利用 CGI 技术实现 WEB 服务器与 DBF 数据库文件的连接

孙 琨 曹 勇 (南开大学计算机与系统科学系 300071)

摘要:本文详细讨论了采用 CGI 技术实现 WWW 服务器与流行的 DBF 格式数据库文件之间动态数据交互的方法。

关键字:CGI URL 数据库连接

一、CGI 技术的实现方法

浏览器可通过 GET 和 POST 两种方法把数据从用户站点上的 HTML 表格传输给服务器,在对这两种方法详细介绍之前,首先让我们先研究一下网络环境下 URL 的编码方式。

1. URL 编码

虽然根据表格指定方法的不同,浏览器可以用不同的方法(GET 或 POST)把数据发送到服务器上,但这些浏览器用于数据编码的方式却是一致的。客户机收集用户的输入数据并用相应的变量名包装,变量之间用 & 号隔开,变量名与其对应值之间以等号隔开,如下例:

name = blacksmith&e-mail = city @ shoop. com&comments = you%27ll + go

从中可看出该表格包含三个 HTML 变量:name、e-mail、comments,分别取值为 blacksmith, city@shoop. com 和 you%27ll + go。

由于 GET 命令是把表格数据加到 URL 末尾,故必须对表格信息进行专门编码,以使其不与 URL 标准中的特殊字符相混淆。URL 标准保留了多个特殊字符,如 &,它们对客户机和服务器有特殊含义。另外有些特殊字符在 URL 中是非法的,具体定义如下:

·在 URL 中下列保留字符有特定含义：= ；/ # ，? ：'

·任意不可打印的 ASCII 控制字符为在 URL 中为非法

·空格为非法字符

浏览器用加号来代替原来的空格字符，用百分号后接上相应的十六进制 ASCII 码值替代保留的控制字符。出于这个原因，前面例子中的 you'll go 将编码成 you%27ll+go。

CGI 脚本接收到表格数据后，首先对其进行 URL 解码的，它包括 4 个步骤：

(1) 把原来用 & 符号连接在一起的不同 HTML 表格变量及其相应值分开。

(2) 把原来用等号连接在一起的 HTML 表格变量同其对应值分开。

(3) 把各个变量和数据中包含的十六进制序列(通过百分号转义处理)转换成相应的 ASCII 等价表示。

(4) 把各变量和数据中所有加号替换成空格。

依次执行完这些步骤并把 HTML 变量及其值还原成原始形式后，便可以对这些信息进行各种形式的处理了。

2. POST 方法

如果 HTML 表格采用的是 POST 方法，那么服务器从标准输入接收客户机数据，并将它发送给相应的 CGI 脚本，同时把 REQUEST-METHOD 环境变量设置为 POST。CGI 脚本程序首先检查 REQUEST-METHOD 变量，确保其处于接收 POST 数据的状态，然后读取这些数据。

正常情况下，一个实用程序在读取完来自标准输入的所有数据后，将收到一个特殊的文件结束标志，但使用 POST 方法传输完数据后，HTTP 并不发送相应的结束标志，因此 CGI 无法判断何时接收结束。作为一种替代方法，HTTP 服务器将设置环境变量 CONTENT-LENGTH，该变量给出脚本程序应读取的输入数据量(以字节为单位)。当脚本程序已读完这个数目的数据后，将停止阅读。

类似于服务器发送给客户机的数据，客户机用 POST 方法发送的信息也有一个相应的 MIME 类型。目前客户机发送的数据只有一种可能的 MIME 类型：application/X - www - form - unlencoded，该类型定义来自 HTML 表格的数据。服务器把客户机发送数据的 MIME 类型记录在环境变量 CONTENT-TYPE 中，尽管目前对于来自 HTTP POST 的数据只有一种类型，但你应该检查该变量的值，以确保你的 CGI 脚本程序能适应未来的变化。

在确认环境变量 REQUEST-METHOD 和 CONTENT-TYPE 具有合乎要求的值，并从标准输入读取完由变量 CONTENT-LENGTH 指明的数据量后，接下来的工作便是进行前面提过的 URL 解码过程，然后 CGI 脚本程序便可以对这些信息进行各种形式的处理了。

3. GET 方法

基于 GET 方法的 CGI 脚本与基于 POST 方法的脚本程序的运行原理基本相同，只不过服务器发送数据所采用的方式稍有差别。这时服务器不再把浏览器提交的数据发送到脚本的标准输入，而是把这些数据编码发送到环境变量 QUERY-STRING 中。另一个差别是 CGI 变量 REQUEST-METHOD 的缺省定义为“GET”，而非“POST”。当数据传输结束后，如果使用 GET 方法，传输内容紧接着脚本程序的 URL 后面被传送和显式出来，而 POST 方法不显式传送的内容。

GET 方法比较简单，易于控制，但它限制了可传送的数据量，通常限制在一千字节以内；POST 方法允许传送较大量数据。究竟选用 GET 还是 POST 方法对编写 CGI 脚本程序的工作量没有太大的影响，许多脚本程序根据 REQUEST-METHOD 的值来确定程序的行为，进而提供对所有这两种方式的支持。

二、利用 CGI 实现 WEB 服务器与 FoxPro 数据库的连接

FoxPro 数据库是一种简便易用的数据库，在当今社会中有着广泛的应用。WWW 网如果不能访问该数据库中的数据，将造成大量的资源浪费。使用 CGI 技术可以很好的解决 Web 服务器与 FoxPro 数据库的连接问题，从而拓宽了 WWW 网的应用范围。

1. 数据库存盘文件 DBF 的文件格式

在我们介绍具体的连接方法之前，有必要对 FoxPro 数据库存盘文件(后缀名为 DBF)进行简要的了解。DBF 文件格式如表 3 所示。

表 3　FoxPro 数据库存盘文件格式

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
03/F5	年	月	日	总记录数				首记录偏移量			各字段长度和		00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
第一字段名											类型	偏移	00	00	00
长度	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
第二字段名											类型	偏移	00	00	00
长度	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

…… ……

最后字段名											类型	偏移	00	00	00
长度	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	第一条记录长度(= 各字段总长度)														
		20	第二条记录长度(= 各字段总长度)												
				……				20	最后条记录长度						
(= 各字段总长度)											1A				

上图中所有数为十六进制，其中字节 0 中的值为 03 或 F5，当数据库中包含 MEMO 类型的域时，值为 F5，否则值为 03；字节 1 至 3 中为最近修改的日期；后面 4 个字节为数据库包含的总记录数，低位在前，高位在后；字节 8 至 9 中的值非常重要，由该值可得到第一个记录在文件中的位置，即其偏移量；每个记录包含的字节数由字节 10 和 11 中的值决定；在每条记录开始前有一个字节，其值为 20 或 2A，20 表示该记录未被删除，2A 表示记录已被删除；在文件头中从字节 16 开始，每 32 个字节用于描述一个字段的属性；文件最后以 1A 结尾。

2. **用 C 语言编写 Web 服务器与 FoxPro 数据库连接的 CGI 脚本程序**

对于存在于服务器上的 DBF 数据库文件，用户可以通过 Web 浏览器对其进行各种操作，例如查询某一记录或修改添加记录。下面以查询记录号为 n 的记录内容为例，结合 DBF 文件存储结构加以介绍。上面介绍的表格程序提供的信息，已经包含在各 CGI 的环境变量中。

首先打开 DBF 文件，得到数据库的结构消息，并通过 CONTENT-METHOD 环境变量来判断使用 POST 方法还是 GET 方法：

```
printf ("Content-type: text/htm \ n") ; //服务器通知用户反馈信息的类型
Filename = getenv("PATH-INFO"); //获取路径信息
f = fopen ( Filename, "r+"); //打开数据库文件
if ( f = = "NULL") // 判断是否成功
  { printf ( " <HEAD><TITLE> Error - can not open file </TITLE></HEAD> \ n");
  printf (" <BODY> The file %s could not be opened \ n", Filename </BODY>");
  exit(0 );
  }
  fseek ( f, 4 , SEEK-SET);
  recordnum = (int ) getc(f); //获取记录总数
  fseek ( f, 8, SEEK-SET);
  fread ( &inipos, sizeof (int), 1, f ); //获取首记录起始偏移地址
  fseek ( f, 10, SEEK-SET);
  fread ( &setlength, sizeof (int) , 1, f); //获取所有字段总长度
  setnum = (inipos - 1)/32 - 1; //计算文件包含的字段数目
  RequestMethod = getenv ("REQUEST-METHOD"); //获取 REQUEST-METHOD 中内容
```

采用 POST 方法的 CGI 处理程序由 CONTENT-LENGTH 环境变量得到用户输入的数据长度，对其进行

解码,得到用户要查询的记录号 n,然后从数据库文件中找到相应记录,将其读出并返回到用户的浏览器上,程序如下:

```
if ( strcmp(RequestMethod,"POST") = = 0 )
  { length = getenv( "CONTENT-LENGTH"); //获取用户输入的数据长度
  if ( length! = NULL)
    ContentLength = atoi(length)
  else
    ContentLength = 0;
    i = 0;
    while (i< ContentLength) //将长度为 ContentLength 的用户数据读入 InputBuffer 中
      { x = fgetc(stdin);
        if ( x = = EOF) break;
        InputBuffer[i + + ] = x; //InputBuffer 是用户定义的数据缓冲区
      }
      InputBuffer[i] = '\n';
```

下面开始对用户数据进行解码操作:

```
Ctype = getenv("CONTENT-TYPE");
if ( strcmp ( Ctype, "application/x - www - form - unlencoded" = = 0)
  { Item = strtok (InputBuffer, "&"); //分离不同的变量
    p = strchr(Item, '='); //p 中为变量的对应值
    *p + + = '\0';
    urlDecode(Item); //将 Item 及 p 中的百分号后的十六进制数转换成对应的 ASCII 码
    urlDecode(p);
    strcvrt(p, '\n', ' ');
    strcvrt(p, '+', ' '); //用空格替代"+"和"\n"号
    strlength = strlen(Item);
    str = strlen(p);
    num = 0;
    for(i = 0;i<str;i + + ) //获取用户查询的记录号
    { num1 = (int)(p[i] - '0');
    num = num * 10 + num1;
    }
    recordini = inipos + recordnum * setlength + 1; //计算该记录在数据库中的偏移量
    fseek(f, recordini, SEEK-SET);
    printf ("< HEAD > < TITLE > data </TITLE > </HEAD> \n");
    printf("<BODY><H1>Data in recordnumber %d are:</H1> \n", num);
    for(k = 0;k<setnum;k + + ) //将记录中数据反馈给用户
    { numread = fread(listbuf, sizeof(char), setlength[k], f);
      printf(" %s = %.99s \n", setname[k], listbuf);
      printf("<pre><BR></pre>");
    }
  }
  URL = getenv("HTTP-REFERER");
  printf("<h3><A href = \"%s">Return</a></h3></BODY> \n", URL);
  printf("</BODY> \n");
  exit(0);
  }
```

当操作发生错误时,产生如下反馈信息:

```
if ( ferror(f))
{
  printf( <HEAD><TITLE> SERVER FILE I/O ERROR </TITLE> < HEAD> \n");
  printf( <BODY> The information you supplied could not be accpted due to \n");
  printf( "a file I/O error at the server \n");
  printf( " <A href = s"> Return </a> </BODY> \n", URL);
}
```

采用 *GET* 方法的程序与对应的 *POST* 程序大体相同,此时环境变量 *REQUEST-METHOD* 中的值为 *GET*,只是在读取用户输入数据的方式上有所不同,它由下面的两条程序完成:

```
p = getenv( "QUERY-STRING");
strncpy( InputBuffer, p, sizeof(InputBuffer));
```

其余的程序段与 *POST* 的程序段基本一样。

参考文献

[1] 兰达尔,<<*HTML* 用户使用指南>>,科技出版社

[2] *J. Magid P. Jones*,<<*WEB* 服务器技术指南>>,机械工业出版社

[3] *P.* 肯特,<<*Netscape for windows* 95 用户使用指南>>,科技出版社

[4] 易文韬、陈颖平,<< *JAVA* 手册>>,科学出版社

(来稿时间:1998 年 4 月)