基于 Delphi Web应用开发

康祥顺 黄显堂 王巍 编著
杨继峰 审校

1CD-ROM

清华大学出版社

基于 Delphi Web 应用开发

康祥顺　黄显堂　王　巍　编著
杨继峰　审校

清 华 大 学 出 版 社
北 京

内 容 简 介

本书主要讲述 Delphi 的功能组件在 Web 开发中的应用，作者采用“原理－实例－分析”的叙述模式，对 Delphi 组件的 Web 应用开发进行了精彩剖析。

全书共分 9 章，内容涉及 WebBroker、MIDAS、Internet Express、MTS/COM+、ASP、ActiveX、WebSnap、Web Services、IntraWeb 等 9 个功能组件，既分析了如何创建传统的 CGI、ISAPI/NSAPI 和客户端/服务器结构的 Web 应用，又分析了如何创建当前最炙手可热的 Web Services、MIDAS 和 MTS/COM+等多层结构的 Web 应用，技术全面、实用性强。随书光盘附赠书中所有实例的源代码。

本书适用于 Delphi 程序开发人员、Borland 产品爱好者以及希望用 Delphi 开发 Web 应用的所有人员。

图书在版编目（CIP）数据

基于 Delphi Web 应用开发/康祥顺等编著. －北京：清华大学出版社，2005.6

ISBN 7-302-11219-3

Ⅰ. 基…　Ⅱ. 康…　Ⅲ. 软件工具—程序设计—教材

Ⅳ. TP311.56

中国版本图书馆 CIP 数据核字（2005）第 064327 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦

http://www.tup.com.cn　　**邮　　编**：100084

社 总 机：010-62770175　　**客户服务**：010-62776969

组稿编辑：夏非彼

文稿编辑：刘秀青

封面设计：林陶

版式设计：科海

印 刷 者：北京科普瑞印刷有限责任公司

发 行 者：新华书店总店北京发行所

开　　本：185×260　　**印张**：22.75　　**字数**：554 千字

版　　次：2005 年 7 月第 1 版　2005 年 7 月第 1 次印刷

书　　号：ISBN 7-302-11219-3/TP・7406

印　　数：1~4 000

定　　价：39.00 元（1CD）

前　言

Delphi 是 Borland 公司推出的优秀的前端开发工具。自 Delphi 7.0 问世以来，其友好的集成开发界面、可视化的双向开发模式、良好的数据库支持以及高效的程序开发和程序运行效率，备受广大程序设计师的好评。

近几年，Delphi 已逐渐成为使用最广泛的编程语言之一。Delphi 不但能够帮助开发者快速创建 Windows 应用程序，还简化了 Windows 与浏览器、Web 服务器、中间件以及后台数据库系统等的集成。Delphi 7.0 是目前惟一支持所有工业标准（XML、SOAP、WSDL、XSL 等）的开发工具，同时它还支持基于 Web 服务的 Microsoft.NET 和 SUN ONE 体系。Delphi 7.0 中包含了 BizSnap、DataSnap、WebSnap 和 IntraWeb 等功能组件，用户可以利用这些组件快速开发出支持 Web 服务的服务器端和客户端应用程序。在网络应用日益普及的今天，利用 Delphi 7.0 提供的开发模式，企业不用丢弃原来的开发方式、开发技巧以及源代码，仅仅是拖放组件和写几行 Delphi 代码，就可以将系统快速转移到基于 Web 服务的网络应用上来，系统在编译时会自动进行转换。

一、本书内容

本书是目前最全面、最深入、最切合实际地分析用 Delphi 7.0 开发 Web 应用的书籍。通过阅读本书，您将学习到以下几个方面的知识：

- WebBroker 在 Web 开发中的应用，包括如何创建 ISAPI/NSAPI、CGI 等类型的 Web 应用；
- DataSnap 在 Web 开发中的应用，包括如何在 Web 应用中使用 MIDAS 技术以及如何开发多层 Web 应用；
- MTS/COM+在 Web 开发中的应用，包括如何在 Web 应用中使用安全、高效的 MTS/COM+中间件技术；
- Internet Express 在 Web 开发中的应用，包括如何结合 MIDAS 和 WebBroker 技术开发多层 Web 应用；
- ASP 在 Web 开发中的应用，包括如何结合 ASP 技术开发基于 Delphi 组件的 Web 应用；
- ActiveX 在 Web 开发中的应用，包括如何开发 Microsoft 提出的 ActiveX 控件的 Web 应用；
- WebSnap 在 Web 开发中的应用；
- SOAP/Web Services 在 Web 开发中的应用；
- IntraWeb 在 Web 开发中的应用，包括如何开发可视化的“所见即所得”的 Web 应用；

- 其他更多内容。

二、开发环境基本配置需求

由于现在 Microsoft 的产品已经占据了市场 90%以上的份额，多数的企业和读者使用的是 Microsoft 的产品，因此，本书所讨论的开发环境配置主要是以 Microsoft 的产品为主。

1. Web 服务器

（1）服务器操作系统

服务器操作系统是整个系统安全、稳定运行的保障。对于用 Delphi 7.0 开发的 Web 应用来说，其操作系统至少需要 Windows NT 4.0 以上版本；如果需要处理中文的话，还应该是中文版本。另外，出于安全的考虑，还应该安装 Service Pack 6。

（2）IIS（Internet Information Server）信息服务器

IIS 信息服务器是整个系统能够成功服务于客户的基础。对 Delphi 7.0 来说，需要 IIS 4.0 以上版本，另外还应该安装 Option Pack 4。

2. 客户端

客户端操作系统应该是 Windows 98 以上，浏览器应该是 IE 4.0 以上。

3. 数据库服务器

本书采用 Microsoft 的 SQL 数据库系统。由于 SQL 数据库与 Windows NT 服务器系统是紧密联系在一起的，它的很多安全管理机制要依靠服务器操作系统，因此如果需要单独使用数据库服务器，请安装 Windows NT 4.0 以上版本的操作系统。至于 SQL 服务器的安装方法，请依照安装光盘的提示。

上面所说的是基本的系统需求。由于目前服务器操作系统不断升级，而且客户端操作系统自 Windows 2000 开始已经集成了大部分的服务器功能，因此，最简单的环境可以是使用一台机器，把本地机器既当服务器又当客户端：先为其安装 Windows 2000 以上的操作系统，然后再安装随盘附带的 IIS 信息服务器以及 SQL Server 2000 系统。本书中的所有实例都可以在单机系统上使用。

但要提醒大家，如果把本地机器既当服务器又当客户端的话，则只能是做开发试验。如果要开发实际上线的 Web 应用，则需要先依照上面的最低配置组建网络系统，经过严格的测试后再上线运行。如果只在本地测试，将有很多意外无法检测到。

4. 开发工具

Inprise Delphi Enterprise 7.0
IntraWeb 7.19
Microsoft FrontPage 4.0

三、光盘源代码使用说明

本书中的实例代码都存放在随书附带的光盘内。光盘内容按照章节存放，比如，第 1

章第 1 节的例子的位置为：第 1 章\1-1，每个例子所用到的文件以及源代码都在该目录下，而数据表则为系统自带的数据表。如果要使用光盘中的实例，可以将相应目录下的所有文件复制到系统中，然后编译运行即可。

本书由康祥顺、黄显堂、王巍编著，由杨继峰审校，参加本书编写工作的还有张小丽、康祥琴、李正非、保春艳、李欣、张庆、崔竞、王涛、刘亮等同志。

本书最后的统稿、协调工作由康祥顺完成。在本书的编写过程中，西北大学的张小刚老师给出了很多的具体指导和宝贵意见，科海电子出版社的王金柱和刘秀青老师为本书的最后出版做出了不懈努力，在此一并表示最诚挚的谢意！对于本书中的所有实例，编者都在 Delphi 7.0 的开发环境中进行过测试。由于水平有限，尽管做了严格的审核和测试，书中难免仍有一些错误，敬请广大读者不吝赐教，编者在此表示感谢！作者邮箱：

KANGXS@21CN.COM

编　者

2005 年 6 月

目　录

第 1 章　WebBroker 技术在 Web 开发中的应用

在 Web 应用开发早期，主要采用静态的 HTML 描述语言编写 Web 页面，然后将 Web 页面以文件的形式保存在 Web 服务器的虚拟目录下。当有客户端向 Web 服务器发出请求的时候，Web 服务器就将相应的 Web 页面传递给客户端，然后在浏览器中显示出来。后来，一些大的浏览器公司（如 NetScape），定义了一套脚本语言（如 JScript、VBScript），运用这些脚本语言便能够开发出动态的 Web 页面。但是，无论哪种脚本语言，其处理复杂问题的能力都较差，而且开发效率十分低下。随着 Delphi 的出现，这些问题就迎刃而解了，特别是 WebBroker 技术，它为用 Delphi 开发 Web 应用奠定了坚实的基础。

早在 Delphi 3.0 中，就已经支持 WebBroker 技术了，它是 Delphi 中 Web 系统开发技术的核心，也是 CGI、ISAPI/NSAPI 等大多数类型的 Web 应用程序的开发基础。只有深入了解了 WebBroker 技术的运作机制，才能够为后面学习 Delphi 的其他 Web 技术打下良好的基础，因此本章的知识尤为重要，它是你进入 Delphi Web 开发的成功之门。通过对 WebBroker 技术的学习，能够使你理解 Delphi Web 应用系统的基本运作原理及过程。

1.1　WebBroker 运作模式及其组件

一个 Web 应用系统基本上包括客户端、Web 服务器和数据库服务器 3 个部分。从系统开发的角度来说，Web 服务器的开发是开发整个系统的关键。WebBroker 技术是采用可视化组件来编写应用程序的，省去了开发人员编写繁琐的 HTML 代码的劳动，而只专心于编写功能应用。它采用了 Web 数据模块的方式，当客户端浏览器发出请求时，Web 服务器就将 HTTP 请求封装在 TWebRequest 对象中，然后传递给 WebModule 对象的 TWebDispacth，它专门负责分配和处理 HTTP 请求。根据请求，呼叫具有相同 PathInfo 的 TWebActionItem 处理程序，并将处理结果封装在 TWebResponse 对象中，返回给 Web 服务器，最后通过 Web 服务器传递给客户端浏览器。图 1-1 是对 WebBroker 运作模式的简单示意。

对于 Web 开发人员来说，仅需要处理 WebModule 部分即可，而 WebModule 部分又可以应用 WebBroker 提供的组件可视化地编辑数据，然后这些组件就可以将与之连接的数据集组件的数据自动转化为 HTML 代码页面，而真正需要我们动手写代码的地方也许仅仅是 TWebActionItem 的处理。下面来看看 WebBroker 都提供了哪些开发组件。WebBroker 可视化开发组件中共有 6 大组件，如表 1-1 所示。

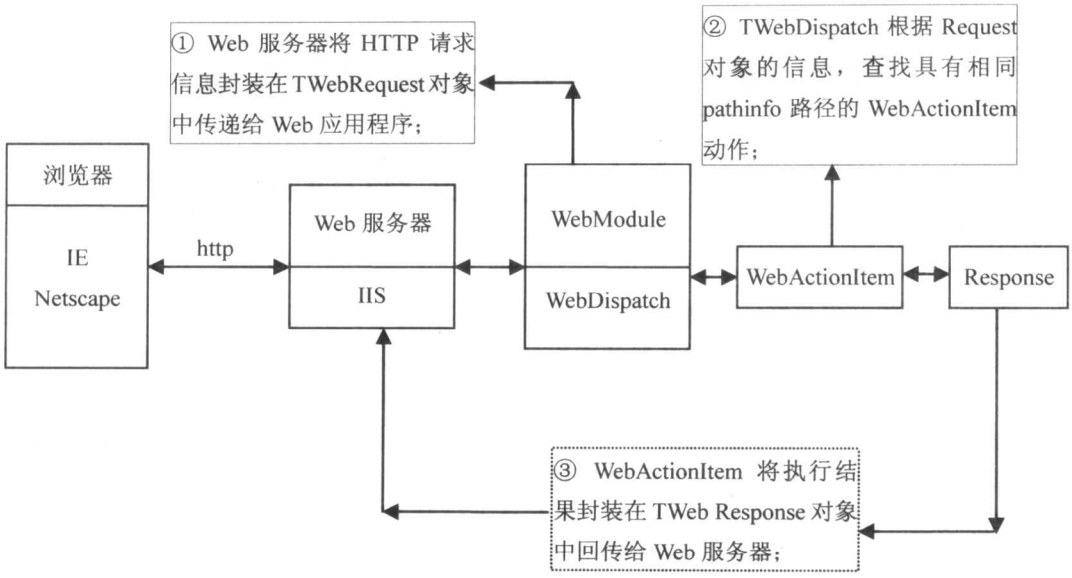

图 1-1 WebBroker 技术运作流程图

表 1-1 WebBroker 组件及其功能描述

组件图形	组件名称	功 能 描 述
	TWebDispatcher	接收和处理 HTTP 请求信息
	TPageProducer	产生静态 Web 页面，也可根据 HTMLDoc 或 HTMLFile 属性生成动态 Web 页面，它是最简单的 Web 页面生成方式
	TDataSetTableProducer	将与之连接的数据集组件的数据自动转换为 HTML 代码页面
	TQueryTableProducer	将与之连接的查询数据集组件 Query、ADOQuery 等的数据自动转化为 HTML 代码页面，有利于处理查询应用
	TDataSetPageProducer	一种 TPageProducer 组件，能够连接任何的数据集组件，详细应用请参照第 7 章相关内容
	TSQLQueryProducer	用于处理 SQL 查询，并将查询结果转换为 HTML 代码页面

这些组件都位于 Internet 组件面板中，后面我们将用专门的例子来讲解如何应用这些组件开发出功能复杂、页面漂亮的 Web 应用程序。

1.2 TWebRequest 对象和 TWebResponse 对象

WebBroker 处理 HTTP 请求的信息都是由 TWebRequest 对象和 TWebResponse 对象完成的。这两个对象是在 httpapp.pas 单元声明的，而 WebBroker.pas 又引用了 httpapp.pas 单元，根据 Object Pascal 语言的多态性，WebBroker 所有组件都封装了这两个对象。

1.2.1　TWebRequest 对象

TWebRequest 对象封装了客户端的 HTTP 请求信息，从图 1-1 可以知道，它主要是将浏览器的请求信息传递给 TWebModule 组件，以便进一步处理请求。在应用程序中，可以根据 TWebRequest 对象读取客户端的请求信息。该对象有两个主要的属性，在高级 Web 开发中会经常用到。

1. QueryFields 属性

该对象封装了客户端 URL 传递的信息，类型为 TStrings，其值的存在形式为 name=value，因此可以通过 Request.QueryFields.values['name']来取得每一个参数值。

例如，客户端传递了一个 URL 请求信息：

```
HTTP://www.Delphi.COM/runQuery?first=kang&lastname=consir
```

那么它被封装在 TWebRequest 对象的 QueryFields 属性中的形式就是：

```
firstname=kang
lastname=consir
```

因此，可以用如下的方式解析出客户端浏览器传递的参数信息：

```
firstname:=Request.QueryFields.values['firstname'];
lastname:=Request.QueryFields.values['lastname'];
```

然后再根据取得的参数信息到数据表中查询信息。另外，该对象还封装了其他的客户端浏览器信息，如 host、address、referee 等等，但是这些信息不一定是完整的，有的可能并没有完全处理，在应用程序中一定要注意这一点。

2. ContentFields 属性

该属性与 QueryFields 属性的功能一样，使用方法也一样。

ContentFields 和 QueryFields 的区别在于：如果 HTML 程序使用的是 get 方法，就应该使用 QueryFields 属性取得客户端请求的信息；如果 HTML 程序使用的是 post 方法，就应该使用 ContentFields 属性取得客户端的请求信息。HTML 程序究竟是使用的那种方法呢？可以在 WebModule 的 ActionItem 组件内的 MethodType 属性中指定。由于默认情况下该属性为 mtAny，也就是可以适用于任何方法，因此在程序中既可以使用 ContentFields 属性，也可以使用 QueryFields 属性。

1.2.2　TWebResponse 对象

TWebResponse 对象封装了 Web 服务器返回的信息。它的最主要也是最常用的属性就是 content，该对象封装了 Web 服务器返回给浏览器的 Web 页面内容。

例如：

```
Response.content:=Request.QueryFields.values['username']+',welcome
to Delphi's world! ';
```

在客户端就会显示信息：kangconsir,welcome to Delphi's world!

另外，TWebResponse 对象也封装了大部分的 Web 服务器的属性和方法，比如 Cookies、version、host 等属性和 SetCookieFields、SendRedirect 等方法。

1.3 TWebAction 的 TWebActionItem 对象

客户端传递的请求信息通过 Web 服务器传递给 WebModule 数据模块的时候，WebModule 会根据所请求的 PathInfo 路径信息来执行相应的动作，即 TWebAction。在图 1-2 中，下面部分就是 WebBroker 的 TWebAction 编辑器。

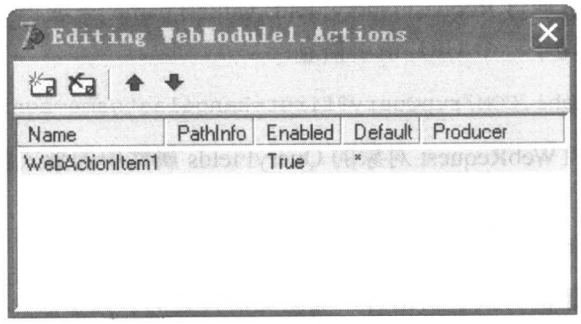

图 1-2 TWebAction 编辑器

TWebAction 包含的最重要属性就是 TWebActionItem 对象。TWebActionItem 包括 PathInfo、MethodType、Producer 等属性。当 WebModule 调用该对象时，首先分析请求信息的 PathInfo 属性，然后将其与 TWebActionItem 列表中的 PathInfo 做对比，如果具有相同的 PathInfo 路径，就执行该 TWebActionItem 的动作，并将执行结果封装在 TWebReponse 对象中返回给 Web 服务器。处理流程如图 1-3 所示。

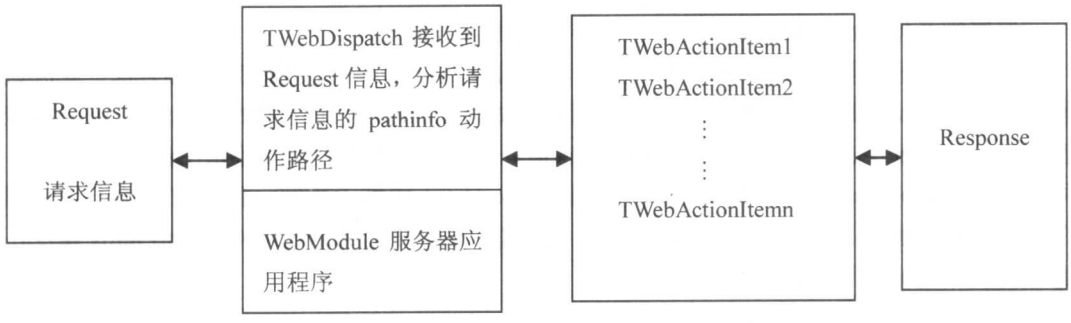

图 1-3 TWebActionItem 对象处理流程图

可以看出，每一个 WebModule 中可以定义多个 TWebActionItem 对象，同时要求有一个且只能有一个 TWebActionItem 对象为默认对象，当没有任何 TWebActionItem 可以处理客户端的请求时，那么就执行默认 TWebActionItem 动作。一般来说，默认的 TWebActionItem 动作会处理第一个返回给浏览器的 Web 页面。若要把一个 TWebActionItem 对象设为默认项，只需要在 TWebAction 编辑器中选择该对象，然后在对象观察器中将它的 Default 属性设为 True 即可。

如果没有在 WebModule 中定义 TWebActionItem 执行动作，那么当浏览器向该 Web 应用程序发出请求时，将不会返回任何页面内容。

当定义了一个 TWebActionItem 对象后，必须定义一个 PathInfo 路径信息（默认对象可以不定义），这样 WebModule 处理客户端浏览器的请求信息时，才能够找到该对象并执行相应动作。这个 PathInfo 属性可以指定任何字符串，比如：/runQuery、/detailes 等。然后要定义这个 TWebActionItem 对象处理的方法类型 MethodType，其适用的类型有 Mtany、Mtpost、Mtget、Mtput、Mthead，各类型的功能如表 1-2 所示。

表 1-2　TWebActionItem 对象处理的方法类型 MethodType 表

类型	功能描述
Mtany	可以处理任何类型的请求信息
Mtpost	浏览器以 post 方式传递信息。通常是客户端向 Web 服务器传递数据
Mtget	使用 get 方式传递信息。通常是客户端以 URL 方式向 Web 服务器发出请求
Mtput	适合 put 类型的请求。通常是当客户端需要以特定的资源来代替请求的 URL 返回资源时使用
Mthead	适合 head 类型的请求。与 get 方式相似，但是它只需要表头信息，不需要主要内容

决定一个 TWebActionItem 对象是否执行的因素，除了 PathInfo 路径信息外，MethodType 也很重要。如果请求方法类型与定义的方法类型不一致，对象也将不会执行。例如：

```
<form method="post" Action="/pwb26.wb26/runQuery">
```

在这个代码中，定义的是 post 方法。但在 TWebActionItem 中如果定义 MethodType 为 get，那么当应用程序处理用户请求时，将不会执行该动作。因此，最好将 MethodType 定义为 Mtany，以便适用所有的方法类型。

还有就是要为 TWebActionItem 对象定义处理方式。处理方式有两种。一种方式是指定 Producer 属性，例如将 Producer 属性指定为 PageProducer 对象，那么当执行该动作时，就会返回 PageProducer 对象的内容给浏览器；另一种方式就是指定该动作的处理事件 OnAction，该事件定义如下：

```
procedure TWebModule1.WebModule1WebActionItem1Action(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: BoOLEan);
```

例如，可以在该事件中指定返回的页面内容：

```
procedure TWebModule1.WebModule1WebActionItem1Action(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: BoOLEan);
Begin
    Response.content:=PageProducer1.content;
    Handled:=True;
end;
```

在上面代码中，就是将 PageProducer1 页面的内容返回给客户端，而且最后必须告诉 WebModule 是否完成了页面处理：Handled:=True。如果不指定 Handled 属性，WebModule 会以为该动作没有处理完请求信息，它会执行默认的 TWebActionItem 动作，返回默认页面

给浏览器，此时无论哪个路径的请求都始终返回同样的 Web 页面。

1.4 实际开发一个 WebBroker 应用程序

前面说了很多的内容，这些对于 WebBroker 开发甚至后面的 Web 开发都非常重要。但是你也许还意识不到这一点，没关系，现在就来实际开发一个 WebBroker 应用程序，以便验证上面所讲的内容。

要开发一个 WebBroker 应用程序，先在 Delphi 集成开发环境下选择菜单栏的 File→New→Other 命令，在打开的 New Items 对话框中选择 New 选项卡，然后选择 Web Server Application 图标，如图 1-4 所示。

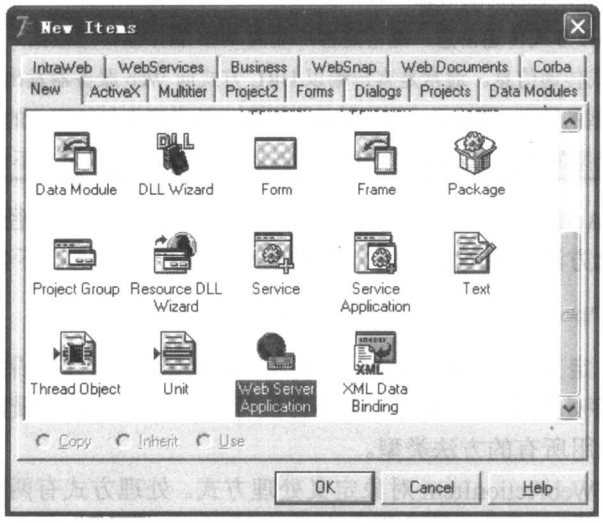

图 1-4 创建一个 Web Server Application 类型的应用程序

单击 OK 按钮后，系统将弹出 New Web Server Application 对话框，如图 1-5 所示。在这个对话框中，所有类型的 Web 应用程序都以相同的开发界面呈现出来，使得开发任何类型的 Web 应用程序的工作都是一样的，而你要做的就是选择开发何种类型。

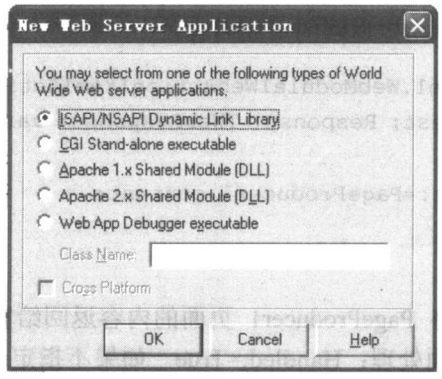

图 1-5 New Web Server Application 对话框

在图 1-5 中可以看到，我们可以开发的 Web 应用程序有 ISAPI/NSAPI、CGI、Apache、WAD 等类型。各种类型的应用程序各有优缺点。

- ISAPI 是在 Microsoft 浏览器中执行的应用程序，是 DLL 形态的进程内（in-process）执行文件，对象一旦创建，直到服务器关闭该应用程序才会停止，这样所有的客户端都使用同一对象，因此服务器处理客户请求的效率较高。但是一旦该应用程序出现故障，那么所有的客户端都将无法再访问。
- NSAPI 与 ISAPI 功能相同，只不过它是在 NetScape 浏览器中执行的应用程序。由于 Delphi 系统作了相关处理，使得这两种类型的应用程序可以在对方的浏览器中执行。
- Apache 类型的应用程序是专门在 Apache 服务器中使用的。
- CGI 类型的应用程序实际上是一个 exe 可执行文件，它为每一个访问的客户端创建一个应用程序的实例，各自在自己的空间中运行，相互间不干扰，因此性能比较稳定，但是，每次访问它都需要重新创建应用程序的实例，其执行效率较低，也比较浪费资源，是目前很多服务器支持的 Web 应用程序类型。
- Web App Debugger（WAD）类型的应用程序是调试类型的应用程序。在 Delphi 6.0 以后版本中，为了方便调试 Web Server 应用程序，加入了 Web App Debugger 调试工具。该工具相当于一个模拟的 Web Server 服务器，只不过它可以自动结束一个进程。该类型的应用程序实际上是一个 COM 组件，可以在 Web 服务器上执行，但比较麻烦的是必须将 WAD 调试器打开，因此最好将应用程序开发成 WAD 类型，然后在调试器中进行测试，通过之后再转换为其他类型的应用程序。不过要注意，WAD 调试器对于有的错误信息会自动跳过，因此，如果要真正测试，还应该转换为其他类型后在 Web 服务器环境下测试，一切无误后，再正式上线运行，以免发生不必要的系统错误。

如图 1-6 所示，创建一个 WAD 类型的应用程序，命名为 WB24。

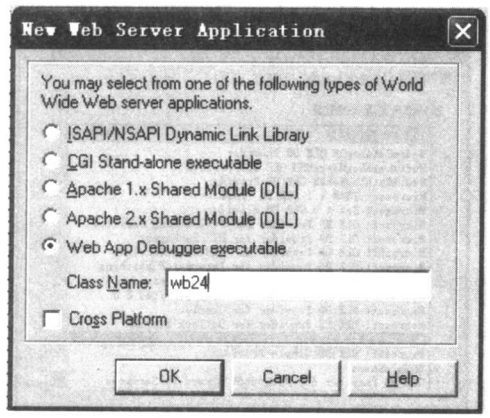

图 1-6　创建一个 WAD 类型的应用程序，命名为 WB24

单击 OK 按钮，系统将自动创建一个空白的 WebModule 数据模块，我们可以在这个模块中添加任何数据集组件和 PageProducer 对象。由于本例是将 SQL 服务器中 NorthWind 数