.NET 互操作下 Gecko 浏览器组件①

周昊明, 李 扬

(广东工业大学 信息工程学院, 广东 广州 510006)

摘 要: 在讨论.NET 互操作技术的平台调用, COM 组件互操作, 数据封送与 Gecko 内核浏览器的运行环境 XULRunner 及跨平台对象模型 XPCOM 的基础上, 根据其操作思路, 通过互操作把.NET 与 Gecko 内核两者相结合, 创建了.NET 环境下通过 XPCOM 组件控制的 Gecko 内核浏览器控件, 使其同时具备异构复用、可扩展、可操作的特点.

关键词: .NET; 互操作; Gecko; XULRunner; XPCOM; 浏览器组件

.NET Interop-Based Implementation of Gecko-Browser Component

ZHOU Hao-Ming, LI Yang

(School of Information Engineering, Guangdong University of Technology, Guangzhou 510006, China)

Abstract: From discussion .NET interop technology platform invoke COM components interop data marshaling and runtime environment XULRunner and cross-platform object model XPCOM of Gecko-based browser, base on its operation ideas, combine the .NET and Gecko, create a Gecko-browser control under the .NET environment through XPCOM component controls, to make it has heterogeneous reuse, scalability, operational characteristics.

Key words: .NET; interop; Gecko; XULRunner; XPCOM; web browser component

随着互联网的高速发展与网络应用规模的扩大, 应用软件具备网络浏览与交互操作能力可丰富软件的样式与功能. 但微软.NET 开发工具中的 Web 浏览器控件对 Trident 内核功能封装局限于简单的浏览网页, 对 HTML 与 CSS 标准及 javascript 脚本支持有限, 未能全面展示网络应用. 以 Mozalla Firefox 为代表的 Gecko 内核浏览器具有安全、快速、开源等特性, 能够满足网络应用的交互需求, 其功能可通过跨平台对象模型实现跨平台调用. 而互操作技术的出现为不同语言与技术的交互, 兼容托管与非托管代码间的接口及方法, 公共语言运行时(CLR)与 COM 或 DLL 组件之间的复用提供了有效可靠的解决方案[1]. 本文通过分析互操作技术与 Gecko 运行环境组成的研究基础上, 设计基于.NET 平台下具备异构复用性、可扩展、可操作的浏览器组件.

1 互操作技术

为了保持与利用现有的资源, 托管和非托管代码间的相互调用, .NET 互操作主要有平台调用 COM 互操作两方面.

1.1 平台调用

平台调用技术可以让托管代码调用动态链接库(如 Win32.dll 或 C/C++创建的 DLL)中的非托管函数, 使用时要指定非托管函数 DLL, 创建函数原型, 用 static 和 extern 修饰符标记方法, 设置平台调用的相关属性, 根据需要跨越互用边界封送其参数.

1.2 COM 互操作

用与非托管的 COM 代码交互, 这种机制的大多数基本功能是由平台调用机制实现的, 并对其做了进一步的封装和扩展以兼容 COM 组件[2]. 在.NET 中使用 COM 组件通常是用 TlbImp 工具为 COM 组件生成互操作程序集早期绑定 COM 对象, 或用反射后期绑定 COM 对象. 如果已知组件对象的类型与其全局唯一标识符 GUID, 可在自定义互操作程序集中的接口或类的声明前添加 ComImport(说明类或接口定义已在

① 收稿时间:2013-05-12;收到修改稿时间:2013-06-13

COM 组件中)与 Guid 来包装组件, 从而直接调用 COM 组件实现通讯. 并且当程序只使用 COM 接口的部分方法时, 将不会嵌入没定义的方法, 可提高编译效率及简化程序的大小.

1.3 数据封送

数据在托管与非托管内存之间传递是双向过程, 托管与非托管代码间传递参数或返回值时都会经过封送拆收器进行封送处理[3]. 封送过程会涉及数据类型(基本数据类型或结构数据类型)转换, 以及非托管和托管内存之间来回复制与转换数据, 由于托管与非托管数据类型内存结构可能不一致, 如处理不当将会导致更费时的数据操作、数据转换异常或内存泄露等问题, 这样数据封送就需要严格按照 COM 组件与应用软件双方的数据类型约定(表 1 是本设计涉及的主要数据类型转换约定), 才能避免数据封送处理成为影响性能的瓶颈.

表 1 数据类型转换表

托管数据类型	非托管数据类型	说明
string	LPWStr	若[In,Out]用StringBuilder代替
bool	Bool	转换为4字节的整数值
int, long, double 等数值类型	Int, Long, Double等	如实际使用值没超过int类型的最大值, 可用int替代
结构体类型	LPStruct	结构体数据为连续
接口类型	Interface	如nsIWebBrowser
Object对象	IUnknown	包装对象转换为实现接口

2 Gecko运行环境介绍

2.1 Gecko 内核

现今网络浏览器应用程序市场主要由 Trident、Gecko、Webkit 三种浏览器内核占据, 本设计用 Mozilla 的 Gecko 为浏览器的核心模块. 浏览器使用 Gecko 内核能有效完成 DOM 文档解析, 可靠的 javascript 脚本运行能力, 高效的 CSS 样式处理系统, 并有完善的图形渲染组件, 能在运行中实现硬件加速, 有效支持 Html5 与 SVG 图型, 具备隐私安全防护力.

2.2 Gecko 浏览器运行环境

XULRunner 是 Mozilla 公司开发的程序运行时环境, 它由众多特定功能的 XPCOM 组件构成, 其组件通过接口可在多种平台下灵活调用[4]. XULRunner 整合了排版引擎 Gecko、网络通信引擎 Necko、JavaScript 解析引擎 SpiderMonkey、XML 解析器等功能模块部分[5], 为网络应用提供渲染, DOM 编辑和事务支持等功能, 允许把 Mozilla 技术嵌入到其它项目和产品中. 此外 XULRunner 有丰富的扩展与开发的潜力, 并可在 about:config 中详细配置软件的浏览功能.

2.3 XPCOM

Cross Platform Component Object Model 是 Mozilla 公司的跨平台组件对象模型, 包含在 XULRunner 中, 它与微软的 COM 组件相似, 其实现与接口分离[6]. XPCOM 提供生存期管理, 接口查询, 文件抽象, 消息传递及内存管理的功能方法. 其每一个接口都继承自 nsISupports 并有可阅读的 Contract ID, 支持多语言开发调用和开放的组件版本管理. 在 XULRunner 中, XPCOM 组件对底层的功能进行封装, 其组件涉及 DOM 文件访问、界面布局、网络通信、用户身份存取、扩展管理、图片编解码、字符集转换等方面. 不同的 XPCOM 组件可以被其他 XPCOM 组件、程序或网络应用的 javascript 调用. 本设计需要通过对 XPCOM 的互操作实现与 XULRunner 中的 Gecko 组件的交互调用.

3 组件解决方案

根据互操作的原理及 Gecko 运行环境的特点, 本浏览器组件属于多层结构(见图 1), 分为表示层, 应用层以及业务逻辑层.

图 1 控件体系结构图

3.1 构建业务逻辑层

本浏览器组件与 Firefox 一样使用 XULRunner 作为底层运行环境. 根据浏览器需要的功能选择在 XULRunner 对应的接口组件, 在程序中根据其组件接口定义文件来构建统一的组件接口. 表 2 是主要接口组件介绍, 其功能可分为浏览模块(包括窗口, DOM 文档对象模型, 转跳, 事件等)与组件控制(XPCOM 的服务、注册、查找、管理).

由于.NET 的接口不包含字段, 若原接口定义涉及到接口属性及结构体数据类型, 将根据需要在软件中进行重定义, 使之与 XULRunner 底层组件接口的数据结构类型相一致.

表 2 主要接口组件

接口名称	接口说明
nsIComponentManager	该接口提供了来访问对象实例化的方法
nsIComponentRegistrar	提供来访问和修改 XPCOM 组件注册的方法
nsIDirectoryServiceProvider	用于获取文件位置目录服务。
nsIInterfaceRequestor	该接口为对给的对象提供可访问的通用接口
nsIServiceManager	为应用程序提供访问全局服务，取决于要用的组件库与要获得服务的实例
nsISupports	所有 XPCOM 接口都继承自这个接口
nsISupportsWeakReference	nsIWeakReference 通过该接口产生相应的实例
nsIWeakReference	此接口表示 XPCOM 对象的代理，它允许应用间接拥有 XPCOM 对象的引用
nsIBaseWindow	通用的窗口框体对象，可执行基本操作
nsIWebProgress	接口用于添加或删除 nsIWebProgressListener 的实例与观察 DOM 中的异步加载请求
nsIWebProgressListener	可实现浏览器进程状态的监听的接口组件
nsIRequest	用于控制启动器的请求
nsIEmbeddingSiteWindow	提供可视的 Gecko 内核窗口与控制方法
nsIWebBrowser	此接口由 Web 浏览器对象创建，可为其添加事件监听与获取其 DOM 的内容
nsIWebBrowserChrome	为最顶层的 Gecko 的 Web 浏览器窗口组件。
nsIWebBrowserFocus	用于浏览器焦点控制和交互的接口
nsIWindowCreator	创建新 Gecko 浏览器窗口的回调接口，应用程序要启动该组件并通知 WindowWatcher 组件
nsIWindowWatcher	接口用于 Gecko、DOM 窗口的持久化，允许程序对它们操作。
nsIContentViewer	展示内容的句柄接口组件
nsIDocShell	用于检索浏览器中的 nsIDocshell 接口组件或 DOM 文档的元素
nsIMarkupDocumentViewer	用于设置 Html 或 Xml 文档的属性
nsIWebNavigation	浏览转跳网页接口，控制浏览器转跳

3.2 XPCOM 中间应用层

应用层则为调用 XULRunner 中的 Xpcom.dll 与运行环境进行交互操作, 需要在程序中设计对应的静态 Xpcom 类. 类中包括静态接口成员组件管理(nsIComponentManager)、组件注册(nsIComponent Registrar)、服务管理(nsIServiceManager)为 CLR 与 XULRunner 之间交互提供组件控制管理服务.

3.2.1 在.NET 中初始化 XPCOM

利用平台调用 Xpcom.dll 库中的启动 XPCOM、获取组件管理与获取注册组件函数, 实现 XPCOM 在.NET 下的初始化, 大致流程为:

① 读取 Xpcom.dll 并加载到内存中;

② 通过 NS_InitXPCOM2 函数启动 XPCOM;

③ 分别调用 NS_GetComponentManager 与 NS_GetComponentRegistrar 完成组件管理与注册的初始化;

④ 用"@mozilla.org/file/directory_service;1"指令通过服务管理启动 nsIDirectoryService 接口组件, 为后续的组件使用指定运行环境的位置.

3.2.2 启动组件服务与创建实例

由于 Xpcom 组件有字符串类型的 Contract ID(由域, 模块名, 组件名及版本信息构成)与 GUID 标识 COM 对象和接口, 而组件注册与服务管理接口组件包含了可根据其中之一创建组件实例或启动服务的方法. 这样程序不需要知道组件的位置, 也能向底层 XULRunner 启用不同的模块服务或创建组件实例, 同时得到这个组件的接口对象.

3.2.3 获取 Xpcom 对象

类中包括 CLR 向 XPCOM 查找接口的方法, 用于获取已在 XULRunner 中启动的组件接口对象, 过程为:

① 获取 CLR 中托管对象接口的指针;

② 用 Marshal.QueryInterface 方法查询接口在内存中的指针, 如不能直接找到时可用 nsIInterface Requestor 接口组件的 GetInterface 方法间接查找目标指针;

③ 最后 Marshal.GetObjectForIUnknown 把指针转为对象.

通过以上对 XPCOM 的初始化、创建组件服务实例、查找组件实例的功能方法, 完成.NET 下对 Xpcom.dll 调用的再封装, 实现对 XPCOM 的跨平台操作.

3.3 浏览器组件 UI 表示层

浏览器为使用者与软件之间进行交互的窗口, 其功能与 Visual Studio 自带的 Web 浏览控件相似, 具备外观样式, 行为状态属性及功能事件, 其主体功能为与网络应用的基本交互操作.

3.3.1 构造浏览器组件

为使控件具备可视的界面, 其基类继承自系统窗体命名空间下的 Control 类, 并接口继承 XULRunner 的 nsIWebBrowserChrome、nsIEmbeddingSiteWindow、nsISupportsWeakReference、nsIWeakReference 等组件, 使浏览器 UI 控件具备 Gecko 内核上层可视化窗口、事件及 CLR 与底层组件间交互引用的能力. 类中包括全局关键接口成员 WebBrowser(nsIWebBrowser)、BaseWindow(nsIBaseWindow)、WebBrowserFocus(nsIWebBrowserFocus)、WebNav(nsIWebNavigation), 组件

通过 XPCOM 完成在 XULRunner 中的实例化, 以下是本浏览器组件构造函数的部分代码:

```
Xpcom.Initialize(); //XPCOM 初始化, 通过 XPCOM 启动浏览器组件及类中成员实例
WebBrowser = Xpcom.
    CreateInstance<nsIWebBrowser>("@mozilla.org/embedding/browser/nsWebBrowser;1");
WebBrowserFocus =(nsIWebBrowserFocus)WebBrowser;
BaseWindow = (nsIBaseWindow)WebBrowser;
WebNav = (nsIWebNavigation)WebBrowser;
WebBrowser.SetContainerWindow(this); //设定自身为底层窗口容器
BaseWindow.InitWindow(this.Handle, IntPtr.Zero, 0, 0, this.Width, this.Height);    //初始化窗口
BaseWindow.Create();    //根据初始化设置启动窗口, 并通过 nsIWindowWatcher 组件使窗口可受控制
nsIWindowWatcher watcher = Xpcom.
    GetService<nsIWindowWatcher>("@mozilla.org/embedcomp/window-watcher;1");
watcher.SetWindowCreator(new WindowCreator());
BaseWindow.SetVisibility(true); //设置窗口可见
WebBrowserFocus.Activate();//浏览器组件获得焦点
```

3.3.2 控件包含功能

浏览器 UI 包含基本的页面转跳, 停止, 调整缩放率等功能, 其中转跳停止由类中 nsIWebNavigation 接口成员提供 LoadURI 与 Stop 方法来完成操作, 缩放功能则要去内存中获取浏览器的 DOM 文档组件对象, 再通过接口方法改变 DOM 文档的大小.

由于组件在 XULRunner 中分配的内存实例 CLR 不会直接释放, 需要重写类的释放方法, 通过成员 WebNav 停止浏览器组件与 BaseWindow.Destroy 通知底层释放组件内存, 否则会引起内存泄漏.

通过调用以上方法, 实现浏览器 UI 控件的缩放、转跳、定向功能与组件的生存期控制.

3.3.3 浏览器组件的事件

类中事件分为接口组件响应事件, UI 控件事件, 及自定义事件. 浏览器 COM 组件响应事件可包含进度事件与 DOM 事件, 但需要 WebBrowser 成员调用其组件 AddWebBrowserListener 方法通知底层加载事件监听, 其事件机制通过 Connection Points 使组件响应触发事件, 从而进入事件函数; 浏览器控件事件主要是继承自基类, 有操作、布局、行为、按键、属性变更等事件, 若涉及类中接口组件值改变, 要调用组件接口方法来重载原方法; 自定义事件则为自定义属性值改变或由某控件响应事件引发的自定义伴随事件, 可通过在方法中添加自定义事件原型来触发自定义事件.

3.4 组件运行测试

使用.NET 的 WinForm 窗体应用程序与设计的组件搭建一个简单的网络浏览器程序, 对本设计的浏览器组件进行了运行测试, 效果如图 2, 发现本设计通过互操作 XPCOM 能与 Gecko 内核功能相结合, 浏览能力优于 VS 中 Web 浏览器组件, 与同内核版本的 Firefox 浏览器差距不明显. 若结合 XULRunner 还能调整浏览器中部分机能的样式效果, 扩展功能插件, 开启 GPU 对浏览器视觉效果的硬件加速等功能.

图 2 浏览器组件效果测试图

(下转第 84 页)

5 结束语

鉴于培训评估体系无法量化培训效果问题，提出了量化评估体系方法，按照该方法要求对培训流程进行改进，设计了基于评估反馈的培训流程，最后实现了支持培训流程的组织培训管理系统. 当前组织培训管理系统在我所平稳运行，已帮助其通过软件能力资质认证. 进一步的研究应着眼于如何根据历史培训评估数据策划今后的培训工作.

参考文献

1 赵杰.企业培训效果评估体系研究[硕士学位论文].天津:天津大学.2007.

2 Kirkpatrik DL, Kirkpatrik JD. Evaluating Training Programs(Third Edition). San Francisco, California. Berrett-Koehler Publishers. 2006.

3 Philips JJ, Stone RD, Philips PP. 人力资源积分卡:计量与评价 HR 投资回报率.北京:人民邮电出版社.2006.

4 李刚.整合 STRUTS+HIBERNATE+SPRING 应用开发详解.北京:清华大学出版社.2007.

5 胡奇.JBPM4 工作流应用开发指南.北京:电子工业出版社.2007.

6 高杰.深入浅出 JBPM.北京:人民邮电出版社.2007.

7 PRC-OT-1.2-2011.组织培训规范.北京.北京计算机技术及应用研究所.

8 魏钧.绩效考核指标设计.北京:北京大学出版社.2010.

9 张鹤飞,蒋晓舰,刘旭等.中国软件行业生产力报告.http://www.doc88.com/p-805243940787.html,2007.

10 盛骤,谢式千,潘承毅.概率论与数理统计.北京:高等教育出版社.2005.

(上接第 125 页)

4 结束语

.NET 互操作技术实现了 COM 组件及 C/C++动态链接库在.NET 平台下的相互调用，使得原有的成果可以在不同语言结构的项目中继续运用. 本设计通过与 XPCOM 结合，实现了 XULRunner 中 Gecko 内核的浏览器组件与.NET 的关联操作，使得浏览能力得到提升. 如软件程序若能充分应用互操作技术，可使程序功能更加全面丰富，从而更有效地支持不同结构层次的应用与满足功能上的需求.

参考文献

1 黄际洲,崔晓源.精通.NET 互操作.北京:人民邮电出版社. 2009:2–20.

2 高明..NET Framework 对 COM 组件的调用机制研究.科学技术与工程,2006,6(14):2177–2179.

3 Gordon A. The .NET and COM Interoperability Handbook. New York. Prentice Hall PTR. 2002. 377–432.

4 Stearn B. XULRunner:a new approach for developing rich internet applications. IEEE Internet Computing, 2007, 11(3): 67–73.

5 张令臣,王雷,向继,周健.基于 XPCOM 代理的浏览器扩展行为分析技术研究.信息网络安全, 2012,(8):223–225.

6 张立杰,黄迪明.跨平台的开发环境——Mozilla 简介.计算机与数字工程,2005,6(33):62–65.