且维护起来不方便。所以，文中提出基于XML[2]的Web服务，它能够利用XML schema来抽象业务，屏蔽SQL语言与XML[2]数据访问以及普通的文件访问之间的差异，提供方便、快捷的二次开发能力。

针对具体行业的业务处理系统的主要特点是业务的多样性、不确定性和可扩充性，即该行业随着时间的推移，所需要增加的业务是多样的，不可预知的。因此，抽象、统一多种业务受理流程，提供方便的二次开发能力是业务系统的关键所在。文中就在陕西邮政185客户服务中心业务系统中应用XML[2]来解决其关键问题，并进行阐述和研究。

1 XML在Web Services系统中的应用与研究

经过调研，笔者认为邮政客服业务可以归纳为以下5类：咨询业务（如邮政资费咨询），查询业务（如邮编查询），录入业务（如投诉），预定业务（如报刊预定），网上业务（如EMS查询，是直接连接到邮政综合网EMS查询网站）。由于咨询类业务相对比较固定、简单，所涉及的操作多为查询，插入一条咨询记录，网上业务与本地应用逻辑没有关系，故文中重点讨论查询、录入、预定3类业务。

收稿日期：2006－03－03
作者简介：杨振宇（1979－），男，河南鹿邑人，硕士研究生，研究方向为J2EE技术、Web应用和XML技术；武 波，教授，硕士生导师，研究方向为计算机软件理论与应用。

为方便以下论述,定义:

1) 业务信息表:指在该业务受理之前必须要有的记录该业务相关信息的数据库表,如报刊预定业务受理之前,必须要有报刊数据库以供查询;

2) 业务订单表:指在该业务受理之前不必要有记录,但在业务受理结束要插入记录的数据库表,如投诉业务,在业务受理之后才向投诉记录数据库表中插入一条投诉记录。

经过对上述 5 类业务进行分析,按其所需的数据库表将其抽象为以下 4 类:

(1) 只需要有业务信息表,而不需要业务订单表的业务:如咨询、查询业务。

(2) 只需要有业务订单表,而不需要业务信息表的业务:如录入业务。

(3) 既需要有业务订单表,又需要业务信息表的业务:如预定业务。

(4) 既不需要有业务订单表,又不需要业务信息表的业务:如网上业务。

1.1 设　计

JSP 技术规范中给出了一种常用 JSP 开发 Web 应用的方式,称之为 JSP + JavaBeans 模型。在这一模型中,JSP 页面独自响应请求并将处理结果返回给客户,所有的数据通过 JavaBean 来处理,JSP 实现页面的表现。该模型实现了页面表现和业务逻辑相分离。然而使用这种方式就要在 JSP 页面使用大量的 Java 代码,当需要处理的业务逻辑很复杂时,这种情况会变得非常糟糕。大量嵌入式代码使整个页面程序变得异常复杂。对于前端界面设计的网页开发人员来说,这简直是一场噩梦。

根据实际的业务要求和提供方便、快捷的数据交换、二次开发能力,文中决定采用以 XML 为核心的系统模型。图 1 所示是如何将 XML 数据作为应用的核心。

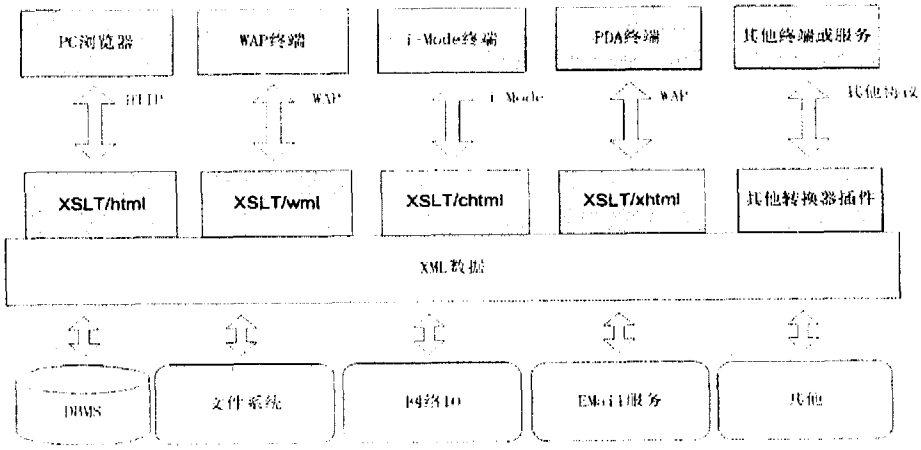

图 1　XML 屏蔽各终端数据

说明:最下层是应用的数据源,包含各种可能的数据介质,通过系统的 Model 的处理,将不同介质的数据源中的数据转换成统一定义的 XML 数据;中间的浅灰色部分是符合应用服务 XML 格式定义的 XML 数据,应用服务的 XML 格式定义是依照应用的处理流程,将每个独立的步骤中所需的数据结构通过 XML 来定义;产生的 XML 数据将通过 View(表示层)的转换,产生服务流程各个状态的页面,这里通过中间上层深灰色的部分描述了 View 的几种转换方式,为了说明一个 XML 数据可能交给不同的 View 处理;View 可以将 XML 转换为一种显示输出,比如 HTML 或者 WML[3],这就是一个服务的一个状态的最终的输出。

由上可见,以 XML 为核心不仅可以屏蔽多种客户终端,而且可以方便地扩充对未来客户终端的支持。

1.2 实　现

1.2.1 作为业务定义语言

为使系统只与业务种类有关,而与具体业务无关,方便提供二次开发能力,系统需要用某种手段对具体业务进行描述和分类。经过分析比较,鉴于 XML 语言强大的数据描述、交换能力,以及其与平台无关,系统采用 XML 对具体业务进行描述。以报刊投诉业务为例其描述文档如下:

```
<Business>
<BusinessNo>000101</BusinessNo> //该业务的编号
<BusinessName>投诉业务</BusinessName> //该业务的名称
<BusinessPresent /> //对于某些需要特殊显示的业务,此处可以指定其显示的 XSL 文件名。
<BusinessProduce /> //对于网上业务,此处直接指定其连接的网址。
<BusinessMessage /> //用于业务受理和业务维护中,显示操作信息
<BusinessQueryResultField />
<BusinessOrderTable>ORDER_TOUSU_000101</BusinessOrderTable> //业务的订单表名
<BusinessOrderPrice />
<BusinessOrderField> //该业务订单表中各字段
    <Field name="MY_BNO" unit="" type="STRING" nullable="false" length="10" lable="被投诉人工号" value="" />
    <Field name="MY_NAME" unit="" type="STRING" nullable="true" length="20" lable="投诉人姓名" value="" />
    <Field name="MY_TEL" unit="" type="STRING" nullable="true" length="15" lable="投诉人电话" value="" />
    <Field name="MY_REASON" unit="" type="STRING" nullable="false" length="300" lable="投诉原因" value="" />
    <Field name="MY_TIME" unit="" type="TIME" nullable="false" length="14" lable="投诉日期" value="" />
```

</BusinessOrderField>
<BusinessProduceField>
<Field name="MY_BNAME" unit="" type="STRING" nullable="true" length="20" lable="被投诉人姓名" value="" />
</BusinessProduceField>
</Business>

根据上述XML文档的描述,系统可以识别某一业务所属业务种类,事后处理和相关数据库表等信息。利用XML对具体业务进行描述,对于以后具体业务的显示,页面信息的收集,数据交换、操作都有非常重要的意义。

1.2.2 作为层间数据交换对象

与前一功能相似,XML文档既可以作为一种业务定义提供给客户端,同时也可以作为系统内部的层间数据交换的载体。接入系统与各个模块之间的数据交换都是通过XML文档完成的。这种设计,最大限度地解除了系统各个模块之间的耦合关系,允许用户能够自由地重载系统的各个模块,创建一个内核可以被更改的开放式系统。用户只要能够保证输入输出的XML文档的格式(在系统的各个模块的输入输出之前,都有相应的Schema的验证,这样能够有力地保证系统的可靠性,而且这一部分代码是用户不能修改的,通过在方法名前加注final关键字实现),就能够按照应用的需要,量身订制各个模块的处理逻辑。

对上述4类业务从业务受理流程角度进行分析,可以抽象为以下两种:

1) 对业务信息表的查询动作;

2) 对业务订单表的订单插入动作。

为使系统在数据库操作方面与业务无关,系统使用XML的Schema技术[4]定义了一组支持数据操作的语法集合。利用该语法集合,系统可以获得对哪个数据库表,哪个字段进行何种操作等信息,从而达到真正地与业务无关。这些语法通过一份Schema[4]文档来体现,它规定了需要用户组装的XML数据库操作文档的类型、元素、属性等语法内容,以及各种元素之间的相互关系,同时借助Schema的标签传达了无二意的语意信息。在该Schema文档中,共定义了4个级别的类型,它们是:

* Filed: 定义了一个数据库字段的抽象描述,包括类型、长度、值等信息。

* Fields: 定义了一组Field的集合,这些Field可能同时表达一个语法单元,例如,一个WhereFields就是一个Fields类型,它表示了一个where语句中的字段集合。

* Segment: 定义了一个数据库操作,如QuerySegement,InsertSegment,它们分别表示查询请求和插入请求。一个Segment由众多的Fields类型的子元素组成,比如一个InsertSegment元素可能包括WhereFields,InsertFields等子元素。

* Transaction: 定义了一组操作集合,这组操作作为一个事务而被执行。它是一个客户端提交请求的最小单元。

按照上述Schema文档语法规定组装的XML数据库操作流程如图2所示。

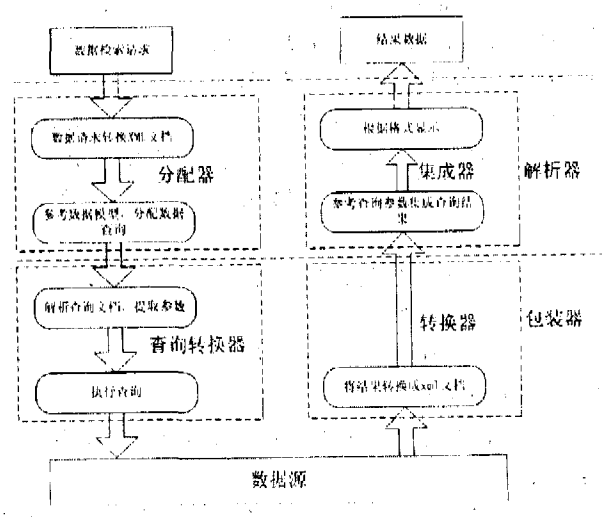

图2 业务系统运行数据流图

系统给用户呈现出一个检索页面,当用户提出数据的检索请求时,系统接收到用户的检索条件后,将做以下的处理:

1)业务系统将用户的检索条件组织成XML查询文档,提交给解析器(Resolver);

2)分配器(Dispatcher)参考系统中的数据模型,确定用户集成视图到各个数据源的映射,并将基于集成视图的查询转换为基于各个数据源的子查询;

3)各个数据源的包装器(Wrapper)收到子查询后,查询转换器(Switcher)解析查询文档,提取出其中的查询参数,并组织成各数据源能识别的查询语句;

4)各个数据源执行查询语句,并返回结果给包装器(Wrapper);

5)结果转换器(Switcher)将子查询结果转换为标准的XML文档,同时提交给集成器(Integrater);

6)集成器在收到各个包装器返回的子结果后,结合用户提交的查询参数,将子结果整合成为一个完整的结果XML文档,返回给用户。

整个处理过程中,各个模块中传递的唯一参数就是相应的XML文件,这样可以最大限度地解除各个层次之间的耦合关系,并且允许对逻辑处理部件的重载,构建了一个可以更改核心的开发式系统。

1.2.3 方便二次开发

为了避免每开发一项新业务,都必须为之建立相应显示页面,系统为每类业务提供一组默认的XSL转换文档,根据该业务的XML描述文档显示该业务相关页面,其显示原理参见参考文献[5]。当某具体业务触发查询动作时,系统根据该业务的XML描述文档收集页面输入参数,组装成XML数据库操作文档,以参数的形式传递给后台数据库操作引擎去解析,完成操作数据库动作。

对于下订单动作,由于其涉及到的客户信息、订单信息、日志、事后处理等其他支撑数据较多、较复杂,故系统

利用 XML 的 schema 技术[4]为其专门定义了一组支持订单操作的语法集合(schema 文档),该文档部分内容如下:

```
<AttributeType name="name" dt:type="string"/> //业务订单表字段的属性定义
……
<ElementType name="Field"> //业务订单表字段定义
<attribute type="name" required="yes"/>
……
</ElementType>
<ElementType name="BusinessOrderField" content="eltOnly"> //业务订单表字段集合定义
<element type="Field" minOccurs="1" mxaOccurs="*"/>
</ElementType>
<ElementType name="KeyValue" dt:type="ui4"/> //该业务信息表中记录的主键,如所预定的报刊的主键
<ElementType name="BusinessNo" dt:type="ui4"/> //涉及的具体业务编号
<ElementType name="item" content="eltOnly"> //订单子项定义
<element type="BusinessNo" minOccurs="1" mxaOccurs="1"/>
<element type="KeyValue" minOccurs="0" mxaOccurs="1"/>
<element type="BusinessOrderField" minOccurs="1" mxaOccurs="1"/>
</ElementType>
<ElementType name="items" content="eltOnly"> //订单子项集合的定义
<element type="item" minOccurs="1" mxaOccurs="*"/>
</ElementType>
<ElementType name="UserID" dt:type="string"/> //客户编号的定义
<ElementType name="Order" content="eltOnly"> //订单根元素的定义
<element type="UserID" minOccurs="1" mxaOccurs="1"/>
<element type="items" minOccurs="1" mxaOccurs="1"/>
</ElementType>
```

以预定的报刊为例,其符合上述 schema 文档语法的订单文档如下:

```
<Order>
  <UserID>AUTO10000112</UserID> //下订单的客户 ID
  <Items> //该客户预定的多个订单子项
  <item> //第一个订单子项,例如订阅人民日报
    <BusinessNo>000023</ BusinessNo > //业务编号
    <KeyValue>1</KeyValue> //该业务信息表中记录的主键,即所预定的报刊的主键
    <BusinessOrderField> //该业务订单表中各字段内容,即填写的预定内容
    <Field name="MY_NO" type="STRING" lable="报刊号" value="kh1254" />
    <Field name="MY_NAME" type="STRING" lable="报刊名" value="光明日报" />
    <Field name="MY_MAN" type="STRING" lable="订报人" value="订报人" />
    <Field name="MY_FEE" type="FLOAT" lable="订报金额" value="2" />
    </BusinessOrderField>
  </Item>
  <item> //第二个订单子项,例如订购一本杂志
    ……
  </Item>
  </Items>
</Order>
```

同时,系统为订单处理提供了专门的后台订单处理引擎,该引擎首先利用上述 schema 文档对传递的订单文档进行验证,验证通过后进行解析和相关数据库操作。利用 XML 进行数据库操作,使系统最大程度地与业务无关,对以后二次开发做到不需重新编写代码、编译程序具有重大意义。

在分析、抽象业务种类和业务受理动作之后,其相应的二次开发流程也应运而生。二次开发时,根据页面输入,创建业务的 XML 描述文档,再根据 XML 描述文档创建相应的数据库表。这样,在受理一个新创建的业务时,系统解析其 XML 描述文档,按上述 4 种业务受理模式直接进行查询、下订单操作,而不需再编写代码,再重新编译程序。

2 结束语

在陕西邮政 185 客户服务中心 Web Services 系统的研发中,笔者针对系统的业务多样性、不确定性和可扩充性特点,采用基于 J2EE 标准的 B/S 体系结构,利用 XML 强大的数据描述、交换能力方便地解决了抽象、统一多种业务受理流程,数据交换,提供方便的二次开发能力的问题,同时该系统在陕西邮政 185 客户服务中心工程运行中经受住了实际考验。

参考文献:

[1] Christensen E, Curbera F, Meredith G, et al. Web Services Description Language (WSDL) 1.1 W3C Note[Z]. 2001.

[2] Bray T, Paoli J, Sperberg-McQueen C M, et al. Extensible Markup Language (XML) 1.0[M]. 2nd ed. [s.l.]: W3C Recommendation, 2000.

[3] Engelschall R S. Website META Language 2.0.9[M]. [s.l.]: W3C Recommendation, 2002.

[4] Sperberg-McQueen C M, Thompson H. XML Schema 1.0 [M]. 2nd ed. [s.l.]: W3C Recommendation, 2004.

[5] Adler S, Berglund A, Caruso J, et al. Extensible Stylesheet Language (XSL) Version 1.0[M]. [s.l.]: W3C Recommendation, 2001.