技术文档

版本 1.0.0

昆明光标科技有限公司
翻译：杨义金（Steven Yang）
联系电话：86-0871-5184866
13769174101
Email: kmgbkj@126.com

目录:

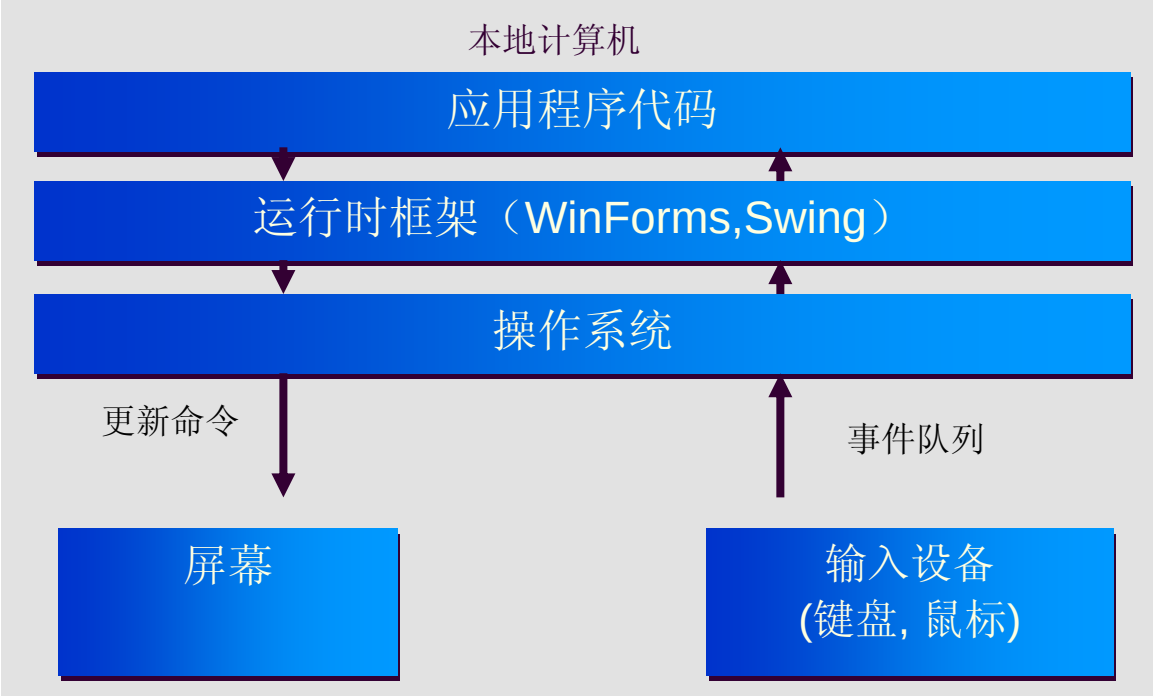
1 简介

1.1 什么 Visual WebGui?

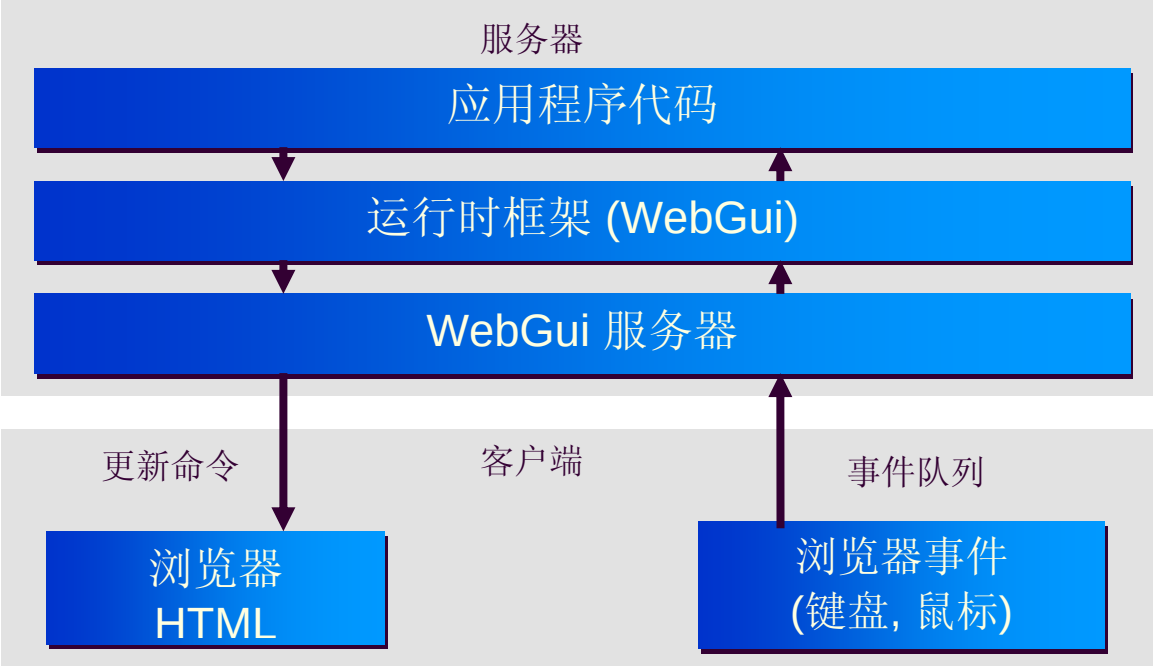
1.2 Visual WebGui 服务扩展

Visual WebGui 是一个实现了 IHTTPHandler 的服务器扩展（ISAPI 过滤器的.NET 执行方式），它提供了另一种 HTTP 管道来更好的适应 WEB 应用程序的开发需要。Visual WebGui 服务器扩展在应用软件代码层和表现层之间提供了一个抽像（层），管理所有的资源和请求。应用软件代码层指依赖 Visual WebGui SDK 创建的代码，表现层指浏览器或其它远程表现层。

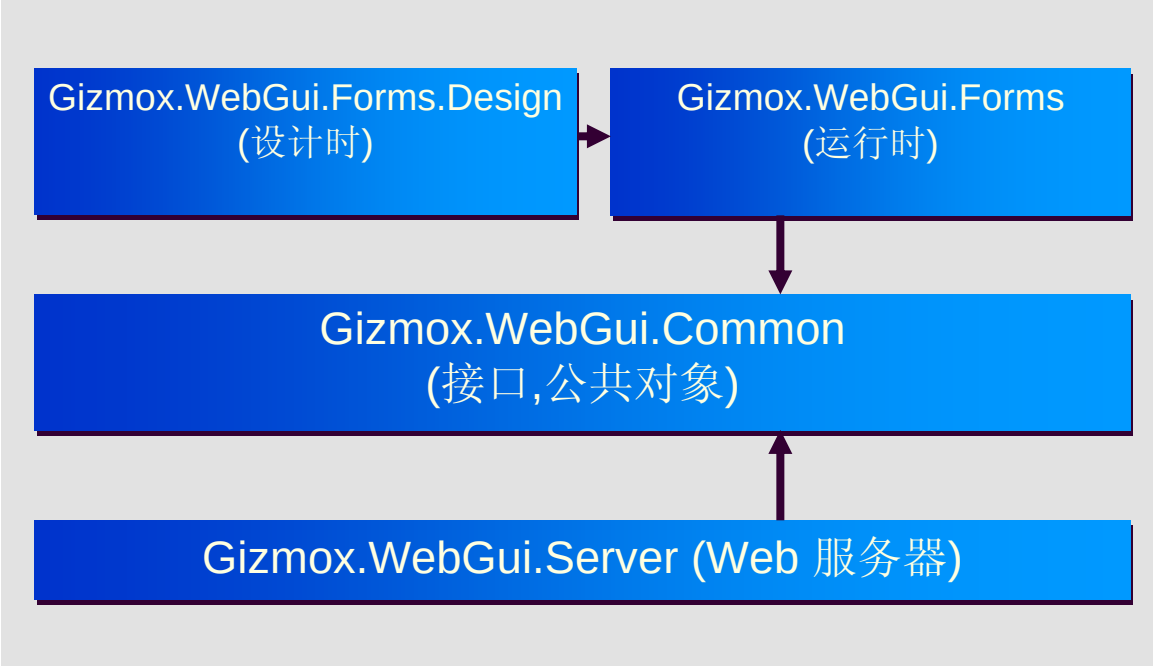
下图展示了桌面环境结构：



下图展示了 Visual WebGui 是如何通过 HTTP 实现桌面架构：



Visual WebGui 服务器扩展完全包含在 WEB 服务器（HTTPHandler）中，使得 Visual WebGui 应用程序就像普通 WEB 应用程序一样安全，并用相同的方式管理他们。运行 Visual WebGui 应用程序的服务器宿主不必安装 Visual WebGui 组件在服务器上，可以当作在 Bin 目录下运行的一般组件，这使得发布 Visual WebGui 应用程序变得非常简单，只需拷贝装配件（Dll 类库）到 Web 应用程序目录下的 Bin 目录即可。Visual WebGui 服务器扩展和 Visual WebGui 的通信机制降低了对服务器资源的使用和后台支撑服务的使用，这使得在一台机器上支撑更多的用户负载成为可能。
Visual WebGui 只产生很少的输出响应和解析非常少的输入请求，带宽使用率非常高效。
Visual WebGui 服务使用每一个 SDK 控件自身的资源来为表现层服务，Visual WebGui 服务通过资源的继承或重载来支持多种设备和不同应用程序不同外观风格的要求，Visual WebGui 服务管理的资源被自动缓存在 WEB 服务器和客户端浏览器中。这些被缓存的资源用于优化带宽占用率和加快客户端响应，Visual WebGui 服务负责处理界面布局、界面增量更新和事件队列，把资源以响应的方式重绘到提定的客户端区域，以及响应客户端事件（非常类似于面向桌面的控件事务处理机制）。



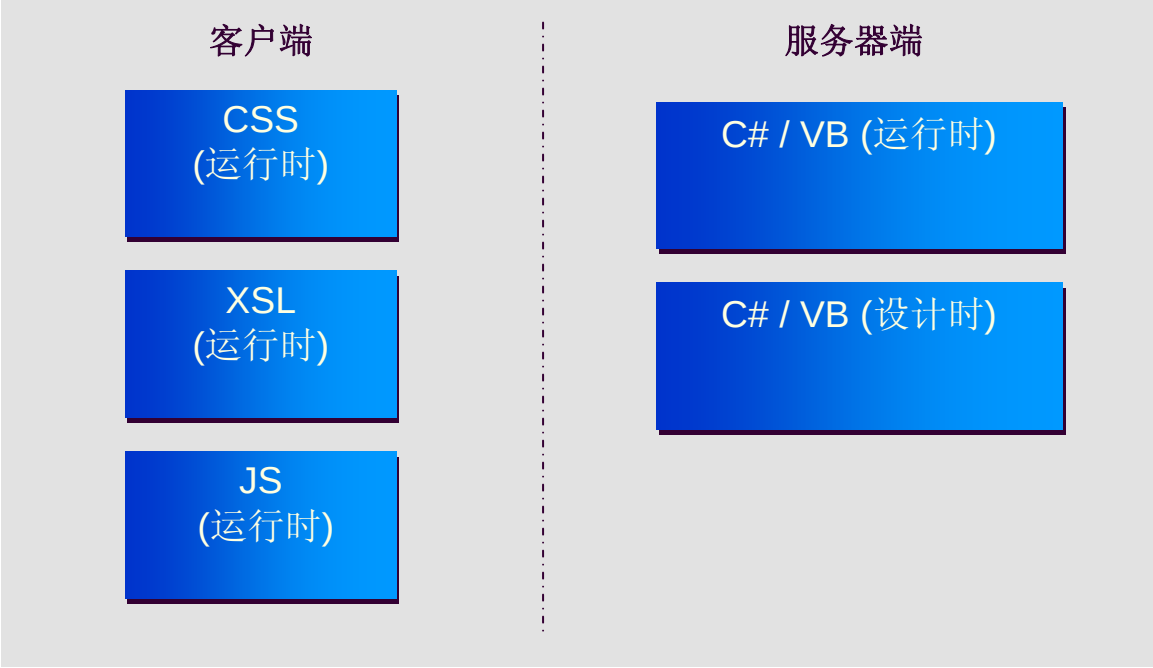
Visual WebGui 服务扩展以接口形式从 SDK 中划分出来，允许应用程序代码作为离线应用程序（此功能是企业版 SDK 的一部分）。由于 Visual WebGui 应用程序被设计为基于 WinForms/Swing API，可通过 Visual WebGui 在线应用程序生成对应的离线版本，这就允许离线版本和在线版本在 GUI 级共享代码，而不仅仅只能共享业务逻辑。在数据访问层使用提供者（Provider）模式，在线应用和离线应用可完全共享相同的用户接口（UI）。

Visual WebGui 服务扩展能够聚合调试追踪器和性能指示器来帮助开发人员排除应用障碍，详细到控件在什么时刻在客户端浏览器被绘制的所有信息都可以被收集和分析，可以节省大量调试、监视追踪时间。由于 Visual WebGui 是基于事件驱动环境的，追踪也在此层次完成所有的工作，在事件处理时和重绘时的事件触发动作过程中收集信息。

1.3 Visual WebGui SDK

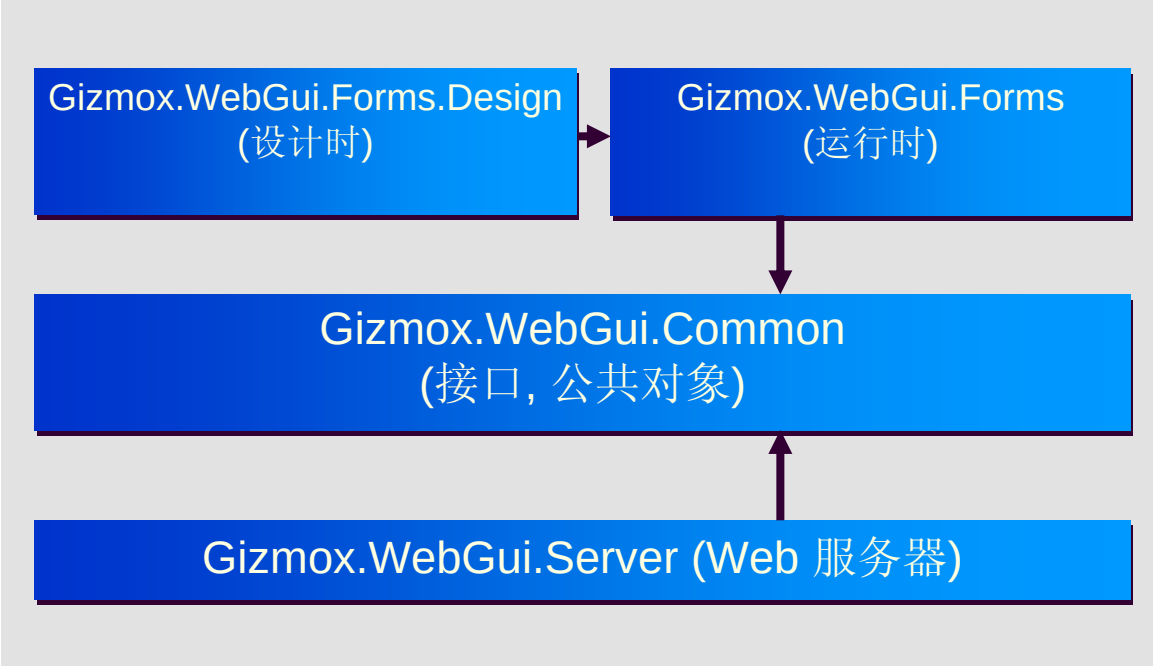
开发人员开发 Visual WebGui 应用程序时使用的就是 Visual WebGui SDK，Visual WebGui SDK 包含了大部分对应的 WinForm/Swing 控件，WinForm/Swing 开发人员很熟悉这些控件。完全支持 WinForm/Swing 控件的设计时行为，WebGui 使得开发人员在两种不同的应用架构（窗体应用和 WEB 应用）下具有相同的“用户体验”。WebGui 开发人员工作在一个抽像层上，此抽像层不用考滤 WebGui 服务的消息通讯和缓存细节。WebGui 服务依赖 SDK 的使用情况（设计时和运行时）来处理和优化消息通讯和缓存，开发人员不用考滤低层是如何构建 WEB 机制等细节情况。

WebGui SDK 为开发 WEB 应用程序提供了全面的抽像，开发人员不需要了解低层的 Web 技术细节，除非他们想扩展 WebGui SDK 或者是想开发额外的外观主题以适应不同的界面外观需求。一旦计划要扩展 WebGui SDK 或新建一个外观注题，开发人员无需从头设计和开发，WebGui 服务器和 WebGui 客户端内核服务已处理了大部分任务。WebGui 客户端内核服务提供的服务有跨浏览器支持、产生服务器端事件、改变客户端状态、样式和快捷菜单等一些常见功能的支持等;WebGui 客户端内核服务显著地降低了扩展和编写 WebGui 资源的总体时间。



WebGui SDK 包含运行时类库、设计时类库和一组客户端内嵌资源。运行时类库用于创建和运行 WebGui Web 应用程序;设计时类库为开发人员提供了类似 WinForms/Swing 控件相同外观和设计时行为的一组设计时控件;客户端内嵌资源是一组为支持客户端控件行为和多设备支持并按指定规则预先格式化的资源。

WebGui SDK 资源借助面向对象的多态机制被设计为很容易编写适应特定浏览器的实现，在基类编写一次所有浏览器的共用代码，并在继承类上添加或重载实现各个特定浏览器代码。而且 WebGui SDK 资源可与 WebGui 客户端浏览器内核服务交互，这减少了开发人员在扩展 WebGui SDK 时所需管理的代码量。



WebGui SDK 并不引用 WebGui 服务器对象，相反，WebGui SDK 只与在 WebGui SDK 和 WebGui 宿主提供者之间的一组公共接口交互，这使得 WebGui 应用程序可通过不同的宿主提供者布署到不同的宿主中，此架构提供对单机桌面安装的支持。

WebGui SDK 内部提供了开发外观主题和多设备支持的灵活机制，WebGui SDK 提供了类似.NET 框架的资源后置机制，因此，在需要的时候重载基类资源集来创建新的资源集成为可能。依据这种资源继承机制，当创建新的 WebGui SDK 外观主题时，不必为所有组件创建实现，只需重载特定组件的设计时或客户端行为就行了。与此类似，如果为了支持多设备而实现的组件资源，也只需从基类代码继承，扩展或创建实现对特定设备（终端）的支持。

1.4 Visual WebGui 和 WinForms 的设计区别

WinForms 应用程序和 Visual WebGui 应用程序最重要的区别就是他们的运行环境不同。Visual WebGui 是一个搭建为多用户服务的多线程环境的 WEB 应用程序平台，而 WinForms 应用程序在特定机器上只服务于单个用户。

WinForms 应用程序和 Visual WebGui 应用程序的一个本质区别就是窗体对象显示模式窗体的（ShowModal）方法的执行方式。在 WinForms 应用程序中调用此方法后将中断正在执行的代码，直到窗体被关闭;而在 Visual WebGui 应用程序环境中，显示模式窗体的方法（ShowModal）调用后会立即返回，并且开发人员必须处理窗体关闭（Closed）事件来继续执行窗体关闭后的事件，也就是说在 Visual WebGui 应用程序中，模式对话框只在客户端是模式的。

除了窗体的显示模式窗体（ShowModal）方法的执行方式不同外，设计一个 Visual WebGui 应用程序和设计一个 WinForms 应用程序的步骤和经验基本上是相同的。只是 Visual WebGui 应用程序是一个多线程执行环境，所以在实践过程中要注意几个方面：静态成员应该是多线程访问安全的;像

数据库连接和内存等资源应该在使用后尽快释放;按照“Visual WebGui 最佳实践”的指导处理事件将优化服务器往返和在服务器上发生的改变而导制发送到客户端的更新命令。

Visual WebGui 消除了 Web 应用程序天生的 请求/响应 模式的特点，取而代之的是一个全面的事件驱动环境。Web 开发人员一般在页面级别定义像数据库连接这样的资源，并在请求处理结束时释放资源; Visual WebGui 没有页面和请求的概念，开发人员应把一个事件处理当作是一个页面。比如说：在一个按钮的事件处理函数中要从数据库中取出数据，并把这些数据更新到 ListView 中，数据库连接应该在按钮事件处理函数的开始打开，并在事件处理函数结束时关闭数据库连接，局部变量将自动在方法执行到最后时被释放，用这种方法来管理资源的生命周期将是一种好的方案。

WinForms 应用程序和 Visual WebGui 应用程序的另一个区别是网关。网关是 WEB 应用程序中根据上下文环境处理请求的处理器。例如：有个列表需要为打印（输出响应）页面预先打开，就需要使用网关来处理。要支持网关的控件必须实现“IGatewayCotnrol”接口，此接口有一个方法返回一个“IGatewayHandler”接口实例，此接口负责处理网关请求。此概念与 ASP.NET 管道中的“IHTTPFactory”和“IHTTPHandler”之间的关系相同。

1.5 Visual WebGui 最佳实践

“Control.Update()”方法用于重绘控件，在控件上调用此方法将把控件及控件的所有下级控件添加到更新命令中。并谨记只在 WebGui 有更新 Bug 时才直接调用此方法，并在 Bug 解决后移除对此方法的直接调用。
“TabControl”是解决浏览器负载和增强局部更新的一个极好的办法。“TabControl”可以是逻辑的，也就是说“TabControl”可以是一个不可见的控件，它使用程序完成页面的切换。在见不到真正的“TabControl”控件的情况下你也可以感受到它的运行性能。
WebGui 应用程序运行在服务器会话中，也就是说每个使用 WebGui 创建的应用程序的用户都在服务器会话中有一个运行的应用程序。不像桌面应用并不以此种方式运行，所以要谨记每个被使用的对象只要不需要了，就应释放其占用的资源，在需要的时候再重新创建。
在用户之间共享的静态变量应该是线程安全的。
应该使用关闭事件取得对话框的返回值（就像 WebGui 的演示片段中所展示的哪样）。
应该小心使用事件，如果定义了事件处理程序，一旦有事件发生，将迫使服务器发送返回数据。考滤使用事件的队列版本，或在可能的情况下就删除事件处理程序。重写事件为队列事件却并不引起用户界面改变将是一种理想状况。
WebGui 自动检查 GUI 的改变并确定更新的最佳方法，GUI 控件更新的范围应该越小越好。
例如：
如果要更新“ListView”控件的一项，最好是只更新需要更新哪个“ListViewItem”，而不是重新加载整个列表。WebGui 会检测更改并创建适合客户端的更新命令，更新“ListView”的整个列表将导制重绘整个控件。

2 安装 Visual WebGui

Visual WebGui 安装分为两个安装包，一个是 SDK 安装包，另一个是服务器组件安装包。

服务器组件安装包只包含 Visual WebGui 运行时组件，只在发布 Web 应用程序时使用这些组件。如果在每个工程项目中嵌入所有 Visual WebGui 运行时组件就可以不用安装服务组件安装包。

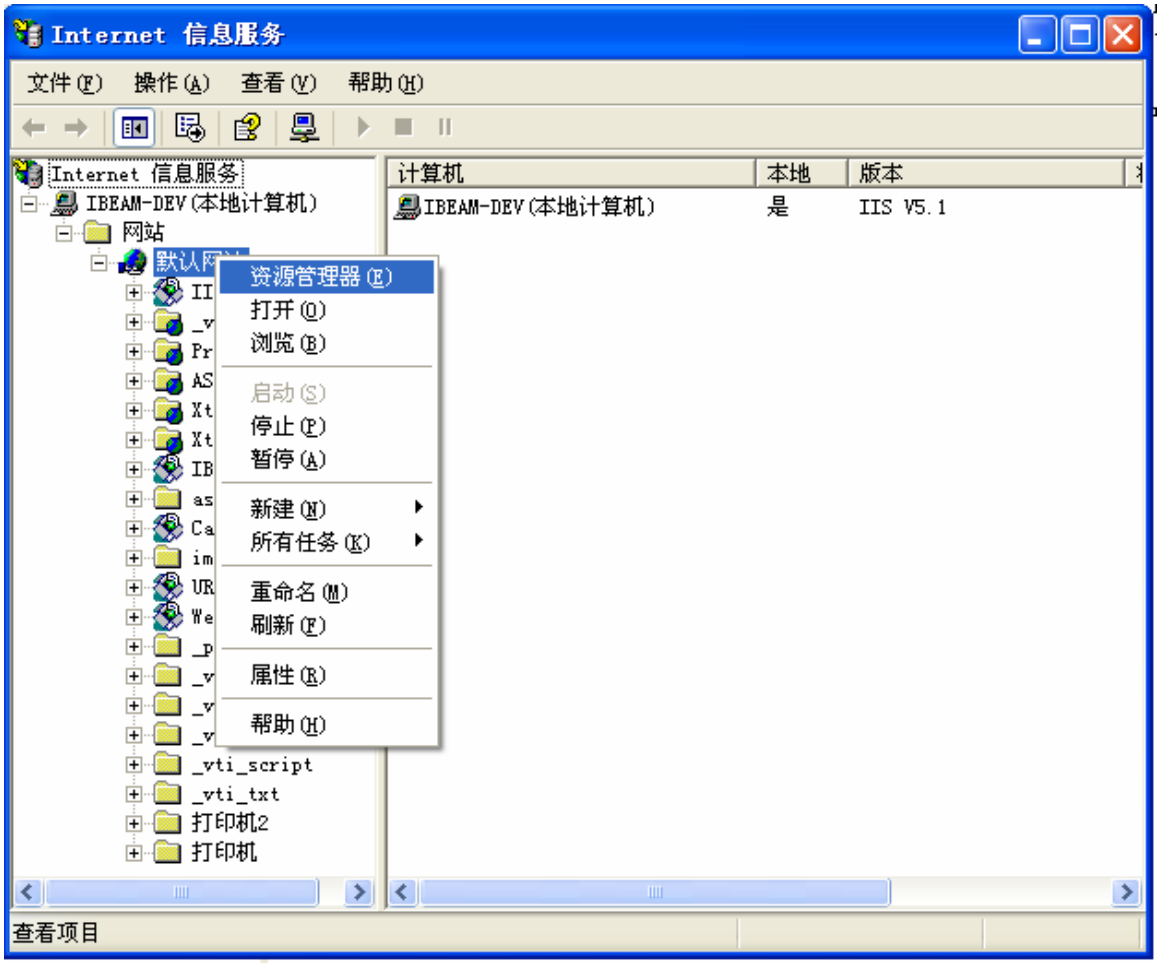
SDK 安装包包含运行时组件和集成到 Visual Studio 的设计时组件，SDK 安装包还分发了在 Visual Studio 中创建“Visual WebGui”应用程序的项目模板。

2.1 安装先决条件

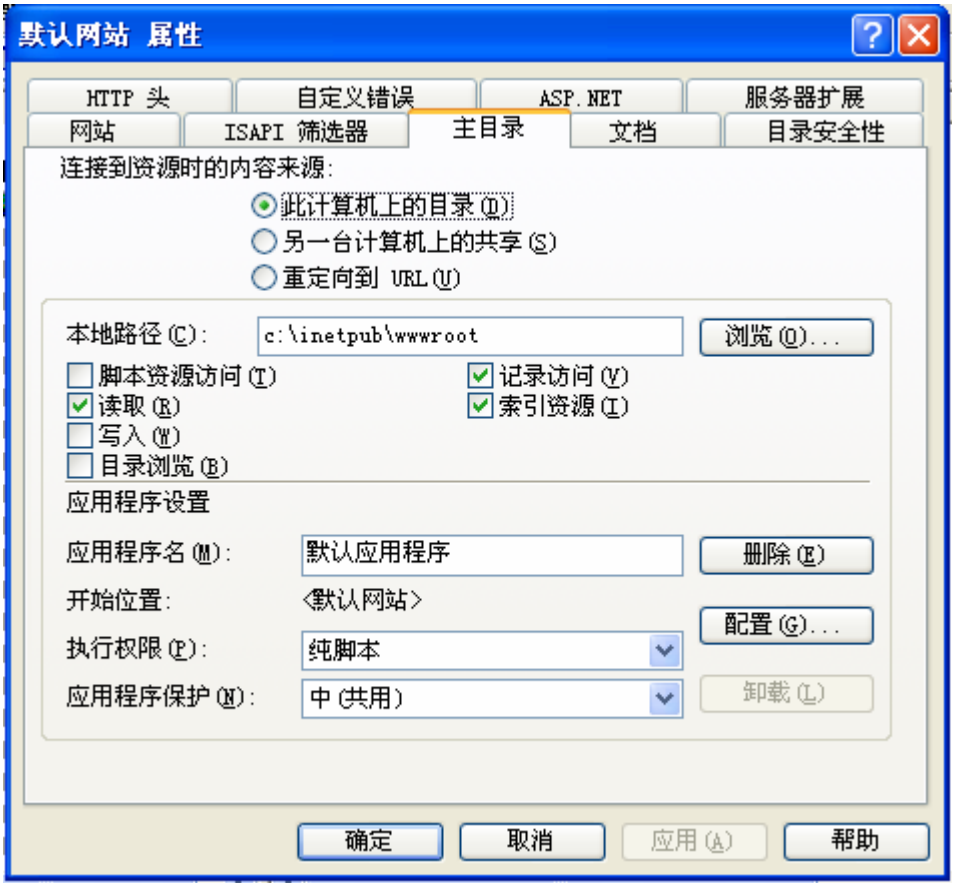
由于 Visual WebGui 应用程序实际上是 ASP.NET 应用程序的扩展，所以 Visual WebGui 的安装要求和 ASP.NET 的安装要求是相同的。在 Visual Studio 2005 平台下还需要安装微软的更新以支持 WEB 应用程序;Visual Studio 2003 也需要安装“wgx”脚本映射扩展，“wgx”脚本映射扩展实际上是 ASP.NET“ISAPI”筛选器的另一个扩展。

2.2 安装注册 Visual WebGui 的"wgx" 扩展

Visual WebGui 需要在“IIS”上注册一个叫做脚本映射的扩展，为了安装“.wgx”扩展需要打开“IIS 管理控制台（Internet 信息服务）”，右键单击工作站点显示如下图所示的图像，单击选择“属性”菜单项。

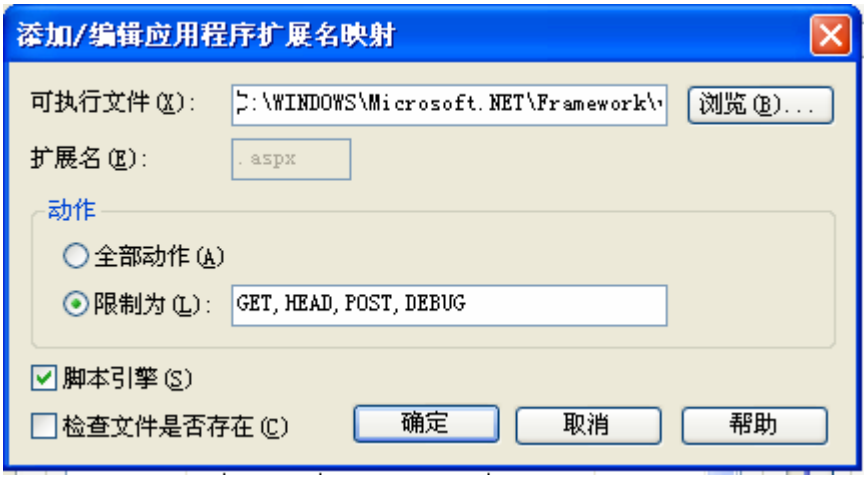


出现对话框后选择“主目录”选项卡并点击“配置”按钮。



单击“配置”按钮后将出现下图所示的对话框，此对话框显示了所有的脚本映射。在“.aspx”扩展名上双击。

显示下图所示的对话框：



从上图所示的对话框中拷贝可执行文件的路径（不知为什么此对话框不支持键盘命令，所以需要在文本框中单击右键从快捷选菜单中拷贝路径）。注意此路径包含了.NET 框架的版本号，应调整到当前计划使用的版本号，可以为同一扩展名配置两个选项（.NET1.1 和.NET2.0），但需要手动为每一个 WEB 目录指定不同的可执行文件路径。

回到第一个对话框并单击“添加”按钮，粘贴可执行文件路径并设置扩展名为“.wgx”，输入如上图所示的动作的“动词”或改变单选按钮到“全部动作”选项，由于 Visual WebGui 并不使用实际的页面，所以取消选择“检查文件是否存在”就变得非常重要。在所有显示的对话框中单击“确定”按钮，这样就成功安装了“.wgx”扩展。

2.3 在 Visual Studio 2003 下安装 Visual WebGui SDK

在 Visual Studio 2003 下安装 Visual WebGui 十分简单，参考“安装先决条件”查看是否可以安装，基本上只要可以创建 ASP.NET 站点就可以安装 Visual WebGui 了。安装 Visual WebGui 的.NET1.1 版本是一件简单而快速的过程，安装过程结束后就具备开发条件了。此安装将安装运行 Visual WebGui 应用程序必须的运行时组件和 Visual Studio 扩展。由于 Visual Studio 2003 使用“IIS 管理控制台（Internet 信息服务）”调试 Web 应用程序，为了方便调试和跟踪，务必要在“IIS 管理控制台（Internet 信息服务）”上添加“.wgx”扩展名的脚本映射。

2.4 在 Visual Studio 2005 下安装 Visual WebGui SDK

在 Visual Studio 2005 下安装 Visual WebGui 稍微有点困难和耗时，因为 Visual Studio 2005 创建了基于页面结构的简单一步式工程项目的新概念来开发 Web 应用程序，和 Visual WebGui 的无实际页面文件的概念并不相同（Visual Studio 2003 是这样工作的）。

实际应用中，为了解决上面的问题，可以把 Visual WebGui 应用程序编译成组件（类库），再在 ASP.NET 项目中引用组件的方法来解决，但使用原来的 WEB 工程项目（Visual Studio 2003 的方法）的方法为最佳方法，幸好微软公司提供了升级以使用开发人员可以按以前的 WEB 工程项目的方式来开发 WEB 应用程序，只是需要花时间安装两个升级。好的方面是这些升级只需要安装一次就可以了，Visual WebGui 安装程序会指出下载这些升级的 URLs，所以您不用担心找不到升级。

安装好升级后就可以安装 Visual WebGuib,安装程序在添加 Visual Studio 模板时要持续一到两分钟时间。一旦安装好必需的运行时组件和设计时组件后，就可以开始开发 Visual WebGui 应用程序了。 如果打算使用“Internet 信息服务（IIS 管理控件台）”来代替内置的调试服务器，还需要在“Internet 信息服务（IIS 管理控件台）”上注册“.wgx”扩展。

2.4.1 安装 Web 应用程序更新 (Visual Studio 2005)

在.NET 2.0 下，Visual WebGui 需要安装“微软 WEB 应用程序更新（Microsoft's Web Application update）”，以使 Visual Studio 2005 可以创建 Visual Studio 2003 哪种在桌面工作环境下更直观的 WEB 应用程序工程，Visual Studio 2005 创建的 WEB 工程文件是以页面为中心的，却并不非常适合 Visual WebGui。也就是说仍然可以在 Visual Studio 2005 下创建基于页面环境的 ASP.NET 工程，同时又可以应用 Visual WebGui，可通过在工程中实现 Visual WebGui 元素或映射到外部工程达到此目的，但推荐使用映射到外部工程的方法，因为这样可以同时获得两者的优点。
安装 Visual WebGui SDK 的.NET 2.0 版本时检查是否已安装了这个更新，如果没有安装此更新，安装程序会引导您下载并安装。

2.5 安装故障解答

2.5.1 为什么 Visual Studio 2005 安装程序会冻结？

Visual Studio 提供了方便的“添加应用程序模板”功能，但此功能唯一的问题是安装好模板后还需要调用更新程序来使“Visual Studio 2005”可以识别新添加的模板，有时候更新模板程序的工作可能很繁重，不过更新模板的任务只需运行一次且只要几分钟。

2.5.2 为什么运行 Visual WebGui 应用程序时会出现“无法找到资源”的提示？

请确认是否正确执行了下面的提示：

是否正确注册了“.wgx”脚本映射？

在“添加/编辑应用程序扩展名”对话框中不要选择 “检查文件是否存在”。

是否运行了正确有效的“.wgx”虚拟页面？

2.5.3 什么是 Visual WebGui 虚拟页面？

Visual WebGui 实际上是 WEB 应用程序的桌面应用，意思是说展现为窗体的一些类替代了通常我们在 WEB 应用程序中见到的页面。在桌面窗口程序中有一个运行主窗体的静态入口函数“Main 函数”，通过映射类到虚拟页面，Visual WebGui 可以有多个主窗体。通过配置“web.config”配置文件中的 Visual WebGui 节的“Applications”配置，可以映射虚拟页面到 Visual WebGui 窗体，每个“Application”元素映射一个虚拟的“.wgx”页面到对应是类。

2.5.4 Visual WebGui SDK 的.NET1.1 版和.NET2.0 版的区别是什么？

Visual WebGui SDK 的.NET1.1 版和.NET2.0 版基于同样的代码和设计，只是编译器的版本不同，为了支持.NET2.0 进行了一些修改。.NET1.1 版要求先安装 Visual Studio 2003，.NET2.0 版要求先安装 Visual Studio 2005。运行发布的应用程序时，请确认 IIS 虚拟目录使用了正确的.NET 版本，否则会出现“您运行的 ASP.NET 站点使用了错误的.NET 框架版本号”的错误信息。

2.5.5 为什么从 Visual Studio 2005 中运行 Visual WebGui 应用程序时得到的是一个目录列表？

在.NET2.0 版本下的空白 Visual WebGui 应用程序模板中没有设置起始页，可以右键单击工程文件，设置工程的属性，在 WEB 标签页手动设置虚拟起始页。当创建 Visual WebGui 应用程序时应手动设置“Form1.wgx”虚拟页对应到默认的类 Form1。

3 发布 Visual WebGui 应用程序

3.1 发布 Visual WebGui 用户程序到服务器

基本上发布 Visual WebGui 应用程序和发布 ASP.NET 程序是相似的。相似的原因是因为 Visual WebGui 应用程序实际上是 ASP.NET 应用程序的一个扩展。一旦您创建了发布工程时，您应该检查 Gizmox.WebGUI.Common 和 Gizmox.WebGUI.Server 是否被正确地发布了。如果您使用 Visual Studio 的发布机制来发布，您应检查并确认引用到 Visual Web GUI 的引用被标记为 拷贝到到本地。除此以为，发布 Visual WebGui 应用程序和发由 ASP.NET 应用程序就是相同的了。

3.1.1 注册“.wgx”扩展

在宿主站点注册“.wgx”扩展是一件简单的工作，并且注册此扩展没有任何安全隐患。实际上和 aspx 扩展是相同的。目前我们知道提供了此扩展的宿主服务器站点有：Brinkster 和 SiteGround。

3.1.2 提供所有必须的组件

为了正确运行 Visual WebGui 应用程序需要三个运行时组件，它们是：“Gizmox.WebGUI.Common”，“Gizmox.WebGUI.Forms”和“Gizmox.WebGUI.Server”。一定要确定这三个组件在工程的 Bin 目录下。

3.1.3 设置静态资源模式

Visual WebGui 的静态资源模式会在站点根目录的“Route”目录下缓存应用程序的大部分资源。因为 Visual WebGui 站点加载后会在“Route”目录下产生缓存文件，所以必须指定此目录的读写权限，应该总是在 IIS 的缓存头文件中定义此目录来防止浏览器重复下载这些资源。要使 Visual WebGui 应用程序工作在静态模式要在配置文件的 Visual WebGui 节中添加键值（***<StaticResources Mode="On"/>***）。

建议在发布产品时工作在此静态资源模式，因为静态模式创建了强大的缓存机制，发布产品时建议使用此静态资源模式，但在开发时并不是很适用。如果在开发时没有创建自定义控件或自定义外观主题，工作在此模式也可以改善运行性能。

3.1.4 设置图标预加载

Visual WebGui可以定义预加载模式来避免图标问题，主要是IE的innerHTML问题，参见（http://support.microsoft.com/kb/319546）。IE has an issue with caching image in innerHTML operation which is the cornet stone of any browser based AJAX environment.激活图标预加载模式可避免IE重复多次加载同一图标，为了激活图标预加载模式需要在配置文件的Visual WebGui节添加键值（***<IconsPreloading Mode="On"/>***）。

4 开发参考

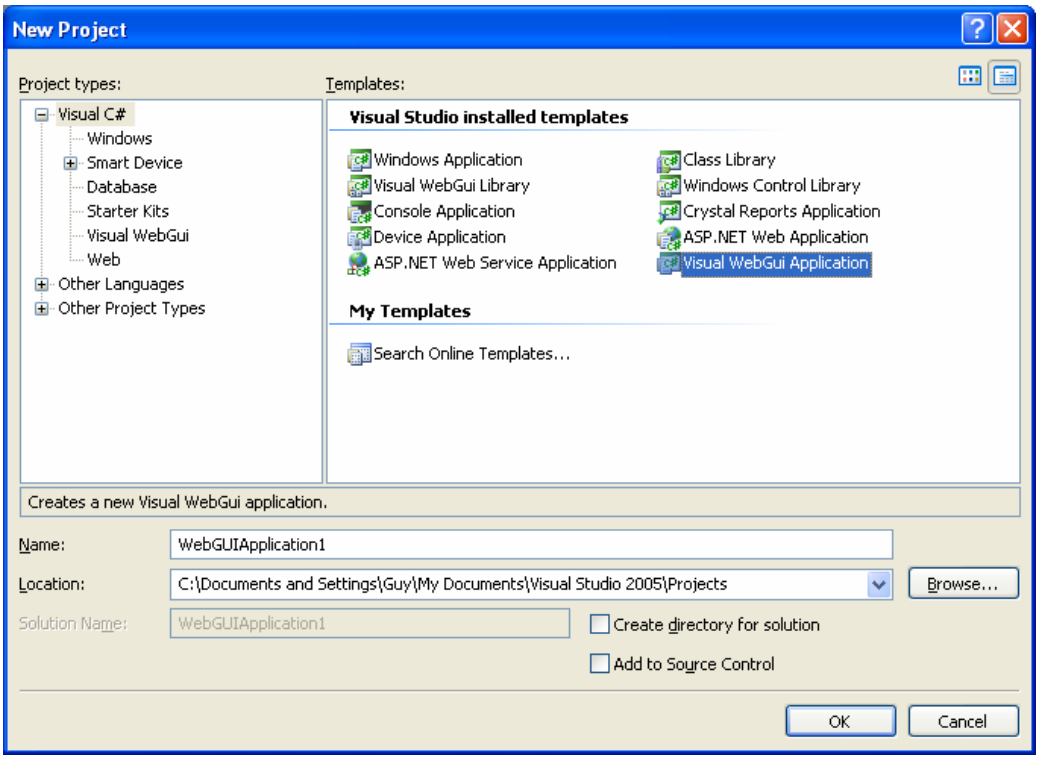
4.1 应用程序

4.1.1 什么是 Visual WebGui 应用程序?

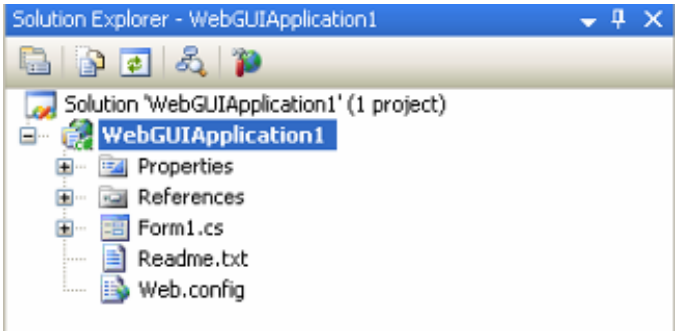
Visual WebGui 应用程序实际上也是 ASP.NET 程序，通过扩展 ASP.NET 支持 Visual WebGui 管道。添加到 Visual WebGui 运行时组件的引用，并在配置文件中创建“Visual WebGui”节，这些就是从 ASP.NET 应用程序转换到启用了 Visual WebGui 支持的应用程序所必须的步骤（即：当 ASP.NET 遇到以.wgx 结尾的请求时，应该用什么组件来处理请求）。

4.1.2 使用 Visual Studio 创建 Visual WebGui 应用程序

当打开“新建项目”对话框时，新增加的了“Visual WebGui application”和“Visual WebGui library”，这两个工程项目类似“Windows 应用程序”和“Windows 控件类库”。

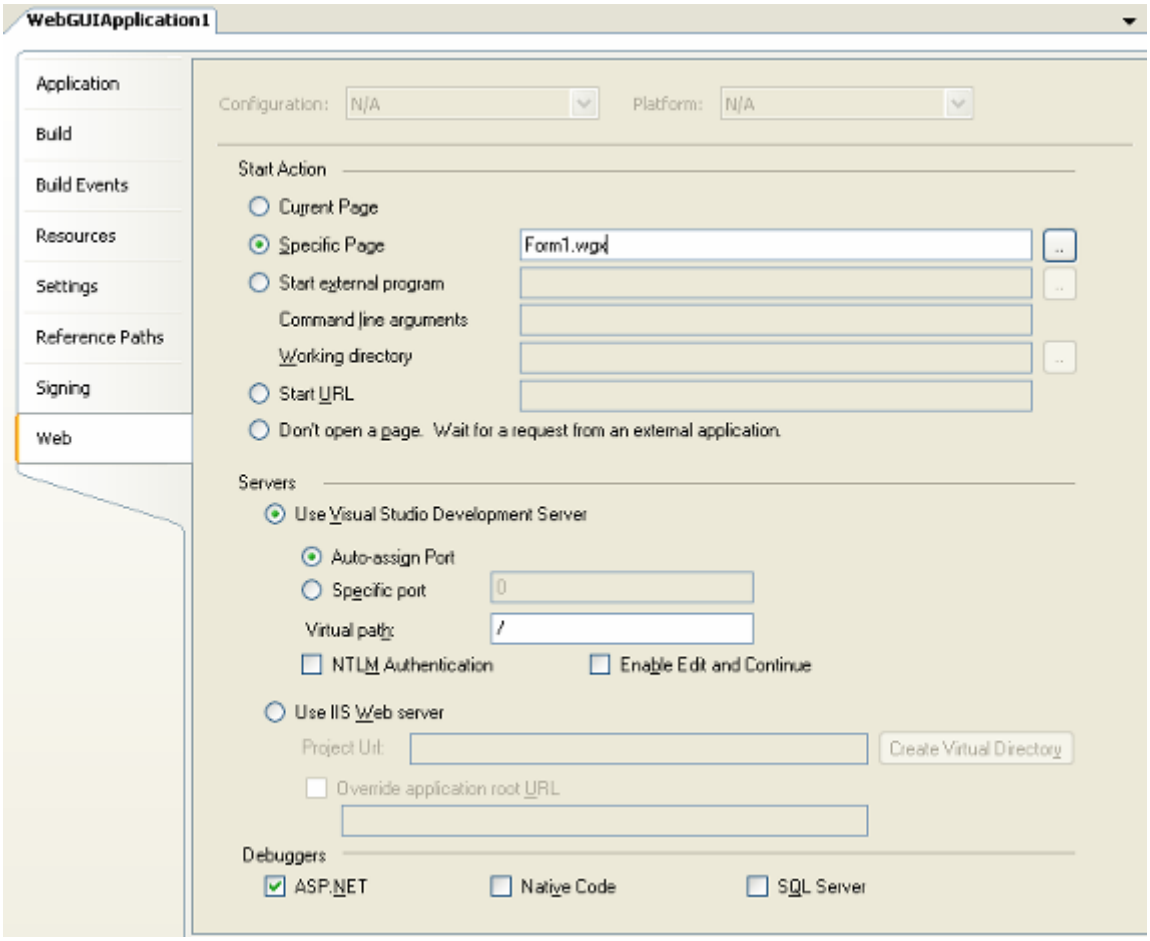


点击“确定”按钮创建一个新的 Visual WebGui 应用程序，Visual Studio 创建的 Visual WebGui 应用程序的内容如下图所示：



Visual WebGui 工程项目实际上是 ASP.NET 页面和 WinForms 窗体的一个混合体。此工程项目完全是一个 ASP.NET 工程项目，可以添加我们熟悉的的 ASP.NET 元素到项目中，这些元素包括：Web 页面、Web 服务、甚至 global.asax 文件，但还可以添加界面和行为（操作方法、接口）类似 WinForms 的扩展项目。Visual WebGui 应用程序默认创建了一个窗体并命名为“Form1”，以后可以更改此窗体的名称，但由于 Viual WebGui 通过在“web.config”配置虚拟页，映射到指定的继承自 Visual WebGui 窗体的类，所以需要在更改了窗体名称时也需要更改对应的虚拟配置。双击窗体文件显示类似 WinForms 的设计器，可以从 Visual WebGui 工具箱上拖动组件到设计器，Visual WebGui 工具箱上的组件具有和 WinForms 组件相似的行为和界面。

注意：Visual WebGui .NET2.0 的当前版本没有添加“Form1”虚拟页作为起始页，可以通过访问工程项目的“属性”来设置项目的起始页，如下图所示：



4.1.3 在现有 ASP.NET 应用程序中添加对 Visual WebGui 的支持

为了在现有 ASP.NET 项目中添加对 Visual WebGui 的支持，需要添加对 Visual WebGui 运行时组件的引用，并在 Web 配置文件中添加 Visual WebGui 配置节和 HTTP 处理器。并且需要注意的是因为在.NET2.0 和.NET1.1 下运行的组件版本是不同的，所以基于这两个版本的 Visual WebGui 应用程序可以同时一起运行。Visual WebGui 的运行时组件包括：“Gizmox.WebGUI.Forms”和“Gizmox.WebGUI.Common”，“Gizmox.WebGUI.Forms”可以在全局装配件（GAC）和安装目录中找到。

4.1.4 手动创建 Visual WebGui 应用程序

创建一个 ASP.NET 站点并依照上一节的介绍就可以手动创建 Visual WebGui 应用程序。

4.1.5 什么是 Visual WebGui 客户端应用程序?

Visual WebGui 客户端应用程序实际上是 WinForms 应用程序，这使得开发人员可以以 Web 应用程序和客户端应用程序两种方式来发布应用程序（但此功能只在企业版中提供）。

4.1.6 创建 Visual WebGui 客户端应用程序

可使用随 Visual WebGui 企业版一起发布的 Visual WebGui 客户端应用程序模板创建 Visual WebGui 客户端应用程序，并添加了运行所必需的 Visual WebGui 运行时组件的引用。在“Program.cs”文件中有个“to-do”的节需要定义，此节定义了哪个窗体将作为应用程序的启动窗体和登录窗体，其中登录窗体是可选的。

4.2 类库

4.2.1 什么是 Visual WebGui 控件类库?

Visual WebGui 控件类库等价于 WinForms 控件类库，提供了以可再发行的格式来包装控件和类的方式。此方式对于第三方控件开发人员和哪些想创建大型应用并把组件分为多个包的开发者非常有用。

4.2.2 从其它组件（装配件）中使用 Visual WebGui 类库

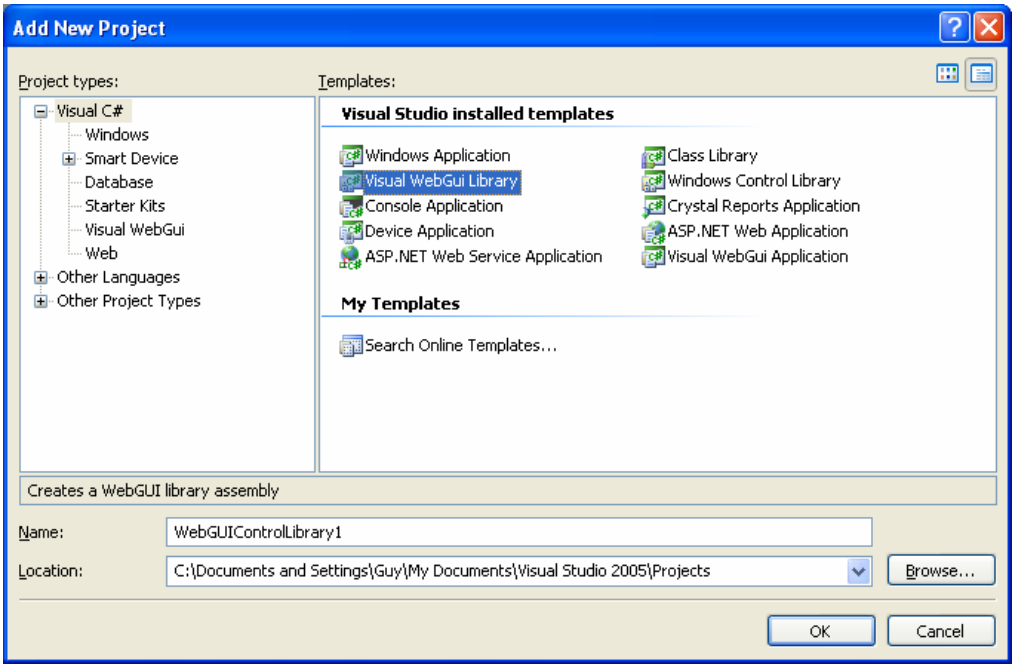
使用 Visual WebGui 类库和使用.NET 组件（装配件）是一样的，一旦添加了对控件或类的外部引用，使用起来就好像它们是项目中的控件或类一样。

4.2.3 从 ASP.NET 工程中使用 Visual WebGui 类库

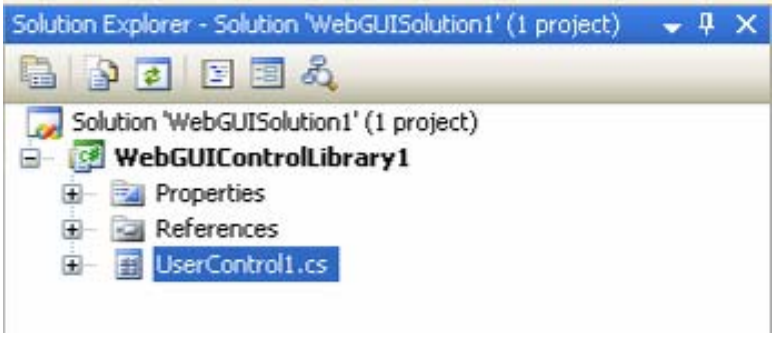
可在现有 ASP.NET 项目的配置文件中添加 Visual WebGui 配置节和 HTTP 处理器来提供对 Visual WebGui 扩展的支持，使用 Visual WebGui 适配器技术可以从遗留的 JavaScript 代码中访问任意 Visual WebGui 窗体，这对于哪些需要在项目中重写单件模式或需要添加特殊功能的工程项目特别有用。

4.2.4 使用 Visual Studio 创建 Visual WebGui 控件类库

打开“新建项目”对话框有两个新增加了的工程项目模板：“ Visual WebGui application”和“Visual WebGui library”。类似于“Windows 应用程序”和“Windows 控件类库”工程项目。



单击“确定”按钮创建 Visual WebGui 控件类库，Visual Studio 创建的 Viusal Web Gui 控件类库如下图所示：



Visual WebGui 控件类库项目实际上也是.NET 类库项目，项目添加了对 Visual WebGui 运行时组件的引用，Visual WebGui 类库项目可以包含用户自定义控件、窗体和引用自其它项目的.NET 类等，但 Visual WebGui 控件类库项目默认创建的是一个叫做“UserControl1”的用户自定义控件。

4.3 窗体

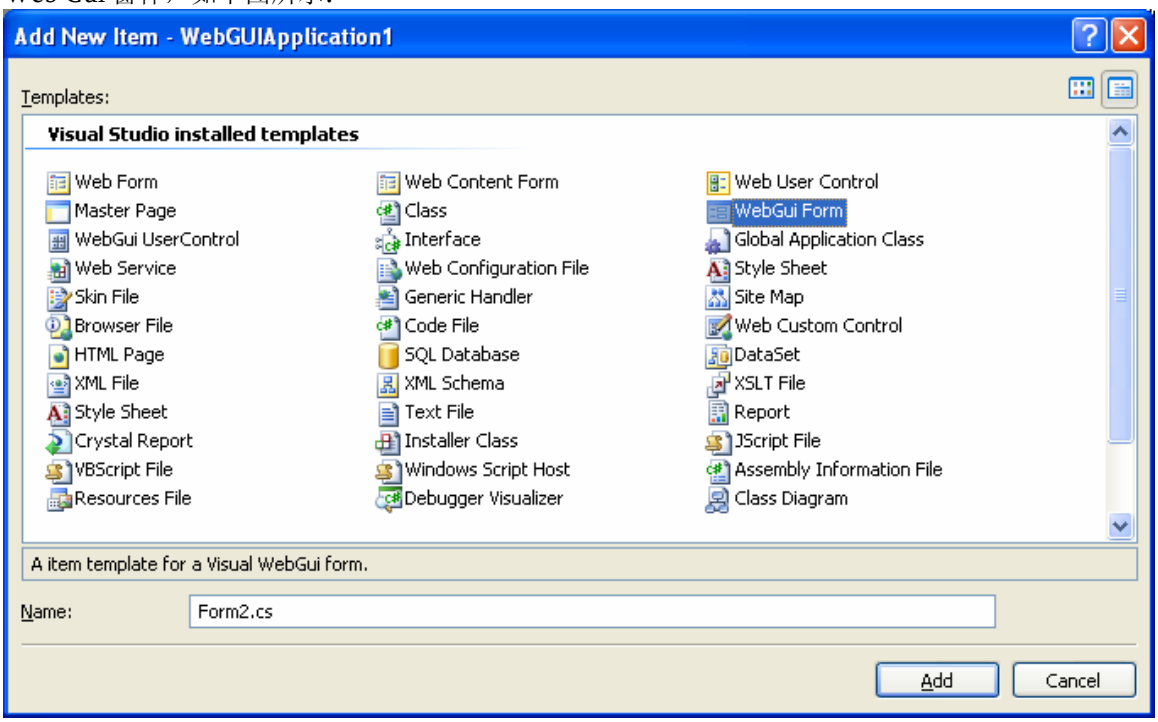
4.3.1 什么是窗体?

在 Visual WebGui 应用程序中，任何显示的窗口都是一个“窗体（Form）”类的表现，通过使用“窗体（Form）”类的“BorderStyle”属性可以创建标准（standard）, 工具箱（tool）,无边框（ borderless）和悬浮（ floating）样式的窗体实例;也可以使用“窗体（Form）”类创建像模式窗口之类的对话框窗体。“窗体（Form）”类本身提供了对键盘（Tab 切换焦点时序）和窗口内容滚动的支持。

在应用程序中创建用户界面时，一般是创建从“Form”类继承的类，然后在类中添加控件、设置属性、创建事件处理程序和编写处理逻辑等。

4.3.2 使用 Visual Studio 创建窗体

可以单击工程项目文件或工程项目下的文件夹打开 Visual Studio 的“新建项目”对话框添加 Viusal Web Gui 窗体，如下图所示：



4.3.3 手动创建窗体

“Forms（窗体）”实际上只是个继承自“Gizmox.WebGUI.Forms.Form”类的一般.NET 类，所以基本上手动创建 Visual WebGui 窗本只是新建一个从“Gizmox.WebGUI.Forms.Form”继承的新类。

4.4 控件

4.4.1 什么是控件?

添加到窗体的每一个组件，象：“Buttons（按钮）“TextBox（文本框）”、“RadioButton（单选

```
public class HelloWorldForm : Gizmo.WebGUI.Forms.Form
{

    private Button button1 = new Button();
    private TextBox textBox1 = new TextBox();

    public HelloWorldForm()
    {

        this.Text = "Hello Windows Forms World";
        this.ClientSize = new Size(392, 117);
        this.AcceptButton = button1;

        button1.Location = new Point(256, 64);
        button1.Size = new Size(120, 40);
        button1.TabIndex = 2;
        button1.Text = "Click Me!";

        button1.Click += new System.EventHandler(button1_Click);

        textBox1.Text = "Hello Windows Forms World";
        textBox1.TabIndex = 1;
        textBox1.Size = new Size(360, 20);
        textBox1.Location = new Point(16, 24);

        this.Controls.Add(button1);
        this.Controls.Add(textBox1);
    }
}
```

按钮）”这些都是控件。Visual WebGui 包含对应到在 Windows 窗体中所有常见的控件，甚至像“DataGridView”这样的控件都有。一般通过设置控件的属性和控件进行交互来改变控件的界面外观和行为，下面的代码展示了如何在窗体中添加一个按钮并设置其大小和位置。

有两种类型的控件：

自定义控件：通过提供自定义 Web 资源和设计时/客户端行为来显示用户界面的控件，自定义控件一般继承自“Control”。“Chart”控件就是一个自定义控件的例子，它只提供有限的设计时支持。

用户控件或组合控件：组合使用其它控件构建的控件称为用户控件，用户控件继承自“UserControl”。使用一个“TextBox”控件来显示客户地址的控件就是一个用户控件的例子。在 Visual Studio .NET 下创建用户控件可以利用 Visual WebGui 设计器得到全面的设计时支持，包括浏览控件属性、控件事件、定义设计时行为和应用许可到自定义控件和用户控件中。

4.5 事件

4.5.1 什么是事件?

Visual WebGui 的编辑模式是基于事件驱动的。控件的状态改变将产生一个事件，比如说：用户单击了按钮。为了处理这些事件要在应用程序中注册特定事件的事件处理方法，在 Visual Basic 中有两种方法注册事件处理方法：

- 如果定义变量时使用了“WithEvents”关键字，可以在方法上使用“Handles”关键字来注册方法作为事件处理程序。
- 可以在运行时使用“AddHandler”注册事件驱动方法。

下面的代码展示了注册事件处理方法的两种方法：

```
....
//Create the button
Button button1 = new Button() ;
button1.Location = new Point(256, 64);
button1.Size = new Size(120, 40);
button1.Text = "Click Me!";
this.Controls.Add(button1);

//Register the event handler
button1.Click += new System.EventHandler(button1_Click);
....

//The event handling method
private void button1_Click(object sender, EventArgs evArgs)
{
      MessageBox.Show("Hello Visual WebGui World!");
}
```

4.5.2 理解事件队列

4.6 用户控件

4.6.1 什么是用户控件?

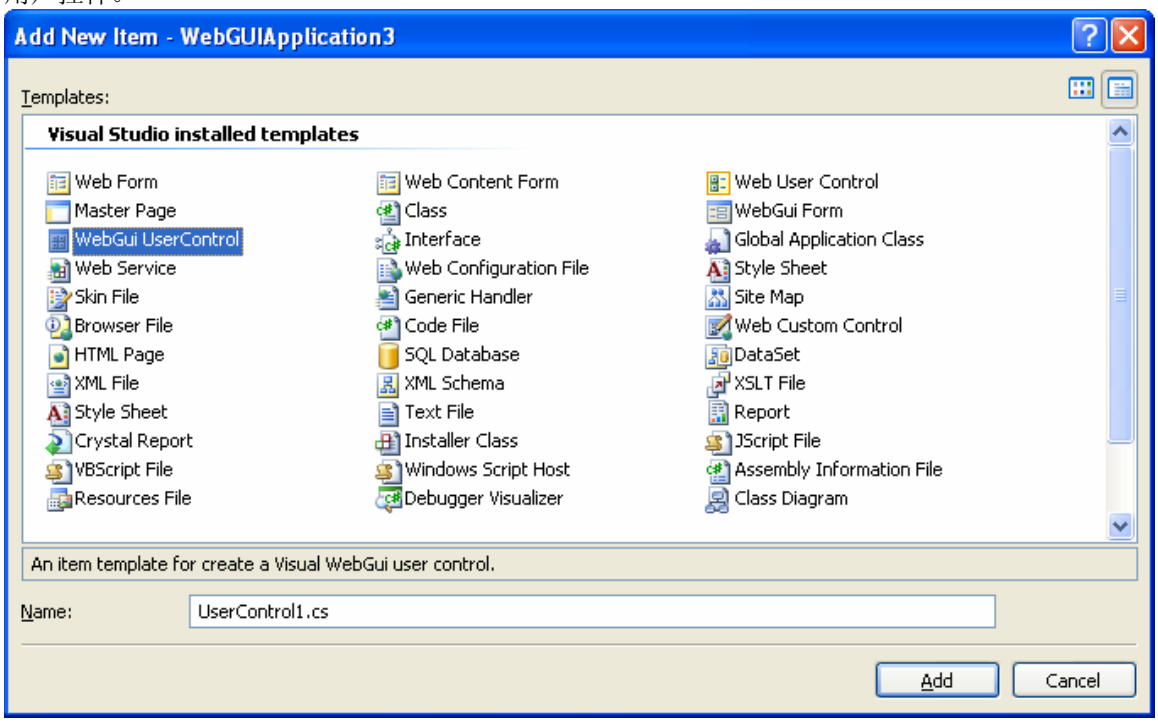
由其它控件组合构成的控件叫做用户控件，也叫作组合控件，用户控件继承自“User Control”类。使用一个“TextBox”控件来显示客户地址的控件就是一个用户控件的例子，在 Visual Studio .NET 下创建用户控件可以利用 Visual WebGui 设计器得到全面的设计时支持。

4.6.2 什么时候我该使用用户控件?

用户控件是把一个大的应用程序分成多个高内聚、低耦合的组件和包装哪些可用于多个应用程序的公用代码的极佳方法。在 Web 开发过程中如果页面中有一部分的变化类似“IFREAME”结构，使用用户控件就非常有用，也就是说，例如有个应用程序它的界面是通过树形结构导航到不同的页面（类似 Outlook），可以使用用户控件来显示这些不同页面的内容。

4.6.3 使用 Visual Studio 创建 Visual WebGui 用户控件

在工程项目文件或工程项目下的文件夹上右键单击调出“新建项目”对话框来创建 Visual WebGui 用户控件。



4.6.4 在 Visual Studio 工具箱中注册用户控件

在包含用户控件的组件中，一旦成功编译了组件，Visual Studio 会自动把用户控件注册到工具箱，但如果使用的是外部组件，在工具箱的选项卡上右键单击选择“添加/移除项…”，选择要添加的组件，单击“确定”按钮添加用户控件到工具箱，此时就可以把添加的用户控件以拖拽的方式添加到设计界面了。

4.7 资源

和 WinForms 相比，Visual WebGui 处理资源的方式稍微有些不同，WinForms 用 Image 对象保存位图资源，可以通过“resource manager”加载资源对象;Visual WebGui 应用了另外一种资源处理方法，“Resource handles（资源名柄）”实际上是指向真正的资源，如果控件需要使用“Image（图像）”资源时，可定义了一个“ResourceHandle”类型的变量。开发人员可以把此变量转换为任何从“ResourceHandle”继承的资源句柄。Viusal Web Gui 发布了一些像“IconResourceHandle”、“UrlResourceHandle”等的一些内建资源句柄，“IconResourceHandle”指向图标资源文件下的图标资源，“UrlResourceHandle”允许开发人员链接到外部资源并通过 HTTP 协议取得。

4.7.1 在设计时使用资源句柄

当前还没有资源句柄属性的可视编辑器，但使用类型转换器允许插入格式化字符串，并根据格式化字符串转换资源句柄为特定的资源句柄实例。例如：如果输入“Icons.MyImage.gif”字符串将得到以“MyImage.gif”参数初始化的“IconResourceHandle”类的对象实例。

4.7.2 IconResourceHandle 类

“IconResourceHandle”指向在“Web.config”配置文件中指定的“Icons”目录下的资源，在 Visual WebGui 配置节中有一个叫做“Directoies”的节映射了已命名目录名称对应的文件目录，其中“Icons”目录就是其中之一。在客户端模式文件目录即可以是相对于工程项目的根目录或“Bin”目录的相对路径，也可以绝对路径，绝对路径可以在工程项目间共享图标，“Icons”目录的默认值是“Resources/Icons”。

在设计时初始化“IconResourceHandle”实例时要为构造函数指定一个内嵌资源格式的参数，此参数的格式是以“.”作为目录之间分隔符的内嵌资源格式，也就是说“dira/dirb/file.gif”要格式化为“dira.dirb.file.gif”。

在设计时可以使用“Icons.DirA.DirB.FileName.gif”这样的格式化字符串作为初始参数传入构造函数来创建“IconResourceHandle”实例，资源路径前面添加了“Icons.”前缀。

4.7.3 ImageResourceHandle 类

“ImageResourceHandle”指向在“Web.config”配置文件中指定的“Images”目录下的资源，在 Visual WebGui 配置节中有一个叫做“Directoies”的节映射了已命名目录名称对应的文件目录，其中“Images”目录就是其中之一。在客户端模式文件目录即可以是相对于工程项目的根目录或“Bin”目录的相对路径，也可以绝对路径，绝对路径可以在工程项目间共享图像，“Images”目录的默认值是“Resources/ Images”。

在设计时初始化“ImageResourceHandle”实例时要为构造函数指定一个内嵌资源格式的参数，此参数的格式是以“.”作为目录之间分隔符的内嵌资源格式，也就是说“dira/dirb/file.gif”要格式化为“dira.dirb.file.gif”。

在设计时可以使用“Images.DirA.DirB.FileName.gif”这样的格式化字符串作为初始参数传入构造函数来创建“ImageResourceHandle”实例，资源路径前面添加了“Images.”前缀。

4.7.4 DataResourceHandle 类

“DataResourceHandle”指向在“Web.config”配置文件中指定的“Data”目录下的资源，在 Visual WebGui 配置节中有一个叫做“Directoies”的节映射了已命名目录名称对应的文件目录，其中“Data”目录就是其中之一。在客户端模式文件目录即可以是相对于工程项目的根目录或“Bin”目录的相对路径，也可以绝对路径，绝对路径可以在工程项目间共享数据，“Data”目录的默认值是“Resources/ Data”。

在设计时初始化“DataResourceHandle”实例时要为构造函数指定一个内嵌资源格式的参数，此参数的格式是以“.”作为目录之间分隔符的内嵌资源格式，也就是说“dira/dirb/file.gif”要格式化为“dira.dirb.file.gif”。

在设计时可以使用“Data.DirA.DirB.FileName.gif”这样的格式化字符串作为初始参数传入构造函数来创建“DataResourceHandle”实例，资源路径前面添加了“Data.”前缀。

4.7.5 UrlResourceHandle 类

UrlResoruceHandle类允许开发人员映射到外部资源，例如：PictureBox控件可通过把UrlResoruceHandle类的对象实例赋值给控件来显示位置在http://www.yahoo.com/yahoologo.jpg的图片。
可在设计时在“属性面板”以“http://”前缀初始化“UrlResourceHandle”类对象实例。

4.7.6 GatewayResourceHandle 类

“GatewayResourceHandle”可以在必要时生成资源，使用此技术的一个常见的用法就是用它生成转为位图的图表并用 PictureBox 显示出来。“GatewayResourceHandle”接收一个“GatewayReference（网关引用）”作为构造函数的参数，此“GatewayReference”实际上是一个指针指向“Gateway（网关）”的句柄。目前“GatewayResourceHandle”还只能使用代码手工创建，还不支持使用交互式界面来创建。
应该按照如下所示代码来使用 GatewayResourceHandle：

```
PictureBox objPicture = new PictureBox();

objPicture.Image = new GatewayResourceHandle(
        new GatewayReference([component], "[action name]")
);
```

4.7.7 AssemblyResourceHandle 类

“AssemblyResoruceHandle”允许从组件（装配件）的嵌入式组件资源集中生成资源，“AssemblyResoruceHandle”接收一个“Assembly reference（组件（装配件））引用”和一个以“.”为分隔符、包含命名空间和目录的全路径名的文件为构造函数参数。
应该按照如下示例代码使用“AssemblyResoruceHandle”：

```
objPicture.Image = new AssemblyResoruceHandle(
      this.GetType().Assembly,
      "Gizmox.WebGUI.Forms.FolderA.File.gif"
);
```

4.7.8 TypeResourceHandle 类

“TypeResourceHandle”允许从嵌入式类型资源集中生成资源，“TypeResourceHandle”接收一个“Type reference（类型引用）”和嵌入式资源代码作为构造函数参数。
应该按照如下示例代码使用“TypeResourceHandle”：

```
objPicture.Image = new TypeResoruceHandle(
      this.GetType(),
      "BackgroundImage"
);
```

4.7.9 SkinResourceHandle 类

“SkinResourceHandle”允许从嵌入式皮肤类型资源集中生成资源，皮肤是一个资源集可以被外观主题重载并嵌入在外观主题组织下的特定的布局中。
应该按照如下示例代码使用“SkinResourceHandle”：

```
objPicture.Image = new SkinResoruceHandle(
      "WindowCloseButton.gif"
);
```

4.8 数据存储

在 Web 应用程序中一般有应用程序级（Application scope）、会话级（Session scope）和 Cookies 级（Cookies scope）数据存储，Visual WebGui 另外还有一个叫做“上下文（Context scope）”级别的数据存储。一个 Visual WebGui 会话可以有多个上下文，一个上下文拥用自己的主窗体并管理自己的状态，上下文的示例应用可以是两个源于同一个 Web 应用程序的在线应用程序服务，每个应用被映射到不同的 URL 并拥有自己的上下文。在上下文中存储数据可以作为在 Windows 应用程序（WinForms 应用程序）中定义静态变量的一种替换方法，静态变量模拟了一个可配置的应用程序级上下文。

4.8.1 会话数据

可通过“VWGContext.Current.Session”静态属性或“Form.Context.Session”属性访问会话级数据，会话级数据服务于 Visual WebGui 应用程序，就像 ASP.NET 中的 HTTP 会话级数据服务于 ASP.NET 应用程序一样。

```
// Set MyParameter equals 3 in the session scope
VWGContext.Current.Session["MyParameter"] = 3;

// Getting the value of MyParameter from the session scope
int myParameter = (int)VWGContext.Current.Session["MyParameter"];
```

4.8.2 应用程序数据

可通过“VWGContext.Current.Application”静态属性或“Form.Context.Application”属性访问应用程序级数据，应用程序级数据服务于 Visual WebGui 应用程序，就像 ASP.NET 中的 HTTP 应用程序级数据服务于 ASP.NET 应用程序一样。

```
// Set MyParameter equals 3 in the application scope
VWGContext.Current.Application["MyParameter"] = 3;

// Getting the value of MyParameter from the application scope
int myParameter = (int)Application.Current.Session["MyParameter"];
```

4.8.3 当前上下文数据

可通过“VWGContext.Current”静态属性和“Form.Context”属性访问上下文级数据。上下文数据（存储）是为了实现 Web 桌面上下文而产生的，此上下文数据（存储）是 Visual WebGui 特有的。在客户端模式运行时，只有一个上下文数据存储。

```
// Set MyParameter equals 3 in the context scope
VWGContext.Current["MyParameter"] = 3;

// Getting the value of MyParameter from the context scope
int myParameter = (int)Application.Current["MyParameter"];
```

4.8.4 Cookies 数据

可通过“VWGContext.Current.Cookies”静态属性和“Form.Context.Cookies”属性访问 Cookies 级数据，在 Web 模式下数据存储在 cookies，在客户端模式下数据存储在 registry hives。

```
// Set MyParameter equals 3 in the cookies scope
VWGContext.Current.Cookies["MyParameter"] = "3";

// Getting the value of MyParameter from the cookies scope
string myParameter = VWGContext.Current.Cookies["MyParameter"];
```

4.9 从 Visual WebGui 应用程序中发送和接收数据

4.9.1 使用参数

在客户端模式，Visual WebGui参数实际上通过查询字符串、Post参数或命令行参数发送到应用程序的命名参数。为了让Visual WebGui能够转换查询字符串和发送数据到参数列表中，需要使用“Post”前缀定位应用程序，也就是说，如果映射到应用程序的URL是“http://localhost/mydomain/myapp.wgx”并且想发送一个UserID参数，访问程序时要使用“http://localhost/mydomain/Post.myapp.wgx?UserID=3”这样的格式;如果是从HTML窗体发送到 http://localhost/mydomain/Post.myapp.wgx，要按如下示例使用参数：

```
string userID = (string)VWGContext.Current.Arguments["UserID"];
```

4.9.2 使用返回值（结果）

返回值是 Visual WebGui 应用程序返回数据到引用应用程序（referring application）的一种方法，引用应用程序（referring application）实际上是当使用“Post”前缀时的 URL。例如：有一个“webform.aspx”页面发送数据到 Visual WebGui 应用程序，Visual WebGui 记录下此 URL，当应用程序的主窗体关闭时就检查其返回值并发送到记录下的 URL 中。
使用返回值的示例代码如下所示：

```
VWGContext.Current.Results["Token"] = token;
```

4.10 执行和调试 Visual WebGui 应用程序

在 Web 模式下调试 Visual WebGui 应用程序和调试常见的 ASP.NET 应用程序所需的知识和技能是一样的。

4.10.1 为调试 Visual WebGui 应用程序的配置

4.10.2 使用.NET 调试

调试 Visual WebGui 应用程序和调试常见的 ASP.NET 应用程序的要求基本上是一样的，下面的链接包含的提示会帮助你在开发 ASP.NET 应用程序时如何配置调试环境。

http://support.microsoft.com/kb/306172
http://support.microsoft.com/kb/319842

4.10.3 使用跟踪调试信息

添加下述诊断选项到“Web.config”以启用 Visual WebGui 应用程序跟踪功能，设置“VWG_TracingSwitch”为非 0 值启用跟踪，如下所示代码片断列出了可选值。
“VWG_TracingTresholdSwitch”定义了过滤起始值，即只有在执行时执行次数大于此值的跟踪信息才会被打印出来。

```
<system.diagnostics>
      <switches>
            <!--
            0 - Disabled
            1 - Gives error messages
            2 - Gives errors and warnings
            3 - Gives more detailed error information
            4 - Gives verbose trace information
            -->
            <add name="VWG_TracingSwitch" value="0" />
            <!--
            Performance tracing limited to this threshold
            -->
            <add name="VWG_TracingThresholdSwitch" value="10" />
      </switches>
</system.diagnostics>
```

4.10.4 使用内置调试查看器跟踪调试 Visual WebGui 应用程序

可以使用“DebugViewer”跟踪调试在Visual Studio之外正在运行的应用程序。使用“DebugViewer”非常简单也不需要安装它，您可以单击此处从微软公司下载。

4.11 与 ASP.NET 共享同一站点

前述章节我们讨论过，Visual WebGui 应用程序实际上也是 ASP.NET 应用程序，Visual WebGui 应用程序是 ASP.NET 的外部扩展。仍然可以在 Visual WebGui 站点下创建和运行 ASP.NET 项目，甚至可以通过“HTTPContext.Current.Session”或“VWGContext.Current.HttpContext.Session”访问用户会话（Session）共享参数。

也可以引用 Visual WebGui 应用程序并在配置文件中添加 Visual WebGui 配置节来扩展现有 ASP.NET 应用程序，即可以添加 DotNetNuke 站点和 Visual WebGui 应用程序到 ASP.NET 站点而不需要重新编译应用程序。

4.12 使用对话框

4.12.1 如何使用模式对话框

当打开 Windows（WinForms）模式对话框，应用程序被挂起并等待对话框关闭，为了节约 CPU 使用率，在 Web 环境下实现模式对话框就不是这样工作了。在多线程环境下并不希望有一个一直工作的线程，虽然 Visual WebGui 隐藏了许多 Web 开发的细节，但 Visual WebGui 是在 HTTP 基础上实现的，因此也具有其固有的限制，每个 HTTP 请求只创建一个线程。也就是说显示一个模式对话框只是在客户端具有模式对用，为了取得对话框关闭后的返回值要注册对话框的关闭事件处理程序。

4.12.2 如何使用消息对话框

因为消息对话框也是模式对话框，所以它也具有模式对话框同样的限制，为了取得消息对话框的返回值要传入处理返回值的事件委托。这和 Windows（WinForms）窗体取得返回值的方法是不同的，在移植 Windows（WinForms）窗体代码时要重构代码，一般要在显示消息对话框的前后修改应用代码，即前面的方法结束时显示消息对话框，对话框关闭后继续执行应用程序的另一个方法。

4.12.3 如何使用弹出式对话框

可以简单地调用每个窗体类的“ShowPopup”方法把窗体作为弹出式窗口来显示，“ShowPopup”方法接收一个作为停靠的组件的参数。

4.13 如何使用快捷菜单（上下文菜单）

4.13.1 创建快捷菜单

在 Visual WebGui 中创建上下文菜单和在 Windows（WinForms）程序中创建上下文菜单相似，当前设计器还不支持可视化编辑上下文菜单，但可以编辑上下文菜单类的“MenuItems”属性在属性编辑器中创建上下文菜单项。

4.13.2 在窗体上创建主菜单

在 Visual WebGui 中创建主菜单和在 Windows（WinForms）程序中创建主菜单相似，在窗体上创建主菜单，并设置窗体的“Menu”属性为刚创建的菜单组件。当前设计器还不支持可视化编辑菜单，但可以编辑菜单类的“MenuItems”属性在属性编辑器中创建菜单项。

4.13.3 MenuClick 事件

菜单单击事件是对 Windows（WinForms）窗体对像模块的扩展，允许在组件级别接收事件通知而不是直接把事件处理方法注册在菜单组件上。基于此，提供了一种更为简单的机制以处理上下文菜单事件，直接在控件上取得与控件相关的菜单事件，也就是说如果在“Panel”控件上使用上下文菜单，“Panel”控件会根据当前位置显示上下文菜单，但在 Windows 窗体中还需要取得菜单在当前控件的位置。每个 Visual WebGui 组件都有一个“ContextMenu”属性，此属性是对 Windows 窗体对象的部分扩展（例如：树形控件目前不支持上下文菜单）。

4.14 Gateways

4.14.1 What are gateways?什么是网关？

Visual WebGui 借助 Windows（WinForms）对像模块提供了开发像“Outlook”这样的富客户端 Web 应用程序的新感受，在开发“Outlook”样式的 Web 访问应用程序时，可以说 90%以上的对象模块都是所需要的。哪么 Windows（WinForms）对象模块是如何与 Web 开发协调的呢？这又是如何做到的呢？为了达到以上目的应用了一种叫作“网关（Gateways）”的技术。
每个 WebGui 组件可以用“IGatewayControl”接口声明自己为一个 WebGui 网关，这允许控件声明方法处理声明的虚拟 URL。“IGatewayControl”接口有一个方法根据传入的动作名称返回对应的网关句柄，网关句柄完全以 HTTP 句柄的方式处理请求，这说明也可以在网关句柄中使用 HTTP 句柄（基于此，可以把 ASPX 页面嵌入在 WinForms 对象模块里并且可以和模块交互，这使得 WebGui 和遗留程序协同工作变得非常容易）。

其它可以使用网关的地方：

- 为“IFRAMES”提供基于 HTML 的内容。
- 为当前视图提供可打印版本。
- 与“applets”、“ flash”、“ activeX”等交互。
- 使用像“Janus grid”这样的 ASP.NET 现有控件。
- 下载文件。

4.14.2 创建网关

“IGatewayControl”接口定义了一个方法叫做“GetGatewayHandler”，网关就是实现了“IGatewayControl”接口的控件。“GetGatewayHandler”不但可以自己处理请求，也可以返回一个“IGatewayHandler”接口对象处理请求，此概念类似于 ASP.NET 中“IHTTPFactory”和“IHTTPHandler”之间的关系。动作参数用于在同一网关上支持多动作，也就是说可以在同一网关上根据不同的动作返回生成的 XML 或 HTML 文件。

4.14.3 引用网关（GatewayReference）

通过“GatewayReference”类引用一个网关，“GatewayReference”类接收网关控件和要处理的动作作为初始化参数，也就是说每个不同的动作需要一个不同的“GatewayReference”类实例。通过“GatewayResourceHandle”类可以把网关当作“ResourceHandle”使用，
“ GatewayResourceHandle”类使用“GatewayRefrence”生成资源。

4.14.4 以网关方式使用 ASPX 页面

为了达到“以网关方式使用 ASPX 页面”的目的最好的方法就是从“ASP.NET”页面继承并实现“IGatewayHandler”接口，命名此类为“GatewayPage”。创建好此类后就可以创建另外一个 ASPX 页面并在页面上使用第三方控件，然后在新建的 ASPX 页面的后置代码中将继承的基类从“Page”改变为“GatewayPage”，这对 ASPX 页面没有任何影响，可以当作普通 ASPX 页面使用，也可以当作网关使用。在应用程序环境下可以访问“VWGContext”对象、存储“IRegisteredComponent”对象和转换打开当前页面的控件为网关控件。例如：可以从“HtmlBox”继承，使用其 URL 属性引用到网关，并为 ASPX 页面添加需要的参数（或是传递给第三方控件的参数）。以这种方式包装显示第三方控件，的确是一个完整、精巧的方法。当继承的控

件的属性改变时，可以调用其“update（更新）”方法迫使控件重绘自身和重新调用网关。因为此网关实际上也是一个 ASPX 页面，所以可以方便的使用和嵌入任何第三方控件。

下面是示例代码：

```
/// <summary>
/// Summary description for GatewayPage.
/// </summary>
public class GatewayPage : Page, IGatewayHandler
{
    private IContext mobjContext;

    public GatewayPage()
    {
    }

    #region IGatewayHandler Members

    public void ProcessGatewayRequest(IContext objContext,
IRegisteredComponent objComponent)
    {
        mobjContext = objContext;
        System.Web.HttpContext objHttpContext =
System.Web.HttpContext.Current;
        this.ProcessRequest(objHttpContext);
    }

    public IContext WGContext
    {
        get
        {
            return mobjContext;
        }
    }

    #endregion
}
```

然后编写“IGatewayControl”的“GetGatewayHandler”方法的代码如下所示：

```
#region IGatewayControl Members

public IGatewayHandler GetGatewayHandler(IContext objContext,
            string strAction)
{
    IGatewayHandler a =
System.Web.UI.PageParser.GetCompiledPageInstance
( @"..\..\..\..\..\Tutorial13\WebForm1.aspx" ,@"C:\Program Files\Janus
Systems\Controls for Microsoft .NET\ASP.NET Server Controls\Tutorials
and Samples\GridEX\tutorials\CSharp\Tutorial13\WebForm1.aspx" ,
System.Web.HttpContext.Current ) as IGatewayHandler;
    return a;
}

#endregion
```

4.15 自定义控件

4.15.1 什么是自定义控件?

自定义控件是继承自“Visual WebGui Control”类的控件，此类使得开发人员不必从头开始创建用户控件，创建自定义控件的典型方法是嵌入一个现存的 DHTML 窗口（控件）或创建一个新的自定义 DHTML 窗口（控件）。

4.15.2 创建自定义控件

创建自定义控件需要引用“Gizmox.WebGUI.Common”和“Gizmox.WebGUI.Forms”，并从“Gizmox.WebGUI.Forms.Control”类继承。自定义控件需要在同一命名空间下以同一名称命名的三种不同类型的资源，这些资源帮助 Visual WebGui 在客户端如何显示此控件。

第一种类型的资源是一个 XSLT 文件，此文件是用于生成控件 HTML 的模板文件，此模板在客户端运行并生成初始化和更新控件的 HTML。为了定义控件与其模板文件的关系需要设置“Cotrol.TagName”属性，此属性指明了控件元数据元素的名称。XSLT 模板应该只处理哪些标签名称（tag name）前缀为“WC”的控件，所有的 Visual WebGui 控件都被设置为“WC”前缀，此前缀表明控件是“Web Gui control”。XSLT 模板应该添加模式属性（mode）并设置其值为“modContent”，此值激活了 Visual WebGui 客户端引擎的绘制布局功能和一些常用样式属性，使开发人员只需提供控件的界面外观。

4.15.3 创建自定义控件设计时/客户端控制器

此部分文档正在编写过程中。

4.15.4 注册自定义控件

在“Web.config”配置文件的“Visual WebGui”配置节中注册自定义控件，下面是注册自定义控件的示例代码：

```
<Controls>
   <Control  Type="KRControls.RichTextEditor, KRControls" />
</Controls>
```

注册多个自定义控件就添加多个注册标签。

4.15.5 创建客户端事件

每个控件都可以创建在服务器端处理的事件，事件可以在创建后马上触发，也可创建后入队等待触发。一旦创建了事件，可以调用“Events_RaiseEvents”方法发送最后创建的事件和事件队列中等待触发的事件到服务器端。使用“Events_CreateEvent”方法创建事件，方法的返回值是一个事件对象，使用“Events_SetEventAttribute”方法添加事件参数，事件可以有多个参数。创建事件时要使用控件的 ID 号，此 ID 号指明是哪个控件产生的事件并直接发送到服务器端对应的控件类。

4.15.6 转换通讯事件到.NET 事件

当服务器处理完事件，就将事件传递到控件的“FireEvent”方法处理事件，可以重载此方法。大部情况下“FireEvent”方法简单地用于激活一般的“.NET”事件。事件类型通常提供了所有检查应该如何处理此事件的所有必须的数据。如果控件不处理事件，应该总是调用基类的“FireEvent”方法;如果控件处理此事件，就应在方法中添加基本的事件行为代码。

4.15.7 理解临界事件（critical events）

临界事件是哪些不能入队而必须马上在服务器处理的事件。如果开发人员添加了事件处理程序（event handler），哪么此事件就变为临界事件。意思是“TextBox.TextChanged”事件只在开发人员在控件实例对象上注册了事件处理程序（event handler）才会产生服务器端的“TextChanged”事件回调;如果没有在控件实例对象上注册事件处理程序（event handler），“TextChanged”事件会被创建但只在下一个临界事件被发送到服务器时一起发送到服务器。也就是说如果一个按钮的单击（Click）事件是在一个非临界“TextChanged”事件之后产生的，此“Click”事件将迫使事件队列（event queue）发送到服务器。事件队列维护着事件发生的先后次序，处理这些事件也将按事件发生的先后次序处理。即当服务器调用按钮的事件处理程序时，文本框（textbox）控件的文本已经被设置了。

4.15.8 理解控件重绘机制

Visual WebGui 控件只在控件的元数据被发送到客户端并要显示在客户端时才会重绘控件。当控件被修改后，为了使控件状态与其在客户端的界面表现一致，需要调用控件的“Update”方法同步更新，此“Update”方法将创建一个完全更新命令，此命令将发送当前控件及其所有子控件的元数据到客户端;也可以使用局部更新命令“UpdateParams”同步更新单个属性的更改，此方法使得只需更新当前控件而不需发送所有控件的元数据，此特性使得 Visual WebGui 显著地降低了带宽使用率，这和其它 AJAX 框架是不同的。

4.15.9 添加自定义控件资源

可以通过以下三种方法添加控件资源：

第一种方法是在控件类文件所在位置处添加与控件名称同名的 XSLT、JS 和 CSS 文件等的嵌入式资源，这些资源将被内部的 Visual WebGui 客户端引擎处理并扩展客户端兼容性。

第二种方法是注册控件页面资源，页面资源可以是任何客户端资源，可以是 HTML、JS、CSS、XML 或 XSLT 等。页面资源使用“WebCompilerPages”属性映射资源，此属性值包含所有与控件在同一位置并作为嵌入资源使用的页面名称，此属性值使用“;”作为资源分隔符。

第三种方法是在控件相同目录下以与控件相同的名称注册图标、皮肤（界面外观）和图片，例如有一个“MyTextBox”控件，就需要在控件所在目录下创建一个名称为“MyTextBox”目录，然后在此目录下为嵌入图标资源创建“Icons”目录，以及为皮肤和图片资源创建“Skins”和“Images”目录。

当创建了包含 Visual WebGui 的应用程序，所有资源被生成在“Generated”目录下，当使用自定义控件开发应用程序时要始终记住应用程序目录需要读写权限，发布应用程序时可以连生成的目录一起发布以避免在发布后设置目录的读写权限。Viusal WebGui 生成资源前首先要检查资源是否还没有生成，这就避免了在应用程序发布后对目录的写操作。

4.16 使用外观主题

4.16.1 什么是 Visual WebGui 外观主题

Visual WebGui 外观主题实际上是一个包含内嵌资源文件并按指定格式预先定义好的.NET 装配件（组件或类库）。例如要重载“TextBox”默认的 XSLT 资源，必须把重载的资源按“Gizmox/WebGUI/Forms/TextBox.xslt”路径内嵌到一个.NET 装配件中，每个内嵌的资源要按其原来的位置（路径）嵌入到.NET 装配件中，即为什么在“Gizmox.WebGUI.Forms”名称空间下的“TextBox”控件重载 XSLT 资源时需要按前面所述的路径嵌入到.NET 装配件中。同样，可以为要重载的皮肤资源创建一个“Skins”目录，在此目录下的文件即要重载的资源，必须和原资源具有相同的名称。

4.16.2 创建 Visual WebGui 外观主题

创建一个空的.NET 装配件（类库）并在装配件中添加资源，添加好资源后要在属性面板上设置资源的“生成操作”属性为“嵌入的资源”。

4.16.3 注册 Visual WebGui 外观主题

在“Web.config”文件的“WebGui”节中添加“Themes”节注册外观主题装配件，如下所示将注册“Ifn.W2.Web.Sites.BLL”外观主题。“Theme”标签用于声明外观主题映射到哪个装配件，“Selected”属性设置了当前外观主题，设置“Selected”属性值为“default”将导制应用程序使用默认的外观主题。

```
<Themes Selected="BLL">
      <Theme Name="BLL" Assembly="Ifn.W2.Web.Site.BLL" />
</Themes>
```

5 Visual WebGui 扩展指南

5.1 Visual WebGui 客户端架构

Viual WebGui 客户端由 JavaScript 脚本、XSLT 模板和 CSS 样式表组成，这些资源由 Visual WebGui 服务器从服务和组件中生成。服务是哪些提供公共（通用）功能的普通注册资源;组件是哪些从“Visual WebGui component”继承开带有附加资源的注册类。生成的资源的名称在全局范围内被放大以避免名称冲突，也就是说在“ComponentA”资源中定义的“MyFunction”与在“ComponentB”资源中定义的“MyFunction”的名称是不同的。此全局重命名机制提供了健壮的唯一性功能;此机制还支持“继承反射（inheritance reflection）”功能，即从“TextBox”继承的“MaskedTextBox”可以访问在基类定义的服务和资源。

Visual WebGui 服务提供了绝大部分的公共（通用）机制，组件行为以内建多分支的方式实现多浏览器支持。注册的资源使用浏览器后缀标明此资源用于特定的浏览器，哪些没有标明浏览器后缀的注册资源被认为是浏览器无关的可用于所有浏览器，哪些标明了浏览器后缀的注册资源只服务于从特定浏览器发出的请求。。

组件资源实际是装配件内嵌资源，一般情况下和组件在同一名称空间下，也就是说组件可以使用一些预定义属性指明从其它装配件或名称空间查找资源。组件可以定义嵌入的资源和页面两种类型的资源，Visual WebGui Web 编译器使用嵌入的资源定义组件在浏览器的行为和界面;页面是外部定义的资源，被嵌入的资源使用。这就是说例如：组件可以定义为界面表现为“IFRAME”，并调用显示在“IFRAME”中由 HTML 和脚本组成的页面资源。此机制允许嵌入 DHTML 控件及控件资源作为一个完整的控件。

5.2 Visual WebGui 客户端服务

5.2.1 Web 服务

Web 服务主要是一些 JavaScript 脚本功能用于执行与 Web 相关的常见（通用）任务，例如通过“Web_GetElementById”服务访问 DOM 元素，通过使用内部资源分支机制“Web_GetElementById”的实现是跨浏览器兼容的，也就是说当调用此服务时不必提心服务是如何实现的，也不必关心当前使用的是什么浏览器。

下面是 Web 服务资源中可以使用的功能列表：

5.2.2 Xml 服务

Xml 服务主要是一些 JavaScript 脚本功能用于执行与 Xml 相关的常见（通用）任务，例如通过“Xml_GetOuterXml”服务访问节点的外部 XML，通过使用内部资源分支机制“Xml_GetOuterXml 的实现是跨浏览器兼容的，也就是说当调用此服务时不必提心服务是如何实现的，也不必关心当前使用的是什么浏览器。

下面是 Xml 服务资源中可以使用的功能列表：

5.2.3 布局服务（Layout）

布局服务定义在 Visual WebGui 的基础 XLST 资源中，此服务也是跨浏览器兼容的，这使得 Viusal WebGui 客户端内核使用此布局服务提供像控件锚定（anchoring）和停靠（docking）这样的基本桌面布局功能成为可能。在已定义的布局中已放置了 HTML 定位器（HTML palce hodler），控件与其布局特性序列化到 HTML 定位器里面，也就是说当开发人员创建 Visual WebGui 控件时不必关心整个控件是如何定位的，只需要处理控件内部的界面表现。应该由容器控件通过使用“tplDrawContained”XLST 模板访问布局功能，例如，“Group”控件在自己内部调用此模板来组织管理其内部的控件。通过提供自定义布局 HTML 代码并在容器控件中调用“applytemplates”使容器控件重新绘制其自身，这样就可以实现自定义布局。“Gizmox.WebGUI.Forms ”装配件里实现的“TableLayoutPanel”就是自定义布局的一个例子，可以在“source forge”的开源工程中查看其源代码。

5.2.4 事件服务

事件服务的实现是一组负责和 Visual WebGui 服务器交互的 JavaScript 脚本功能，Visual WebGui 控件通常使用“Events_CreateEvent”和“Events_RaiseEvents”函数产生事件，事件实际上是被发送到服务器处理的 JavaScript 可序列化脚本对象，事件源是产生事件的控件的 ID 号，当调用“Events_RaiseEvents”方法后服务器接收到一个事件队列，服务器将在相应的控件上触发事件队列中的事件。事件被转换为实现了“IEvent”接口的对象，此接口包含序列化了的事件参数和用于触发普通.NET 事件的事件类型。事件处理完成后服务器将发送增量更新命令到客户端，通知界面表现层的浏览器重绘控件的相关部分。

下面是事件服务资源中可以使用的功能列表：

5.2.5 数据服务

数据服务是一组允许开发人员与离线界面缓存（off screen buffer）交互的 JavaScript 脚本功能。Visual WebGui 客户端实现使用基于 XML 的离线界面缓存来增强浏览器的运行效率，意思是说例如：有一个“tab”控件的每个标签页面（tab page）都填充了数据，浏览器的 DOM 机制一次只允许工作在一个标签页上，即可以在“tab”控件上可以使用很多控件而不会降低运行性能;这也使只加载需要的部分数据的多种优化功能成为可能。例如，“tab”控件的标签页如果不是当前工作页面时其页面数据将不会被加载，此功能将极大地降低带宽占用率。数据服务最常用的用法是为控件行为更新控件的显示属性设置，而不需要与服务器往返来查询控件的属性设置。例如，“Listview”控件的脚本检查它的数据属性设置是否允许多项选择;另一个例子是“tab”控件的脚本通过防止当切换到已预加载的页面时的服务器往返，这极大地优化了带宽占用率。

下面是数据服务资源中可以使用的功能列表：

5.2.6 Auxiliary services 附属服务

附属服务是一组防止浏览器资源 foot print 和提高开发人员效率的常用（公共）JavaScript 脚本功能。在附属服务中可以找到一个名称为“Aux_IsStringEmptyOrNull”的功能，此功能检查字符串是否是 null 或空字符串。
下面是附属服务资源中可以使用的功能列表：

5.3 Viusual WebGui 冲突解决机制（Web 编译器）

Visual WebGui 有一个内部 Web 编译器，实际上是一个资源生成机制。Web 编译器加载系统资源和用户注册资源并产生使用实体名称替换后的合并输出，意思是说如果组件 A 定义一个名称为“MyFunction“的函数，同时组件 B 也定义也一个相同名称的函数，但在输出的资源中将是两个不同名称：“MyFunction1”和“MyFunction2”。为了使此机制在所有类型的资源上都有有效，包括 XSLT 和 HTML 资源，Visual WebGui Web 编译器需要遵守“可搜索名称转换”的规则（在未来可能不必遵守此要求，但当前版本的 Web 编译器实现要遵守此命名转换规则）。

为了使“命名唯一性机制”正常工作，以下是命名转换需要遵守的规则：

1. JavaScript 函数应该以“XXXX_XXXXXX”的格式命名，一般情况下第一部分是控件的名称，例如：ListView_SelectItem。
2. CSS 类应该以“XXX-XXXX”的格式命名，添加“_XXXX”后缀规则在像“ListView-Item”和“ListView-Item_Selected”的类之间切换，同样，就像在上一条规则中提到的一样，一般情况下第一部分是控件的名称。
3. XSLT 模板应该以“tplXXXXXXX”的格式命名，同样第一个动词应该是控件的名称，全如：“tplListViewItemDraw.XSLT”。

哪些名称会被替换为简短有序的名称，所以没有必要使用短名称来降低带宽占用率。为了方便调试，在调试模式下使用的是完整的名称。也没有必要在代码中移除注释，因为在服务客户端之前 Visual WebGui 会先移除注释，但在调试模式下是可看到注释的，这些注释将帮助开发人员理解他们正在调试的代码是做什么的。