

基于 XML - Glue 的 Widget 的开发与研究

黎莹,陈榕

(同济大学基础软件中心,上海 200092)

摘要:XML - Glue 是基于 Elastos(和欣)操作系统开发的编程模型,它利用 Elastos 所提供的各种系统服务,以及 CAR 编程技术,为应用编程者提供了一种网络计算模式的 MVC 计算模型。基于这个编程模型,以 XML 书写界面,以脚本语言书写逻辑,并以 CAR 构件作为底层支持,成功地完成在 XML - Glue 下 Widget 的实现。实验证明,基于 XML - Glue 实现的 Widget 不仅可以完成丰富的用户体验,而且整个开发过程具有简捷方便、代码量小且运行时占用资源少的特点,完全能满足在嵌入式系统开发小应用程序的需求。

关键词:Elastos 操作系统;Widget;CAR 构件技术;XML - Glue

中图分类号:TP319

文献标识码:A

文章编号:1673 - 629X(2007)09 - 0023 - 04

Research and Implementation of Widget Based on XML - Glue

LI Ying, CHEN Rong

(System Software Engineering Centre of Tongji University, Shanghai 200092, China)

Abstract: XML - Glue is a programming paradigm based on Elastos embedded operating system. It makes the best use of the advantages of current system and CAR component technology and thus provides powerful model support, which make the development of widget (small application) in this programming paradigm easier and more efficient. This thesis introduces a practical method of how to develop widget using XML - Glue. It successfully develops a widget application using XML UI language and script language. And it proves that developing widget on XML - Glue is both easy and powerful, what's more, it need less runtime resources, which can perfectly meet the need of developing small application on embedded system.

Key words: Elastos embedded operating system; Widget; CAR; XML - Glue

0 引言

XML - Glue^[1]是 Elastos^[1]操作系统上的一个 Rich Client 应用模式的编程模型,符合 MVC^[2](模型 - 视图 - 控制器)架构思想,通过 CAR^[1]构件、XML^[3]、脚本语言^[4]等来描述应用,通过 XML - Glue 运行支持平台提供运行时支持。各组成部分中,CAR 构件封装应用的逻辑,XML 描述界面,XML 描述及脚本语言完成逻辑与界面的组装,从而形成一个完整的应用。这个思想在桌面应用上已经得到了较广的应用,像 Mozilla 的 XUL^[5,6],Microsoft 的 XAML^[7]等。XML - Glue 也提供了在嵌入式操作系统中相应的技术支持,为嵌入式应用的开发注入新的活力。

这个编程模型具有简单方便、扩展能力强、应用的

维护成本和定制成本低的优点。基于这一模型开发应用程序简单、方便、灵活,容易被大众接受,有助于无线运营商实现 3G 无线网络时代“创意无所不在”的目标。此外,它面向嵌入式系统的设计,使得开发的应用程序代码紧凑、运行时占用资源少。

利用 XML - Glue,可以迅速便捷地开发出外观靓丽、功能灵活实用的 Widget^[8](小应用),非常适合做嵌入式终端(如手机)的小挂件。在功能和外观上可与雅虎的 Widget 相媲美,而在代码量和占用资源方面却大大少于雅虎 Widget。

文中介绍了利用 XML - Glue 技术开发 Widget 的方法。

1 XML - Glue 下 MVC 架构的实现

XML - Glue 是 Elastos 操作系统上的一个 Rich Client 应用的 MVC 编程模型。MVC 模式把一个应用的输入、处理和输出流程按照 Model, View, Controller 的方式进行分离,将一个应用分为三层,即:模型层、视图层、控制层。每一层都有各自的处理任务。其中模

收稿日期:2007 - 01 - 10

基金项目:国家“863”计划资助项目(2001AA113400)

作者简介:黎莹(1982 -),女,上海人,硕士研究生,研究方向为嵌入式操作系统、系统软件支撑技术;陈榕,中心主任,教授,博士生导师,研究方向为嵌入式系统、构件技术。

型层(Model)是业务流程的处理以及业务规则的制定。业务流程的处理过程对其它层来说是黑箱操作,模型接受视图请求的数据,并返回最终的处理结果。业务模型的设计可以说是 MVC 最主要的核心。视图(View)是用户看到并与之交互的界面。MVC 设计模式对于视图的处理仅限于视图上数据的采集和处理,以及用户的请求,而不包括在视图上的业务流程的处理。控制(Controller)可以理解为从用户接收请求,将模型与视图匹配在一起,共同完成用户的请求。划分控制层的作用也很明显,它清楚地告诉你,它就是一个分发器,选择什么样的模型,选择什么样的视图,可以完成什么样的用户请求。控制层并不做任何的数据处理。

MVC 只是提出了一种分层思想以方便模型的重构和提高代码的重用性,但并没有提供具体分层方法。合理的抽取层次和模块设计,可以在很大程度上减少软件计算模型与视图间的耦合和代码重复性。由于 XML - Glue 是基于 Elastos 操作系统和 CAR 构件技术开发的,因此在支持 MVC 架构、对各层次进行分离时有其独特优势,它是通过对多种脚本语言的支持、XML 标记语言的使用以及 CAR 构件技术的运用来支持 MVC 架构,实现表现层和逻辑层的分离。

在 XML-Glue 编程模型中,视图(View)部分是用 XML 标记语言实现的。XML 通过定义标签和设置标签属性来描述整个界面的架构。同其他传统标记语言相比,XML-Glue 对标签的支持更为灵活。在 XML-Glue 中标签并不是被定死的。除了系统本身提供的标签支持,你完全可以用 CAR 构件技术编写自己标签,把它们封装在具体的构件中。使用时通过名域的实体化将标签和具体的构件实体对应起来,然后通过对名域的引用实现对实体中包含的标签的定义和操作。用这种方法可以非常灵活地选择适合自己的标签,如果系统提供的标签无法满足你的需求,你完全可以自己定制一个,这样就可以为视图界面的实现提供强大的支持。对于 MVC 中的控制器(Controller)部分,在 XML-Glue 中是由脚本语言和 Reflection 部分实现的。脚本语言嵌入在 XML 文件中,可以对所有的构件及构件内的资源进行操作,使得应用能够访问独立于界面架构的构件,弥补 XML 在描述逻辑上的缺陷,实现用户与界面的互操作;Reflection 是一个 CAR 构件,它提供了一组接口,通过它可以获取模块、类或者接口下所有信息,XML-Glue 利用这些信息实现对脚本与具体 CAR 构件实现类的对应关系,完成控制器的作用。模型部分(Model)在 XML-Glue 编程模型是通过 CAR 构件技术实现的。Elastos 系统中的所

有 CAR 构件都可以作为基于 XML - Glue 应用开发中的模型,用户自己编写的 CAR 构件也可以作为模型。此外,由于 Elastos 系统提供了构件的自动寻址、自动加载机制,所以,所有用到的 CAR 构件模型也可以在网络上获得,前提是已知构件在网络上的链接地址。XML - Glue 编程模型如图 1 所示。

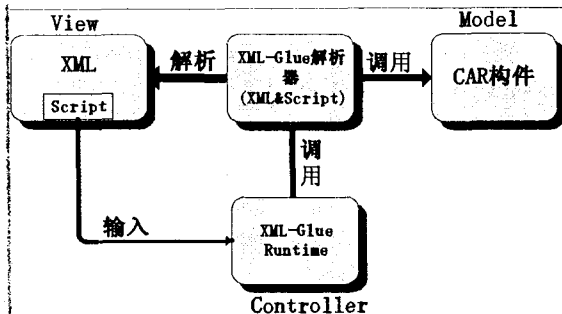


图 1 XML-Glue 编程模型的 MVC 架构描述

2 Widget 具体实现方案

由于 XML - Glue 模型下 MVC 架构的 Controller 部分由 Reflection 机制完成,而 Reflection 是 XML - Glue 内部机制,对于开发者透明,所以在用 XML - Glue 开发 Widget 时需要关注的只有两部分,即:视图部分和模型部分。视图部分供用户浏览和做相应的交互,这部分并没有真正的处理发生,它只是以界面形式输出参数并允许用户操纵数据;模型部分也就是逻辑实现部分,包括界面的交互和数据的交互,它的作用是处理用户和界面的交互,这种交互的内在实现是 CAR 的回调机制。

2.1 视图部分的实现

在 XML-Glue 中视图部分是由一个个标签组成的,每个标签对应一组属性和方法。XML 语言通过对标签的引用和属性的设置完成用户界面即视图部分的实现。

在 XML - Glue 中,每个标签都对应一个具体的构件类,在构件类中实现了所有基于标签的属性和方法。各个标签所对应的构件类封装在构件中,使用时,通过模块名(名域的实体化)指定具体的构件。XML - Glue 本身提供了各类常用的标签支持,你可以使用系统提供的标签,也可以运用 CAR 技术书写自己的标签以满足特殊的需要,这样相对而言就比较灵活。

XML- Glue 为开发人员提供的标签控件分两类: 图形系统标签和系统标签。图形系统标签集中在“elactrl.dll”模块中, 其中包含了完成一般用户界面所需的各类控件, 如 form, button, panel 等。系统标签有: xglue, script, id, event 等。其中 xglue 标签用于标识 XML- Glue 程序; event 是事件标签, 可用于注册回调

函数;id 标签用来标识控件对象的名字,脚本语言可通过此名字来操作标签对象;script 标签标识脚本语言。使用你自己书写的标签的方法和使用系统提供的标签方法一样。使用时,将名域实体化为所需标签所在的构件,通过引用实体完成对标签的各项操作。

利用系统提供或者你自己实现的标签完成视图非常简单。下面用一个例子作具体的说明。该例子很简单:在 form 控件上放置一个 button 控件,点击 button 按钮,退出程序。具体代码如下:

```
<? xml version="1.0" encoding="gb2312"? >
<x: xglue xmlns: x = "http://www. koretide. com/xml - glue"
xmlns: w = "elactrl. dll">
<w: form x: id = "MainForm" caption = "main" style = "Control-
Style_ Client" left = "10" top = "50" width = "180" height = "180"
>
<w: button x: id = "Button" ControlStyle = "" esCaption = "EXIT"
Left = "50" Top = "60" Width = "80" Height = "50">
</w: button>
</w: form>
<script language = "JavaScript">
<![CDATA]
function OnButtonClick()
{
MainForm. Close();
}
Button. mouseDown = OnButtonClick;
Button. SetForeColor(0x8b0000);
MainForm. Show();
]]>
</script>
</x: xglue>
```

这个程序的运行结果如图 2 所示。

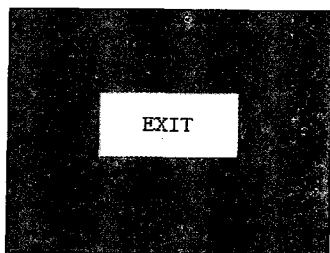


图 2 运行结果图

从这个例子中可以看到整个程序分为三个部分:文件头、XML 标记语言书写的视图部分和脚本语言书写的逻辑部分。文件头指明了 XML 的版本和字符编码属性,代码的主体部分从<x: xglue>到</x: xglue>,xglue 是一个系统标签。代码中“xmlns”就是名域,该名域下定义的 x 的值为 xml-glue 文档自己的标识,w 指定了一个具体的实体,在这个例子中,就是支持图

形系统标签的“elactrl. dll”构件。通过对这个实体的引用,可以定义和操作实体中包含的元素,如 button, form 等等。

代码中视图部分从<form>到</form>。可以看到:在 XML-Glue 中布局标签,首先需引用标签,如 form, button;为它赋一个标识号,即 id,逻辑部分对标签的操作就是通过这个 id 号进行的;指定标签后,设置好它的属性,如位置(left, top)、大小(width, height)等。至此就完成了视图部分的布局。在设置属性时,需要注意的是,XML 中的属性名和 CAR 实现类中的参数名之间是顺序无关、名字相关的,所以在书写属性名时,一定要保证属性名和参数名严格一致,至于顺序则无所谓。此外,对于在实现类中包含而 XML 中没有赋值的属性,系统会自动给它赋一个默认值。至于控件之间的父子关系,在 XML 布局标签时体现在标签的嵌套关系中,如上例中 button 是在 form 控件的子控件,书写时只需要将 button 标签嵌套在 form 标签中即可。在图形系统标签中,只有 form 和 panel 可以作为容器。

类似的可以在 form 上放置 picturebox, panel 等,设置好背景图片等属性,这样就可以很快地开发出赏心悦目的用户界面了。

2.2 逻辑实现部分

逻辑部分的实现是通过在 XML-Glue 中嵌入脚本语言完成的。目前 XML-Glue 支持的脚本有两种:JavaScript 和 LUA,以后还会扩充。要完成 Widget 逻辑部分的实现,需要了解:脚本中对非回调函数、回调函数的使用,多视图的实现和网络数据交互问题。下面分别介绍。

对于 Widget 中靓丽的外观,可能会遇到如换肤、更改颜色、图片加载等操作,对于这些,可以在脚本中调用非回调函数完成。在脚本中使用非回调函数跟一般对象方法的使用没什么差别,语法为:

对象名.方法名();

需要注意的是,脚本中的构建类对象又分为两类:一类是标签类对象。通过对标签的方法的调用,可以操作标签,如改变属性设置,或者关闭、移动标签等,示例程序中的 Button. SetForeColor(0x8b0000);就是如此。另一类构件类对象需要先加载构件模块,然后调用实例化构件类来创建一个对象,再通过对象名调用相应的方法。通过全局对象 Elastos 调用方法 Using 即可加载模块,以加载图形构件 elagdi 并进行实例化:

```
Var elagdi = Elastos. Using ("elagdi. dll");
```

```
Var image = elagdi. Image. New();
```

其中 elagdi. dll 为需要加载的模块名,CImage 为该

模块下的一个构件类,名为 CImage 类。创建这样一个对象后,就可以调用 IImage 接口中的方法,实现如图片的加载、转换等操作。

对于 Widget 中用户与界面的交互,在 XML - Glue 中是通过事件驱动机制完成的,即回调机制。回调函数在脚本中的使用,集中在事件处理函数的编写和函数的注册上。脚本中编写事件处理函数,与其他的函数编写没有太多的区别,只是函数的参数列表要与对应的回调函数严格对应,即参数名要相同、参数类型要对应。函数的注册过程也非常简单,如在示例程序中,注册 Exit 按钮上鼠标按下事件处理函数的语句如下:

```
Button.MouseDown = OnButtonClick;
```

其中 Button 是标识 Exit 按钮的对象名,MouseDown 是被注册的回调函数,OnButtonClick 是事件处理函数,其函数主体如下:

```
function OnButtonClick()
{
    MainForm.Close();
}
```

该事件处理函数也是在脚本中完成的。

对于 Widget 中多视图的实现,可以用 XML - Glue 中内置全局对象 application 的 forward(URI, ForwardMethod, ...) 方法和 view 中 ShowDialog(URI, ...) 方法实现,其中参数 URI 为要跳转的页面,ForwardMethod 标注以何种方式跳转,省略号的意思是后面可以接任意多个参数,它们的作用相当于将当前视图的一些数据传递给跳转页面,供跳转页面使用。

最后,对于 Widget 中与网络数据的交互,XML - Glue 也提供了相应的 XMLHTTP 构件完成网络操作,具体操作规范和 XMLHttpRequest 差不多,客户端通过 CXMLHttpRequest 对象向 HTTP 服务器发送请求并使用 XML 文档对象模型或者是 SAX 处理回应,

在本地解析服务器返回的 XML 文件,提取有用信息供视图显示。

3 总 结

总之,XML - Glue 技术是一种 MVC 架构支持技术,在 XML - Glue 下编写 Widget 只需要用 XML 布局视图部分,用大家熟悉的脚本语言进行逻辑处理,对于网络数据的处理,用 XMLHTTP 实现,该技术目前已经在多部手机上应用,主要用于广告下载等网络数据的下载。XML - Glue 本身非常容易上手,Widget 开发中会遇到的问题也都提供了相应的处理方法,只要你按照文中所述的方法去实践,相信很快就可以开发出了绚丽的 Widget。

参考文献:

- [1] 科泰世纪.《和欣 1.1》资料大全[EB/OL]. 2003. <http://www.51widgets.com/web/guest/downloads>.
- [2] MVC [EB/OL]. 1998. <http://ootips.org/mvc-pattern.html>.
- [3] Martin D. XML 高级编程[M]. 李 喆,严春莹,马 琳译. 北京:机械工业出版社,2001.
- [4] Flanagan D. JavaScript 权威指南[M]. 张铭泽等译. 北京:机械工业出版社,2003.
- [5] Xul Planet. Tutorial with examples[EB/OL]. 2006. <http://www.xulplanet.com/>.
- [6] Pansani A D. XUL and the building processe of a UI[EB/OL]. 2003. <http://luxor-xul.sourceforge.net/download/qatar.pdf>.
- [7] Microsoft Corporation. XAML Overview[EB/OL]. 2006. <http://msdn2.microsoft.com/en-us/library/ms752059.aspx#used-for-ui>.
- [8] 科泰世纪. Koretide. Website[EB/OL]. 2006. <http://www.51widgets.com/web/guest/home>.

(上接第 22 页)

高设计质量,或者增加抽样单元数、减小单元孔径,或者幅值补偿,或者寻求新的编码方法,或者改变全息产品传统的制作步骤,不经过出图打印、缩拍,采用电子束直写全息图技术^[6],与此同时,提高光学元件的制造精度也极其重要。可以肯定的是,随着光技术和加工工艺的进一步发展,计算全息的误差会得到有效降低,在信息光学中应用前景将更加广阔。

参考文献:

- [1] Lohmann A W, Paris D P. Binary Fraunhofer holograms gen-

erated by computer[J]. Appl Opt, 1967, 6(10): 1739 - 1748.

- [2] 王永仲,薛 蕊,张 勇. 计算全息匹配滤波器的研究及性能分析[J]. 激光与红外, 2005, 35(4): 275 - 278.
- [3] 虞祖良,金国藩. 计算机全息图[M]. 北京:清华大学出版社,1984.
- [4] Jennison B K. Iterative approaches to computer-generated holography[J]. Opt Eng, 1989, 28(6): 629 - 637.
- [5] 陈祯培,陈怀新,戴建明. 减少计算全息再现散斑的多图像平均方法[J]. 光电子·激光, 1997, 8(5): 383 - 386.
- [6] 高峰,朱建华,黄奇忠. 电子束直写计算全息图[J]. 中国激光, 2001, A28(6): 556 - 558.