第17章 Web数据库编程

- 使用客户/服务器版本的Delphi可以快速、迅捷的进行Web数据库的开发，比如开发像CGI（Common Gateway Interface，即通用网关接口）、DLL（Dynamic-Link Libraries，即动态链接库）等的服务器程序。这些服务器程序通常会包括一些不对见部件，Delphi的Internet部件组提供了许多这种部件，从而使编写服务器程序不再困难。Delphi中的Internet部件组提供的部件可以用于获得来自某个URI（Uniform Resource Identifier）的请求，然后触发一定的事件对请求进行处理，最后通过编程创建HTML页面将对客户请求处理的结果返回给客户。本章将主要介绍的内容有：Web数据库的基本概念及Delphi中Web服务器程序简介；ISAPI动态链接库的编写；WIN-CGI编写方法以及和ISAPI DLL之间的转换。

  - 17.1 Web数据库编程中的基本概念
  - 17.2 使用Delphi编写ISAPI DLL
  - 17.3 编写Win-CGI和标准CGI的服务器程序

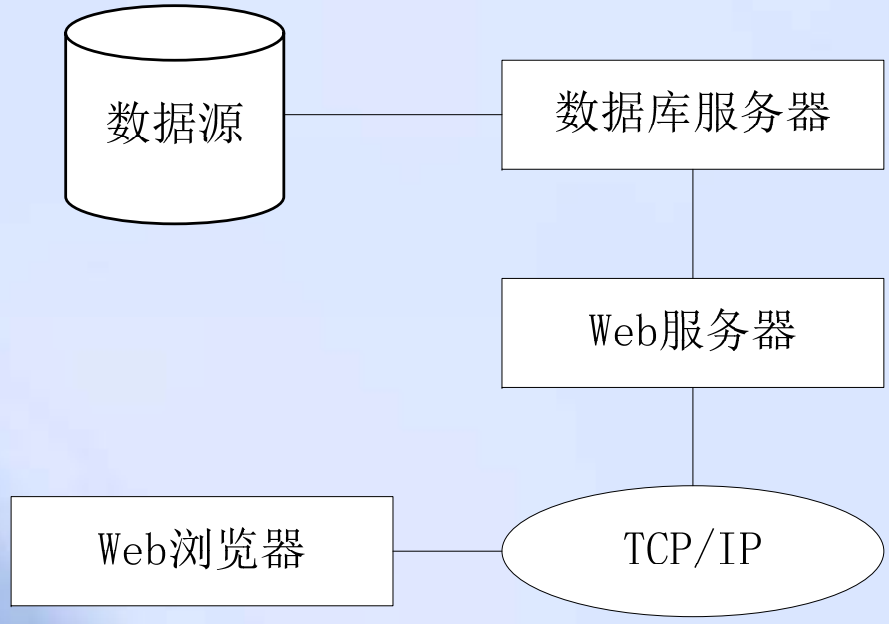
17.1 Web数据库编程中的基本概念

- 随着网络技术的发展，新的技术层出不穷，往往一个问题有多种解决方案。比如Web服务器程序包括ISAPI动态链接库、NSAPI动态链接库、标准CGI、WIN-CGI等，但是又不局限于这几种方式，使用ActiveX技术同样可以完成Web服务器程序的功能。
  - 17.1.1 Web数据库的基本结构
  - 17.1.2 HTML在Web数据库编程中的作用
  - 17.1.3 Web服务器程序

17.1.1 Web数据库的基本结构

- 典型的Web数据库有一个Web浏览器作为用户界面、一个数据库服务器用作信息存储和数据的采集和一个连接两者的Web服务器。Web软件出色的将数据陈述简化和标准化，DBMS（数据库管理系统）则组织和标准化数据的接收与存储。Web浏览器通过TCP/IP（Translation Control Protocol/Internet Protocol，传输控制协议/网际协议）和Web服务器相连，Web服务器把来自Web浏览器的请求或数据传给数据库服务器，然后数据库服务器在数据库中进行相应的操作。这其中对客户请求和有关数据的处理都是由Web服务器程序完成的，Web服务器程序是存在于服务器上的。Web数据库系统简化的结构如图所示。

17.1.1 Web数据库的基本结构



17.1.1 Web数据库的基本结构

- 在这种Web数据库系统中，Web浏览器把Web页请求与数据请求送到Web服务器，Web服务器接收请求并把数据请求送到Web服务器程序。Web服务器程序接受请求，将其转化成数据库服务器能够接受的形式（如ODBC SQL），然后执行数据库操作，诸如查询或插入，并把结果送回服务器扩展程序。最后，Web服务器程序将结果转化成Web浏览器能够接受的形式（如HTML），把它们送给Web服务器。Web服务器则把数据库结果送回到Web浏览器。

- 用户唯一需要在机器上安装的程序是Web浏览器，并且唯一需要学会用的也是Web浏览器。用户通过Web页上显示的表格和数据库进行交互操作。典型的交互操作包括读取页、单击链接、在列表框中进行选择以及查询和输入数据域。从数据库获取的信息能以文本、图像、表、图形或者多媒体对象的形式在Web页上显示。用户在Web页上的表格内输入要查询的内容——接收邮件所消耗的机时；然后，数据库在Web页上用表显示出所要查询的内容。

17.1.2 HTML在Web数据库编程中的作用

- Web数据库的建立基于几种不同的技术，并且编写Web数据库应用程序的人们有着各种不同的背景，这里讲解的是HTML在Web数据库编程中的作用。Web工具和数据库是分开发展的两种不同的技术。

- HTML（超文本语言）在Web数据库中利用表格（Form）接收用户的输入，利用JavaScript进行输入合法性检查，并用表显示数据库查询结果。

17.1.3 Web服务器程序

- Web服务器程序的存在扩展了Web服务器的功能和能力。Web服务器程序接收从Web服务器传输来的请求，根据这些请求进行一系列的操作，然后将操作的结果返回给Web服务器。当然，这些请求都是通过HTTP协议传输来的。这里所说到的HTTP协议和TCP/IP协议并不矛盾，因为HTTP协议是在TCP/IP协议上工作的应用层协议。

- 1. Web服务器程序的种类
  - 这里所说的Web服务器程序的种类是指通常使用的一些，也就是Delphi可以创建的服务器程序：Microsoft Server DLL（ISAPI）；Netscape Server DLL（NSAPI）；标准CGI程序；Windows CGI程序。

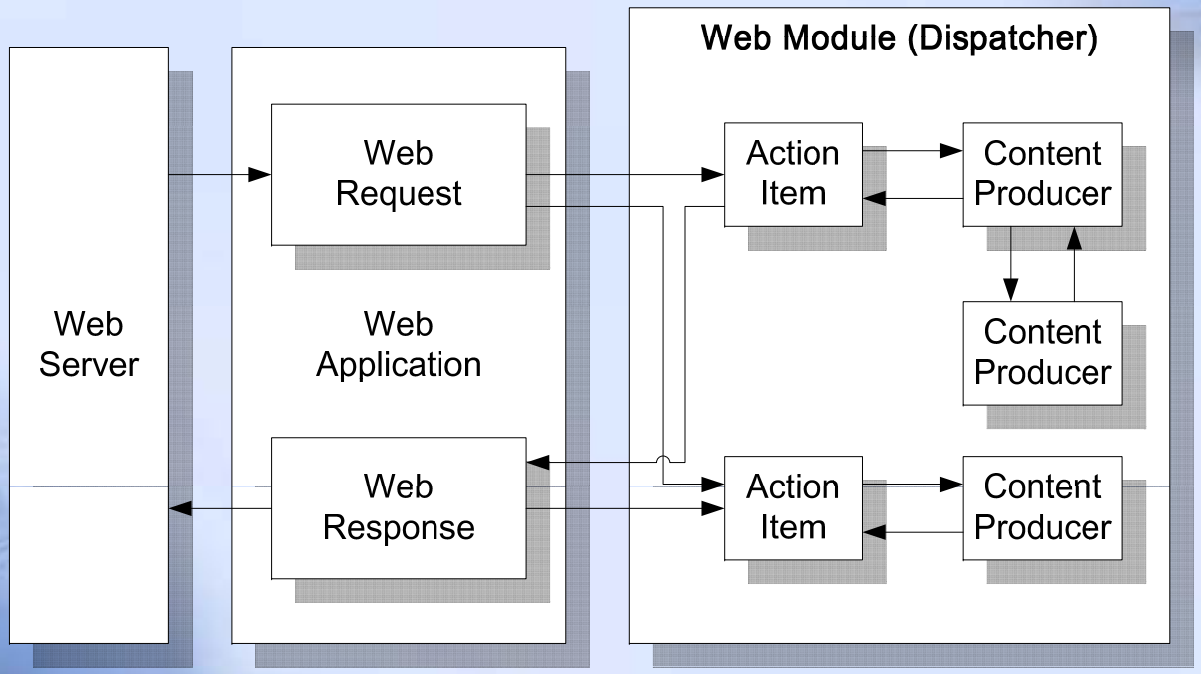
17.1.3 Web服务器程序

- 2. 创建Web服务器程序的基本步骤
  - 在Delphi中可以创建上面介绍的四种形式的Web服务器程序。可供选择的种类是：ISAPI and NSAPI、CGI、Win-CGI。
  - 选择ISAPI and NSAPI后将项目文件设置成一个动态链接库，并且在这个动态链接库中带有服务器支持的方法。在项目文件中将自动加入有关库（即library）的标题，以标志在动态链接库中引用的库的种类。
  - 选择CGI将把项目文件设置成为控制台类型的应用程序，并且在项目文件的uses子句中加入必需的条款。
  - 选择Win-CGI将把项目文件设置成为Windows类型的应用程序，并且也在项目文件的uses子句中加入必需的条款。这些加入到uses子句中的条款都是表明Web服务器程序种类的。

17.1.3 Web服务器程序

- 3. Delphi中Web服务器程序的结构
  - 在Delphi中当Web服务器程序接收到一个用户请求时，将产生一个TWebRequest对象用于代替用户请求信息，产生一个TWebResponse对象来代替处理用户请求的结果。程序然后将这两个对象传送给Web Dispatcher部件。程序的结构如图所示。
  - 图中所示的Web Dispatcher是用来控制整个程序正确执行的。如图包含在Web Dispatcher中的Action Items用来获得有关用户请求的信息。Dispatcher确定应该采取何种行动来获得信息，以及应该触发何种事件来处理用户请求。触发的这种事件通常可以读取用户请求的细节，然后产生回应信息给出服务器对用户请求处理的结果。特定的一些Cement Producer部件可以产生回应信息，回应信息的格式通常是采用HTML或者其它MIME（Multipurpose Internet Mail Extension protocol，多用途的国际邮件扩充协议）格式的文档。

17.1.3 Web服务器程序



17.1.3 Web服务器程序

- 4. 调试服务器程序
  - 编写完程序后．必须对程序进行调试，才能保证编译后的服务器程序可以正常运行。调试服务器程序有一些特殊的步骤，这是因为它们的执行必须要先接收到用户请求的信息。在Delphi的IDE（即集成开发环境）里并不能执行这些服务器程序，它们的执行必须要有Web服务器的支持。离开Web服务器后的服务器程序将一无是处。
  - 调试ISAPI和NSAPI类型的服务器程序必须在支持这种服务器程序的服务器上进行。可以在服务器中编写一定的代码作为用户请求直接调用由这些服务器程序生成的DLL。调试CGI、Win-CGI类型的服务器程序就比较困难一些，需要在服务器上执行这些程序，并且需要编写代码作为用户请求的信息。

17.2 使用Delphi编写ISAPI DLL

- 这里虽然讲的是ISAPI类型的动态链接库的编写力法，们是对于NSAPI同样适用。这是因为Netscape已经宣布支持ISAPI标准，这就意味着ISAPI已经成为Windows平台下的通用标准，NSAPI将逐渐消失。

- 17.2.1 返回静态页面
- 17.2.2 返回动态页面
- 17.2.3 接收用户输入的ISAPI DLL

17.2.1 返回静态页面

- 这里所说的静态页面，就是针对用户不同的请求返回的是同样的HTML页面，即HTML页面的内容不变。

- 【例17-1】按照下面的步骤编写一个能够返回一个静态页面的ISAPI类型的动态链接库。
- 1. 新建一个项目
  - 在主菜单中选择New，并在出现的对话框中选择Web Server Application选项。
- 2. 选择创建ISAPI动态链接库
  - 在选择服务器程序类型的对话框中选择ISAPI/NSAPI选项。

17.2.1 返回静态页面

- 3. 程序框架
  - 这时将得到一个ISAPI动态链接库的框架，整个框架就包括一个Web Module部件。这个部件和Data Module部件没有多大区别。利用它们，用户可以方便地拖放部件。开发ISAPI动态链接库时可以输出许多函数，这些函数可以生成回应信息。函数的生成是通过Web Dispatcher Action Items实现的，这在后面就会提到。

17.2.1 返回静态页面

- 4. 创建ActionItem
  - 每个Web Module中包含一个缺省的Web Dispatcher部件。这个Web Dispatcher部件在Web服务器程序中起着举足轻重的作用：每个从Web浏览器向ISAPI DLL发出的用户请求都是由Web Dispatcher接收，并且确定由哪一个ActionItem来执行；程序中所有可用的ActionItem都由Web Dispatcher记录，由Web Dispatcher确定是否执行某个ActionItem的onAction事件。这里创建一个ActionItem帮助返回一个响应用户请求的静态页面。

① 双击Web Module部件，激活Action Editor对话框。

② 在Action Editor对话框中单击Add按钮，为程序创建一个ActionItem。

17.2.1 返回静态页面

③ 在Action Editor对话框中选中新建的ActionItem，在Object Inspector中编辑它的属性。

④ 改变ActionItem的Name属性，命名为dllStatic。

⑤ 为这个ActionItem指定一个URI（即Universal Resource Identifier，在Internet上用于标识某个特殊实体的通用资源标识符），可以通过修改PathInfo属性来实现。这里将这个ActionItem的PathInfo属性设置为first。

17.2.1 返回静态页面

- 5. 为ActionItem编写代码
  - 为ActionItem指定URI如同命名一个方法，这样根据URI就可以调用动态链接库中的这个ActionItem来响应用户的请求。至于ActionItem响应用户的请求后要完成什么操作就取决于ActionItem的OnAction事件。这个ActionItem的0nAction事件代码如下：

```
procedure TWebModule1.dllStaticAction(Sender:TObject;Request:TWebRequest;
  Response:TWebResponse;var Handle:Boolean)
var
  HTML:String;
begin
  {下面的语句都是用来产生HTML页面}
  HTML:='<HTML>';
  HTML:=HTML+'<BODY>';
  HTML:=HTML+'<H1>HELLO WORLD WIDE WEB</H>';
  HTML:=HTML+'<P>这是—个ISAPI类型的动态链接库例子</P>';
  HTML:=HTML+'</BODY>';
  HTML:=HTML+'</HTML>';
  Response.Content:=HTML;
end;
```

17.2.1 返回静态页面

- 6. 设置服务器
  - 用户调试编写好的ISAPI动态链接库首先需要一个与ISAPI兼容的HTTP服务器来使用编写好的服务器程序。如果用户的HTTP服务器支持ISAPI，它就会在安装时创建一个目录，专门用于存放ISAPI的动态链接库（以.dll为文件后缀名），用户还需要将这个目录设置为允许动态链接库从这个目录执行。
  - 这里使用的HTTP服务器是Windows 2000系统自带的IIS（即Internet Information Server），在IIS中专门用于存放ISAPI动态链接库的目录是scripts。
  - 将scripts目录（在当前机器上的路径）设置为可执行的。

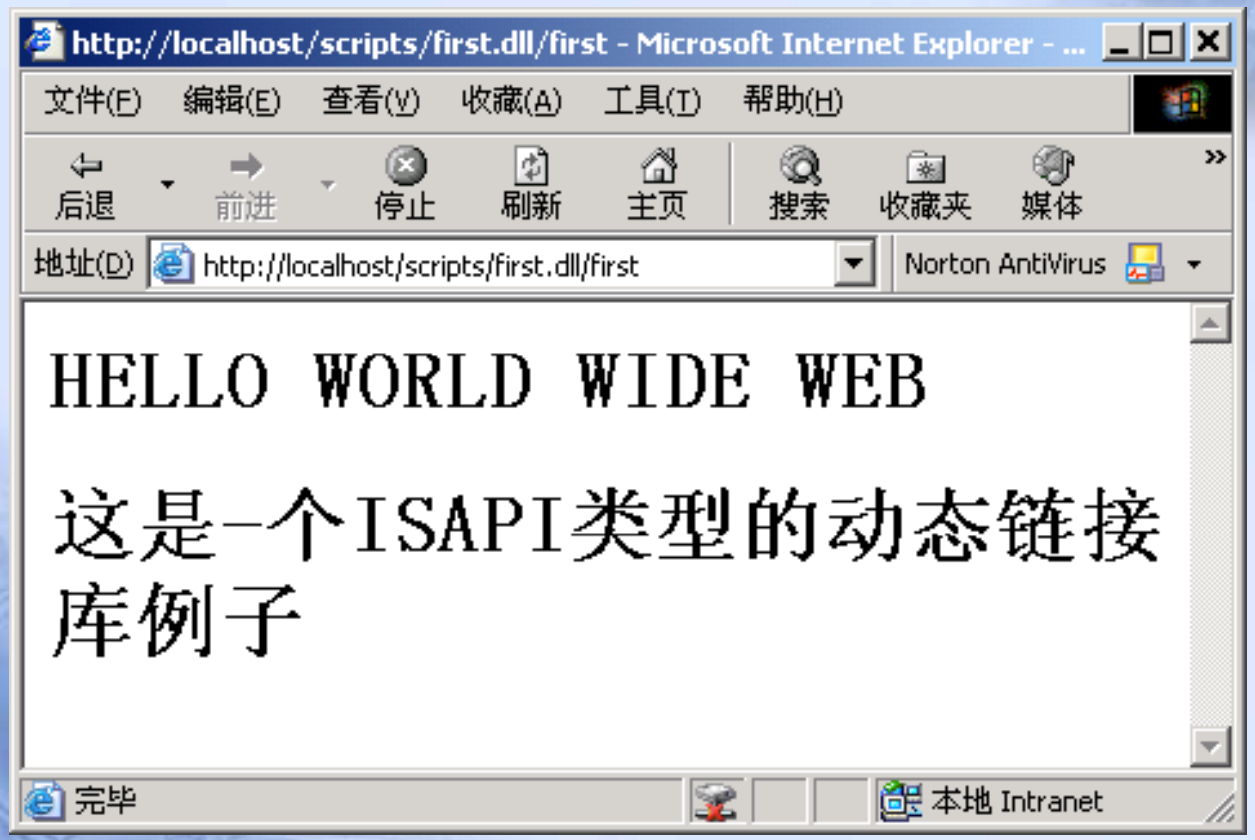
17.2.1 返回静态页面

- 用户调试ISAPI动态链接库还需要知道scripts目录在机器上的实际路径。在调试前，用户需要将编译后的ISAPI动态链接库文件存放在scripts目录。只有在这个目录里，IIS服务器才能执行这个服务器程序。这个scripts目录实际上是一个别名，这个别名表示一定的实际路径，和数据库别名的作用一样。

17.2.1 返回静态页面

- 7. 调试ISAPI动态链接库程序
  - 服务器可以执行的服务器程序是动态链接库，是以.dll为后缀名的文件。在Delphi中编写的ISAPI动态链接库文件还只是一个项目文件，需要进行编译。编译可以选择Delphi主菜单中Project子菜单的Compile first选项(first是项目文件名)，然后将编译后得到的first.dll存放到指定的scripts目录下，实际路径为C:\Inetpub\wwwroot\scripts。
  - 在Internet Explorer浏览器的地址栏中输入http://localhost/scripts/first.dll/first，就会得到如图所示的内容。地址中最后的first就是由ActionItem的PathInfo属性设置的。

17.2.1 返回静态页面


http://localhost/scripts/first.dll/first - Microsoft Internet Explorer - ...
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 前进 停止 刷新 主页 搜索 收藏夹 媒体
地址(D) http://localhost/scripts/first.dll/first
Norton AntiVirus
HELLO WORLD WIDE WEB
这是一个ISAPI类型的动态链接库例子
完毕
本地 Intranet

17.2.2 返回动态页面

- 将服务器程序稍作修改，就可以返回一个动态页面，即在响应每次用户请求时生成的HTML页面不一样。

- 在这个例子里，服务器程序接收到用户请求后，将在生成的HTML页面中显示当前的时间。生成这种动态页面时，用户不需要从用户获取任何输入。日期和时间虽然是动态的，但是和用户请求的类型没有关系。

17.2.2 返回动态页面

- 【例17-2】这个例子完全可以在生成静态页面的例子上进行，需要完成的工作就是在生成HTML页面的语句中加入系统日期和时间。具体步骤如下：

① 在first项目文件中选择Web Module组件，激活Action Editor对话框；

② 在Action Editor中添加一个ActionItem；

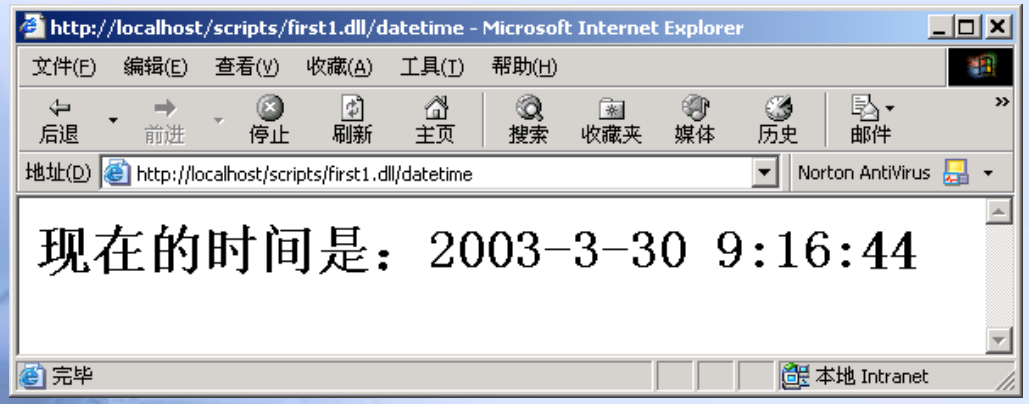
③ 将第二个ActionItem命名为dllDynamic，PathInfo属性设置为DateTime；

④ 为新建的ActionItem的onAction事件编写如下代码：

```
procedure TWebModule1.dllDynamicAction(Sender:TObject;Request:TWebRequest;
  Response:TWebResponse;var Handle:Boolean)
var  HTML:String;
begin
  {下面的语句都是用来产生HTML页面}
  HTML:='<HTML>';
  HTML:=HTML+'<BODY>';
  HTML:=HTML+'<H1>现在的时间是：</H>';
  HTML:=HTML+DateTimeToStr(Now);
  HTML:=HTML+'</BODY>';
  HTML:=HTML+'</HTML>';
  Response.Content:=HTML;
end;
```

17.2.2 返回动态页面

- 加入系统日期和时间可以使用Delphi的Now函数来完成，DateTimeToStr函数是用来将系统日期和时间值转化为字符串用于输出。将编译改进后的ISAPI动态链接库更名为first1，在浏览器的地址选项中输入http://localhost/scripts/first1.dll/datetime，就会得到如图所示的内容。
- 注意：如果运行过first.dll/first，它在服务器的内存中可能还存在着，这是用户需要重新启动一次服务器，然后才可以写入一个新的动态链接库。



17.2.3 接收用户输入的ISAPI DLL

- 1. OnAction事件的Request参数
  - 在前面生成静态、动态页面的例子中，使用了Response参数。OnAction事件的另一个参数Request在接收用户请求、以及确定该进行何种反应起着非常重要的作用。
  - 通过Request参数可以访问用户的输入。同时，Request作为一个TWebRequest对象具有多个属性，这些属性对于获得用户输入的信息特别重要。
  -

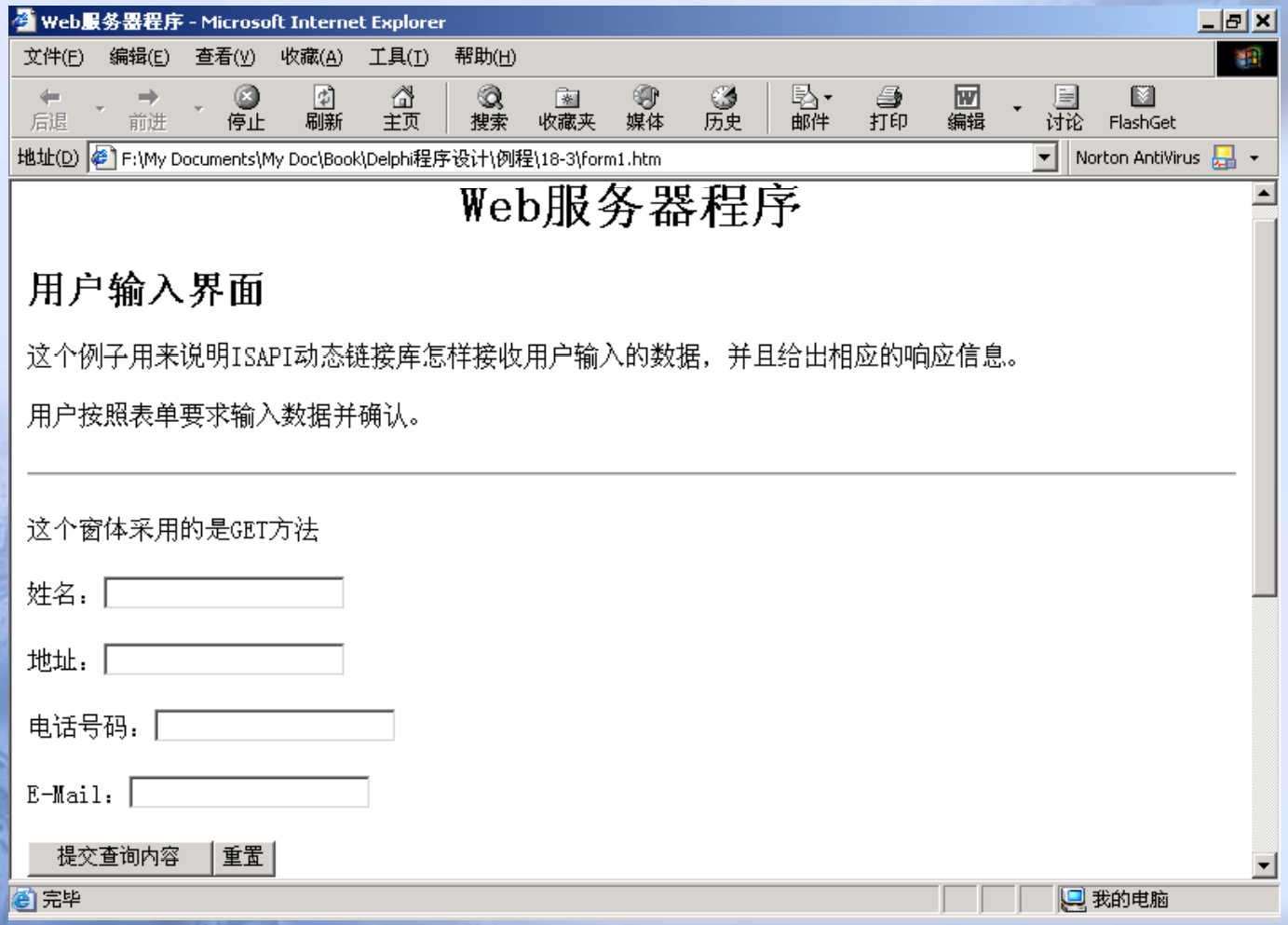
17.2.3 接收用户输入的ISAPI DLL

- GET和POST是HTTP客户向HTTP服务器请求信息时用的方法。GET方法是在浏览器向Internet服务器发送数据时，将数据送入URL端并由HTTP服务程序送入服务器程序的环境变量QUERY-STRING。利用环境变量QUERY-STRING可以获得输入的数据。但由于环境变量QUERY-STRING的最大长度不超过255个字符，所以利用GET方法最多能传送255个宁符。当要传送255个以上的字符时，就需要采用POST方法。缺省情况下浏览器发送给Internet服务器的所有请求均使用GET方法。
- POST方法是更普遍、功能更强大的向脚本传送数据的方法。POST方法在传送数据时，是告诉HTTP服务程序直接将数据送给标准输入STDIN，这正是POST方法和GET方法的区别。使用POST方法的优点是可以传送255个以上的字符。

17.2.3 接收用户输入的ISAPI DLL

- 2. 创建用户输入的HTML页面
  - 为了获得用户的输入信息、确定用户请求的种类，必须将用户输入的数据传送给服务器程序。这就必须在客户端的浏览器中提供给用户输入数据的界面。这种界面通常是HTML页面。
  - 下面这个HTML页面是后面ISAPI DLL服务器程序的用户输入界面。同时为了比较GET方法和POST方法的区别，在这个页面中设置两个窗体并且分别采用这两种方法。创建好的HTML页面如图所示，命名为form1.htm。

17.2.3 接收用户输入的ISAPI DLL



17.2.3 接收用户输入的ISAPI DLL

- 3. 添加PageProducer组件
  - 这个组件通过使用它自己能分析的模板，帮助用户创建动态的HTML输出页面。PageProducer组件将一个文本文件或者一个TStrings对象用作输入，分析输入中特殊的Tags标记，并用运行时刻确定的数据代替这些标记，然后PageProducer组件返回一个完整的HTML页面，由服务器发送给浏览器。
  - 使用PageProducer组件的步骤如下：

① 打开first1项目文件，在原来的基础上进行改进；

② 从Delphi的Internet组件页将PageProducer组件拖入到Web Module中；

17.2.3 接收用户输入的ISAPI DLL

③ 设置PageProducer组件的HTMLDoc属性值输入以下代码：

```
<html>
<head>
<title>Thank you!</title>
</head>
<body bgcolor="#FFFFFF">
<h1 align="center">Web服务器程序对用户请求的响应信息</h1>
<h3>以下是用户输入的数据</h3>
<hr>
图17-13 String List Editor对话框
<table border="0">
 <tr>
  <td>姓名：</td>
  <td><#Name></td>
 </tr>
 <tr>
  <td>地址：</td>
  <td><#Address></td>
 </tr>
 <tr>
  <td>电话号码：</td>
  <td><#Phone></td>
 </tr>
 <tr>
  <td>E-Mail：</td>
  <td><#Email></td>
 </tr>
</table>
</body>
</html>
```

- 这些HTML代码的作用是根据用户输入的数据产生相应的信息。这个文件中的Tags在服务器程序获得用户输入数据和给出响应信息种类起着重要作用。当PageProducer组件处理这个HTML模板时，每遇到一个Tag就触发一个OnHTMLTag事件。OnHTMLTag事件的过程可以用于确定哪个Tag触发这个事件，然后通知PageProducer用哪个值代替这个Tag。

17.2.3 接收用户输入的ISAPI DLL

④ 编写PageProducer组件的OnHTMLTag事件，代码如下：

```
procedure
    TWebModule1.PageProcedure1HTMLTag(Sender:TObject;Tag:TTag;
  const TagString:String;TagParams:TStrings;var ReplaceText:String)
var
  Data:TStrings;
begin
  Data:=nil;
  with Request do
  begin
    case MethodType of
      mtPost: Data:=ContentFields;
      mtGet: Data:=QueryFields;
    end;
    ReplaceText:=Data.Values[TagString];
  end;
end;
```

17.2.3 接收用户输入的ISAPI DLL

- 4. 为项目文件新建一个ActionItem
  - 为项目文件first1增加ActionItem的具体步骤如下：

① 在Action Editor对话框中增加一个ActionItem；

② 将ActionItem命名为dllParseFile，PathInfo属性设置为ParseFile；

③ 在dllParseFile的0nAction事件中增加如下代码：

```
procedure
    TWebModuleI.webModuleIdllParseFileAction(Sender:TObject;
  Request:TwebRequest;Response:TWebResponse;var
   Handled:Boolean)
```

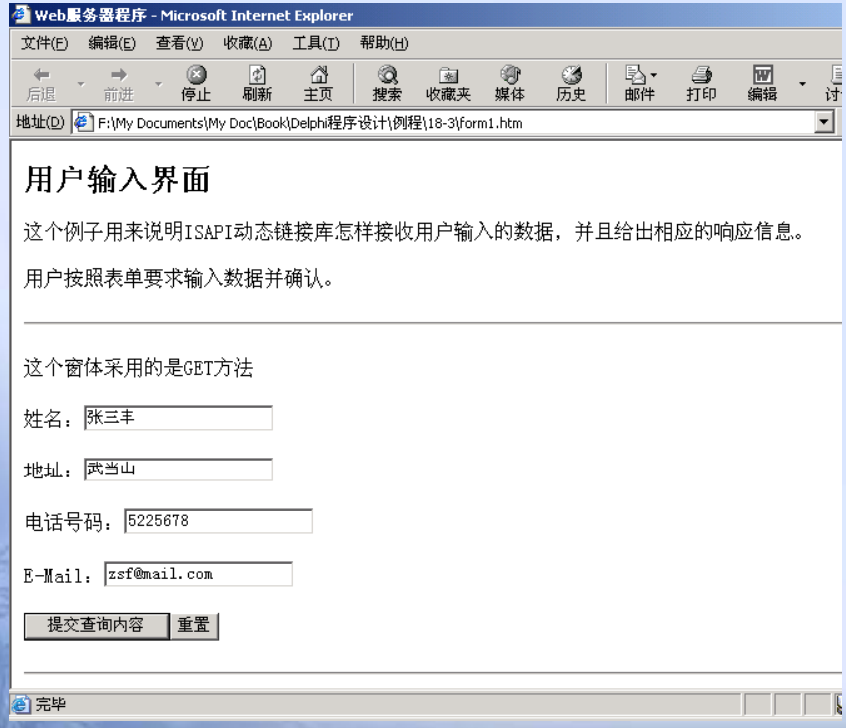
17.2.3 接收用户输入的ISAPI DLL

```
begjn
  Response.Content:=PageProducer1.Content;
end;
```

④ 将修改的项目文件命名为first2；并重新进行编译，将编译后的动态链接库存放到服务器的scripts目录下。

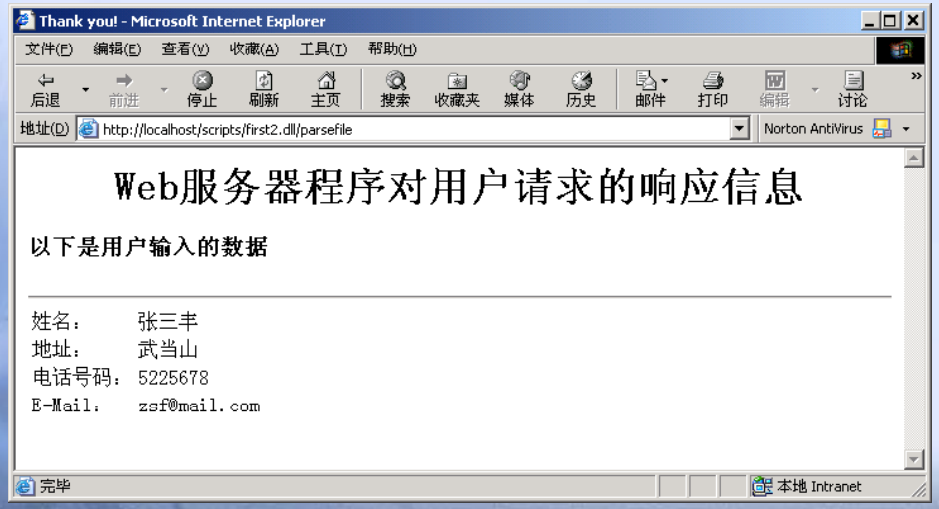
17.2.3 接收用户输入的ISAPI DLL

- 5. 调试程序
  - 将用户输入界面form1.htm存放在服务器上，将窗体Action指向first2.dll；然后在浏览器中访问用户输入界面并输入数据，如图所示。



17.2.3 接收用户输入的ISAPI DLL

- 在用户输入界面的两个窗体中分别输入数据。这两个窗体分别采用POST和GET方式，在提交查询内容后，从图中可以看到这两种方式的区别。从图中可以看到采用POST方式的窗体中参数不会出现在浏览器的地址栏中，这对于保证传送信息的安全是十分重要的。而采用GET方式的窗体中传送所采用的各种参数都出现在浏览器的地址栏中，这对于保证传输信息的安全是十分不利的。



17.3 编写Win-CGI和标准CGI的服务器程序

- 17.3.1 编写Win-CGI和标准CGI的服务器程序
  - 在Delphi中Win-CGI、标准CGI服务器程序和ISAPI DLL、NSAPI DLL拥有一样的程序结构、同样类型的Request、Response对象，从而必然就有一样的编程方法。
  - 编写Win-CGI、标准CGI服务器程序首先也是创建一个新的项目，并选择Win-CGI或者标准CGI类型。接下来的对组件的使用和编写也同ISAPI DLL一样。在程序的调试上，Win-CGI和标准CGI有着不同的方法，这也是在程序设计阶段它们和ISAPI DLL之间的区别所在。

17.3.2 调试Win-CGI和标准CGI服务器程序

- 调试Win-CGI和标准CGI的服务器程序要比ISAPI DLL困难得多，这是因为程序本身必须在Web服务器上运行。因此在调试Win-CGI和标准CGI服务器程序时，必须模拟服务器的行为。

- 调试Win-CGI服务器程序时可以通过写入包括用户请求信息的配置文件来模拟服务器，这种配置文件通常是以INI为后缀名的文件。然后运行Win-CGI服务器程序，它将包含用户请求信息文件的具体路径和Win-CGI要用来产生响应信息的程序的具体路径。这样就能正常调试程序了。

- 另外还有一种调试的方法就是：将Win-CGI或者标准CGI服务器程序先用ISAPI DLL的形式编写一遍，并且进行调试；在调试成功后，将这个ISAPI DLL转化为Win-CGI或者CGI服务器程序。这样做的前提是调试ISAPI DLL要比调试这两种服务器程序简单得多。这种方法的本质就是Win-CGI、标准CGI服务器程序和ISAPI、NSAPI DLL之间的相互转化。

17.3.3 Win-CGI、标准CGI和ISAPI DLL之间的相互转化

- Win-CGI、标推CGI和ISAPI DLL之间的相互转化是基于Delphi中程序模板的创建和对模板的引用。这里通过一个实例来说明问题。
- 这里将先前编写完毕并且调试成功的Web数据库服务器程序second.dll转化成一个Win-CGI的服务器程序，按照如下步骤进行：

① 创建模板。

- 用编写好并且调试成功的second.dll创建一个模扳，以便在Win-CGI中使用。在Web Module组件上单击右键，在如图17-21所示的弹出式菜单中选择Add To Repository选项。
- 创建这个模板后，需要对这个模板进行属性的设置以便在后面进行调用。

17.3.3 Win-CGI、标准CGI和ISAPI DLL之间的相互转化

- 在模板的Title文本框中输入模板的标题dllSecond，在Description文本框中说明这个模板是从ISAPI DLL得来的。在Page列表框中选择Data Modules，因为模板是基于一个Web Module组件创建的，另外，Page列表框中还有Forms、Dialogs、Projects几个选项，根据模板创建的基础进行选择。在Author文本框中输入作者的姓名。通过Browse按钮为这个模扳选择一个图标，以便和其它模板进行区别。确认后，这个模板自动加入到Delphi的模板库中。

17.3.3 Win-CGI、标准CGI和ISAPI DLL之间的相互转化

② 新建Win-CGI项目文件。

- 关闭Second项目，在主菜单中选择New创建一个新的项目，并且选择Web Server Application中的Win-CGI形式。

③ 加入模板。

- 新建项目后，Delphi将自动产生一个缺省的Web Module组件，在加入模板后需要将它删除。首先通过主菜单中的New，然后在对话框中的Data Modules页选择模板dllSecond。

17.3.3 Win-CGI、标准CGI和ISAPI DLL之间的相互转化

- 加入模板dllSecond后，可以看到在程序中增加了如Second项目一样的Web Module组件。删除Delphi产生的缺省Web Module组件Unit1，选择OK按钮确认进行删除。
- 完成加入模板和删除缺省组件的工作后，通过选择主菜单中的Project下的View Source选项，可以查看项目文件内容以确认是否产生需要的Win-CGI服务器程序。

- 标准CGI和Win-CGI的服务器程序要比ISAPI DLL、NSAPI DLL简单。因为标准CGI或者Win-CGI的服务器程序执行的是单线程的操作，不需要面对ISAPI DLL编程中的多线程问题。所以全部的Win-CGI和标准CGI的服务器程序都可以转化成ISAPI DLL进行调试，这样就大大提高Web服务器程序的编写效率。