基于Web的数据采集

唐翔弘[1] 汪林林[1] 文 展[2]

(重庆邮电学院软件学院 重庆400065)[1] (成都信息工程学院电子系 成都610041)[2]

摘 要 在本文中,将讨论使用标准Web技术——HTML、XML和Java——开发的一种基于Web的数据采集方法。万维网是到目前为止世界上最丰富和最密集的信息来源,但其结构使它很难用系统的方法来利用信息。本文描述的方法主要是通过设定目标锚并利用XALAN技术在源信息里获取目标数据,最后生成XML文档。这种方法可使那些熟悉Web最常用技术的开发人员能快速而便捷地获取他们所需的以Web方式发布的信息。

关键词 数据采集,锚,XML,XSL变换

Data Gathering Based on Web

TANG Xiang-Hong[1] WANG Lin-Lin[1] WEN Zhan[2]

(Software Institute, Chongqing University of Post and Telecommunication, Chongqing 400065)[1]

(Electronic Department, Chengdu University of Information and Technology, Chengdu 610041)[2]

Abstract This paper mainly discusses how to use the standard Web technology ——HTML, XML and JAVA to develop a way of data Gathering based on Web. The internet is the most abundant and dense source to get the information. But it is hard to use a systematic way to utilize information due to its structure. The way described in this paper mainly introduces how to setup a anchor and utilize XALAN technology to get the end data from the resource information. This way can make those developers who are familiar to the Web technology get the information published by Web style quickly and conveniently.

Keywords Data gathering, Anchor, XML, XSL transformation

1 引言

在信息时代,快速成长起来的万维网使各种各样的公用信息被大量分发。不幸的是,尽管作为信息主要载体的HTML提供了一种方便的向读者呈现信息的方法,但它可能并不是一个很好的可以从中自动抽取与数据驱动的服务或应用程序相关的信息的结构。

人们已经尝试了多种方法来解决这个问题,但大多数方法会因为以下两个原因变得不切实际:首先,它们需要开发人员花时间去学习一种并不通用的针对Web的专业查询语言,其次,它们还不够强大到能方便的对目标Web页面进行简单的更改。

本文就将从标准Web技术入手,使用流行的Java和XML技术来开发基于Web的数据采集。这种方法即使不比其它专用方法更强大,也和其它方法不相上下,并且对于那些已经熟悉Web技术的人来说,只需要付出很少的努力就可以收到很好的效果。此外,此方法用到的工具例如:Tidy库和XALAN,XERCES库都比较通用,可以从各专业网站下载使用。

2 背景技术

HTML通常是一个很难用程序手段处理的媒体。Web页面中的大多数内容描述与数据驱动的系统无关的格式编排,并且,由于要动态添加标题以及编写其它服务器端脚本,所以文档结构可能在每次连接到页面时都需要进行更改。又因为所有Web页面主要部分的格式编排不合理,所以使问题变得更为复杂,其结果是现在的Web浏览器在进行HTML语法分析时非常不严谨。

尽管存在这些问题,但是HTML特殊的组织格式在数据采集应用方面仍然具有优势。您所感兴趣的数据通常可以用HTML树中深度嵌套的单个〈table〉或〈div〉标记隔离开来。这使得抽取过程可以集中在文档的一小部分内执行。在缺少客户机端脚本的情况下,只有一种定义下拉菜单和其它数据列表的方法。HTML的这些方面允许我们在一旦拥有可用格式的数据时能集中精力于数据抽取。

这里描述的数据采集技术的关键是把现有的Web页面转换成XML,或转换成XHTML可能更适当,并使用众多工具中的一小部分来处理XML结构的数据,以检索出适当的数据。幸好有一个解决方案可以改正HTML页面设计的薄弱之处。Tidy是一个免费使用的产品,可用于改正HTML文档中的常见错误并生成格式编排良好的等价文档。还可以使用Tidy来生成XHTML(XML的子集)格式的这些文档。

3 具体方法概述和实例

我们用示例的方式来介绍数据抽取的方法。假设我们有兴趣跟踪几个月以来每天不同时间测得的某地的温度。如果没有现成的软件用于报告此类信息以满足我们的需求,我们仍然拥有从众多公共网站收集此类信息的机会。

唐翔弘 硕士研究生,从事空间数据库存储管理器的研究和开发网络信息处理。**汪林林** 教授,研究领域:计算机网络、计算机数据库系统、计算机GIS系统、计算机专家系统。**文 展** 讲师。

3.1 概要说明抽取过程

只需要很少的几个步骤，我们就可以拥有一个收集我们信息的合适而可靠的系统。这里列出这些步骤是为了提供该过程的简要概述，具体的流程见图1所示。

1. 标识数据源并把它映射成 XHTML。
2. 查找数据内的引用点。
3. 将数据映射成 XML。
4. 合并结果并处理数据。

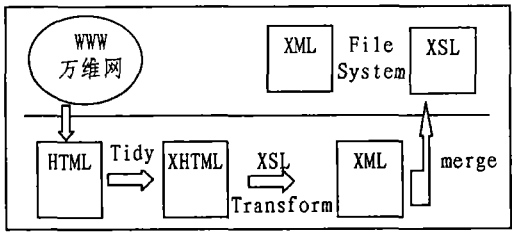

图1

3.2 获取 XHTML 格式的源信息

为了抽取数据，当然需要知道可以在哪里找到它。在大多数情况下，源信息是显而易见的。比如要从网易收集文章的标题和 URL，我们将使用 www.163.com 作为我们的数据源，如果要从网易上抽取旅游信息，我们会使用 travel.163.com 作为数据源。本文将以抽取某网站上的天气信息为例子，具体展现如何从繁杂的数据源中抽取对自己有用的信息。

在考虑信息源时，牢记以下这些要素非常重要：信息源是否是在可靠的网络连接上生成可靠的数据？信息源从现在起将存在的时间长度；信息源的布局结构要稳定。

我们寻求能够在动态环境下工作的健壮的解决方案的过程中，在抽取可用的最可靠和最稳定的信息源时，我们的工作将是最简单的。

一旦确定了信息源，我们在抽取过程中的第一步就是将数据从 HTML 转换成 XML。我们可以通过构造一个由静态函数组成的 Java 类（例如：GatherHelp）来完成这一任务以及其它与 XML 相关任务。在本例中，我们使用 Tidy 库提供的函数在 GatherHelp.tidyHTML()方法中执行转换，转换的具体细节请参考 Tdiy 库提供的 API 说明文档，在这里就不在详细的介绍了。这个方法接受 URL 作为一个参数并返回一个"XM 文档对象"作为结果。当调用此方法或任何其它与 XML 相关的方法时，需要仔细检查是否有任何异常。

3.3 查找数据的引用点

需要注意的是，无论是在 Web 页面还是源 XHTML 视图中的绝大多数的信息都与我们完全无关。我们接下来的一个任务是在 XML 树中找出一个特定区域，我们可从中抽取我们的数据而无需关心外来信息。对于更复杂的抽取，我们可能需要在单个页面上找出这些区域的若干实例。完成这一任务的最简单的办法通常是，首先检查 Web 页面，然后使用 XML。只需要看一下页面，就可以知道我们正在查找的信息位于页面的中上部区域中。即使对 HTML 的熟悉程度非常有限，也很容易推断出我们正在查找的数据可能都包含在同一个<table>元素下，并且这个表可能总是包含象"温度"和"湿度"这样的字，无论当天的数据可能是什么。

记下我们观察到的内容，现在要考虑页面所生成的 XHTML。搜索"温度"和"湿度"的文本（如图2所示）说明该文本确实在一个包含我们所需的所有数据的表中。我们将把该表作为引用点或锚的起点。

```
<table cellspacing = "0" width = "100%" cellpadding
    = "0" border = "0">
<table cellspacing = "0" width = "100%" align
    = "center" border = "0" cellpadding = "0">
<tr>
    <td>          锚      数据值
      <font face = "arial" size = "2">
      <b>温度</b>
      </font>
    </td>
    <td>
        <font face = "arial" size = "2"> 34° </font>
    </td>
    <td>br</td>
    ......
</tr>          锚
<tr>
    <td>
      <font face = "arial" size = "2">
      <b>湿度</b>
      </font>
    </td>                    数据值
    <td>
        <font face = "arial" size = "2"> 57% </font>
    </td>
    <td>br</td>
    ......
</tr>
......
```

图2

现在，我们需要找到这个锚的方法。因为我们正准备使用 XSL 来转换我们的 XML，所以可以使用 XPath 表达式来完成这个任务。首先，我们可以考虑使用以下这个普通的表达式：/html/body/center/table[6]/tr[2]/td[2]/table[2]/tr/td/table[6]。这个表达式指定了从根<html>元素到锚的路径。这个普通的方法将导致我们对页面布局的修改容易遭到破坏。较好的方法是根据周围的内容指定锚。通过使用这个方法，我们把 XPath 表达式重新构造成：//table[starts-with(tr/td/font/b,'温度')]

3.4 将数据映射成 XML

拥有这个锚，我们可以创建实际抽取数据的代码。这个代码将以 XSL 文件的形式出现。XSL 文件的目的是标识锚，指定如何从锚获取我们正在查找的数据（以简短跳跃的方式），并且用我们所需的格式构造一个 XML 输出文件。这个过程实际上比想象的要简单得多。下面给出了将执行这个过程的 XSL 代码，这些代码还可以作为一个 XSL 文本文件获取。其中<xsl:output>元素仅告诉处理器我们希望的变换结果是 XML。第一个<xsl:template>建立名为<xsl:apply-templates>的根元素以搜索锚。第二个<xsl:template>让我们只匹配需要匹配的内容。最后那个<xsl:template>在 match 属性中定义锚，然后告诉处理器跳到锚所指定的位置，也就是我们尝试采集的温度与湿度数据。当跳到我们的目标数据后，则根据 select 所对应的目标树的级数以锚为起点开始往下对应，并取其值。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/
    1999/XSL/Transform">
    <xsl:output version="1.0" indent="yes" encoding="UTF-8"
     omit-xml-declaration="no" method="xml"/>
    <xsl:template match="/html">
      <RESULT>
        <WEATHER>
          <xsl:apply-templates/>
        </WEATHER>
```

```
    </RESULT>
  </xsl:template>
  <xsl:template match="text()"></xsl:template>
  <xsl:template match="table[starts-with(tr/td/font/b,'温度')]">
    <TEMPERATURE>
      <xsl:value-of select="tr/td[2]/font"/>
    </TEMPERATURE>
    <HUMIDITY>
      <xsl:value-of select="tr[2]/td[2]/font"/>
    </HUMIDITY>
  </xsl:template>
</xsl:stylesheet>
```

当然，只编写 XSL，作业将不会完成。我们还需要一个执行转换的工具。因此，我们得自己写几个函数对 XSL 进行语法分析并执行这个转换。执行这些任务的方法分别实现解析 XML 文档和根据 XSL 代码转换输出结果 XML 文档。

3.5 合并与处理结果

如果只执行一次数据抽取，那么任务现在已经完成了。但是，我们并不只是想知道某一时刻的温度和湿度，而是要知道若干不同时刻的温度和湿度信息。现在，我们需要做的是反复执行抽取过程，把结果合并到单个 XML 数据文件中。通过定义一个方法比如：mergeXML()，它通过将新采集的数据以 XML 文档对象的模式合并到原来已经存在的目的 XML 数据文件中，这样我们就可以把在当前抽取过程获得的数据合并到包含以前抽取数据的档案文件中。这样，我们可以看到例如下例中连续四天抽取的温度、湿度信息，它是每天运行一次抽取函数并合并到已经存在的 XML 文档中，以这种方式连续运行四天后的结果 XML 文档内容如下：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<RESULT>
  <WEATHER Retreive = "2003-9-12 12:56:34">
    <TEMPERATURE>34</TEMPERATURE>
    <HUMIDITY>57%</HUMIDITY>
  </WEATHER>
  <WEATHER Retreive = "2003-9-13 12:56:23">
    <TEMPERATURE>37</TEMPERATURE>
    <HUMIDITY>48%</HUMIDITY> </WEATHER>
  <WEATHER Retreive = "2003-9-14 12:56:12">
    <TEMPERATURE>35</TEMPERATURE>
    <HUMIDITY>63%</HUMIDITY>
  </WEATHER>
  <WEATHER Retreive = "2003-9-15 12:56:37">
    <TEMPERATURE>30</TEMPERATURE>
    <HUMIDITY>52%</HUMIDITY>
  </WEATHER>
</RESULT>
```

结束语 在本文中，我们已经描述并证明从目前存在的最大信息来源——万维网抽取信息的强壮方法的基本原则。我们还讨论了能够使任何 Java 开发人员花最少的精力和具备最少的抽取经验就可以开始他们自己抽取工作所必需的编码工具。这种方法最大的优点就是通过明智的选取可靠的数据源以及在这些数据源中选取与内容相关但与格式无关的锚，可以使您拥有一个维护成本低廉、可靠的数据抽取系统。并且，根据经验级别和要抽取的数据量，您可以在短时间内就能安装与运行它。

另外需要提到的是，尽管本文以一个采取天气信息的例子来展示这种方法，但是只要稍微修改一下 XSL 文件，就可以为其它的数据采集项目服务。

参 考 文 献

1 Marchal B. XML 示例程序导学. 北京：清华大学出版社，2002

2 Becker D. 使用 java 编程利用 Web XML 数据. IBM developerWorks，2002. 8

3 Kankure P. 使用 Java 和 XSLT 生成动态 Web 页面. IBM developerWorks，2001. 4

4 Darugar P T. 在 Java 中使用 DOM 和 XPath 进行有效的 XML 处理. IBM developerWorks，2001. 12

(上接第48页)

络节点，构成部件内存，如在 Windows 环境下由各种原语形成的动态链接库可作为部件内存。原语对于主动网络节点是非常重要的，它不仅决定着封包程序的功能范围并能防止对节点产生有意或无意的破坏。Active IP 的原语可分为四类：

(1) 环境资源访问：用于查询网络节点地址、连接状态、路由表、主机时间等；

(2) 数据报处理：用于对数据报本身数据更新操作。

(3) 控制操作：用于数据报的创建、发送、复制、放弃等；

(4) 节点存储空间的访问：用于访问封包程序所执行的临时空间。

Active IP 的原语决定着封包程序的功能范围，并且能够防止对节点产生有意或无意的破坏，因此它对于主动网络的节点是非常重要的。封包程序的简洁性和执行效率都会受到这些原语的影响。

采用上述的方案，将主动技术与现有网络有机地结合起来，可以使主动结点或局部主动网络更好地融入当前的网络中，从而在现有的网络上实现主动方案，为用户提供可定制环境，以满足用户的各种需求。采用主动 IP 选项方案后，传统的结点分不出主动包和被动包，也就是说它可以对所有的数据包进行存储和转发，主动结点则可区分主动包和被动包，采取不同的处理方式。

结束语 主动网络是一种崭新的网络结构，它采用的主动技术涉及到编译技术、操作系统、网络技术等各个方面。它使得网络可以动态地配置和动态控制，极大地提高了网络的性能并增加了网络的灵活性和扩展性，并为宽带网络的发展提供了广阔的前景。目前虽然还没有实用的产品推出，但是它得到了广泛的关注，现在正在对其关键性的技术如路由、资源分配、安全性、开发语言和平台等进行研究，可以相信，主动网络已经开始改变了传统网络的概念，它对未来技术的影响将起非常重要的推动作用。

参 考 文 献

1 Wetherall D, Legedza U, Guttag J. Introducing New Internet Services: Why and How. IEEE Networks Magazine, May/June, 1998

2 Moore O T, Nettles S M. Towards Practical Programmable Packets: [Technical Report MS-CIS-00-12University of Pennsylvania]. May 2000

3 Schwartz B, et al. Smart Packets: Applying Active Networks to Network Management. ACM Transanctions on Computer Systems, February 2000, 18(1): 67～88

4 Di Fatta G, Lo Re G. Active Networks: An Evolution of the Internet. In: Proc. of AICA2001 - 39th Annual Conference, Cernobbio, Italy, 19-22 Sept. 2001

5 Mills D L. On the accuracy of Clocks Synchronized by the Network Time Protocol in the Internet System. ACM Computer Communication Review, Jan. 1990, 20(1): 65～75

6 Brunner M, Stadler R. Service Management in Multi-Party Active Networks. IEEE Communications Magazine, Special Issue on Active and Programmable Networks, March 2000, 38(3)

7 Di Fatta G, Gaglio S, Lo Re G, Ortolani M. Adaptive Routing in Active Networks. IEEE Openarch 2000, Tel Aviv Israel 23-24 March 2000