China-pub.com
下载

第7章 用户界面

- 怎样把XML数据表示给用户？
- 什么技术是有用的？
- 怎样使用这些技术？

数据库最终是要给人们来用的。Web发送的用户界面特别适合XML数据库。Web用户界面可以被用作客户端来为终端用户表示数据，或者可以用在应用程序管道的中间阶段以检查两个应用之间传送的XML数据流。在任一种情况下，用户都可能需要查询或浏览数据。

本章描述怎样使用XML/XSL为普通Web浏览器（例如Microsoft Internet Explore）从数据库中生成HTML表单和报表。为了解析XML和生成通用或自定义报表，这里还将讲述一个（Java）客户端结构。

7.1 概述

通过Web来表示数据的两种方法分别是HTML和Java applet。XML与XSL样式表结合可以生成HTML报表，该报表可以通过Web浏览器来查看。使用HTML表单可以创建数据输入和查询表单，并且使用嵌入在HTML页面中的Javascript可以实现简单的交互功能。对于更复杂的用户界面来说，可以使用Java applet来解析（和/或生成）XML，还可以创建合适的报表、表单和实现可视化。

HTML具有简单和轻型的优点。只需要一个Web浏览器，因为客户端和用户界面可以通过Web浏览器来得到。HTML非常容易设计，并且基本的数据操作可以使用HTML表单来实现。另一个优点是大多数用户都已经熟悉了HTML表单的外观和感受，因此只需要很少的培训或指导。

HTML的缺点是它仅限于具有超文本链接（和一些漂亮的图片）的普通文本，并且用户的交互也是比较基础的，例如单击或移动鼠标、填充文本框或从一个下拉菜单中进行选择。与用程序设计语言（例如Java）开发的报表相比，文本报表的布局和显示会受到非常多的限制。

Java可以用来开发相当复杂的用户界面，包括HTML所有的文本操作，也包括图形显示、旋转3D图象以及与数据库的逐步交互。它的缺点就是在开发和运行时都需要更高的开销：必需编写Java代码来创建和布局页面以支持更复杂的交互，且必须下载applet，然后在客户端Web浏览器中运行。

如果HTML可以满足当前和近期的系统需求，那么继续向前开发且至少把HTML当作原型来使用，这可能比较好。否则，在开发Java界面的过程中需要相当大的工作量，虽然仅仅只是复制HTML中已有的界面。当需要比HTML还要多的交互时，但是并不需要Java的全部功能，那么JSP或Javascript可能是有用的。Javascript可以用在HTML页面中增强交互功能，且JSP允许在HTML文档中嵌入Java代码以生成HTML。

后面两节将讲述怎样使用XSL和Java来给数据库生成一个可浏览的用户界面。7.4节将描述

需要全本请访问 www.ertongbook.com

China-pub.com
下载

怎样扩充用户界面功能，包括数据输入、编辑和搜索。

7.2 基于XSL的用户界面

基于XSL的用户界面可以用来生成 HTML。这一节将介绍 XSL样式表，描述怎样把 XML文档显示成表格或记录，并解释怎样扩展那些带有探测（ drill-down）、代理和自定义的描述。

当数据的表示需要根据客户需求来变化时，具有 XSL的XML特别有用。例如，你可能使用不同的样式表来利用不同的 Web浏览器的特性，或者你可以使用 XSL来生成另一种标记语言，例如手机或其他无线设备的 WML。

7.2.1 XSL样式表

XSL样式表是一个可以转换文档的模板集合。每个模板都具有一个可被称为文档元素的模型和一个用在新文档中创建片段（如果模式相匹配的话）的主体所组成。例如，模板

```
<xsl:template match="para">
   <p>
</xsl:template>
```

将转换文档

```
<page>
   <para>Welcome to my Web site!</para>
</page>
```

成下面的文档

```
<page>
   <p>Welcome to my Web site!</p>
</page>
```

模板的模型部分由一个代表节点集的字符串组成。可以被匹配的字符包括元素类型名、元素属性和子元素的特性。例如：

- Emu —— 匹配一个元素类型名为“ emu ”的节点。
- Walrus —— 匹配一个元素类型名为“ Walrus ”的节点。
- Frog/fish —— 匹配任何元素类型名为“ frog ”或元素类型名为“ fish ”的节点。
- node() —— 匹配任何节点（它不匹配根节点或属性）。
- / —— 匹配文档的根节点。
- circle/coordinate —— 匹配“ circle ”元素的子元素“ coordinate ”。
- text() —— 匹配任何文本。
- * —— 匹配任何元素。
- @name —— 匹配元素的“ name ”属性。
- @id —— 匹配元素的“ id ”属性。
- @* —— 匹配元素的任何属性。

模板的主体可以是：

- 逐字被插入到目标（结果）文档中的元素。

China-pub.com 下载

- 可以用源文档数据来构建目标文档的命令，例如字符数据、元素和属性值。
- 用简单的程序设计语言结构（例如条件分支、迭代或模式匹配）来处理的前面数据项的组合。

XSL支持的一些命令如下：

- value-of —— 得到一个节点的文本，或者得到由一个类似于模型（用在匹配模板中）的表达式所指定的属性。
- if —— 如果布尔表达式的值为真，那么得到“ if ”元素的内容。
- choose —— 求出一系列的“ when ”布尔条件，并且得到第一个为真值的条件的内容，或者如果条件都不为真，那么得到一个（可选）默认值。

利用这些命令，可以用XSL样式表从单一XML文档中创建各种各样的HTML文档。

本书描述XSL样式表的两个版本。第一个版本是 XSL的1998年12月版，它已被 Microsoft Internet Explore（IE5）使用。现在，这个版本是惟一已经广泛用于具有本地 XSL处理的Web浏览器中。第二个版本是 XSLT 1.0，即最近的W3C版本。Microsoft在msdn.microsoft.com上为XSLT 1.0提供一个XSL处理程序。两个版本之间的不同之处包括怎样引入样式表，新版本还增加了其他的命令。本书中的大多数例子都使用 IE5版本。

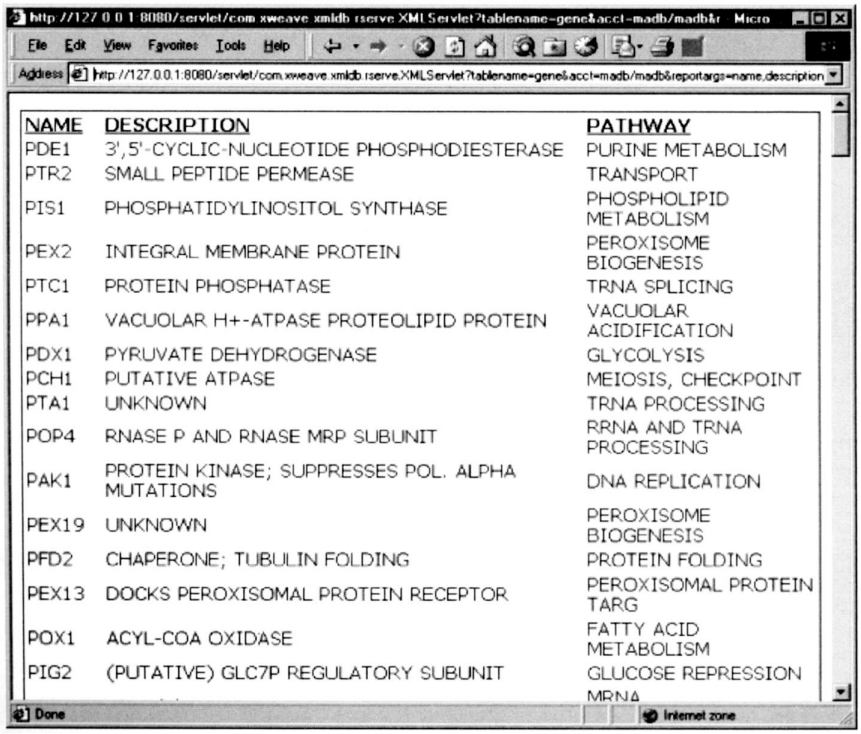
7.2.2 把XML显示成表格

使用XSL样式表可以把XML显示成一个表格，如图7-1所示。示例7-1中的示范样式表文档把一个XML文档显示成一个表格，文档中的元素具有表格特征的子元素结构。示例 7-2中的样式表是同一个样式表，只是它使用的是 XSLT。将在本节的结束部分描述它们的不同之处。

示例7-1 把XML数据转换成HTML表格的简单样式表（IE5）

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <TABLE STYLE="border:1px solid black" BORDER="1">
      <TR STYLE="font-weight:bold; text-decoration:underline">
        <xsl:for-each select="collection/*[1]/*">
          <TD><xsl:node-name/></TD>
        </xsl:for-each>
      </TR>
      <xsl:apply-templates />
    </TABLE>
  </xsl:template>
  <xsl:template match="collection">
      <xsl:for-each select="*">
        <TR>
          <xsl:for-each select="*">
            <TD><xsl:value-of select="."/></TD>
          </xsl:for-each>
        </TR>
      </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

China-pub.com
下载



NAME	DESCRIPTION	PATHWAY
PDE1	3',5'-CYCLIC-NUCLEOTIDE PHOSPHODIESTERASE	PURINE METABOLISM
PTR2	SMALL PEPTIDE PERMEASE	TRANSPORT
PIS1	PHOSPHATIDYLINOSITOL SYNTHASE	PHOSPHOLIPID METABOLISM
PEX2	INTEGRAL MEMBRANE PROTEIN	PEROXISOME BIOGENESIS
PTC1	PROTEIN PHOSPHATASE	TRNA SPLICING
PPA1	VACUOLAR H+-ATPASE PROTEOLIPID PROTEIN	VACUOLAR ACIDIFICATION
PDX1	PYRUVATE DEHYDROGENASE	GLYCOLYSIS
PCH1	PUTATIVE ATPASE	MEIOSIS, CHECKPOINT
PTA1	UNKNOWN	TRNA PROCESSING
POP4	RNASE P AND RNASE MRP SUBUNIT	RRNA AND TRNA PROCESSING
PAK1	PROTEIN KINASE; SUPPRESSES POL. ALPHA MUTATIONS	DNA REPLICATION
PEX19	UNKNOWN	PEROXISOME BIOGENESIS
PFD2	CHAPERONE; TUBULIN FOLDING	PROTEIN FOLDING
PEX13	DOCKS PEROXISOMAL PROTEIN RECEPTOR	PEROXISOMAL PROTEIN TARG
POX1	ACYL-COA OXIDASE	FATTY ACID METABOLISM
PIG2	(PUTATIVE) GLC7P REGULATORY SUBUNIT	GLUCOSE REPRESSION
		MRNA

图7-1 用XSL生成的生物学数据表格

示例7-2 把XML数据转换成HTML表格的简单样式表（XSLT）

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
  <xsl:output method="html"/>
  <xsl:template match="/">
    <TABLE STYLE="border:1px solid black" BORDER="1">
      <TR STYLE="font-weight:bold; text-decoration:underline">
        <xsl:for-each select="collection/*[1]/*">
          <TD><xsl:value-of select="name()"/></TD>
        </xsl:for-each>
      </TR>
      <xsl:apply-templates />
    </TABLE>
  </xsl:template>
  <xsl:template match="collection">
      <xsl:for-each select="*">
        <TR>
          <xsl:for-each select="*">
            <TD><xsl:value-of select="."/></TD>
          </xsl:for-each>
        </TR>
      </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

China-pub.com 下载

示例7-1中的样式表文档由XML标题和一个XSL样式表元素组成。样式表元素由一个属性（它把命名空间定义成与IE5所使用的版本一样）和两个模板（与子元素一样）组成。第一个模板在源文档（由斜线“/”字符所表示）的根节点处执行，且当第二个模板匹配任何具有元素类型名“collection”的元素时，就执行第二个模板。列举如下：

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-
    xsl"> <xsl:template match="/">
    ...
   </xsl:template>
   <xsl:template match="collection">
    ...
   </xsl:template>
</xsl:stylesheet>
```

根模板通过分散的HTML和XSL命令来创建一个HTML表格。“xsl:”命名空间前缀用来区分XSL命令和HTML命令。根模板如下：

```
<xsl:template match="/">
<TABLE STYLE="border:1px solid black">
  <TR STYLE="font-weight:bold; text-
    decoration:underline">
      <xsl:for-each select="collection/*[1]/*">
        <TD>
          <xsl:node-name />
        </TD>
      </xsl:for-each>
    </TR>
   <xsl:apply-templates />
  </TABLE>
</xsl:template>
```

为了理解模板，可以分别查看HTML代码和XSL代码来检查模板。HTML代码如下：

```
<TABLE STYLE="border:1px solid black">
  <TR STYLE="font-weight:bold; text-
    decoration:underline"> For each column name,
      <TD>
        The name of the column goes here.
      </TD>
  </TR>
</TABLE>
```

HTML命令创建一个具有特定样式的“TABLE”，并且创建表的第一行（又具有特定的样式）。行由一个元素类型名（变成列名）集合组成。

XSL代码如下：

```
<xsl:template match="/">
  <xsl:for-each select="collection/*[1]/*">
    <xsl:node-name />
  </xsl:for-each>
  <xsl:apply-templates />
</xsl:template>
```

需要全本请访问 www.ertongbook.com

China-pub.com
下载

XSL代码是一个合法的不包含HTML命令的模板，它将把表标题表示为普通文本。模板主体由一个“for-each”语句（反复迭代列名，直观的引用“collection/*[1]/*”）和一个“apply-templates”命令（在引用该命令的地方，把任何匹配模板应用于样式表）组成（在这个例子中，执行了“collectioin”元素，同时发布其余的行）。

列名是从XML文档中解析出来的，例如：

```
<?xml version="1.0"?>
<collection>
<GENE>
  <NAME>PDE1</NAME>
  <DESCRIPTION>PHOSPHODIESTERASE</DESCRIPTION>
  <PATHWAY>PURINE METABOLISM</PATHWAY>
</GENE>
<GENE>
  <NAME>PTR2</NAME>
  <DESCRIPTION>SMALL PEPTIDE PERMEASE</DESCRIPTION>
  <PATHWAY>TRANSPORT</PATHWAY>
</GENE>
</collection>
```

“collection/*[1]/*”模型由三个部分组成。整个路径引用元素树的第三层深度上的一个元素。每个部分引用一个元素（或者多个元素），斜线“/”意味根据已被选中元素（或多个元素）的孩子匹配下一部分。这三个部分别是：

1) 匹配元素类型“collection”。

2) 匹配这个元素的第一个孩子，由“/*[1]”来表示。逐次的匹配这个元素的所有孩子，然后选择元素的第一个兄弟。

3) 匹配这个元素的所有孩子，由“/*”表示。

因此，第一步匹配：

```
<collection>
 ...
</collection>
```

第二步匹配：

```
<GENE>
  <NAME>PDE1</NAME>
  <DESCRIPTION>PHOSPHODIESTERASE</DESCRIPTION>
  <PATHWAY>PURINE METABOLISM</PATHWAY>
</GENE>
```

第三步匹配三个元素：

```
<NAME>...</NAME>
<DESCRIPTION>...</DESCRIPTION>
<PATHWAY>...</PATHWAY>
```

这三个元素被“for-each”命令迭代，并且每个元素类型名是用“node-name”命令来析取的，最后得到列标题：“Name”、“Description”和“Pathway”。

样式表的第二个模板实际上比较简单。它是：

China-pub.com 下载

```
<xsl:template match="collection">
  <xsl:for-each select="*">
    <TR>
      <xsl:for-each select="*">
        <TD>
          <xsl:value-of select="." />
        </TD>
      </xsl:for-each>
    </TR>
  </xsl:for-each>
</xsl:template>
```

HTML代码为collection的每个元素生成单独的一行：

```
For each element in the collection,
  <TR>
    For each subelement (i.e., column element),
      <TD>
        Value of the character data in the column goes
          here.
      </TD>
  </TR>
```

XSL代码是两个嵌套的“ for ”循环：

```
<xsl:template match="collection">
  <xsl:for-each select="*">
    <xsl:for-each select="*">
      <xsl:value-of select="." />
    </xsl:for-each>
  </xsl:for-each>
</xsl:template>
```

外层“ for-each ”循环匹配collection的所有元素，由select属性中的“ * ”表示。对这个基因集合例子来说，这个循环匹配每个“ gene ”元素，包括已经被根模板用来析取列名的元素：

```
<GENE>
 ...
</GENE>
<GENE>
 ...
</GENE>
<GENE>
 ...
</GENE>
```

内层“ for-each ”循环匹配“ gene ”中的所有元素，例如：

```
<NAME>PDE1</NAME>
<DESCRIPTION>PHOSPHODIESTERASE</DESCRIPTION>
<PATHWAY>PURINE METABOLISM</PATHWAY>
```

“ <xsl:value-of select="." /> ”命令返回当前节点的文本值，实际上是开始与结束标记之间的字符数据。当前节点由“ . ”来表示。在这个例子中，循环中匹配的值是：“ PDE1 ”、“ PHOSPHODIESTERASE ”和“ PURINE METABOLISM ”。

China-pub.com
下载

示例7-1中的整个样式表，当用来处理示例 7-3中的示范数据时，最后结果会得到示例 7-4所给出的HTML。

示例7-3 用XML示范的生物学数据

```
<?xml version="1.0"?>
<?xml:stylesheet type="text/xsl" href="http://127.0.0.1/ss/
simple.xsl"?>
<collection>
<GENE>
  <NAME>PDE1</NAME>
  <DESCRIPTION>3',5'-CYCLIC-NUCLEOTIDE PHOSPHODIESTERASE</DESCRIPTION>
  <PATHWAY>PURINE METABOLISM</PATHWAY>
</GENE>
<GENE>
  <NAME>PTR2</NAME>
  <DESCRIPTION>SMALL PEPTIDE PERMEASE</DESCRIPTION>
  <PATHWAY>TRANSPORT</PATHWAY>
</GENE>
<GENE>
  <NAME>PIS1</NAME>
  <DESCRIPTION>PHOSPHATIDYLINOSITOL SYNTHASE</DESCRIPTION>
  <PATHWAY>PHOSPHOLIPID METABOLISM</PATHWAY>
</GENE>
</collection>
```

示例7-4 生物学数据的HTML表格

```
<TABLE STYLE="border:1px solid black" BORDER="1">
   <TR STYLE="font-weight:bold; text-decoration:underline">
       <TD>NAME</TD>
       <TD>DESCRIPTION</TD>
       <TD>PATHWAY</TD>
   </TR>
     <TR>
         <TD>PDE1</TD>
         <TD>3',5'-CYCLIC-NUCLEOTIDE PHOSPHODIESTERASE</TD>
         <TD>PURINE METABOLISM</TD>
     </TR>
     <TR>
         <TD>PTR2</TD>
         <TD>SMALL PEPTIDE PERMEASE</TD>
         <TD>TRANSPORT</TD>
     </TR>
     <TR>
         <TD>PIS1</TD>
         <TD>PHOSPHATIDYLINOSITOL SYNTHASE</TD>
         <TD>PHOSPHOLIPID METABOLISM</TD>
     </TR>
</TABLE>
```

示例7-2中的XSLT版本在以下四个方面不同于示例 7-1中的IE5 XSL版本：

- XSLT命名空间改为：

```
"http://www.w3.org/1999/XSL/Transform".
```

China-pub.com 下载

- 给样式表处理命令添加了一个版本属性，实际上是：

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

- 定义了一个输出元素，它看起来像下面这个HTML输出：

```
<xsl:output method="html"/>
```

- node-name命令被替换为一个“ name() ”函数，由“ value-of ”调用。实际上，“ <xsl:node-name /> ”被替换为：

```
<xsl:value-of select="name()"/>
```

7.2.3 把XML片段显示成记录

除了数据的面向表格视图以外，当数据集合仅包含一条记录时，或者当用户希望更加详细的检查表的一行时，把数据表示成一个面向记录的视图也是有用的。如果表格表示比浏览器能够显示的宽度还要宽很多，那么面向记录的视图特别有用。

示例7-5给出了一个可以创建图 7-2所示记录的XSL分片。为了在XSL的IE5版本中判断一个集合是否多于一个元素，可以做一个测试，用“ <xsl:when test="*[1]"> ”来看看第二个孩子是否存在。孩子是以数字0作为开始的，因此第二个孩子的位置是1。XSLT版本有一个叫做count的函数，可以计算孩子的数量，而XSLT的孩子的数量是从“ 1 ”开始的。示例7-5中的XSLT版本片段在示例7-6中给出。

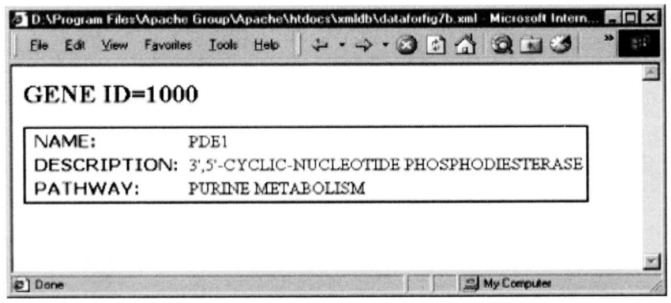


图7-2 使用XSL生成的生物学数据的记录

示例7-5 把XML数据转换成HTML记录的样式表（IE5）

```
<h2><xsl:for-each select="*[0]"><xsl:node-name/></xsl:for-each></h2>
<TABLE STYLE="border:2px solid black">
   <xsl:for-each select="*[0]/*">
   <TR>
     <TD STYLE="font-weight:bold">
       <xsl:node-name/>:</TD>
       <TD>
         <xsl:value-of select="."/>
       </TD>
   </TR>
   </xsl:for-each>
</TABLE>
```

China-pub.com
下载

示例7-6 把XML数据转换成HTML记录的样式表（XSLT）

```
<h2><xsl:value-of select="name(*[1])"/> </h2>
<TABLE STYLE="border:2px solid black">
   <xsl:for-each select="*[1]/*">
   <TR>
     <TD STYLE="font-weight:bold">
      <xsl:value-of select="name()"/>:</TD>
       <TD>
         <xsl:value-of select="."/>
       </TD>
   </TR>
   </xsl:for-each>
</TABLE>
```

7.2.4 把标识符和代理显示为超文本链接

当数据库中的记录具有惟一的标识符时，这个标识符可能被当作表的一部分表示给用户，以允许只选择和显示这个记录的数据。在示例 7-7的样式表中，假设标识符是相应元素的一个“ID”属性，且被表示在表的第一列。可以修改样式表为标识符创建一个超文本链接，这样可以从数据库中检索出完整的记录数据。就如图7-3所示的那样。示例7-8给出了样式表的XSLT版本。

示例7-7 把XML数据转换成具有标识符的HTML表格的样式表（IE5）

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
      <xsl:apply-templates />
  </xsl:template>
  <xsl:template match="collection">
    <h2><xsl:for-each select="*[0]"><xsl:node-name/></xsl:for-each>
LISTING</h2>
    <TABLE STYLE="border:5px solid black" BORDER="1">
      <TR STYLE="font-size:12pt; font-family:Verdana; font-
weight:bold; text-decoration:underline">
        <TD STYLE="background-color:lightgrey">ID</TD>
          <xsl:for-each select="*[0]/*">
          <TD><xsl:node-name/></TD>
            </xsl:for-each>
      </TR>
      <xsl:for-each select="*">
        <TR STYLE="font-family:Verdana; font-size:12pt; padding:0px 6px">
          <TD STYLE="background-color:lightgrey">
            <xsl:value-of select="@id"/>
          </TD>
          <xsl:for-each select="*">
                  <TD>
             <xsl:value-of select="."/>
            </TD>
          </xsl:for-each>
        </TR>
      </xsl:for-each>
    </TABLE>
```

China-pub.com 下载

```
  </xsl:template>
</xsl:stylesheet>
```

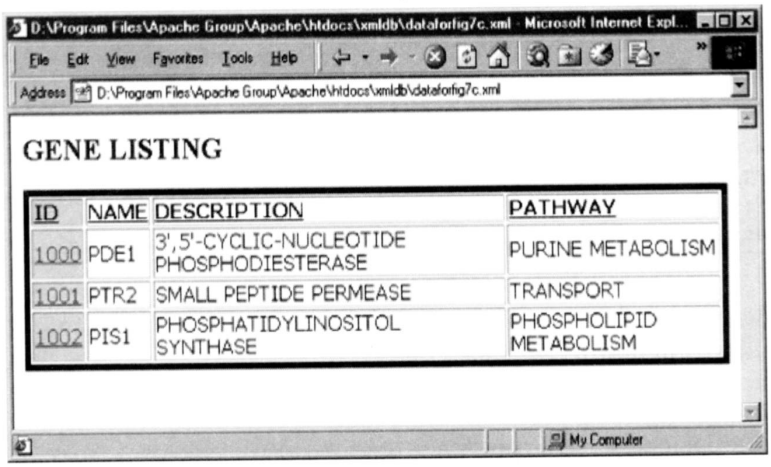


图7-3 用XSL生成的具有标识符的生物学数据表格

示例7-8 把XML数据转换成具有标识符的HTML表格的样式表（XSLT）

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
  <xsl:output method="html"/>
  <xsl:template match="/">
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:template match="collection">
    <h2><xsl:value-of select="name(*[1])"/> LISTING</h2>
    <TABLE STYLE="border:5px solid black" BORDER="1">
      <TR STYLE="font-size:12pt; font-family:Verdana; font-
weight:bold; text-decoration:underline">
        <TD STYLE="background-color:lightgrey">ID</TD>
          <xsl:for-each select="*[1]/*">
          <TD><xsl:value-of select="name()"/></TD>
            </xsl:for-each>
      </TR>
      <xsl:for-each select="*">
        <TR STYLE="font-family:Verdana; font-size:12pt; padding:0px 6px">
          <TD STYLE="background-color:lightgrey">
            <xsl:value-of select="@id"/>
          </TD>
          <xsl:for-each select="*">
            <TD>
              <xsl:value-of select="."/>
            </TD>
          </xsl:for-each>
        </TR>
      </xsl:for-each>
    </TABLE>
  </xsl:template>
</xsl:stylesheet>
```

需要全本请访问 www.ertongbook.com

China-pub.com
下载

提取一条具有惟一标识符的记录的超文本链接是应用探测的一种特殊情况。探测是一种机制，可以用来更加详细的检查一个集合数据的某些部分。当想要获得比二维表格所表示的数据还要多的数据时，探测是有用的。

当一个文档具有很多级别的嵌入元素时，把所有级别的数据都发送给用户界面是比较浪费的，特别是如果显示只需表示最高层的数据时。这时可以使用一个 proxy元素，它代替一个XML分片，并且包含足够的信息，如果需要，可以用这些信息来检索这个分片。对关系型数据库来说，包含表名和主键值就已经足够了。对 XML数据库来说，使用惟一的分片标识符。

在用户界面中显示数据时，通过检索所需的合适的分片，应用程序可以扩展代理。 HTML浏览器通过超文本链接提供一个简单的机制。使用超文本链接，用户可以探测由代理所包含的信息。例如，XML元素

```
<proxy tablename="exper_result" id="1024"/>
```

可以被下面的XSL样式表模板转换成一个超文本链接。

```
<xsl:template match="proxy">
    <xsl:element name="a">
        <xsl:attribute name="href">
        http://127.0.0.1com.xweave.xmldb.rserve.XMLServlet?
          tablename=<xsl:value-of select="@tablename"/>&amp;
          id=<xsl:value-of select="@id">
          &amp;
          stylesheet=
          http://127.0.0.1/ss/generic.xsl
        </xsl:attribute>
        <xsl:value-of select="@id"/>
          </xsl:element>
        </xsl:template>
```

模板创建一个元素类型名为“ a ”、属性为“ href ”的HTML元素，这样就可以创建一个具有文本值（就是代理的属性“ ID ”的值）的超文本链接。 HTML包含以下几个部分：基本 URL、表名、ID和样式表参数。“ &amp；”编码被转换成“ & ”字符，最后得到的HTML（为了便于阅读添加了换行符）如下所示：

```
<a href="http://127.0.0.1/com.xweave.xmldb.rserve.XMLServlet?
tablename=__________&id=_______________&
stylesheet=http://127.0.0.1/ss/generic.xsl">
id goes here as text
</a>
```

HTML导致超文本链接被显示在Web浏览器中。

示例7-9中的XSL样式表生成一个具有代理和探测的 HTML表格。

示例7-9 把XML数据转换成具有探测的HTML表格的样式表

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
```

China-pub.com 下载

```
      <xsl:apply-templates />
  </xsl:template>
  <xsl:template match="collection">
    <h2><xsl:for-each select="*[0]"><xsl:node-name/></xsl:for-each>
LISTING</h2>
    <TABLE STYLE="border:5px solid black" BORDER="1">
      <TR STYLE="font-size:12pt; font-family:Verdana; font-
weight:bold; text-decoration:underline">
        <TD STYLE="background-color:lightgrey">ID</TD>
          <xsl:for-each select="*[0]/*">
          <TD><xsl:node-name/></TD>
            </xsl:for-each>
      </TR>
      <xsl:for-each select="*">
        <TR STYLE="font-family:Verdana; font-size:12pt; padding:0px 6px">
          <TD STYLE="background-color:lightgrey">
             <xsl:element name="a">

            <xsl:attribute
            name="href">http:/servlets/com.xweave.xmldb.rserve.xmlservlet?
            tablename=<xsl:node-name/>&amp;
            id=<xsl:value-of select="@id"/>&amp;
            stylesheet=http:/ss/generic.xsl
                </xsl:attribute>
                <xsl:value-of select="@id"/>
            </xsl:element>
          </TD>
          <xsl:for-each select="*">
          <TD>
            <xsl:choose>
            <xsl:when test="text()"><xsl:value-of select="."/></xsl:when>
            <xsl:when test="proxy"><xsl:apply-templates /></xsl:when>
            <xsl:when test="node()">
               <xsl:element name="a">
                   <xsl:attribute
name="href">http:/servlets/com.xweave.xmldb.rserve.xmlservlet?
tablename=<xsl:for-each select="*[0]"><xsl:node-name/></xsl:for-each>&amp;
id=<xsl:value-of select="./*/@id"/>&amp;stylesheet=http:/ss/generic.xsl
                </xsl:attribute>
                <xsl:choose>
                <xsl:when test="./*/@id"><xsl:value-of select="./*/@id"/>
                </xsl:when>
                <xsl:otherwise><xsl:value-of select="."/></xsl:otherwise>
                </xsl:choose>
               </xsl:element>
            </xsl:when>
            <xsl:otherwise><xsl:value-of select="."/></xsl:otherwise>
            </xsl:choose>
          </TD>
          </xsl:for-each>
        </TR>
      </xsl:for-each>
    </TABLE>
  </xsl:template>
</xsl:stylesheet>
```

China-pub.com
下载

7.2.5 根据内容改变表示

当集合恰好有一条记录时，示例7-10中的样式表提供一个记录的表示；当集合中有多条记录时，它提供一个表格表示。它也支持代理的使用。

示例7-10 把XML数据转换为带有探测的HTML记录或表格的通用样式表

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
      <xsl:apply-templates />
  </xsl:template>
  <xsl:template match="collection">
  <xsl:choose>
  <xsl:when test="error">
     <xsl:copy>Error: <xsl:value-of select="*/@msg" /></xsl:copy>
  </xsl:when>
  <xsl:when test="*[1]">
    <h2><xsl:for-each select="*[0]"><xsl:node-name/></xsl:for-each>
LISTING</h2>
    <TABLE STYLE="border:5px solid black" BORDER="1">
      <TR STYLE="font-size:12pt; font-family:Verdana; font-weight:bold;
text-decoration:underline">
        <TD STYLE="background-color:lightgrey">ID</TD>
          <xsl:for-each select="*[0]/*">
          <TD><xsl:node-name/></TD>
            </xsl:for-each>
      </TR>
      <xsl:for-each select="*">
        <TR STYLE="font-family:Verdana; font-size:12pt; padding:0px 6px">
          <TD STYLE="background-color:lightgrey">
             <xsl:element name="a">
                <xsl:attribute
name="href">http:/servlets/com.xweave.xmldb.rserve?
tablename=<xsl:node-name/>&amp;id=<xsl:value-of select="@id"/>&amp;
stylesheet=http:/ss/generic2.xsl
                </xsl:attribute>
                <xsl:value-of select="@id"/>
          </xsl:element>
          </TD>
          <xsl:for-each select="*">
          <TD>
            <xsl:choose>
            <xsl:when test="text()"><xsl:value-of select="."/></xsl:when>
            <xsl:when test="proxy"><xsl:apply-templates /></xsl:when>
            <xsl:when test="node()">
               <xsl:element name="a">
                   <xsl:attribute
name="href">http:/servlets/com.xweave.xmldb.rserve?
tablename=<xsl:for-each select="*[0]"><xsl:node-name/></xsl:for-each>&amp;
id=<xsl:value-of select="./*/@id"/>&amp;stylesheet=http:/ss/generic2.xsl
                   </xsl:attribute>
                   <xsl:choose>
                    <xsl:when test="./*/@id"><xsl:value-of select="./*/
@id"/>
                    </xsl:when>
                    <xsl:otherwise><xsl:value-of select="."/></
```

China-pub.com 下载

```
xsl:otherwise>
                    </xsl:choose>
                </xsl:element>
              </xsl:when>
              <xsl:otherwise><xsl:value-of select="."/></xsl:otherwise>
              </xsl:choose>
            </TD>
            </xsl:for-each>
          </TR>
        </xsl:for-each>
      </TABLE>
    </xsl:when>
    <xsl:otherwise>
      <h2><xsl:for-each select="*[0]"><xsl:node-name/></xsl:for-each>
 ID=<xsl:value-of
 select="*[0]/@id"/></h2>
      <TABLE STYLE="border:2px solid black">
          <xsl:for-each select="*[0]/*">
          <TR>
            <TD STYLE="font-family:Verdana; font-size:12pt; font-weight:bold;
 padding:0px 6px">
              <xsl:node-name/>:</TD>
        <TD>
        <xsl:choose>
          <xsl:when test="text()"><xsl:value-of select="."/></xsl:when>
          <xsl:when test="proxy"><xsl:apply-templates /></xsl:when>
          <xsl:when test="node()">
            <xsl:element name="a">
         <xsl:attribute
name="href">http:/servlets/com.xweave.xmldb.rserve?tablename=<xsl:for-each
select="*[0]"><xsl:node-name/></xsl:for-each>&amp;id=<xsl:value-of
select="./*/@id"/>&amp;stylesheet=http:/ss/generic2.xsl
            </xsl:attribute>
            <xsl:choose>
            <xsl:when test="./*/@id"><xsl:value-of select="./*/@id"/></xsl:when>
            <xsl:otherwise><xsl:value-of select="."/></xsl:otherwise>
              </xsl:choose>
                 </xsl:element>
              </xsl:when>
        <xsl:when test="node()">
        <xsl:element name="a">
        <xsl:attribute
name="href">http:/servlets/com.xweave.xmldb.rserve?
tablename=<xsl:value-of select ="./@tablename"/>&amp;
id=<xsl:value-of select="./@id"/>&amp;stylesheet=http:/ss/generic2.xsl
                  </xsl:attribute>
             <xsl:value-of select="."/>
                </xsl:element>
              </xsl:when>
             <xsl:otherwise><xsl:value-of select="."/></xsl:otherwise>
             </xsl:choose>
             </TD>
          </TR>
          </xsl:for-each>
      </TABLE>
      </xsl:otherwise>
    </xsl:choose>
```

China-pub.com
下载


```
  </xsl:template>
  <xsl:template match="proxy">
     <xsl:element name="a">
         <xsl:attribute
name="href">http:/servlets/com.xweave.xmldb.rserve?
tablename=<xsl:value-of select ="@tablename"/>&amp;
id=<xsl:value-of select="@id"/>&amp;stylesheet=http:/ss/generic2.xsl
         </xsl:attribute>
     <xsl:value-of select="@id"/>
     </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

把XML与XSL样式表结合一起使用的一个优点是除了通用的转换以外，还可以自定义表示。对于把数据显示在一个图形表单中、图示数据或者增加数据来说，元素中的信息可能已经足够了。

一个扩充XML的方法就是为具有特殊标记的元素创建一个可选择的转换。例如，在图 7-2中，通用样式表中的个别记录的表示可以根据元素的元素类型名来添加一些可选项，这样可以创建通过CGI程序提供附加功能的HTML表单。例如，下面的XSL片段可以用来给一个试验结果添加一些注释。

```
<xsl:choose>
  <xsl:when test="EXPER_RESULT">
  <form action="/cgi-bin/add_comment" method="POST">
    <hidden name="tablename" value="EXPER_RESULT" />
    <xsl:element name="hidden">
      <xsl:attribute name="name">id</xsl:attribute>
      <xsl:attribute name="value">
        <xsl:value-of select="EXPER_RESULT/@id"/>
      </xsl:attribute>
    </xsl:element>
    <textarea name="comment" rows="5" cols="60">
    </textarea>
    <p>
    <input type="submit" name="Submit" value="Add
      comment to Experimental Result"/>
    </p>
  </form>
  </xsl:when>
</xsl:choose>
```

结果显示如图7-4所示。

可以根据数据库的信息给显示添加一些附加信息。除了 CGI脚本以外，模式中的附加信息也可能被包括进来。例如，基于关系型数据库的外键约束的探测可以提供一个连接到明细表的链接，它提供当前表中值的附加信息。可以从相反的方向跟随外键约束来提供引用当前记录的表的信息。例如，公司中一个职工数据库可能具有支持多种应用程序的多个表，这些应用程序都引用职工ID为外键约束。“ 弹出 ”信息可以用来检查应用程序。而职工又想参与此应用程序，即

China-pub.com 下载

使是在当前应用程序实现以后再开发共享应用程序。在一个生物学数据库中，基因表可能被几个其他数据库组件所引用，并且对“弹出”信息的浏览可以阐明前面未知的关系。

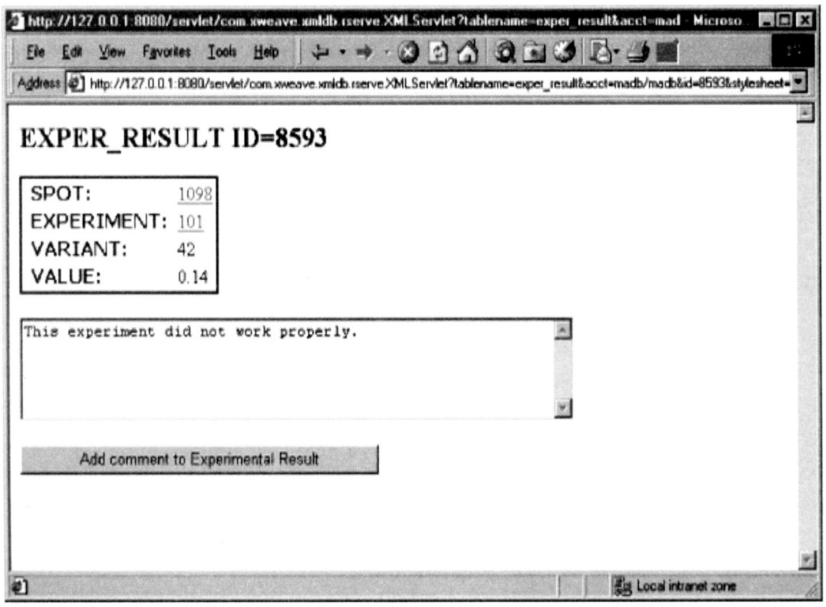


图7-4 带有输入注释表单的记录

7.3 基于Java的可视化

基于Java的用户界面提供另一种可以与XML数据库进行交互的方法。Java或其他程序设计语言可以用来实现编辑器、数据输入应用程序、可视化工具、报表生成器、过程管理软件或其他与XML数据库交互的客户软件。Java applet或应用程序可以用来实现客户程序，这取决于客户程序是否采用Web传送方式。客户程序使用命令和XML与数据库进行交互，并且通过用户界面与用户进行交互。DBMS与客户程序之间的交互已经在第5章中描述过了。本章将主要讲解内部结构和客户程序的功能。这一节描述显示数据库中XML的应用程序，下一节描述与数据库进行交互的应用程序的创建过程。

7.3.1 客户程序结构

最简单的客户程序结构只是给用户显示XML，没有编辑或进一步与DBMS交互的功能。Java客户程序首先必须在显示用户界面之前解析XML。现有两种主要的XML解析器类型，第一种是基于树的解析器，例如DOM解析器，它创建一棵XML文档树。在解析过程完成后，客户程序创建一个显示。第二种类型的解析器是SAX解析器，它是一个基于事件的解析器（它在解析文档的时候创建事件）。在文档的关键位置上——例如开始和结束标记或字符数据区域——解析器创建事件，并把事件传递给一个特定应用的文档处理程序。特定应用的处理程序使用与事件相关联的数据（例如元素类型名或字符数据）来创建一个自定义的数据结构或执行其他动作。

需要全本请访问 www.ertongbook.com