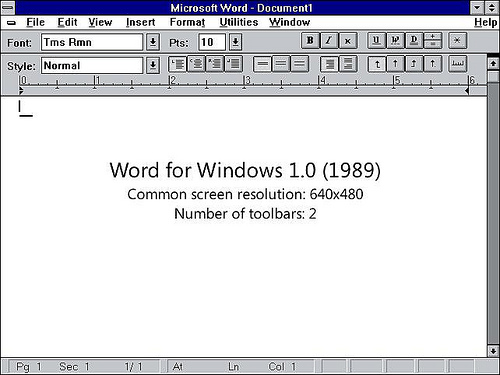
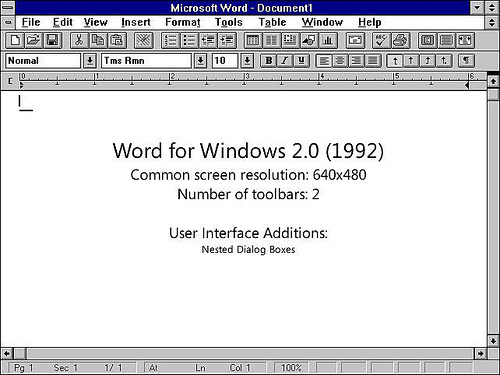
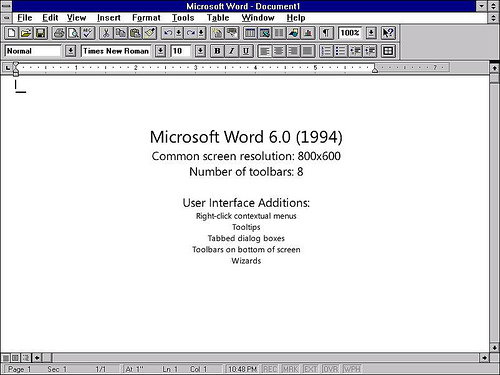
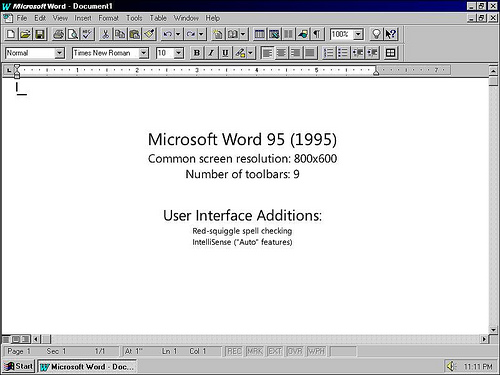
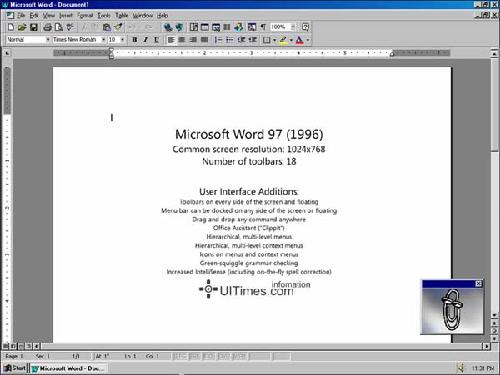
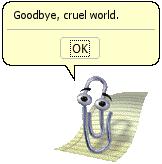
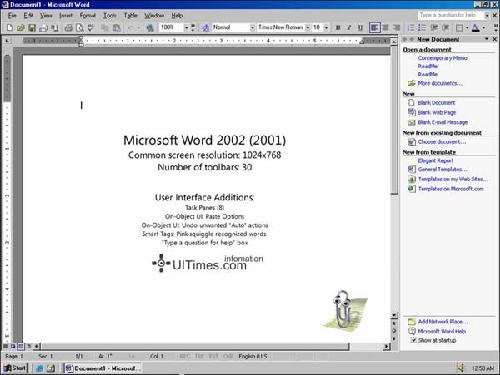
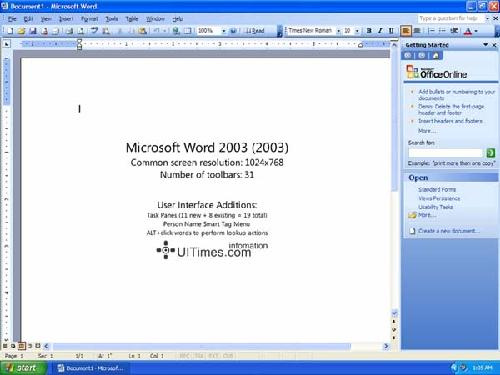
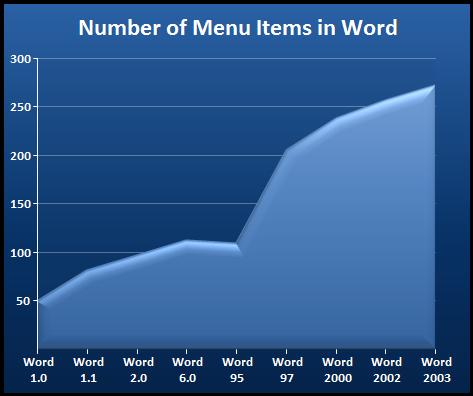
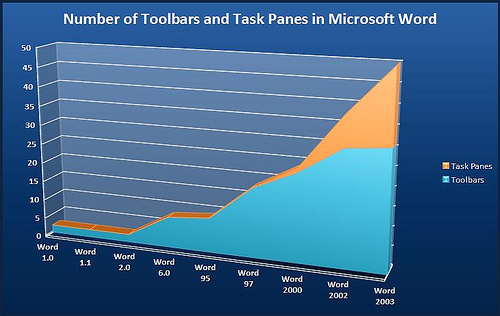
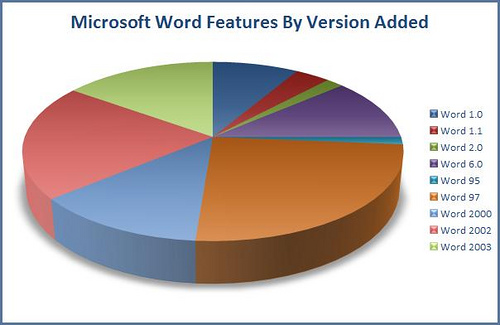
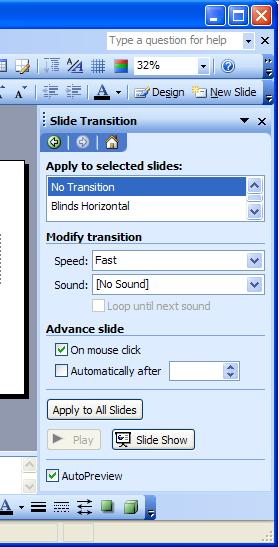
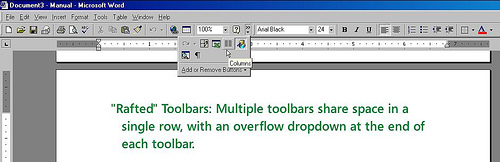
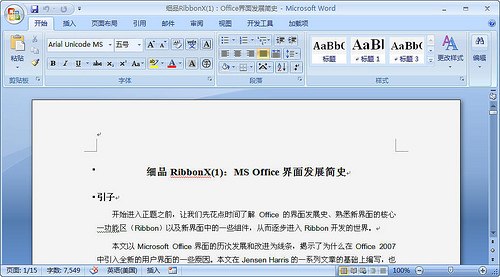
[**细品RibbonX(1)：MS Office界面发展简史**](http://www.excelperfect.com/index.php/2009/03/23/ribbonxstudy1/)

2009年03月23日, 2:04 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**5** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

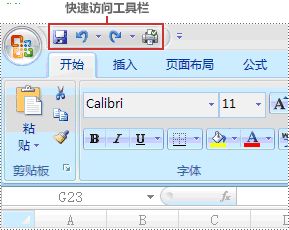
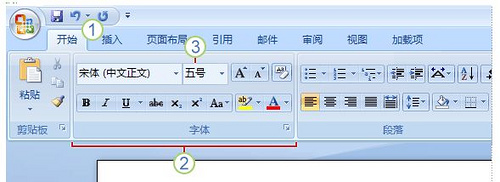
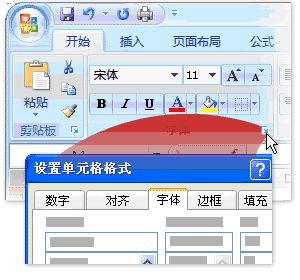
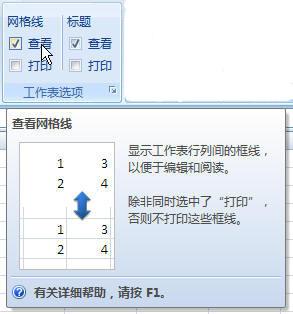
* **引子**功能区（Ribbon）是Microsoft Office的新一代用户界面，完全改变了Office界面风格。从推出以来就饱受争议，褒贬不一，我们估且不论其好坏，自待时间来评说，我们关注的是如何有效地使用功能区，如何掌控功能区，从而提升Office的使用效率。  
  从现在开始，我将细细研究Ribbon及其开发，并形成一系列文章，与大家分享自已的学习成果。  
  开始进入正题之前，让我们先花点时间来了解Office的界面发展史、熟悉新界面的核心——功能区（Ribbon）以及新界面中的一些组件，从而逐步进入Ribbon开发的世界。
* **前言**  
  本文以Microsoft Office界面的历次发展和改进为线条，揭示了为什么在Office 2007中引入全新的用户界面的一些原因。本文在Jensen Harris的一系列文章的基础上编写，也参考了Office系统的帮助知识，大部分截图直接采用了原文中的截图，参考文章链接附后。  
  Jensen Harris是掌管MS Office UI开发团队的程序设计负责人，在Microsoft工作已近八年。开始的5年从事Outlook邮件和PIM设计程序，之后领导团队重新设计了发布在Office 2003中的Outlook 2003用户界面。在完成了Office 2003后，来到Office 用户体验团队工作。
* **Office界面的源头**  
  Microsoft使用早期的Apple雏形开发了Word 1.0，它于1984年发布在最初的Mac中。Multiplan和Chart也在512K Mac下开发，最后它们于1985年合在一起作为Microsoft Excel 1.0发布：第一个在Macintosh上使用的轰动一时的零售程序。  
  因此，早期的Microsoft Office程序根源于Mac，当然，也反映在用户界面上。作为Mac的第一个和最大的软件提供者，在最初的Macintosh上做的一些UI决定受Microsoft开发团队的要求影响。当然，Office图形化用户界面(特别是顶级菜单条)的最基本的轮廓有它在第一个Macintosh版本中的根源。
* **Office界面的演变**  
  下面首先以Word为例，介绍Office用户界面的演进过程。  
  (1) Microsoft Word 1.0 for Windows于1989年发布，运行在Windows 386上。Word 1.0有一个应用程序级的菜单栏，这是从Mac的顶级菜单栏和显示在Microsoft DOS程序底部菜单发展形成的。同时，Word 1.0也包含了新的界面元素：工具栏（在Microsoft Excel中首先使用过）。在Word 1.0的界面中，看起来好像有两个工具栏，实际上仅仅上面的一栏称作工具栏。有趣的是，底部的按钮行称作“Ribbon”。Word 1.0的界面如下图1所示。  
  [](http://www.flickr.com/photos/35750775@N02/3378542370/)  
  图1 Word 1.0界面  
  (2) Word 2.0于1992年投放市场，此时，Word用户界面的基本结构已经固定下来了，就像当前的Word 2003一样。菜单栏为：文件、编辑、视图、插入、格式、工具、表格、窗口、帮助，下面是“常用”工具栏和“格式”工具栏。下图2所示为15年前设计的Word 2.0界面，一直沿用到今天。  
  [](http://www.flickr.com/photos/35750775@N02/3378542372/)  
  图2 Word 2.0界面  
  (3) Word 6.0运行在当时流行的Windows 3.1上，它的发布给Office市场带来了巨大的冲击，这也是Word与WordPerfect竞争的转折点。Word 6.0中增加了一些新的用户界面：右击上下文菜单(right-click context menu)、标签选项卡对话框(tabbed dialog box)、向导(wizard)、以及沿屏幕底部的工具条。Word 6.0的工具栏数量从原来的2个增加到了6个，因为添加了新功能，菜单也变得更加丰富。Word 6.0的界面如下图3所示。  
  [](http://www.flickr.com/photos/35750775@N02/3378542376/)  
  图3 Word 6.0界面  
  (4) Word 95是第一个32位Word应用程序版本，虽然它只是Word 6.0的过渡版本，但也引入了受大多数人欢迎的一个小的创新的功能：带有红色曲线的下划线的拼写检查。有些人将Word 95作为简单的、缩减的、互联网之前的最后一代文字处理器。  
  [](http://www.flickr.com/photos/35750775@N02/3378542380/)  
  图4 Word 95界面  
  (5) Office 97是一款轰动一时的产品，不仅对个人还是公司都带来了巨大的冲击，创下了软件销售的记录。Word 97标志着一个新的超丰富生产力应用程序时期的开始。  
  Office 97引入了“命令栏”，一个完全定制的用户界面，实际上与菜单和工具栏相同。每个菜单和工具栏能够拖放到屏幕的任何地方，可以浮动或者停靠。但是可以认为，在Word 97中最重要的用户界面(UI)是引入了层级菜单。而在所有以前的Word版本中，菜单是一个单独的项目列表，容易浏览和导航。而Excel从1-2-3复杂的用户界面(UI)中吸取了线索，已经引入了层级菜单，最后Excel模式成功了，于是Word 97采取了多层次的层级菜单。Word 97的界面如下图5所示。  
  [](http://www.flickr.com/photos/35750775@N02/3378542382/)  
  图5 Word 97界面  
  Office 97也引入了“Office 助手”，它更像是为用户提供上下文的帮助而设置的。“Office助手”如图6所示。  
  [](http://www.flickr.com/photos/35750775@N02/3378542384/)  
  图6 Office助手  
  (6) Office 2000引入了几个新的UI机制，即：  
   “自适应菜单(Adaptive Menus)”或“拟人化菜单(Personalized Menus)”。在菜单中首先显示的是最常用的菜单项(“短”菜单)，而当光标停留在菜单中一会儿后，会自动展开显示所有的菜单项(“长”菜单)，或者单击下方向下的双向箭头，也会展开显示所有的菜单项。并且，当用户使用菜单中不在默认的“短”菜单中的功能时，下次会自动将该菜单项添加到“短”菜单中。用户可以在“自定义”选项框中选中“始终显示整个菜单”关闭该功能。“自适应菜单”见图14所示。  
   “压缩的工具栏(rafted toolbars)”。两个或多个工具栏能够容纳在屏幕的同一行中，默认情况下，“常用”工具栏和“格式”工具栏“被压缩”在同一行中。由于在同一行中没有空间监控两个工具栏，因此最少使用的命令会在最后溢出的区域中，当用户使用该命令后又会出现在可见的工具栏空间中，就像“自适应菜单”一样。“压缩的工具栏”见图15所示。  
  [](http://www.flickr.com/photos/35750775@N02/3378548244/)  
  图7 Word 2000界面  
  (7) 在Office XP和Office 2003中引入了任务窗格(Task Pane)，在其中放置新的功能。  
  [](http://www.flickr.com/photos/35750775@N02/3378548246/)  
  图8 Office XP界面  
  [](http://www.flickr.com/photos/35750775@N02/3378548250/)  
  图9 Office 2003界面
* **为什么要改进界面**  
  正如Jensen Harris先生所说，Office UI开发团队的目的就是帮助用户能够“无缝地”找到和使用Office中的功能。我想，这可能是界面不断改进的一个最主要的原因。  
  (1) 功能越来越多，用途也越来越复杂  
  在Word 2.0中，有不到100个命令，工具栏也是简单有效地复制了在菜单结构中能找到的功能，没有仅存在于工具栏中的任何其它功能，因此，当时的Word开发团队能够较容易地计划理想的菜单结构，并使它们组织得更有意义。在Word 2.0的界面中能直接且快速地浏览菜单，因为大多数菜单少于10项，而且没有子菜单项。  
  而Word 6.0的工具栏数量从原来的2个增加到了6个。  
  在Office 97中，丰富的功能使它变得更复杂。其工具栏的数量迅速达到了18个，在顶级菜单中的命令数几乎增加了双倍。再以Word为例，其顶级菜单已满，虽然一些一直增加的功能仅仅能在工具栏中执行，但是一些功能仍然需要添加到菜单项中，可是已没有空间给它们了，而将命令添加到多级菜单中将为新的命令提供更多的空间，这样也能提供更多的功能，再加上在Excel中的成功应用。因此，在Word 97中也引入了层级菜单。  
  虽然功能越来越多，但用户的需求是无止境的。因此，Office的每个版本都使开发者集中精力开发新的功能，但这些新功能放在哪里呢？每次都很难决定这些新功能的位置。  
  如果仍然按照以前的用户界面结构，则在Office 2007中将有比Office 2003多10倍的任务窗格，而在Word 2007中则将发布约100个任务窗格。如何排列和组织呢？因此，任务窗格是最后一次试图找到调整老样式UI的一种方法。  
  下面的图10和图11演示了从Word 1.0到Word 2003中所包含的顶级菜单项、工具栏和任务窗格的数据对比。  
  [](http://www.flickr.com/photos/35750775@N02/3378548252/)  
  图10 Word各版本中的顶级菜单项数  
  [](http://www.flickr.com/photos/35750775@N02/3378548254/)  
  图11 Word 各版本中的工具栏和任务窗格数  
  可以看出，在每个版本中功能数持续增加。下图采用饼图来演示Word在过去15年中的成长过程，可以很清楚地对比各版本Word所带有的功能。  
  [](http://www.flickr.com/photos/35750775@N02/3378548258/)  
  图12 Word在每个版本中所添加的功能所占的比例  
  (2) 难以发现、查找和不方便使用一些命令  
  前面提到过，在Word 97中引入了层级菜单，但这也带来了一些不利的方面，即增加了复杂性。层级菜单使得用户难以浏览命令，每次都必须记住已访问过哪层菜单，没有访问过哪层菜单。曾经一个可视化的简单的菜单结构，现在由于带有分支结构而变得更复杂。浏览功能已不像查看商品列表，更像在研究一个复杂的数据结构。这样，对命令不仅难以查找，也不方便使用。  
  在Office 2000中的菜单栏和工具栏实质上都是充满的，这样在其中所添加的额外的项目只是整个结构中的很小的一部分，从一个版本到另一个版本时，用户根本不会注意到新添加的命令，也使得用户难以发现他们曾经要求的已添加到Office中的功能。总之，这种菜单结构使用户难以发现和使用新功能。  
  Office XP引入的任务窗格试图通过在屏幕中增加一个新的矩形框并在其中放置新的功能。任务窗格完全是一个附加物，它不试图改变现有的菜单栏和工具栏结构，大多数传统的功能都在菜单和工具栏中，而在任务窗格中放置新的功能。PowerPoint开发团队在利用任务窗格方面做了大量的工作，一些传统的功能例如幻灯片切换也迁移到了任务窗格中。但增加的任务窗格并未改进用户界面，相反，用户又多了一个要查找功能的地方，给查找某个命令带来了不方便。  
  [](http://www.flickr.com/photos/35750775@N02/3377733907/)  
  图13 任务窗格  
  (3) 越来越给人一种“臃肿的”感觉  
  Office 97是一个里程碑式的产品，增加了很多新的功能，但随之也给用户界面增加了复杂性，使人开始觉得Office是“雍肿的”。实际上，应用程序本身并不是“雍肿的”，至少，用户所需求的大量功能表明人们希望UI团队在这个空间做更多的事情。然而，菜单和工具栏开始显得与产品的丰富功能不太相称，这使用户界面开始感到臃肿。这样，在Office 2000中引入了新的UI机制来减少这种“臃肿的”感觉，这也标志着最终在Office 2007中重新设计新UI的开始。  
  Office 2000中的“自适应菜单”试图通过首先显示最常用的菜单项使顶级菜单更短，当用户使用菜单时，经常使用的菜单项将创建在“短”菜单中，而很少使用的菜单项会创建在“长”菜单中，如下图14所示。然而，自适应菜单是不成功的，实际上给界面添加了复杂性。首先，默认的“短”菜单中包含的菜单项是建立在“每个人在Office中仅仅使用同样的一些功能”假设的基础上的，但事实是，用户使用一个很广范围内的功能，而不仅仅限于“短”菜单中的功能，因此，某人的理想的“短”菜单并不代表着其它人也常常使用这样的“短”菜单。其次，如果用户不能在默认的“短”菜单中找到所需的功能，则必须浏览整个菜单，而这个过程需要先浏览“短”菜单，再单击下方的双向箭头浏览“长”菜单，而在“长”菜单中的菜单项在“短”菜单项之间，因此需要重新浏览，这使得工作效率降低。再次，自动定制的功能(即上次使用的菜单项将会出现在默认的“短”菜单中)也使得菜单项更混乱，因为开始时某菜单项在某个位置，之后某个时候它又不在那里了。因此，即使在Office 2007中的一些应用程序(例如Publisher、Project和Visio)仍使用老的用户界面样式，但已关闭了默认的“自适应菜单”。  
  [](http://www.flickr.com/photos/35750775@N02/3378553974/)  
  图14 自适应菜单  
  Office 2000中“压缩的工具栏”(如图15所示)也是为了减少“臃肿的”感觉，但与“自适应菜单”一样，也给用户界面添加了复杂性。命令的顺序不再固定，浏览功能也变得低效。  
  [](http://www.flickr.com/photos/35750775@N02/3377737551/)  
  图15 压缩工具栏  
  因此，大多数用户，特别是公司用户，关闭了这两项功能。  
  Office XP和Office 2003中的任务窗格也有一些缺点。增加了任务窗格后，所有的菜单和工具栏仍然必须存在，这样，任务窗格占用了空间而没有取代任何现有的用户界面(UI)。并且，Office中现在有“短”菜单、“长”菜单、层级菜单、可视的工具栏、工具栏列表，还有任务窗格，这使得用户仍然要在多处查找命令，增加了产品的复杂性。任务窗格的出现，也使得每个设计团队都想将他们的功能作为一个任务窗格，毕竟任务窗格是一个全新且干净的可放置功能的矩形框。这虽然使得设计者方便了，但却给用户带来了不便。一些任务窗格类似带多页的向导，一些事实上是对话框，一些只是带有一串说明文本的命令。没有真正考虑如何使所有的任务窗格协调一致，如何在老的UI中发现相关联的功能，如何一次使用两个功能。实际上，每个单独的功能都需要它自已的矩形框。因此，为减少复杂性，在Office 2003中已将任务窗格作为一个可管理的用户界面(UI)。  
  (4) Microsoft一直在努力改进，使产品适应发展的需求  
  正如在前面所看到的，Office界面一直在改进，在向着更方便用户的方向改进。例如，在Office 2000中引入的“自适应菜单”和“压缩的工具栏”，原本是试图改变应用程序给人的“臃肿的”感觉，但适得其反，未获成功。就像整理房间一样，只是把一些自认为不常用的东西藏到了床底下，乍一看是好的，但实际上不是。事后分析，这可能与没预料到计算机硬件和软件的迅速发展有关。还有更重要的一点是，仅考虑到了一个非常“窄”的范围的用户的需求，并且Office也是“保守的”，这点可以从自1989年以来其顶级菜单结构从未发生过变化看出来。但这对一些用户来说也是好事，因为节省了他们需要重新培训所付出的代价。但是，无论怎样改变，如果用户界面的结构没有发生变化，所有的功能扩充都处在矛盾之中，换句话说，考虑到了这一方面，可能会造成另一方面的不便。这样，给人“臃肿的”感觉仍然会存在。而且每次添加一个新的UI机制时，它都会被塞满，因为我们仅仅是添加，而从没有重新组织和移除它们，所以每个版本的复杂性都在持续增加。
* **Office 2007用户界面设计基础和思路**菜单和工具栏并不是不好，只是Office已经取得了长足的发展，使得菜单和工具栏不能很好的适应，因此，需要对新版本的Office界面进行重新设计，目标是为这些具备完整功能的高效的应用程序创建一个新标准的用户界面。而今天的Office 2007的用户界面在很大程度上受益于之前的版本，汲取了以前版本对用户界面所作改进的经验和教训。  
  (1) Office 2007改进的思路  
  有几种方式来改进菜单和工具栏不成比例的问题。一种是从产品中裁剪掉一半的功能，剩下的作为用户界面，这实际上要花大量的功夫来重新设计取消的另一半功能命令，但要取消哪一半命令呢？这很难取舍，因为无法想象哪些功能可能是一些人特别喜爱的。  
  另一种方式是将产品分解，例如，Word将会分成8个独立的应用程序：文本编辑器、打印/页面预览程序、表格制作、图片编辑、画图程序、拼写/语法检查器、邮件合并引擎和信封/标签打印机，每一个应用程序都有一个易管理的菜单和工具栏结构。这样，当安装Office时，将会在开始菜单中放置65个图标。但这样做的话，将完全背离了用户的要求。因为用户经常希望UI开发团队能对应用程序做更多更好的整合。  
  因此，Office UI开发团队决定不能够使Office 2007变得更笨重或更“支离破碎”，相反，要使应用程序成为一个整体，并重新创建用户界面以便于在未来十年里能够增加更丰富的功能。这也是为什么诸如上下文相关(contextualization)和图形目录(组，galleries)等概念对新的UI是如此重要的原因，它们将Office的功能分成更多可管理的单元但又维持着其整体性，从而使产品更强大。  
  (2) 大量收集和分析数据，为界面设计提供依据和基础  
  在用户安装Office 2003后不久，一个小气球会弹出来询问您是否想“帮助Office做得更好”，如果您单击它，将会登录到Microsoft Office用户体验改进程序(Microsoft Office Customer Experience Improvement Program)中，此后，您如何使用Office的一些数据信息(匿名的)就会间断地传送到Microsoft(注：不必担心，只是收集使用信息，如软件如何被使用、在什么硬件上使用等，您个人的数据信息不会被采集)。这些数据汇集到一个巨大的服务器中，供分析和作出决定。  
  下面是所收集的数据类型：命令的使用频率、经常使用的键盘快捷键、是否定制了工具栏、……等等。简而言之，所收集的只是软件本身的使用，决不会涉及到用户私人的信息。  
  在Office 2003发布后，共收集了大约13亿次会话。在前90天Word中超过3.52亿单击的命令栏。当然，这些数据只是一小部分，不可能保留收集到的所有数据。  
  因此，这些数据也是帮助Office UI开发团队做出明智的决定，即为Office 12开发新的用户界面的最主要的原因之一。  
  当然，数据只是辅助，但从中也可以了解到：用户经常使用哪些命令和不使用哪些命令、哪些命令在其它一些命令之前使用、哪些命令是使用键盘操作而很少使用鼠标、用户的屏幕多大、多少次使Excel最大化、哪些文档仅使用一次、哪些命令从未使用、东亚地区的用户经常使用哪些命令、……等等。  
  (3) Word 2003中最常使用的5个命令  
  经过大量的数据分析和调查，在Word 2003中最常用的5个命令是(依先后顺序列出)：  
  粘贴、保存、复制、撤销、加粗  
  在Word 2003中，使用这5个命令的次数大约占所有使用的命令数的32%，仅“粘贴”就占11%，是“保存”所使用次数的2倍。  
  然而，最常使用的10个命令的统计曲线是相当不平坦的。在使用率排第100位的命令(“接受变化”)和第400位的命令(“重设图片”)之间的命令的使用比例是不同的，但与排在第1位的命令和第11位的命令(“改变字体大小”)之间的命令的使用比例不同大致相同。这表明用户事实上使用Office的幅度的相当大的，这也给创建新的用户界面带来了挑战。  
  (4) 使用数据来影响和组织Office 2007用户界面  
  下面是使用这些数据帮助设计Office 2007用户界面的一个例子。最初，Office UI开发团队并没有认真考虑将剪切/复制/粘贴按钮放在“带形功能区(Ribbon)”中，每个人都自认为用户最常的就是使用快捷键Ctrl+X/C/V来做这些操作，鼠标用户则使用单击右键后弹出的上下文菜单项来执行这些操作。直到Office UI开发团队分析数据后才知道，用户虽然使用Ctrl+V和在上下文菜单中使用“粘贴”菜单项，但是使用工具栏中的“粘贴”按钮更多。因此，一旦认识到了“粘贴”按钮的重要性后，它就被放置在了界面中第一个选项卡左侧的第一个大按钮中。这也是Office 2007界面设计的一个特点，即把最常用的命令放在最显眼的位置。  
  如果将Office应用程序所使用的命令绘制在一张图表中，将会得到一条曲线，一些命令的使用将占得多数，而慢慢地每个命令的使用数将逐渐变缓。通过曲线可得知，用户经常使用哪些命令，也表现出了大多数用户使用产品的总的格局和平均“深度”。而在一些方面，Office 2003用户界面已经很好的实现了。因此，现在所需要做的是分析数据并理解它们，使用它们配合设计以达到要实现的更好的目的。  
  一是要寻找值得保留但却很少使用的功能，帮助用户找到它们不知道放在哪里的那些有用的功能。这些功能一般是Office已有的，但用户却由于不知道而没有使用，而是在别的地方寻求帮助或者自定义。因此，如果用户能发现这些功能，他们可能就会喜欢使用它们。例如，在Word中可以在文档后设置水印图案，很多人不清楚该功能而询问如何实现，因此在Word 2007中将其放在了突出的位置，这使得很多用户将该功能视为“很好的新功能”。当然也有一些例外，如低质量或设计差的功能不管在UI中能够如何容易发现它们，都将是不成功的，因此，重新增强一些旧功能而使得它们更引人注目；又如，用户很少会使用有着不合适名称的功能，因此，在新UI中变换了一些功能的名称，让更多的用户了解和使用它(但感到矛盾的是，有些用户已经在使用这些功能，若更换名称，会给他们带来不变)。  
  二是寻找不在显眼的界面中却频繁使用的功能。例如，Word中的“上标”，在Word 2003中，虽然它必须通过自定义手动添加到工具栏中，但它的单击量占“格式”工具条的30%。新的用户界面不仅突出了用户喜欢的功能，而且让更多的用户能知道并使用它。
* **现在的Office 2007界面**  
  [](http://www.flickr.com/photos/35750775@N02/3378555950/)  
  图16 Office 2007界面  
  (1) 简洁明了的设计。程序更简单易用，命令更容易找到。  
  各个按钮分布在窗口顶部且以分组形式显示，所需命令就是呈现在用户眼前，也不会让包含命令的隐藏工具栏隐藏在菜单或单独的窗格中，而是提供了一个控制中心，将各种要素集中在一起，使得它们极易被看到。按钮总是显示在相应位置，也不会消失。  
  (2) 将命令集形式提供的命令精简为最常用的命令。用户最喜欢使用且反复使用的命令被放在了最显眼的位置。  
  (3) 在“功能区(Ribbon)”中划分核心任务，每个任务由一个选项卡表示。  
  (4) 许多命令按功能排列形成各个命令集(即组)，将完成某类任务的所有可能需要的命令集中在一起。  
  (5) 许多命令仅在需要时才显示。即某些命令仅在执行相应操作时才会出现，而不是始终显示每个命令。  
  (6) 可以自定义“快速访问工具栏(QAT)”，使它像以前版本中的工具栏一样。  
  (7) 可以方便的在窗口中切换视图。  
  (8) “功能区”会自动调整以适应屏幕的大小。
* **结语**  
  从上面所述可以看出，Office的每一个版本都是尽量融合用户的需求，朝用户满意的方向改进，虽然说有时候可能适得其反。此外，随着Office的发展，必须有新的用户界面相适应，而重新设计的Office 2007用户界面，可以说是一个质的飞跃和创新。  
  Office软件本身就是一门艺术，是发展和智慧的结晶。

[**细品RibbonX(2)：认识Office的新界面—功能区(Ribbon)**](http://www.excelperfect.com/index.php/2009/03/26/ribbonxstudy2/)

2009年03月26日, 1:59 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

在Office 2007中，Microsoft使用了称作Ribbon的全新界面，大胆放弃了沿用已久的菜单命令栏式界面。可以说，这是办公软件的一次重大创新。从Office 97开始，Microsoft就一直尝试着更友好的用户界面，历经十年的努力，终成正果。  
**看看Ribbon的样子**  
我们先来看看Ribbon的样子，以及Ribbon中的一些组件或元素，还有一些相关的组件或元素。同时，看看RibbonX能够做些什么。  
Ribbon(功能区)  
[](http://www.flickr.com/photos/35750775@N02/3386232629/)  
图1  
*功能区：*界面顶部的大矩形区域，Office应用程序的控制中心，所有命令均集中于此，包含标题栏、Office按钮、快速访问工具栏及选项卡。  
Ribbon（功能区）将相关的命令和功能组合在一起，并划分为不同的选项卡，以及根据所执行的任务出现的选项卡。例如，如果想插入一个对象，则选择“插入”选项卡，其中包括可以在工作表中插入的对象和相关命令；如果处理表，Ribbon将添加一个与表相关的选项卡。也就是说，用户可以很容易找到当前需要使用的命令，而不必像原先一样，一个菜单一个菜单地查找。RibbonX主要应用于功能区及其内部的每一项。  
Quick Access Toolbar(快速访问工具栏)  
[](http://www.flickr.com/photos/35750775@N02/3387045418/)  
图2  
*快速访问工具栏：*Office 2007提供的一个工具栏，与显示的Ribbon选项卡独立，用于存放经常使用的命令，接受共享控件（可用于所有文档）和文档控件（仅用于所在文档）。默认情况下，在窗口的顶部、功能区的上方、Office按钮的右侧，提供最频繁使用的命令，是唯一与老版本的Office菜单和工具栏相似的地方。可将其放置于功能区的底部，在快速访问工具栏上右击，从弹出的菜单中选择“在功能区下方显示快速访问工具栏”即可。也可在其中添加或删除命令，按个人爱好和习惯定制快速访问工具栏，显示自已经常使用的命令，是最终用户进行自定义的主要位置。  
用户可以右键单击任何功能区控件（包括自定义的RibbonX控件）并将其添加到快速访问工具栏中。除非已启用了StartFromScatch模式，通常不允许RibbonX加载项改变快速访问工具栏。  
Office按钮  
[](http://www.flickr.com/photos/35750775@N02/3387045448/)  
图3  
[](http://www.flickr.com/photos/35750775@N02/3386232743/)  
图4  
*Office按钮：*位于界面左上角的一个圆形按钮，单击此按钮将显示Office菜单，与以前的Office版本中的“文件”菜单大致相同，集中了处理文档的方方面面，如共享文档、保护文档、打印文档、发布文档和发送文档等。这些命令对文档进行操作，而不是对文档的内容进行操作。使用RibbonX加载项可以随意改变Office菜单的内容，但不能自定义Office按钮本身。  
在Office按钮菜单的右下角，可以找到“Excel选项”，允许自定义Excel中的一些功能。在这里，也可以找到老版本的Excel中“工具”菜单下“选项”命令里的大多数命令。  
组件或元素  
[](http://www.flickr.com/photos/35750775@N02/3386232769/)  
图5  
*①选项卡（Tab）：*集中相关的命令，完成某种特定的任务。位于Ribbon的顶部，是构成功能区的主要内容，包含用于处理文档内容的UI控件。  
标准的选项卡为“开始、插入、页面布局、公式、数据、审阅、视图、加载项”，缺省的选项卡为“开始”选项卡，用户可以在想选择的选项卡上单击选择该选项卡。还有一个“开发工具”选项卡，必须在“Excel选项”中选中“常用”下的“在功能区显示’开发工具’选项卡”时才会出现在Ribbon中，该选项卡包含了与程序开发和XML功能相关的命令。  
RibbonX加载项可以创建自已的自定义选项卡，并改变内置选项卡的可见性和标签。  
*②组（Group）：*位于每个选项卡内部，集中最相关的命令，完成特定的子任务。  
选项卡包含组的集合，组中则包含各个UI控件，这些控件按相关逻辑组合在一起。RibbonX加载项可以改变内置组的可见性，并创建自已的自定义组，但不能改变内置组的内容。有些组的右下角包含有对话框启动器，点击时可以显示与组相关的对话框。  
*③命令（Command）：*一个命令或一个命令菜单，代表希望执行的操作，有不同的表现形式，例如按钮、库、菜单、编辑框等。  
（1）按钮（Button）  
单击按钮——单击执行一项操作。  
切换按钮（ToggleButtons）——在两种状态中来回变换。  
拆分按钮（SplitButtons）—— 联合了单击按钮和一组可供选择的菜单（menu）。  
在自定义Ribbon时，可以定制大的或小的按钮，可以显示或隐藏其标签，也可以有工具提示或超级提示，可以从内置按钮中复制图像，使用自已的图标图像。也可以隐藏内置按钮的标签。还可以创建按钮组。  
（2）复选框（CheckBox）  
具有与切换按钮(toggleButton)相同的行为，但是样式完全不同。复选框用于设置或选项，不会改变文档中的内容。与内置粗体按钮一样，切换按钮在某种情况下更好地反映所选择的状态。Office 2007最明显的内置复选框在“视图”选项卡中。  
（3）编辑框（editBox）——允许用户输入文本。  
（4）组合框（comboBox）——显示预定数据文本列表，也可以输入文本。  
（5）下拉控件（dropDown）——显示预定的数据列表供用户选择。  
*辅助元素*——起提示、分隔和修饰作用，美化布局，不执行任何操作。  
（1）标签控件（LabelControl）  
提供一些额外的信息，确保用户正确使用命令。  
（2）分隔条（Separator）  
在逻辑上分开组中的控件，没有标签，只是一个单独的ID，可以很恰当将控件分开成几部分。  
（3）盒（Box）——其中包含着一系列控件作为一个单元。  
（4）菜单分隔条（menuSeparator）——像是标签控件和分隔条的组合。分隔条为一条水平线分开不同的菜单。  
上下文选项卡  
  
[](http://www.flickr.com/photos/35750775@N02/3386232805/)  
图6  
*上下文选项卡：*仅在编辑特定对象时才出现在界面中。即在用户需要时，为了响应操作才出现在界面中的选项卡，其中包含所有用于处理这些对象的UI元素。例如：  
①选择文档中的一个项目。  
②适用的上下文工具的名称将以强调文字颜色出现，上下文选项卡将出现在标准选项卡集旁边。  
③上下文选项卡提供用于处理所选项目的控件。  
在使用、艺术字、图表或表时，都会出现上下文选项卡。当没有选中对象时，与之相关的上下文选项卡也将隐藏。  
RibbonX加载项可以改变内置选项卡集的可见性，并向其添加自定义选项卡。Office 2007不支持创建自定义上下文选项卡集。选项卡集包含上下文选项卡，其作用方式与常规选项卡相同。  
程序集选项卡  
[](http://www.flickr.com/photos/35750775@N02/3386232845/)  
图7  
*程序集选项卡：*在某种模式或视图下，替换标准选项卡集。如上图7，切换到打印预览下的程序集选项卡。  
对话框启动器（dialogBoxLauncher）  
[RibbonX(2)-8](http://www.flickr.com/photos/35750775@N02/3387045618/)  
图8  
*对话框启动器：*一个箭头，表示提供更多的选项，出现在某些组中。单击对话框启动器将打开相关的对话框或任务窗格，提供更多的选项，如下图9所示。  
[](http://www.flickr.com/photos/35750775@N02/3386232917/)  
图9  
库(Gallery)  
  
[](http://www.flickr.com/photos/35750775@N02/3386232949/)  
图10  
*库：*应用了一组命令后的结果，可以很方便地一次性完成所需的样式，并能提供实时预览(即用户在库所提供的结果上移动指针时，会显示应用该结果后的实际效果)。  
其目的是将功能积累到库中以供选择，便于用户只需单击一次就能得到相同的结果。库可大也可小，能够自动调整图像。可以按比例增加图像，也可以指定库显示的行数和列数，还可选择在库的条目中显示标签，在库的底部添加按钮。  
增强的屏幕提示  
当将鼠标悬浮在功能区中某命令上时，不仅显示该命令的名称（屏幕提示Screentip）而且将提示其详细的功能或使用描述（超级提示Supertip），例如该命令的快捷键、该命令执行的操作、典型使用情况，有图形化演示命令执行的操作或显示可以打开来访问更多选项的对话框。如图11所示。  
[](http://www.flickr.com/photos/35750775@N02/3387045716/)  
图11  
由于Excel 2007中工具提示的上述特征，因此现在称它们为增强的屏幕提示或超级工具提示。主要提供给用户特定命令的一个快速预览和通常的用途，而无须使用快速向导搜索或使用帮助功能。如果需要更详细的信息，则可以按F1键打开帮助。  
可以在Office按钮下的“Excel选项”中改变屏幕提示。在“Excel选项”对话框的“常用”选项卡中，可以选择“屏幕提示样式”为“不在屏幕提示中显示功能说明”或“不显示屏幕提示”。  
其他界面组件  
*任务窗格* Office 2007中仍存在一些任务窗格，而且现在支持同时打开多个任务窗格。COM加载项现在可以创建托管ActiveX控件或Windows窗体控件等内容的CustomTaskPane。  
*MiniToolbar（微型或迷你工具栏）* MiniToolbar是选定文本和右键单击上下文菜单时出现的常见格式命令集合。RibbonX加载项不能修改MiniToolbar的内容，但它们可以禁用或重用其内置命令。  
*上下文菜单* 这些上下文菜单与Office先前版本中的上下文菜单相同。在Office 2007中，RibbonX不应用于上下文菜单，但使用CommandBars对象模型可以像以前版本那样扩展和自定义它们。  
*状态栏* 状态栏包含了几个方便使用的新控件，例如页面视图和显示比例（微调控件）及录制宏。可以使用VBA隐藏状态栏，但使用RibbonX不能自定义状态栏。  
**Ribbon的由来**老式的菜单命令栏结构存在诸多不便  
导致Ribbon诞生的，还是Office沿用十多年的老式菜单命令栏结构。正是因为其越来越膨胀带来诸多不便，才使得Microsoft一直探索着改进，最终出现了Ribbon。下面列出了老式菜单存在的一些缺点：

* 菜单是层级嵌套的，使得很多有用的命令或功能很难被发现。
* 使用某项功能命令往往需要多次的单击。
* 每一次版本的升级带来的功能增加，使得Office变得越来越复杂，结构显得越来越臃肿。
* 对于新用户来说，很多方便的命令和选项都不知道，熟悉并熟练使用各种操作命令需要长时间的学习。
* 新功能或很多有用的功能被添加到更深层次的菜单中，更难以被用户发现。
* 软件有着丰富且强大的功能，但是很多使用软件很长时间的用户仍然只是使用着很少的一部分功能。
* 很多用户自已开发了一些实用功能，但随着使用软件的深入，发现软件自身早已存在该功能。
* 菜单和命令栏可以随意定制，可以悬浮在界面的任何地方，但这会造成不同用户间使用的混乱，也会造成屏幕的混乱，同时对于新用户说不知道如何恢复。譬如，某一用户改变了菜单或命令栏的顺序，或者自已定制了菜单或命令栏，另一用户使用时往往需要重新适应这种结构，或者该用户到其他计算机上使用时，也会给自已造成不便。

为了创造新的Ribbon界面，Microsoft花费了大量的时间和投资，包括调查用户实际工作中如何使用Office，他们使用Office的习惯、喜欢使用的命令、操作命令的顺序、最常使用的命令等。在大量调查研究的基础上，设计了新的Ribbon界面。  
**为什么称作Ribbon**  
早在2003年秋，Office UI开发团队努力创作出一些原型来试图决定新UI的方向，他们画了一些图片并讨论不同方向的优点。在进行办公室头脑风暴的过程中，Jensen Harris提出了“Ribbon”命令的想法，像老式的卷轴画卷一样，一长条的纸绕着两个轴，通过旋转其中一个轴来回滚动纸。在计算机中，它看起来像放在细条带中的一个极长的一系列滚动的命令。正如Office 12的Ribbon，带有选项卡，并相互挨着排列。  
总之，在Office 12中采用了与Ribbon相同的思想，组织成一组图片的形态称作Ribbon，只是分成了不同的选项卡。  
**Ribbon的目的**简化Office界面，使界面整洁、协调、美观、丰富、一目了然，从而用户更容易发现和使用所需的功能。  
**Ribbon的特点**

* 将命令按逻辑分组，将最常使用的命令放到用户最容易看到的地方且以最醒目的方式显示，将以前深藏的或难以找到的功能移到了界面上，大大增强了命令的可访问程度。
* Ribbon始终固定在屏幕上方，不能随意定制，因而界面是固定的，不会因某些操作而造成混乱。如果要定制Ribbon，则需要编程或第三方工具，且需要理解XML及其规则。
* 一般来说，大多数用户只使用了软件20%的功能，而这20%的功能对于特定的用户来说可能又各自不同，在Excel 2003中，各人可以根据自已的喜好自定义菜单和工具栏方便自已使用最常用的功能。在Excel 2007中，如果不使用RibbonX，则只能在快速访问工具栏（QAT）中添加自已常用的命令。
* 由于不能随意定制，因此命令不总是能按照某人所需要的顺序随意排列。有时，实现一项任务可能需要花时间单击不同的选项卡，使用不同组中的命令。
* 自定义Ribbon只能在设计时，而不是在运行时，通常不能动态添加选项卡、组或控件。
* 很容易移除定制的Ribbon，只要文档中不包含定制Ribbon的代码，都将恢复默认的Ribbon界面。而不像以前版本的Office，自定义菜单和命令栏后难以恢复，尤其对于新用户。并且，除了加载项、全局模板外，在某文档中的Ribbon定制不会影响新文档的Ribbon。
* 一些选项卡或命令是上下文敏感的，在需要时才显示。而不像以前版本的Excel，在界面中包含了几乎所有的命令。例如，只有选中图表后，才会出现图表工具设计、布局、格式选项卡。这样，简化了功能区。
* Ribbon能够自动适应窗口的大小，但主要命令始终可见。屏幕分辨率越高，Ribbon可视化越好。并且，Ribbon相对于菜单来说，可能占用更多的屏幕空间，但可以将其最小化。
* 当鼠标放置在功能区某命令上时，会有更详细的操作和信息提示。
* 具有实时预览功能，即光标放置在库或样式上时，会自动应用到相关元素上。
* 以前版本的加载宏或自定义菜单仍然可用，只不过放在了“加载项”选项卡中，相应添加在“菜单命令”组和“工具栏命令”组。
* 当选中文本或者右击时会出现一个“迷你”工具栏，方便常规的格式设置。
* 状态栏放置了更多的显示功能。

**Ribbon的一些使用技巧**

* 可以在任一选项卡标题中双击鼠标，将隐藏Ribbon，再次双击将出现。在隐藏状态下，可单击某选项卡来查看Ribbon并选择其中的命令。
* 可以在快速访问工具栏中放置要使用的但在功能区选项卡中找不到的命令。在快速访问工具栏中，不仅可以添加常用的命令按钮，还可以添加复选框、库等元素。
* 访问键允许使用键盘而不是键盘快捷键来访问功能区、Office按钮和快速访问工具栏。按下Alt键将进入使用访问键的模式。这时，在各个部分将出现小方框中包含的字母，代表需使用某功能要按下的键盘字母。例如，如果想使用“插入”选项卡，则按下N键，此时“插入”选项卡中各组按钮出现相应的访问键，按下某键将使用该按钮的功能。  
  在访问键模式下，可以使用箭头键和Tab键在功能区中移动：  
  (1)使用左箭头和右箭头在选项卡间移动。  
  (2)使用向下箭头键激活该选项卡，然后使用箭头键在组和命令中移动。  
  (3)激活选项卡后，使用向上箭头键移动到快速访问工具栏。  
  (4)从快速访问工具栏，使用向左箭头在工具栏中移动，然后到Office按钮。  
  (5)使用向下箭头键访问Office按钮，使用箭头键在命令之间移动。  
  (6)使用Tab键在活动选项卡的各组中每个命令间从左向右依次移动，到达最后一组的最后一个命令后，按Tab键将到达“帮助”按钮，然后是“Office按钮”、“快速访问工具栏”。使用Shift+Tab键从右向左依次移动。  
  (7)使用Enter键将执行所选择的命令。  
  一旦使用箭头键在屏幕中导航后，键提示将消失。按Alt键两次才再次显示键提示。要移除键提示并返回到文本输入模式，则再次按Alt键。  
  如果想使用原先的键盘快捷键，例如Alt+E组合键来打开“编辑”菜单，则会显示一条消息，表示正在使用Office 2003访问键，如果知道要使用的键的顺序，则可以继续输入，或者按Esc键取消。
* 可以使用鼠标滚轮，在功能区选项卡之间切换。
* 双击Office按钮将关闭应用程序。如果之前对文档进行过更改，则会提示是否保存更改。

**自定义Ribbon**Ribbon并不是绝对不能定制的，使用一些工具并了解XML规则，也能够自已改造Ribbon。在自定义Ribbon前，您需要：

* 配置好Office选项，如调出“开发工具”选项卡，在Office选项中选中“显示加载项用户接口错误”选项。
* 能够编辑和调试Open XML的工具
* 会编写XML代码——实现Ribbon的结构
* 会使用VBA创建并使用回调（Callbacks）——实现Ribbon控件的功能
* 或者会使用一些高级编程语言如C#——实现Ribbon控件的功能

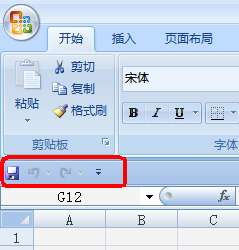
当然，自定义Ribbon可能需要比自定义菜单和工具栏更多的代码，并且要编写XML和VBA代码。因而，首先进行合理的规划，然后创建代码是一个良好的习惯。

[**细品RibbonX(3)：认识快速访问工具栏(QAT)**](http://www.excelperfect.com/index.php/2009/04/03/ribbonxstudy3/)

2009年04月3日, 1:20 下午

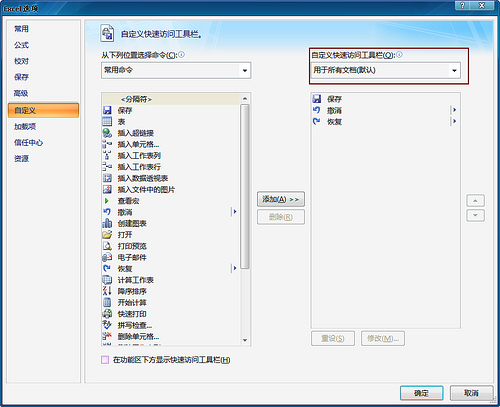
1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

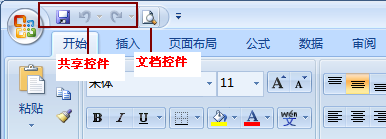
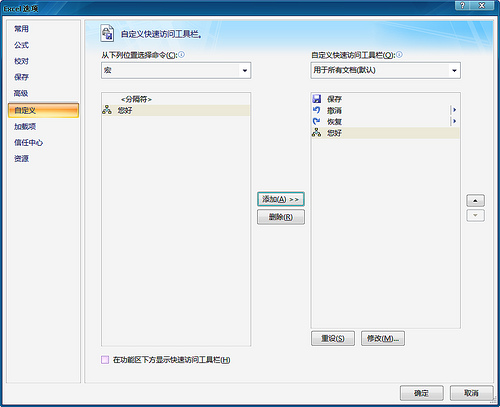
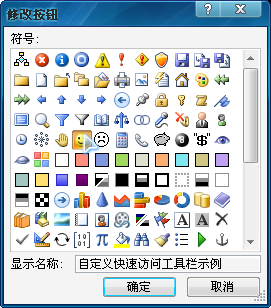
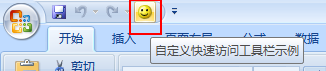
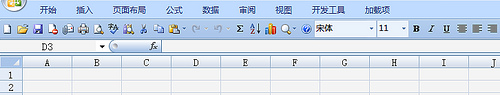
**QAT概述**  
快速访问工具栏是唯一与以前版本的工具栏相仿的Office 2007界面元素，默认情况下，快速访问工具栏位于界面上方、Office按钮右侧，如下图1所示。可以在快速访问工具栏中放置经常需要使用的命令，以方便快速访问。  
[](http://www.flickr.com/photos/35750775@N02/3408076483/)  
图1：快速访问工具栏（QAT）  
与以前版本的工具栏相似，您可以自定义快速访问工具栏，在其中放置经常使用的命令控件，以方便通过直接单击来使用其功能。然而，与以前版本工具栏不同的是，快速访问工具栏只能放置在功能区的上方或下方，不能随意放置或悬浮在界面上，从而消除了管理工具栏的复杂性。  
**放置QAT**默认情况下，快速访问工具栏在功能区上方，您也可以将其放置在功能区的下方。在快速访问工具栏中单击右键，选择“在功能区下方显示快速访问工具栏”，将快速访问工具栏移至功能区下方，如图2所示。  
[](http://www.flickr.com/photos/35750775@N02/3408076513/)  
图2：在功能区下方放置快速访问工具栏（QAT）  
**在QAT中添加命令**在QAT中可以随意添加常用的或自已喜欢的命令，且添加的过程比以前的版本更简单和容易。可以使用下列任一方式在QAT中添加命令：



* 单击QAT右侧下拉图标（如图1所示），选择“其他命令”。
* 单击“Office按钮——<应用程序>选项——自定义”。
* 在QAT或Ribbon或选项组中单击右键，选择“自定义快速访问工具栏”。

上述三种方式都会出现如图3所示的“<应用程序>选项”对话框，在其中可以选择要在快速访问工具栏中添加或删除的命令。  
[](http://www.flickr.com/photos/35750775@N02/3408076563/)  
图3：自定义快速访问工具栏（QAT）对话框

* 右击希望添加到快速访问工具栏的命令，选择“添加到快速访问工具栏”。这可能是最快捷的方式。

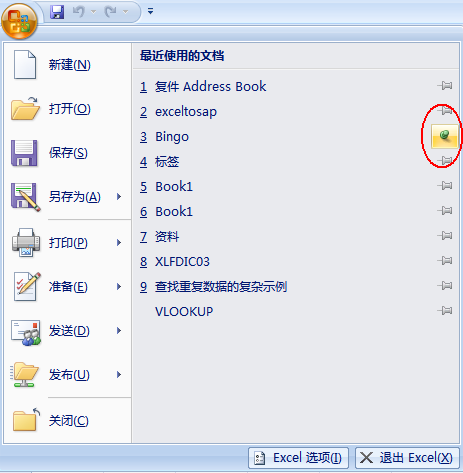
**看看QAT中的命令**  
在图3中，有一个自定义快速访问工具栏下拉列表框，从中可以确定将添加的命令用于所有文档，还是只用于指定的文档。是共享控件（Shared Control）还是文档控件（Document Control）也可从QAT中看出来，如图4所示。共享控件和文档控件的图标稍微有点差异，控件周围的小框表明该控件是文档控件。  
[](http://www.flickr.com/photos/35750775@N02/3408076591/)  
图4：QAT中的命令作用范围不同  
共享控件能够用于所有打开的文档，而文档控件只能用于该控件所在的文档。  
此外，在QAT中不仅可以添加命令按钮，而且还可以添加组、库等，并且也可以添加自定义的命令。还可以在图标之间添加分隔符，从逻辑上对图标分组。  
提示：在图3所示的选项对话框中，右侧指向下的箭头代表组、向右的箭头代表库、带有向右箭头的垂直条代表拆分按钮。  
**在QAT中添加自定义功能**如果是从头开始创建功能区，那么可能需要使和XML来自定义快速访问工具栏。然而，如果您仅仅希望在快速访问工具栏中添加带有自定义功能的按钮，那么可以按照以下步骤实现：  
1、调出如图3所示的“自定义快速访问工具栏”对话框。  
2、在对话框左侧的“从下列位置选择命令”组合框中，选择“宏”，此时在下方将显示所有可用的宏列表。  
3、在对话框右侧的“自定义快速访问工具栏”组合框中选择宏的作用范围（即用于当前文档还是用于所有文档）。  
4、从宏列表中选择所需要的宏，单击“添加”按钮。如图5所示。  
5、单击“确定”按钮。  
[](http://www.flickr.com/photos/35750775@N02/3408076641/)  
图5：在快速访问工具栏（QAT）中添加自定义命令  
您可能对自定义命令的图标不太满意，此时可以单击图5中的“修改”按钮来修改图标。单击“修改”按钮后，出现如下图6所示的“修改按钮”对话框，从中您可以选择喜欢的图标，在下方的“显示名称”文本框中，可以修改当鼠标悬浮在自定义按钮上时所显示的提示文本。  
[](http://www.flickr.com/photos/35750775@N02/3408884512/)  
图6：修改自定义按钮的图标及提示  
[](http://www.flickr.com/photos/35750775@N02/3408076737/)  
图7：在QAT中添加的自定义按钮，鼠标悬浮在其上方时会显示在图6中输入的文本提示  
**模仿Excel 2003界面**可以在Excel 2007中模仿Excel 2003的菜单和工具栏界面，如图8所示。  
[](http://www.flickr.com/photos/35750775@N02/3408884590/)  
图8：模仿Excel 2003界面  
实际上，在图8中，将Excel 2007的功能区最小化，然后在功能区下方放置快速访问工具栏，并按照以前的Excel版本中的“常用”和“格式”工具栏顺序在快速访问工具栏中依次放置相应的命令，从而达到模仿Excel 2003界面的效果。  
**重置快速访问工具栏**  
如果您设置了快速访问工具栏，而现在希望将其恢复到默认状态，那么您可以单击图5所示的对话框中的“重设”按钮即可。  
也可以重新排列快速访问工具栏中图标的顺序。在图5所示对话框的最右侧，有一个箭头向上和一个箭头向下的按钮，在选中命令后，单击这两个按钮即可调整命令放置的顺序。  
当然，您也可以很容易删除快速访问工具栏中的命令，在图5所示对话框中有一个“删除”按钮，选中需要删除的命令后，单击此按钮即可。最简单的方法是，直接右击该命令，选择“从快速访问工具栏删除”。  
**QAT文件**  
Excel会将自定义QAT的信息存储在名为excel.qat的文件中，该文件位于：  
C:\Documents and Settings\<用户名>\Local Settings\Application Data\Microsoft\Office  
文件夹内。  
可以使用记事本等文本编辑器或XML编辑器来查看该文件，也可编辑该文件。并且，一旦修改了快速访问工具栏，该文件就会立即更新。  
如果您已经定制好的快速访问工具栏，可以将该文件复制给他人，与他人共享您定制的成果。

[**细品RibbonX(4)：认识Office的其他界面元素**](http://www.excelperfect.com/index.php/2009/04/08/ribbonxstudy4/)

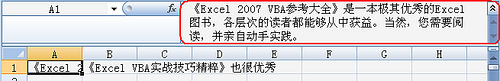
2009年04月8日, 1:39 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

在前面两节中，我们讲述了Office 2007新的用户界面——功能区（Ribbon）以及与以前版本十分相似的快速访问工具栏（QAT）。现在，让我们来简单看看Office 2007其他常见的界面元素（主要以Excel 2007为例）。  
**Office按钮**  
Office按钮包含了以前版本的“文件”菜单中的大部分文档操作功能，并且单击其右下角的“<应用程序>选项”按钮还可以进入应用程序选项对话框，控制应用程序外观，对应用程序进行设置。  
操作技巧1：可以在应用程序选项对话框中自定义显示最近使用文档的数目。以Excel为例，单击“Office按钮——Excel选项——高级”，在“显示”中可以调整“显示此数目的最近使用的文档”的数目，最多可以显示50个文档。  
操作技巧2：可以将文档固定在最近使用的文档列表中。如图1所示，在最近使用的文档列表中，每个文档的右侧有一个灰色的图钉，单击该图钉，图钉会变成绿色且像钉在纸里面一样，表示该文档已被固定，此后该文件将会一直出现在最近使用的文档列表中。如果需要取消对某文档的固定，只需再次单击图钉图标。  
[](http://www.flickr.com/photos/35750775@N02/3422607641/)  
图1：将某文档固定在最近使用的文档列表中  
操作技巧3：如果希望Office按钮闪烁，则请参见日志“[使Office按钮闪烁](http://www.excelperfect.com/wordpress/2009/03/19/officebuttonflash/)”。  
操作技巧4：如果希望禁用最近使用的文档列表，请参见日志“[禁用最近使用的文件列表](http://www.excelperfect.com/wordpress/2009/03/18/disablefilelists/)”。  
**Excel编辑栏**  
Excel 2007极大地增强了编辑栏的功能：

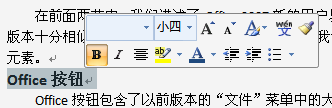
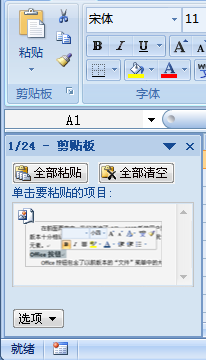
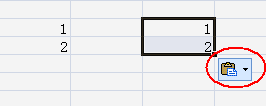
* 可以调整左侧名称框的大小。
* 可以展开或折叠编辑栏，也可以调整编辑栏的大小。当展开编辑栏并调整编辑栏的大小后，可以看到更多的内容，而不必担心会遮住单元格。如果没有调整编辑栏，也可以单击其右侧的上下按钮查看其中的内容。

[](http://www.flickr.com/photos/35750775@N02/3422607679/)  
图2：Excel 2007编辑栏可以展开且可以调整大小。

**Excel状态栏**Excel 2007也极大地增强了状态栏的功能：

* 状态栏左侧增加了“录制宏”按钮，直接单击即可以开始录制宏。
* 在状态栏中可以查看所选单元格区域的数据信息，如图3所示。这里有一个技巧，您如果想要知道单元格区域数值之和，只需选中该单元格区域，状态栏中会显示数值的和。当然，也可以知道数值的平均值、最大值、最小值等。Excel 2007可以自定义在状态栏上显示的统计数据，只需在状态栏中单击右键，选择相应的项目即可。
* 在状态栏中间的三个图标可以在普通视图、布局视图和分页视图之间快速切换。
* 状态栏右侧的缩放滑块可以方便地调整页面显示比例。

[](http://www.flickr.com/photos/35750775@N02/3422607783/)  
图3：在状态栏中可以显示所选单元格区域的平均值、包含数据的单元格数、包含数值的单元格数、最大值、最小值、计数求和。

**右键菜单（mini工具栏）**  
在选择文本后，在屏幕上会出现一个若隐若现的浮动工具栏（mini工具栏），如图4所示。将光标移动到mini工具栏上，它就会清晰地显示出来。mini工具栏中是一些常用的格式设置命令，以方便对文本进行操作。  
[](http://www.flickr.com/photos/35750775@N02/3422607865/)  
图4：选择文本后出现mini工具栏  
在文档中选择文本后单击右键，会出现带mini工具栏的右键菜单，包含更多常用的操作命令。  
可以在任何界面元素上单击右键，将出现快捷菜单，包含了与所选择的对象相关的常用操作功能。  
**任务窗格**Excel 2007也具有以前版本相同的任务窗格功能。例如，单击“开始”选项卡“剪贴板”组右下角的对话框启动器，将在界面左侧出现“剪贴板”任务窗格，如图5所示。  
[](http://www.flickr.com/photos/35750775@N02/3423414734/)  
图5：“剪贴板”任务窗格  
单击某些命令后，也会出现相应的任务窗格。例如，单击“插入”选项卡“插图”组中的“剪贴画”按钮，将在界面右侧出现“剪贴画”任务窗格。  
**智能标记（Smart Tag）**当用户执行某项操作后，在界面中会自动出现一个小图标，单击该图标会出现一些命令选项，如图6所示。  
[](http://www.flickr.com/photos/35750775@N02/3423414768/)  
图6：复制/粘贴后出现的智能标记，单击下拉按钮出现粘贴选项

[**细品RibbonX(5)：开放的XML文件格式**](http://www.excelperfect.com/index.php/2009/04/25/ribbonxstudy5/)

2009年04月25日, 3:06 下午

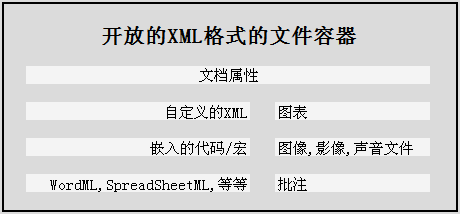
1 颗星2 颗星3 颗星4 颗星5 颗星(**3** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

众所周知，Office 2007采用了新的文件格式。例如，在Excel 2007中，当您保存文件时会默认以xlsx为扩展名来保存，采用这种格式的文件不包含宏；如果要保存带有宏的工作簿，那么应该以xlsm为扩展名保存。（当然，您可以在Excel选项中更改保存文件时默认的文件格式）。  
新的文件格式有：

* xlsx：不带有宏的工作簿文件。
* xlsm：包含宏的工作簿文件。
* xltx：不带有宏的工作簿模板文件。
* xltm：包含宏的工作簿模板文件。
* xlsa：加载项文件。
* xlsb：与旧的xls文件格式相同，但是可以兼容新特性的二进制文件。
* xlsk：备份文件。

除xlsb文件外，其他的文件都是可以打开的XML文件，意味着其他文件可以读写这些文件。  
Office 2007的新文件格式称为开放的XML文件格式，这是一种新的基于XML的文件格式。新的文件格式改进了文件和数据管理、数据恢复和可交互操作的能力。采用这种格式的文件实际上是压缩文件，您可以将其解压缩到文件夹中查看文件内部的情况。任何支持XML的应用程序都能访问和处理Office 2007文件信息，即只要提供了对XML的支持，那么非Office应用程序软件也可以方便地创建和操纵Office文件。即便是在系统没有安装Office软件的情形下，也可以查看文件的内容。  
下面，主要以Excel 2007为例，并使用[Jan Karel Pieterse](http://www.jkp-ads.com/)研究Excel 2007新的文件格式的工作簿示例，来了解Office 2007开放的XML文件格式。

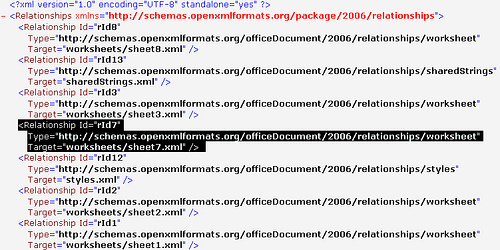
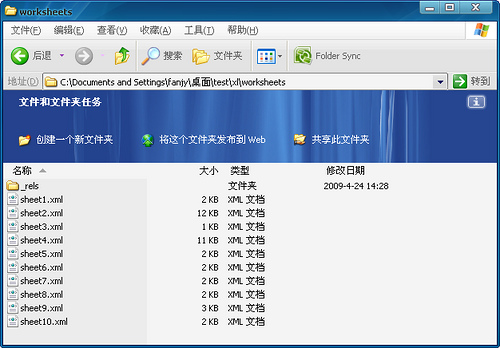
**探索Office 2007 XML文件格式**新的XML文件格式基于压缩的ZIP文件格式规范,每个ZIP容器由许多包含Microsoft称作“部件”的XML文件组成。大多数部件是描述工作簿数据、元数据、文档信息的XML文件，非XML文件例如代表图片或OLE对象的二进制文件也包含在ZIP文件中。也有描述部件之间关系的相关联的部件，从而提供工作簿文件的结构，这些部件是工作簿的内容和内容部件如何组合在一起的关系详情。  
下面演示了开放的XML格式的文件容器。  
[](http://www.flickr.com/photos/35750775@N02/3472053919/)

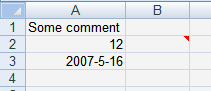
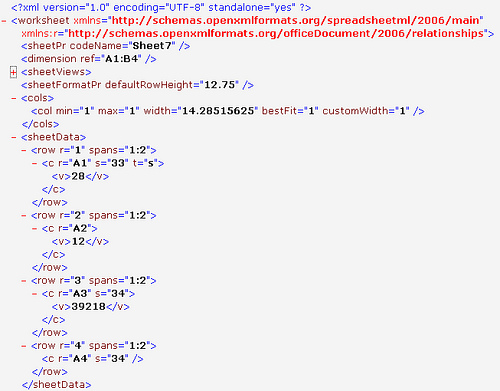
**分析Excel 2007文件**  
一个Excel 2007文件实际上就是一个压缩了的容器（ZIP文件），其中包含一些文件夹和文件。您可以将Excel 2007文件的扩展名更改为.zip或.rar，然后双击该文件来或者将其解压缩到特定的文件夹中来查看其包含的文件夹和文件。在ZIP文件中，可以看到一些XML组件，其中一些独立的XML文件存储了文档所有的设置（setting）、样式（style）、绘图（drawing）和数据等内容。  
例如，新建一个Excel 2007工作簿，在该工作簿中输入数据、插入图表、放置控件和图片及图示、输入VBA代码等，然后将该工作簿命名为test1.xlsm，关闭该工作簿。将工作簿test1.xlsm的扩展名更改为test1.xlsm.zip，并解压缩到文件夹中。可以看到其XML文件包结构如图1所示。  
[](http://www.flickr.com/photos/35750775@N02/3472053923/)  
图1：Excel 2007的.xlsm文件的内容  
该工作簿文件包含3个根文件夹和1个名为[Content\_Types].xml的文件。其中：

* \_rels文件夹：有一个名为.rels的文件，包含关于程序包关系的信息。其中有三个关系属性：Id为任意字符串，在.rels文件中必须是唯一的；Type表示关系的类型；Target指定包含关系的目标的文件夹和文件。  
  很多子文件夹内部都包含有一个名为\_rels的文件夹，这些文件夹都包含有定义与其他部分之间关系的XML文件。  
  关系：Excel 2007文档由部件组成。这些部件通过定义在不同的XML文件里的关系联系在一起。
* docProps文件夹：包含各种XML文件，描述文件属性和应用程序设置。类似于单击Office按钮，选择“准备——属性”后看到的内容。docProps文件夹包含着至少一个名为app.xml的文件和一个名为core.xml的文件。这些文件包含着Excel文件的元信息，例如创建者的名字、修改的和创建的日期。
* xl文件夹：包含文件的核心部分。根据工作簿中的内容，该文件夹中含有一些子文件夹，并且还含有工作簿设置的XML文件。如果工作簿中含有VBA代码，那么文件夹中还有以bin为扩展名的二进制文件。
* [Content\_Types].xml文件：列出了包括在Excel文件程序包中其它部件的内容类型。

打开名为“xl”的文件夹，其包含的内容如图2所示。  
[](http://www.flickr.com/photos/35750775@N02/3472053927/)  
图2：文件夹“xl”中的内容  
可以看出，除VBA工程、图片等特殊情形以二进制格式存储外，几乎所有文件都是XML文件。下面列举一些。

* \_rels：其中的关系文件workbook.xml.rels定义了工作簿、数据和格式部件的关系。
* activeX：包含控件设置的XML文件。
* charts：包含图表设置的XML文件。
* chartsheets：包含工作簿中每个图表数据的XML文件。
* diagrams：描述工作簿中形状（SmartArt）的XML文件。
* media：包含内嵌的媒体文件，例如GIF文件、JPG文件等。
* tables：包含带有每个表的数据的XML文件。
* theme：包含工作簿主题的数据的XML文件。
* worksheets：包含工作簿中每个工作表的XML文件。

除一些文件夹外，还有一些单独的XML文件。例如，workbook.xml文件包含的部分内容如图3所示。  
[](http://www.flickr.com/photos/35750775@N02/3472053931/)  
图3：文件“workbook.xml”包含的部分内容  
在这个文件中，我们可以找到各个工作表对应的id号，即r:id。如果希望对某工作表进行操作，那么先找到其r:id。例如，假设我们想在工作表Comments中添加内容，则需要找到包含该工作表内容的xml文件，即要找到图3中工作表对应的r:id。在图3中该工作表的r:id为rId7。  
在文件夹“\_rels”中，包含着一个名为“workbook.xml.rels”的文件，包含着XML代码，如图4所示。  
[](http://www.flickr.com/photos/35750775@N02/3472053935/)  
图4：在文件“workbook.xml.rels”中，指明Id值所对应的工作表  
如图4所示，名为“sheet7.xml”的文件代表工作表“Comments”(rId7)。下图5为文件夹“worksheets”中的文件列表。  
[](http://www.flickr.com/photos/35750775@N02/3472053937/)  
图5：文件夹“worksheets”中的文件列表

**使用压缩包中的XML文件来操作Office文档**  
读取单元格的值  
工作表“Comments”中的内容如图6所示。  
[](http://www.flickr.com/photos/35750775@N02/3472057663/)  
图6：工作表Comments中的内容，单元格B2中包含批注  
sheet7.xml文件的内容如图7所示。  
[](http://www.flickr.com/photos/35750775@N02/3472057673/)  
图7：文件sheet7.xml中的内容，显示了单元格A1、A2、A3  
如何知道单元格A1中的内容呢？看看图7中的内容：

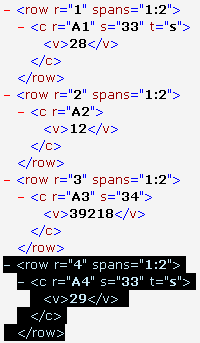
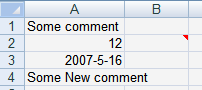
**<c** r="A1" t="s"**>**

，可以知道单元格A1中包含字符串(t=”s”)。下一行：

**<v>**28**</v>**

告诉我们需要在“xl”文件夹中名为sharedStrings.xml的文件里找到项目数值28。该文件顶部的内容如图8所示。  
[RibbonX(5)-8](http://www.flickr.com/photos/35750775@N02/3472057681/)  
图8：文件sharedStrings.xml顶部的内容  
表明在该工作簿中有45个唯一字符串。由于索引值是以0为基数开始，图7中的内容告诉我们需要的索引值是28，因此我们需要找到sharedStrings.xml文件中第29个

**<si>**

条目。如图9所示。  
[](http://www.flickr.com/photos/35750775@N02/3472057685/)  
图9：文件sharedStrings.xml中的第29个si即为工作表Comments中单元格A1的内容  
在单元格中添加文本  
现在，让我们在工作表Comments中添加新的文本数据。首先，修改sharedStrings.xml文件顶部的字符串数，将图8中的代码修改如图10所示。  
[RibbonX(5)-10](http://www.flickr.com/photos/35750775@N02/3472057687/)  
图10：将文件sharedStrings.xml顶部的字符串数修改为46  
接下来，在该文件中添加新字符串。在该文件的相应添加新字符串，如图11所示。  
[](http://www.flickr.com/photos/35750775@N02/3472057689/)  
图11：在文件sharedStrings.xml中添加新字符串  
然后，修改sheet7.xml文件，以添加带有新字符串的行，如图12所示。  
[](http://www.flickr.com/photos/35750775@N02/3472870336/)  
图12：在文件sheet7.xml中添加代码以添加带有新字符串的行  
此时，工作表如图13所示。  
[](http://www.flickr.com/photos/35750775@N02/3472870390/)  
图13：编辑XML文件后的结果  
在单元格中添加数字  
在单元格中添加数字与添加文本相似，但有两点不同：数字是直接保存在工作表xml文件中并且可以添加数字格式。  
例如，在前面的sheet7.xml文件中，数字单元格的格式为“常规”并且没有设置专门的格式。  
通过索引值来定义单元格格式，例如下面的代码表示在单元格A3中显示的是日期：

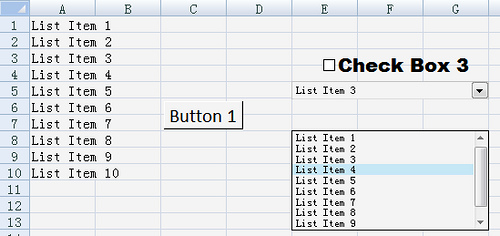
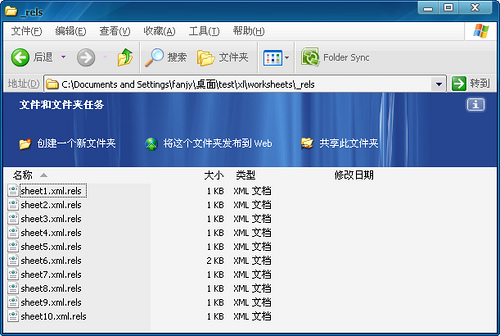
**<row** r="3" spans="1:2"**>**

**<c** r="A3" s="34"**>**

**<v>**39218**</v>**

**</c>**

**</row>**

索引值34指向文件包中的另一个文件：styles.xml，该文件的开始部分是自定义数字格式列表，下面一部分是称作cellXfs的单元格格式交叉引用转到自定义数字格式列表。在工作表xml文件中的s=”34″指向该节点（cellXfs节点）的某一元素。由于索引值基于0，因此索引值34代表列表中的第35项（即最后一项），如图14所示。  
[](http://www.flickr.com/photos/35750775@N02/3472870422/)  
图14：索引值34在列表中的项目  
单元格格式交叉引用依次引用numFormatId列表、FontId列表、fillId列表以及borderId列表，都包含在Styles.xml文件中。  
因此，不能直接添加格式化单元格到工作表，必须检查特定的单元格格式组合是否已经存在于单元格格式交叉引用中（在styles.xml中），这涉及到要为各个元素检查所有单个的格式引用。同时，如果格式组合是新的，则必须扩展带有新格式的格式元素列表，并更新样式交叉引用。当然，也需要更新列表的所有“count”参数。  
上文演示的示例工作簿下载：  
  
改变窗体控件的属性  
通常，窗体控件与控件工具箱中的控件不同，例如不能改变按钮的背景色、不能改变组合框的字体和颜色等。下面来探索如何利用工作簿文件包更改这些属性。  
例如，工作簿FormControls.xlsm的工作表Sheet1如图15所示。  
[](http://www.flickr.com/photos/35750775@N02/3472870444/)  
图15：包含一些窗体控件的演示工作簿  
现在，来分析这些窗体控件存放的位置。将工作簿FormControls.xlsm重命名为FormControls.xlsm.zip。双击该文件，然后打开文件夹“xl”，接着打开文件夹“worksheets”。打开“sheet1.xml”文件，可以找到如图16中的条目。  
[RibbonX(5)-16](http://www.flickr.com/photos/35750775@N02/3472062767/)  
图16：文件sheet1.xml中指定窗体控件的条目  
要找到“rId2”，打开“\_rels”文件夹，如图17所示。  
[](http://www.flickr.com/photos/35750775@N02/3472062793/)  
图17：打开文件夹“\_rels”  
打开“sheet1.xml.rels”文件，找到rId2，如图18所示。  
[RibbonX(5)-18](http://www.flickr.com/photos/35750775@N02/3472870544/)  
图18：文件“sheet1.xml.rels”中Id为rId2的条目  
这告诉我们应该查找文件“..\xl\drawings\vmlDrawing1.vml”。  
每个工作表中有一个vmlDrawing?.vml文件，包含着控件或者在该vml文件中捕捉到的任何其它对象。  
现在，将vmlDrawing1.vml文件拖到桌面上，然后打开它。找到定义复选框的部分，如图19所示。  
[](http://www.flickr.com/photos/35750775@N02/3472062849/)  
图19：文件vmlDrawing1.vml中定义Excel工作表中复选框的部分  
通常，我们不能改变窗体控件中复选框的字体。现在，让我们在vmlDrawing1.vml文件中改变其字体，将图19中的内容：

**<span** style="color: #a00000; font-family: Arial Black;"**>**CheckBox 3**</span>**

修改为：

**<span** style="color: #a00000; font-family: Courier New;"**>**CheckBox 3**</span>**

保存并关闭该vml文件，然后将其拖回压缩文件夹的drawings文件夹中，再将文件名称FormControls.xlsm.zip改为FormControls.xlsm。现在，打开工作簿FormControls.xlsm后，复选框如图20所示。  
[RibbonX(5)-20](http://www.flickr.com/photos/35750775@N02/3472870578/)  
图20：修改字体后的窗体复选框控件  
在这个vml文件中，还可以看出控件的其它信息，例如控件的大小和位置，即Anchor。如图19中的内容：

**<x:Anchor>**4, 30, 2, 0, 8, 39, 3, 18**</x:Anchor>**

其中，前2个数字指定控件左上角的水平位置：第1个数字是所在列数（基于0），第2个数字代表控件基于该列左侧的偏移量；随后的2个数字指定控件左上角的垂直位置：第1个数字是所在行号（基于0），第2个数字代表控件基于该行顶部的偏移量；最后的4个数字指定控件的右下角的位置。  
也就是说，在上面的xml中，控件的左上角位于列E、第3行、向右移29像素，右下角位于单元格I4，向右移38像素，相对于左上角下移18像素。  
也可以改变复选框字体的颜色，将内容：

**<span** style="color: #a00000; font-family: Courier New;"**>**CheckBox 3**</span>**

修改为：

**<span** style="color: #ff0066; font-family: Courier New;"**>**CheckBox 3**</span>**

修改后的效果如图21所示。  
[RibbonX(5)-21](http://www.flickr.com/photos/35750775@N02/3472870594/)  
图21：修改窗体复选框控件中字体的颜色  
上文演示的示例工作簿下载：  
  
通过上面的介绍后，大家可以寻找并更改相关的xml文件内容，以实现在工作表界面中无法实现的操作。

**结语**XML是一种可扩展标记语言，可显示任何平台或应用程序可读的数据，并且XML文件是一种简单的文本文件，可以使用记事本创建，任何程序都可以很容易地从中读取数据，而且这种文件格式也不易损坏。而Microsoft Office引入的新的开放的XML文件格式，就具备了这些优点。它使得我们可以使用其他应用程序软件来读取或编辑Office文档，例如无须打开Excel就能修改工作簿、插入新工作表等。此外，与旧的二进制格式文件相比，新的文件格式也减少了工作簿损坏的机率，并且能够根据文件包内的相关文件来修复文件，例如删除或破坏了XML文件的一些字节后，在Excel中仍然可以读取数据的其余部分。使用新文件格式的文档数据实际上是文本文件，这使得文件大小也比相应的二进制文档要小。同时，文档组织也是结构化的，因此可以提取其中单独的元素。  
还有很重要的一点是，新的XML文件格式使得可以通过自定义功能区来重新调整用户界面，实现新增的功能。

[**细品RibbonX(6)：初识RibbonX**](http://www.excelperfect.com/index.php/2009/04/28/ribbonxstudy6/)

2009年04月28日, 7:03 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**2** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

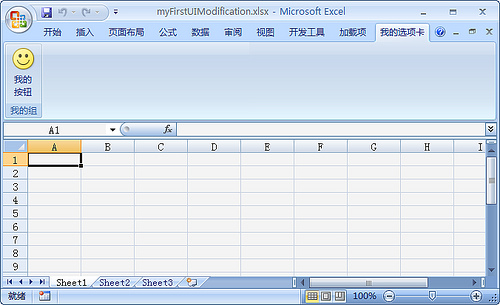
当初次看到Office 2007时，您会立即注意到界面中没有了菜单和工具栏，这是Office2007最大的改变之一，使用功能区替换了菜单和工具栏。您可能意识到自定义用户界面也变得更加困难，其实设置功能区并不会像您想像的那样困难，因为Microsoft也意识到必须有一种方式供开发者或最终用户定制功能区，所以设计了RibbonX，即功能区的可编程机制。  
事实上，正如前面几节所述，功能区具有很多老式界面无法比拟的优势。下面，让我们先简单地认识什么是RibbonX，以及看看如何自定义Office 2007功能区。

**RibbonX是什么**  
在早期版本的Office中，CommandBars对象模型能使开发者在他们的自定义Office应用程序中创建丰富的解决方案。Microsoft在充分了解传统用户界面定制的优缺点后，在Office 2007用户界面扩展中引入了一种创新的模型，可以增强用户体验。新的用户界面定制使用可扩展的标识语言（XML）和几种传统的编程语言之一来操纵组成用户界面的组件。  
功能区（Ribbon）是Office 2007应用程序界面的一部分，而RibbonX是指为自定义的应用程序来创建功能区元素（组件）的编程接口。通过使用RibbonX编程接口，定义如何在应用程序中显示功能区。  
RibbonX使用的XML是简单易懂的，并不需要深入学习和理解XML。RibbonX也依赖至少一个命名空间，而URL定义了指定的RibbonX条目如何工作的位置。通常，您并不需要知道关于命名空间的太多信息，只需要将其包含在RibbonX代码的适当位置。  
在设计自定义功能区时，使用XML定义功能区并在文件格式中存储为自定义部分。当打开文档时，自动读取XML部分并将定制应用到功能区中。无论何时关闭文档，总会自动删除其功能区定制，因此除了自动加载的加载项外，对特定文档的功能区定制不会影响其他文档。  
功能区（Ribbon）的每个元素在RibbonX编程接口中都有相应的元素，用户只能看到由代码修改后的结果。与以前版本的Office不同，RibbonX不依赖于一组层次对象来控制界面。相反，界面取决于描述不同元素的XML文件，该文件遵守下面的XML层次：

* 选项卡（Tabs）
* 组（Groups）
* 控件（Controls）

RibbonX的层次是相当绝对的。一个选项卡不能包含另外的选项卡，仅能包含组。同样，控件能够放置在组中，而不能直接放置在选项卡中。本系列文章将会详细介绍如何创建良好的功能区界面，本文只是对这方面的内容作简单介绍，让大家能够初步认识RibbonX。

**Office 2007 UI定制模型概览**  
下图为Office 2007 UI定制模型示意：  
[](http://www.flickr.com/photos/35750775@N02/3483069952/)  
RibbonX代码  
RibbonX代码是XML代码，遵守Microsoft提供的XML架构。根据维基百科的解释，“XML是描述数据的一种方式，一个XML文件也可以包含数据”。要定制Office Ribbon UI，需要使用XML描述每件事情，所有关于尺寸、位置、可见性、标签、ID等等，习惯于在代码中指定这些操作的开发者应在RibbonX XML格式中指定。  
Ribbon应用程序  
所选择的Ribbon应用程序（Access、Excel、Outlook、PowerPoint或Word）将读取RibbonX代码并应用到自已的UI中。RibbonX代码通过文档或者模板，或者COM加载项来提供。嵌入RibbonX代码到Office 2007文件格式（Open XML）文档或模板中，是供用户或VBA开发者所选择的方法。  
对使用这种新模型的开发者来说，仅仅有一次机会提供RibbonX代码。这意味着，一个加载项可以装载Ribbon定制一次，通常是在启动时。在加载项执行期间修改Ribbon仅在一种有限的方式内是可能的。即在大多数静态情况下，开发者可以指定UI。当卸载加载项时，或者带有RibbonX代码的文档/模板关闭时，通过RibbonX提供的UI修改也将删除。这与以前的Office版本中的CommandBars模型相反，因为在CommandBars模型中，开发者必须确保所有的UI定制被正确地删除。  
加载项  
如果仅仅想要为某文档或模板提供一个定制的Ribbon，那么只需使用Microsoft已经提供的功能。然而，如果您想添加自已的功能，例如，正开发一个加载项，并且要在Ribbon中使用该项功能，那么必须要在VBA或COM加载项中嵌入“回调”。  
在RibbonX代码中，指定这些回调。回调是加载项代码里的函数，Office在某些条件下调用这些函数。回调与事件有些类似，但不相同。例如，可以指定一个回调，当想知道特定的控件是否可见时通过Office来调用。或者，更像事件，当用户单击添加到Ribbon中的自定义按钮时，与这项操作相关的回调将被执行。也可以指定一个回调，使Office获取控件的标签。

**认识RibbonX的关键元素**选项卡（Tabs）  
在功能区层次中，选项卡是最上层的元素，如下图1所示，有一个名为“我的选项卡”的自定义选项卡，它包含一个名为“我的组”的组，其中包含一个名为“我的按钮”的按钮。  
[](http://www.flickr.com/photos/35750775@N02/3483069916/)  
图1：一个简单的自定义选项卡  
通常，创建自定义选项卡用来集中完成某项特定的任务。创建图1所示的自定义选项卡的XML文件如下：

**<?xml** version="1.0" encoding="UTF-8" standalone="yes" **?>**

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon>**

**<tabs>**

**<tab** id="myTab" label="我的选项卡"**>**

**<group** id="myGroup" label="我的组"**>**

**<button** id="myButton"

label="我的按钮"

imageMso="HappyFace"

size="large"

onAction="myButton\_ClickHandler" **/>**

**</group>**

**</tab>**

**</tabs>**

**</ribbon>**

**</customUI>**

这是一个具有所有通用功能的标准的XML文件，包含处理指令、根元素（customUI）、子元素（ribbon）如以及属性。并且，必须为自定义界面定义命名空间，这里，命名空间存在于http:// schemas.microsoft.com/office/2006/01/customui，这是为每个所创建的自定义界面元素使用的位置。  
创建一个新选项卡所需要做的只是编写带有id属性和label属性的元素，使用id属性从应用程序访问该选项卡，而label属性则是用户看到的选项卡的文本。  
组（Groups）  
组将相似的控件集中在一起，使用户不需要花太多的时间查找正确的控件。使用组减少了用户的困惑，并更容易地向用户展示如何执行一项特定的任务。例如，在Excel中想要设置字体时，有一个“字体”组，包含设置字体类型、大小、加粗、背景、边框等，而不需要逐个查找每个命令。  
在上图1中，有一个“我的组”，仅包含有一个控件。在XML中通过使用元素创建组，必须包括id属性，以便在代码中标识该组，以及label属性，用来提供用户在界面中所看到的识别该组的文本。  
控件（Controls）  
一个控件执行一项特定的任务。例如，可以使用按钮控件帮助用户执行某任务，可以使用复选框让用户作出选择，也可以使用下拉列表框让用户在多个选项中选择而不是简单地选择“是/否”。  
控件有多个大小。大尺寸的控件吸引用户注意，执行主要的任务，而小尺寸的控件则帮助用户执行不常用的任务。  
控件总是需要比其它功能区元素更多的代码，因为它们不是静态的，它们执行某项任务。在上文所示的XML清单中，简单的button元素需要多个参数，除了如前所述的元素一样，必须提供id属性和label属性。当按钮包括图像时，图像也必须出现在引用它的文件里，而size属性定义了按钮的大小。绝大多数按钮也包括onAction属性，以执行与按钮相关的代码。当用户单击按钮时，触发onAction事件。

[**细品RibbonX(7)：有效的RibbonX设计**](http://www.excelperfect.com/index.php/2009/05/02/ribbonxstudy7/)

2009年05月2日, 9:29 上午

1 颗星2 颗星3 颗星4 颗星5 颗星(**3** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

Microsoft开发功能区作为Office应用程序的主要界面，其首要目的是使用户更容易使用Office。我们自定义功能区也一样，设计的界面必须考虑用户如何实现特定的任务，必须使操作更有效，更容易完成工作，并且满足用户的所有需求。

**开发RibbonX元素的目标**  
当处理RibbonX开发时，总体目标是创建面向任务的应用程序。按下面的内容来逐步达到这一目标。

* 将任务分成若干步，每一步代表用户一次执行的一项操作。
* 观察用户执行任务的实际过程。
* 将过程拆分为单个的步骤，每一个步骤需要且最多需要一个控件来完成。
* 确定用户可能执行某特定步骤的次数。
* 创建用户需要执行某项任务的控件列表。
* 确定是否适合将控件放置在已有的选项卡里。无论何时尽可能考虑使用上下文选项卡集，以便于仅当用户需要它们时才出现。如果没有找到适用于控件的现有的选项卡，那么定义一个新选项卡来包含它们。决不要为包含多个控件的某项任务使用多个选项卡。
* 确定是否任一现有的组中能够包含控件。如果没有，则定义一系列面向任务的组来包含控件。
* 按用途排列组。用户的注意力应该从屏幕的左边向右移。
* 将先前已定义的控件放置到它们相应的组中。
* 如果每次都要使用某特定的控件，那么该控件应该比很少需要使用的控件有更大的外观，并且放置在更好的位置。
* 确定是否需要包括对话框以显示更高级的或可供替换的选项。此时，可以在组的右下角添加一个称作对话框启动器的控件。

对某一特定的应用程序，可能还会有其它目的。例如，可能需要对高级用户设置一组步骤，而对标准用户设置另一组步骤。此外，高级用户访问的某些功能，不允许标准用户使用。

**考虑RibbonX元素的易访问性和明显性**当创建RibbonX应用程序时，要达到的最重要的目标之一是功能的易访问性和明显性，集中于任务而不是功能。通常，将某项功能放置在合适的位置，并提供可视化的用法提示。  
使用工具提示（tooltip）  
在功能区中，将鼠标悬浮在某功能控件的图标上时，将看到标准的工具提示。例如，将鼠标悬浮在“粘贴”控件上时，会出现提示“粘贴（Ctrl+V） 粘贴剪贴板上的内容”。因为大多数人都理解“粘贴”控件的功能，所以使用了标准的工具提示。然而，即使是标准的工具提示，所包含的信息内容也比以前版本的提示要多。  
此外，功能区也支持超级工具提示（super-tooltip）。此时，有多行显示帮助文本，为用户提供了关于此功能更详细的操作信息。并且在底部还包含帮助图标指示，当显示超级工具提示时按F1键，将显示关于此主题的上下文帮助信息。  
还有一种工具提示为，当用户将鼠标指针悬浮在对话框启动器上时，在提示信息中会显示小的对话框图像。此外，还可以在特定的区域看到这种工具提示，例如将鼠标指针悬浮在Office按钮上时。  
使用已有的Office功能  
有时，您可能想使用某些以前版本的Office中残留的功能。例如，从编程角度来说，任务窗格与以前版本相同，可以在任务窗格中添加已有的应用程序功能，以便在Office 2007和老版本的Office中都显示相同的功能。然而，不能使用VBA实现这种解决方案。  
另外，还可以考虑使用上下文菜单，这依赖于已有的CommandBars对象模型，因此对所有版本的Office均有用。在某对象上右击时，会显示上下文菜单，包含对所选取的对象执行操作任务。因此，可以认为这是克服版本差异的一个较好的解决方案。并且，可以使用VBA实现这种解决方案。  
使用Office菜单  
当必需满足应用程序需求时，也可以更改Office菜单。Office菜单就像过去版本中高级的文件菜单。不应该使用Office菜单来包含应用程序中基于任务的功能。然而，如果应用程序执行某种特定的保存或其他文件相关的任务，那么可以在Office菜单中放置这些功能，而不是将其放置在功能区。  
可以在已有的条目中添加新的选项，例如在“打印”项中添加可供选择的打印方法，或者创建全新的条目。此外，Office菜单是放置非任务相关的项目的最好的地方。  
使用上下文选项卡集（Contextual Tabsets）  
当在Office中执行某项任务时，会出现上下文选项卡集。例如，当在Excel 2007中创建表时，会出现“表工具”上下文选项卡集。注意，上下文选项卡集出现在正常的选项卡上方。在上下文选项卡集下方是与其相关的选项卡。  
在Office 2007中不能创建自已的上下文选项卡集，Microsoft可能已经计划将此功能添加到未来版本中，但可以修改上下文选项卡集中已有的选项卡、隐藏已有的选项卡以及添加自已的自定义选项卡。  
重利用微型工具栏（MiniToolbar）  
微型工具栏提供了一组格式命令，用于处理文档中的文本或其他对象。但是不能在微型工具栏中添加任何新内容，但可以隐藏已有的功能或者重利用其中的元素执行另外的任务，这意味着应该将微型工具栏考虑作为添加新功能的最后选择。  
重利用（Repurposing）是控制某控件的行为供自已使用。例如，可以决定控制“格式刷（Format Painter）”控件来添加公司要求对文档所必需的自定义格式。事实上，还可以重利用功能区中的控件和Office菜单中的控件。  
除非万不得已，否则不要重利用已有的控件。因为很多用户已熟悉已有控件所执行的操作，当他们发现有些控件与预想的操作不同时，往往会导致操作混乱。但是，当您希望某控件执行您不想让用户执行的操作时，可以考虑重利用该控件。  
无论何时重利用某控件，都应该考虑使用户仍然可以使用该控件原有的功能。例如，可能重利用Office菜单中的打印控件，使用户指向网络上特定的打印机，但应该仍然可以让用户选择原有的打印功能，避免指定的打印机因为打印较大的文档而被长时间占用的情形。

**定义有效的RibbonX设计**  
上面的内容介绍了如何定义Ribbon设计目标，以及在哪里放置定义的不同的应用程序元素。现在，应该考虑Ribbon元素的准确的设计了。例如，可以使用文本、图标或者两者的组合来描述大多数控件的目的。用于表达控件目的的方法取决于应用程序用户以及该控件在应用程序里的目的。  
有效地使用名字  
无论何时创建新的选项卡或组，都必须同时提供名字。名字应该满足下列标准：

* **简短：**名字应该只含有尽量少的词，否则当用户调整应用程序界面大小时会看不到名字。
* **集中于任务：**确保名字反映了希望用户来执行的任务。
* **唯一：**虽然可以创建具有两个相同名字的选项卡，但这会导致用户混淆。
* **简单：**使用尽可能最简单的术语来定义任务。

不是所有的控件都提供有名字，很少使用的控件经常呈现为更小的按钮并且仅仅是一个图标。而经常使用的控件，往往使用更大的控件尺寸和文本以吸引用户对该控件的注意。正如为选项卡和组提供的名字一样，应该使控件的标签简短、唯一且简单，此时，应该确保标签反映了控件的用途，而不是用户执行的任务。  
考虑选项卡中项目的数量  
功能区能够自动调整以显示在用户允许的空间里能够适应的信息数量。在大多数情形下，用户能够看到提供的所有的选项卡和组，只要显示区域不是太小。  
如果放置在某选项卡中的项目比Excel放置在“开始”选项卡中的项目还要多，那么会发生什么情况呢？有效的尺寸大小开始增加，用户不能够使应用程序界面足够大以显示放置在该选项卡中的所有控件。此时，功能区成为冗长的。通常，应该使用应用程序中带有最多缺省控件的选项卡作为自定义变化的基础。对于大多数Office应用程序来说，“开始”选项卡包含最多的控件。  
从用户的角度看待组  
用户的目标是使用应用程序来完成工作，而不想考虑太多计算机方面的事情。当为应用程序创建组时，总是考虑连续的任务——一步接着另一步，使用动词表达功能区任务的意图，或者使用名词描述组完成的任务行为。总是从左向右排列组，就像阅读一行文本一样。  
使用正确的控件  
功能区设计面临的重要问题之一是如何选择正确的控件来与用户交互。与过去的菜单和工具栏设置不同，功能区提供了大量可用的控件，使得开发者只需给用户提供正确的控件。例如，考虑简单的按钮。功能区支持许多按钮类型，包括：

* **按钮：**执行某动作或者显示一个对话框。
* **切换按钮：**打开或关闭某功能。
* **拆分按钮：**基于用户选择，执行几种动作中的某动作。包括可以接受的动作的下拉列表。

实际上，因为可以修改基本的按钮，所以不仅仅只有这三种按钮类型。下面是一些可能的修改示例：

* **大小：**可以使按钮更大或更小，以改变其重要性。
* **顺序：**以特定的顺序放置按钮来改变它们的重要性。以特定的顺序放置的按钮可以使用户考虑相关的功能或特定的过程。
* **自定义图像：**无须使用图标或文本来描述按钮。在大按钮中使用自定义图像能够明显改变其可视性，用户更容易看到它。

可以使用下表中列出的任何图形格式，表中也介绍了使用每类图形格式的优点。

|  |  |
| --- | --- |
| 表：功能区支持的图形格式 | |
| **文件类型** | **优点** |
| BMP | 容易创建这类图形，有许多操作这类图形的工具 |
| JPG | 容易在网页中使用，对图片或其它复杂的图形是有用的 |
| GIF | 容易在网页中使用。这种格式不会遭受数据存储损耗，也为功能区中特殊的效果提供了单一的透明级别 |
| PNG | 容易在网页中使用，用于存储复杂的图形。这种格式也为所有的功能区透明效果提供了完全支持 |

提供用户提示  
功能区提供了通过提示来改善用户体验的功能。某些提示是内置的，例如当用户将鼠标悬浮在控件上方时，功能区将使选择的控件非常清楚。  
某些提示在上文已作过介绍。例如，可以使用工具提示来改善与用户的交流。添加超级工具提示将明显地改善与用户的交流。  
另一种提示不是非常明显。例如，当鼠标悬浮在Excel功能区中的单元格样式组中时，会在单元格中看到不同样式的实际外观，这使得为用户选择正确的样式以满足特定的需要变得更容易。在样式外观的下面，有实际的样式名，使得容易通过名字选择样式。  
有效地使用功能隐藏  
功能隐藏的一个示例是上下文选项卡集的使用。上下文选项卡集是一组选项卡，仅当执行特定的任务时才出现在屏幕中，例如编辑表。当创建使用上下文选项卡集来工作的应用程序功能时，可以让Office隐藏该功能。  
然而，上下文选项卡集不会响应每个功能隐藏需求。例如，可能需要基于组织中用户的角色或者Microsoft没有提供在上下文选项卡集中的某项任务来隐藏功能。仍然可以编程来隐藏选项卡、组或单个控件，也可以根据模板或特定的文档来隐藏功能。全局功能可以显示为加载项，以确保用户总是能找到这些功能。  
如果打算基于某条件在应用程序中隐藏功能，那么应该将控件放置在单独的组中或者单独的选项卡中，并且只要可能就立刻隐藏。如果用户在使用应用程序时偶尔需要某功能，那么不要在没有使用时就显示该功能，然后再将其隐藏。

**理解XML连接关系**  
RibbonX依赖于选项卡、组和控件的层次结构描述来修改功能区的外观。Microsoft没有为Office 2007提供图形化的设计工具，因此必须手工编写实际的XML。有时，手工编写代码可能导致潜在的错误，因此彻底的测试和修改是重要的。应该绘制所期望的最终结果的功能区，以减少潜在的错误。  
随着添加的新功能的增加，XML里属性的数量也增加，XML将会变得相当复杂。例如，下面的XML描述了一个包括图像的按钮：

**<button** id="MyButton"

imageMSO="ButtonImage"

size="Large"

label="我的按钮"

onAction="ClickMe" **/>**

imageMSO属性定义了在按钮中使用的图像，可以将图像存储为所创建的加载项、模板或文档的一部分，因为RibbonX不会在硬盘中查找图像。如果决定装载图像，那么必须创建特定的代码。当处理图形时，特别是自定义的图像时，需要使用上表列出的图像格式。  
也可以使用内置图像。如果希望控件使用下划线图标，那么只需在描述该控件的XML中添加imageMSO=”Underline”。关于Office 2007控件识别符的完整列表，可以在下一篇文章中找到。  
在XML文件中放置条目的顺序是重要的，应用程序将按照指定的顺序显示选项卡、组和控件。因此，如果希望“按钮1”控件显示在第一位，必须在XML文件中首先放置它。然而，如果不细心的话，将很容易得到不合适的、奇怪样式的布局。下面的XML就是这样的一个例子。

**<tab** id="myTab" label="我的选项卡"**>**

**<group** id="Group1" label="第一组"**>**

**<button** id="Button1"

label="大按钮1"

imageMso="HappyFace"

size="large"

onAction="myButton\_ClickHandler" **/>**

**<button** id="Button2"

label="按钮1"

imageMso="Bold"

onAction="myButton\_ClickHandler" **/>**

**<button** id="Button3"

label="大按钮2"

imageMso="HappyFace"

size="large"

onAction="myButton\_ClickHandler" **/>**

**<button** id="Button4"

label="按钮2"

imageMso="Italic"

onAction="myButton\_ClickHandler" **/>**

**<button** id="Button5"

label="按钮3"

imageMso="Underline"

onAction="myButton\_ClickHandler" **/>**

**</group>**

**<group** id="Group2" label="第二组"**>**

**<button** id="Button6"

label="按钮1"

imageMso="Bold"

onAction="myButton\_ClickHandler" **/>**

**<button** id="Button7"

label="按钮2"

imageMso="Italic"

onAction="myButton\_ClickHandler" **/>**

**<button** id="Button8"

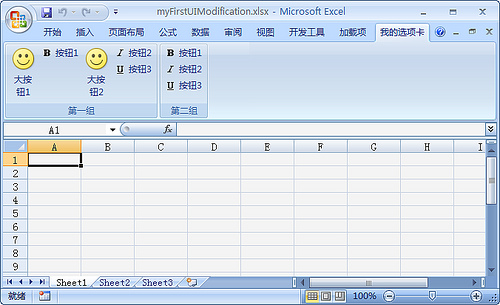
label="按钮3"

imageMso="Underline"

onAction="myButton\_ClickHandler" **/>**

**</group>**

**</tab>**

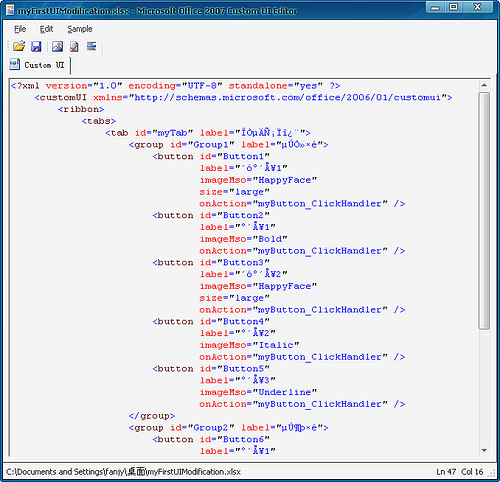
上面的XML中，包含有大按钮和正常的按钮。然而，因为大按钮和正常按钮出现的顺序，所以在第一组中的布局很不理想。下图显示了该XML在Excel 2007中所显示的布局，这都是由于以错误的顺序放置按钮所致。  
[](http://www.flickr.com/photos/35750775@N02/3493005628/)  
图：以错误的顺序放置按钮而出现的无效的布局  
注意到布局中大的间隙，实际上一个单独的大按钮占用了三个正常按钮的空间，如上图中的“第二组”。对于“第一组”来说，更好的布局就是将三个正常按钮放置在一起，让两个大按钮吸引用户的注意，并且该组也有效地使用了功能区空间。由于功能区提供了很多灵活性，因此不可能说哪个布局是最好的。

UI[**细品RibbonX(8)：自定义Ribbon的“武器库”**](http://www.excelperfect.com/index.php/2009/05/06/ribbonxstudy8/)

2009年05月6日, 1:37 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

“工欲善其事，必先利其器。”这里，我们先介绍自定义Ribbon的一些工具和资源，以方便后续的RibbonX开发。  
**自定义Ribbon的工具**  
可以使用记事本来自定义Ribbon，也可以使用第三方工具更方便地自定义Ribbon。  
记事本  
Office 2007采用了Open XML文件格式作为新标准，而新的文件实际上是包含着XML文件的压缩文件。XML文件实际上就是文本文件，因此能够使用记事本方便地打开这些文件并修改，从而自定义Ribbon。  
显然，使用记事本作为自定义Ribbon的工具不是太方便，需要做很多工作。并且，由于没有提供代码验证，编写XML时很容易发生输入错误。  
Microsoft Office 2007 Custom UI Editor  
Microsoft Office 2007 Custom UI Editor（简称“Custom UI Editor”），是一款用来编辑Open XML文件的小工具，方便实用。此外，该工具还提供了代码验证和其它工具来辅助开发。您可以在下面的站点免费获得该工具：  
<http://openxmldeveloper.org/articles/customuieditor.aspx>  
  
在安装Custom UI Editor之前，确保系统中已经安装了Microsoft .Net Framework 2.0及以上版本。下载并解压后，双击OfficeCustomUIEditorSetup.msi即可安装该工具。  
在Custom UI Editor中打开对功能区进行定制过的文件，例如上篇文章中的示例文件，如下图1所示。  
[](http://www.flickr.com/photos/35750775@N02/3505986933/)  
图1：在Custom UI Editor中打开自定义功能区的Excel文件  
打开Custom UI编辑器，您将会看到在工具栏上有5个按钮，如上图1所示。  
Open（打开）：打开想要编辑以改变其功能区的Excel文件。  
Save（保存）：保存xml到已在UI编辑器中打开的文件中（注意当该文件已在Excel中打开时不可能在编辑器中保存该文件）  
Insert Icons（插入图标)）：在文件中插入自已的图标（图标将会保存在文件中）  
Validate（验证）：验证xml以便确保没有语法问题  
Generate Callbacks（产生回调）：为xml中的每个onAction创建宏（回调）。选择所有的回调并将它们复制到工作簿里的标准模块中，然后可以在宏（回调）中的Sub和End Sub行之间插入相应的代码。  
细心的读者可能会发现，在图中有很多乱码，那是因为在自定义功能区时使用了中文，而该编辑器目前版本还不支持中文。  
与我们在记事本中看到的文本文件相比，在Custom UI Editor中的文件使用了颜色来区分不同的标记文本，并且对代码自动缩排，这些都能很好地帮助阅读和理解代码。同时，该编辑器还能容易地嵌入图片、验证代码、存储通用的代码片断，甚至生成响应回调（Callbacks）的代码框架。（回调，即单击定制的功能区控件时触发的自定义VBA程序。我们将在随后的文章中详细介绍回调）  
XML Notepad 2007  
XML Notepad 2007是另一款用于编辑或编写XML代码的工具，可以从Microsoft的站点免费下载该工具。XML Notepad能够捕捉验证代码的XML架构。  
从下面的站点下载并安装XML Notepad 2007：  
<http://www.microsoft.com/downloads/details.aspx?familyid=72d6aa49-787d-4118-ba5f-4f30fe913628&displaylang=en>  
  
接着，从下面的站点下载并提取Office 2007 XML架构：  
<http://www.microsoft.com/downloads/details.aspx?familyid=15805380-F2C0-4B80-9AD1-2CB0C300AEF9&displaylang=en>  
  
最后，将架构链接到XML Notepad。打开XML Notepad，选择“View-Schemas”，在“XML Schemas”中选择“File-Add schemas”，然后找到要提取的XML架构文件夹，即C:\2007 Office System Developer Resources\Office2007XMLSchema\CustomUI.xsd。如下图2所示，单击“OK”。  
[](http://www.flickr.com/photos/35750775@N02/3505986937/)  
图2：添加XML架构到XML Notepad  
**生成自定义功能区选项卡的XML代码的工作簿**  
[Ron de Bruin](http://www.rondebruin.nl/tips.htm)制作了一个工作簿，可以用来生成在功能区中自定义选项卡的XML代码。单击下面的链接下载此工作簿：  
  
**Excel 2007内置控件、组和选项卡列表**

* 在Microsoft网站上提供有Office所有内置控件名称（idMso属性）列表，其下载地址如下：  
  <http://www.microsoft.com/downloads/details.aspx?familyid=4329d9e9-4d11-46a5-898d-23e4f331e9ae>  
    
  除了XML外，在使用VBA操作功能区控件时，可以使用控件名称。
* 下面的工作簿Built-in\_Controls\_Excel\_2007.xlsm由[Ron de Bruin](http://www.rondebruin.nl/tips.htm)创建，包含Excel 2007内置控件名称和相应的图标，非常直观：

**Excel 2007内置控件按钮图像名称**

* 在XML中，会使用imageMso属性来使用想要的内置图标。Jim Rech创建了一个显示Office 2007按钮图像的加载项。安装该加载项后，在工作簿中将添加一个名为Control Image的新选项卡。在该选项卡中，可以在文本框中输入一个索引值，在右侧会显示自该索引值后的50个图标。单击某图标，将会复制该图标控件的名称到剪贴板，然后可以将其粘贴到RibbonX中。从下面的地址下载该加载项BtnImages.xlam：
* John Walkenbach的mso image browser工作簿：
* 从Microsoft的网站下载图标库：  
  <http://www.microsoft.com/downloads/details.aspx?familyid=12B99325-93E8-4ED4-8385-74D0F7661318>  
    
  打开该工作簿后，在“开发工具”选项卡中将会添加一个名为“Office Icons”的组，共有9个图标库，单击每个库的下拉按钮后单击一个图标，将会出现带有图标和图标名称的窗体。

**Excel 2007中的内置菜单**  
在Excel 2007中，仍然可以创建和使用CommandBar对象，但与以前版本相比，其支持的属性和方法已大为减少。例如，没有控制工具栏大小和位置的属性，没有浮动工具栏。

* Excel 2007中有65个快捷菜单，下面的工作簿显示了所有快捷菜单的索引值和名称：
*   
  下面的工作簿显示了所有快捷菜单中第一层级的控件，包括快捷菜单的索引值、名称，控件的ID、Caption、Type、Enabled和Visible属性的值。

**Excel 2007中的内置图像ID**

* 可以使用内置的图像确定快捷菜单中的图标，当然也可以用于功能区中。下面的工作簿显示了所有内置图像的FaceID属性值。
* 下面的工作簿的第一个工作表列出了Excel 2007中所有第一层级的控件及相应名称、ID、FaceId，第二个工作表列出了所有层级控件及相应名称、ID、FaceId，第三个工作表列出了所有图标FaceID，第四个工作表列出了所有弹出式菜单及控件的名称、FaceId、ID。

**Excel 2003中菜单和工具栏名称及ID编号**

* 下面的工作簿列出了Excel 2003中菜单和工具栏的名称以及相应的ID编号：
* 下面的工作簿列出了Excel 2003中所有命令栏清单：
* 下面的工作簿按索引顺序列出了Excel 2003中所有的命令栏：

**Excel97-2003中的命令栏按钮图标及其相应的ID号**

* 下面的工作簿列出了Excel 2003及以前版本中所有命令栏按钮的图标及其ID编号：

**列出Excel 2003命令栏清单的实用加载宏**

* 由erlandsendata设计的一个非常实用的一个加载宏。安装后，将在Excel菜单栏中添加一个名为EDC Tools的新菜单。可以列出当前工作簿中所有的命令栏、列出所有命令栏控件,命令栏按钮图标及相应的ID等。
* 下面的站点包含有列出97-2003版的Word、Excel和PowerPoint图标ID的加载宏：  
  <http://skp.mvps.org/faceid.htm>  
  例如，下面的加载宏列出了Excel 97-2003图标及其ID：  
    
  关于图标的COM加载项：

**VSTO**

* VSTO 2005下载：  
  <http://www.microsoft.com/downloads/details.aspx?familyid=5e86cab3-6fd6-4955-b979-e1676db6b3cb>
* VSTO 2005 SE下载：  
  <http://www.microsoft.com/downloads/details.aspx?familyid=F5539A90-DC41-4792-8EF8-F4DE62FF1E81>  
    
  注意，如果Visual Studio 2005安装的是中文版，那么语言应选择简体中文。
* 如果未安装Office主互操作程序集，则需要下载安装：  
  Office 2003：  
    
  Office 2007：

**免费图标编辑软件**

* 可以在下面的站点下载免费图标编辑软件Paint.NET：  
  <http://www.getpaint.net/download.html>

**RibbonX相关的网站资源**RibbonX论坛：



* <http://pschmid.net/office2007/forums/viewforum.php?f=2>
* <http://www.vbaexpress.com/forum/forumdisplay.php?f=96>
* <http://www.microsoft.com/office/community/en-us/default.mspx>
* <http://www.microsoft.com/office/community/en-us/default.mspx?dg=microsoft.public.office.xml&lang=en&cr=US>
* <http://forums.microsoft.com/MSDN/default.aspx>
* <http://help.lockergnome.com/office/>
* <http://www.mcse.ms/>
* <http://openxmldeveloper.org/forums/12/showforum.aspx>
* <http://openxmldeveloper.org/forums/default.aspx?GroupID=9>

RibbonX官方站点：

* <http://blogs.msdn.com/jensenh/>
* <http://msdn.microsoft.com/en-us/office/aa905530.aspx>
* <http://msdn.microsoft.com/en-us/library/bb187362.aspx>
* <http://msdn.microsoft.com/en-us/library/aa432025.aspx>
* <http://msdn.microsoft.com/en-us/office/aa905359.aspx>
* <http://msdn.microsoft.com/en-us/office/aa905362.aspx>
* <http://msdn.microsoft.com/en-us/library/aa432026.aspx>
* <http://msdn.microsoft.com/en-us/architecture/aa699447.aspx>

RibbonX相关站点：

* <http://blogs.msdn.com/frice>
* <http://blogs.msdn.com/kathleen/>
* <http://blogs.msdn.com/brian_jones/>
* <http://www.excelguru.ca/blog/category/the-ribbon/>
* <http://www.rondebruin.nl/>
* [http://www.google.com/search?q=RibbonX site:blogs.msdn.com](http://www.google.com/search?q=RibbonX%20site:blogs.msdn.com)
* <http://www.devsource.com/>
* <http://www.devx.com/>
* <http://news.softpedia.com/cat/Microsoft/>

免费的RibbonX共享资料：

* 《Excel 2007 VBA参考大全》第14章 RibbonX

您有更好的资源吗？欢迎在此留言推荐。

[**细品RibbonX(9)：层次分明的RibbonX代码结构**](http://www.excelperfect.com/index.php/2009/05/21/ribbonxstudy9/)

2009年05月21日, 1:31 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**2** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

在前面的一些文章中，我们已经介绍过定制功能区的代码。在详细探讨功能区各元素之前，我们将先了解完整的RibbonX代码结构、如何使用各类工具定制功能区、以及至关重要的XML知识。下面，让我们先看看完整的RibbonX代码结构。  
如果忽略定义控件的XML元素，那么完整的RibbonX结构如下：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui" ...**>**

**<commands>**

**<command** ... **/>**

**</commands>**

**<ribbon** ...**>**

**<officeMenu>**

可用于Office菜单中的任何控件类型

**</officeMenu>**

**<qat>**

**<sharedControls>**

**<control>**,**<button>** or **<separator>**控件类型

**</sharedControls>**

**<documentControls>**

**<control>**,**<button>** or **<separator>**控件类型

**</documentControls>**

**</qat>**

**<tabs>**

**<tab** ... **>**

**<group** ... **>**

所有控件类型

**</group>**

**</tab>**

**</tabs>**

**<contextualTabs>**

**<tabSet** idMSO="TabSetChartTools"**>**

**<tab** ... **>**

**<group** ... **>**

所有控件类型

**</group>**

**</tab>**

**</tabSet>**

**</contextualTabs>**

**</ribbon>**

**</customUI>**

其中，省略号表示一个或多个可选的属性。可以看出，RibbonX代码结构是层次分明的。现在，让我们初步了解代码结构中的元素。

**<customUI>**元素是XML的根容器，命名空间将其识别为RibbonX文档。

**<commands>**元素用来重复利用内置控件。

**<ribbon>**元素包含功能区中所有可以利用的元素。可以包含下列元素以控制功能区的相应部分。

**<officeMenu>**元素用来定制Office菜单。

**<sharedControls>**元素代表共享控件。

**<documentControls>**元素代表文档控件。

**<qat>**元素用来定制快速访问工具栏。

**<tabs>**元素表示选项卡的集合。

**<tab>**元素创建选项卡。

**<contextualTabs>**元素创建上下文选项卡。

**<group>**元素用来创建组。

下面，让我们使用Excel示例来看看完整的RibbonX结构能够实现的用户界面。  
禁用命令  
使用command元素能够重载命令或者禁用命令。例如，下面的XML禁用“保存”命令：

**<commands>**

**<command** idMso="FileSave" enabled="false" **/>**

**</commands>**

结果如图1所示：  
[](http://www.flickr.com/photos/35750775@N02/3550886992/)  
图1：禁用“保存”命令，该命令已变灰  
从头开始创建功能区  
将ribbon元素的StartFromScratch属性设置为True，从而去除所有的功能区界面元素，然后从头开始设计功能区。其XML为：

**<ribbon** startFromScratch="true"**>**

在Office按钮菜单中添加自定义项  
可以向Office按钮菜单中添加合适的自定义项。例如，下面的XML在Office按钮中添加按钮和菜单：

**<officeMenu>**

**<button** id="rxOMBtn1" label="My Office Button1" **/>**

**<button** id="rxOMBtn2" label="My Office Button2" **/>**

**<menu** id="MyOfficeMenu" label="My Office Menu"**>**

**<button** id="rxSend1" label="My Menu1" **/>**

**<button** id="rxSend2" label="My Menu2" **/>**

**<button** id="rxSend3" label="My Menu3" **/>**

**<button** id="rxSend4" label="My Menu4" **/>**

**<button** id="rxSend5" label="My Menu5" **/>**

**<button** id="rxSend6" label="My Menu6" **/>**

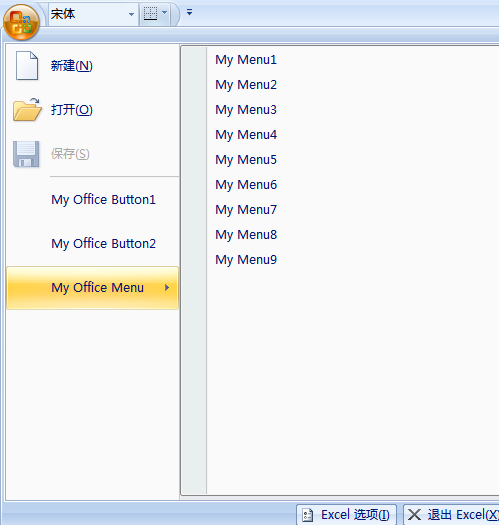
**<button** id="rxSend7" label="My Menu7" **/>**

**<button** id="rxSend8" label="My Menu8" **/>**

**<button** id="rxSend9" label="My Menu9" **/>**

**</menu>**

**</officeMenu>**

结果如图2所示：  
[](http://www.flickr.com/photos/35750775@N02/3550886998/)  
图2：在Office按钮中添加项目  
自定义快速访问工具栏（QAT）  
当将ribbon元素的StartFromScratch属性设置为True后，我们可以在设计时重新自定义快速访问工具栏。例如，下面的XML在QAT中放置“字体”组合框和“边框”拆分按钮：

**<qat>**

**<sharedControls>**

**<control** idMso="Font" **/>**

**</sharedControls>**

**<documentControls>**

**<control** idMso="BordersGallery" **/>**

**</documentControls>**

**</qat>**

其中sharedControls表示放置在其中的控件为共享控件，而documentControls表示放置其中的控件为文档控件（注意，周围带有边框）。结果如图3所示：  
[RibbonX9-3](http://www.flickr.com/photos/35750775@N02/3550887002/)  
图3：自定义快速访问工具栏  
自定义功能区  
下面的XML在功能区中添加一个名为“Menu”的选项卡。

**<tab** id="rxExcelVBA" label="Menu"**>**

接着，使用下面的代码在该选项卡中添加组。

**<group** idMso="GroupInsertChartsExcel" **/>**

上述代码添加内置的“图表”组。结果如下图4所示：  
[](http://www.flickr.com/photos/35750775@N02/3550887006/)  
图4：在自定义选项卡“Menu”中添加“图表”组

**<group** id="rxAuditMisc" label="My Sample"**>**

**<box** id="rxSortBox"**>**

**<control** idMso="SortAscendingExcel" showLabel="false" **/>**

**<control** idMso="SortDescendingExcel" showLabel="false" **/>**

**<control** idMso="SortDialog" showLabel="false" **/>**

**</box>**

**<control** idMso="Copy" **/>**

**<control** idMso="PasteMenu" **/>**

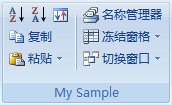
**<separator** id="rxAuditMiscSeparator1" **/>**

**<control** idMso="NameManager" **/>**

**<control** idMso="ViewFreezePanesGallery" **/>**

**<control** idMso="WindowSwitchWindowsMenuExcel" **/>**

**</group>**

上述代码添加一个名为“My Sample”的自定义组，box元素用来组织控件的排列，separator元素用来放置分隔条。结果如图5所示：  
[](http://www.flickr.com/photos/35750775@N02/3550887022/)  
图5：在自定义选项卡“Menu”中添加“My Sample”组，包含一些内置的功能

**<group** id="rxDemo" label="My Menu"**>**

**<menu** id="rxMenu" label="Menu" imageMso="ReviewCompareMenu" size="large"**>**

**<menu** id="subMenu1" label="SubMenu1"**>**

**<menu** id="subMenu11" label="3-SubMenu1"**>**

**<button** id="rxMenu11" label="Up" imageMso="OutlineMoveUp" **/>**

**<button** id="rxMenu12" label="GoTo" imageMso="GoTo" **/>**

**<button** id="rxMenu13" label="Down" imageMso="OutlineMoveDown" **/>**

**</menu>**

**</menu>**

**<menu** id="subMenu2" label="SubMenu2"**>**

**<menu** id="subMenu21" label="3-SubMenu2"**>**

**<button** id="rxMenu21" label="Up" imageMso="OutlineMoveUp" **/>**

**<button** id="rxMenu22" label="GoTo" imageMso="GoTo" **/>**

**<button** id="rxMenu23" label="Down" imageMso="OutlineMoveDown" **/>**

**</menu>**

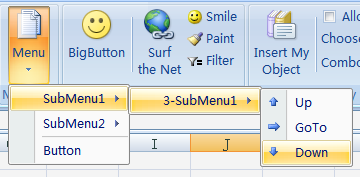
**</menu>**

**<menuSeparator** id="MySeparator" **/>**

**<button** id="rxMenuButton" label="Button" **/>**

**</menu>**

**</group>**

上述代码添加一个名为“My Menu”的组，其中放置了嵌套的层级菜单。结果如下图6所示：  
[](http://www.flickr.com/photos/35750775@N02/3550887024/)  
图6：在自定义选项卡“Menu”中添加“My Menu”组，包含层级菜单

**<group** id="rxMyGroup" label="My Sample Group1"**>**

**<button** id="Button1" imageMso="HappyFace" size="large" label="BigButton" **/>**

**<separator** id="rxSeparator1" **/>**

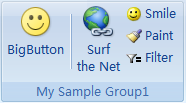
**<button** id="b1" imageMso="HyperlinkInsert" size="large" label="Surf the Net" onAction="surf"**/>**

**<button** id="b2" imageMso="HappyFace" label="Smile" onAction="smile"**/>**

**<button** id="b3" imageMso="FormatPainter" label="Paint" onAction="paint"**/>**

**<button** id="b4" imageMso="AutoFilterClassic" label="Filter" onAction="filter"**/>**

**</group>**

上述代码添加一个名为“My Sample Group1”的组，其中放置了一些带有内置图像的控件并进行了合理布局。结果如下图7所示：  
[](http://www.flickr.com/photos/35750775@N02/3550893868/)  
图7：在自定义选项卡“Menu”中添加的“My Sample Group1”组，包含一些带有内置图像的控件

**<group** id="rxMyGroup1" label="My Sample Group2"**>**

**<toggleButton** imageMso="TranslationToolTip" id="MyToggleButton" size="large" label="Insert My Object"**/>**

**<separator** id="rxSeparator2" **/>**

**<checkBox** id="AllowChanges" label="Allow Changes" **/>**

**<dropDown** id="ChooseDepartment" showLabel="true" label="Choose Department"**>**

**<item** id="Dept1" label="Shipping" **/>**

**<item** id="Dept2" label="Accounting" **/>**

**<item** id="Dept3" label="Engineering" **/>**

**</dropDown>**

**<comboBox** id="ComboBox1" label="ComboBox"**>**

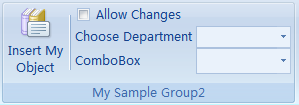
**<item** id="item1" label="one" imageMso="\_1" **/>**

**<item** id="item2" label="two" imageMso="\_2" **/>**

**<item** id="item3" label="three" imageMso="\_3" **/>**

**</comboBox>**

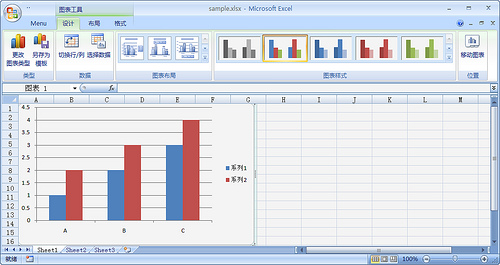
**</group>**

上述代码添加一个名为“My Sample Group2”的组，其中包含切换按钮、复选框、组合框、下拉框，并添加了项目。结果如图8所示：  
[](http://www.flickr.com/photos/35750775@N02/3550087889/)  
图8：在自定义选项卡“Menu”中添加的“My Sample Group2”组，包含切换按钮、复选框、下拉框和组合框，并在其中添加了选项

**<contextualTabs>**

**<tabSet** idMso="TabSetChartTools" **/>**

**</contextualTabs>**

上述代码添加上下文选项卡，当选中工作表中的图表时，会出现“图表工具”选项卡，如图9所示：  
[](http://www.flickr.com/photos/35750775@N02/3550893942/)  
图9：选中图表后出现“图表工具”上下文选项卡  
完整的代码  
上述示例完整的XML代码如下：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<commands>**

**<command** idMso="FileSave" enabled="false" **/>**

**</commands>**

**<ribbon** startFromScratch="true"**>**

**<officeMenu>**

**<button** id="rxOMBtn1" label="My Office Button1" **/>**

**<button** id="rxOMBtn2" label="My Office Button2" **/>**

**<menu** id="MyOfficeMenu" label="My Office Menu"**>**

**<button** id="rxSend1" label="My Menu1" **/>**

**<button** id="rxSend2" label="My Menu2" **/>**

**<button** id="rxSend3" label="My Menu3" **/>**

**<button** id="rxSend4" label="My Menu4" **/>**

**<button** id="rxSend5" label="My Menu5" **/>**

**<button** id="rxSend6" label="My Menu6" **/>**

**<button** id="rxSend7" label="My Menu7" **/>**

**<button** id="rxSend8" label="My Menu8" **/>**

**<button** id="rxSend9" label="My Menu9" **/>**

**</menu>**

**</officeMenu>**

**<qat>**

**<sharedControls>**

**<control** idMso="Font" **/>**

**</sharedControls>**

**<documentControls>**

**<control** idMso="BordersGallery" **/>**

**</documentControls>**

**</qat>**

**<tabs>**

**<tab** id="rxExcelVBA" label="Menu"**>**

**<group** idMso="GroupInsertChartsExcel" **/>**

**<group** id="rxAuditMisc" label="My Sample"**>**

**<box** id="rxSortBox"**>**

**<control** idMso="SortAscendingExcel" showLabel="false" **/>**

**<control** idMso="SortDescendingExcel" showLabel="false" **/>**

**<control** idMso="SortDialog" showLabel="false" **/>**

**</box>**

**<control** idMso="Copy" **/>**

**<control** idMso="PasteMenu" **/>**

**<separator** id="rxAuditMiscSeparator1" **/>**

**<control** idMso="NameManager" **/>**

**<control** idMso="ViewFreezePanesGallery" **/>**

**<control** idMso="WindowSwitchWindowsMenuExcel" **/>**

**</group>**

**<group** id="rxDemo" label="My Menu"**>**

**<menu** id="rxMenu" label="Menu" imageMso="ReviewCompareMenu" size="large"**>**

**<menu** id="subMenu1" label="SubMenu1"**>**

**<menu** id="subMenu11" label="3-SubMenu1"**>**

**<button** id="rxMenu11" label="Up" imageMso="OutlineMoveUp" **/>**

**<button** id="rxMenu12" label="GoTo" imageMso="GoTo" **/>**

**<button** id="rxMenu13" label="Down" imageMso="OutlineMoveDown" **/>**

**</menu>**

**</menu>**

**<menu** id="subMenu2" label="SubMenu2"**>**

**<menu** id="subMenu21" label="3-SubMenu2"**>**

**<button** id="rxMenu21" label="Up" imageMso="OutlineMoveUp" **/>**

**<button** id="rxMenu22" label="GoTo" imageMso="GoTo" **/>**

**<button** id="rxMenu23" label="Down" imageMso="OutlineMoveDown" **/>**

**</menu>**

**</menu>**

**<menuSeparator** id="MySeparator" **/>**

**<button** id="rxMenuButton" label="Button" **/>**

**</menu>**

**</group>**

**<group** id="rxMyGroup" label="My Sample Group1"**>**

**<button** id="Button1" imageMso="HappyFace" size="large" label="BigButton" **/>**

**<separator** id="rxSeparator1" **/>**

**<button** id="b1" imageMso="HyperlinkInsert" size="large" label="Surf the Net" onAction="surf"**/>**

**<button** id="b2" imageMso="HappyFace" label="Smile" onAction="smile"**/>**

**<button** id="b3" imageMso="FormatPainter" label="Paint" onAction="paint"**/>**

**<button** id="b4" imageMso="AutoFilterClassic" label="Filter" onAction="filter"**/>**

**</group>**

**<group** id="rxMyGroup1" label="My Sample Group2"**>**

**<toggleButton** imageMso="TranslationToolTip" id="MyToggleButton" size="large" label="Insert My Object"**/>**

**<separator** id="rxSeparator2" **/>**

**<checkBox** id="AllowChanges" label="Allow Changes" **/>**

**<dropDown** id="ChooseDepartment" showLabel="true" label="Choose Department"**>**

**<item** id="Dept1" label="Shipping" **/>**

**<item** id="Dept2" label="Accounting" **/>**

**<item** id="Dept3" label="Engineering" **/>**

**</dropDown>**

**<comboBox** id="ComboBox1" label="ComboBox"**>**

**<item** id="item1" label="one" imageMso="\_1" **/>**

**<item** id="item2" label="two" imageMso="\_2" **/>**

**<item** id="item3" label="three" imageMso="\_3" **/>**

**</comboBox>**

**</group>**

**</tab>**

**</tabs>**

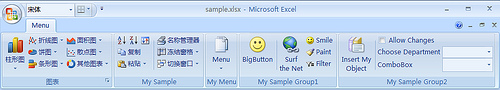
**<contextualTabs>**

**<tabSet** idMso="TabSetChartTools" **/>**

**</contextualTabs>**

**</ribbon>**

**</customUI>**

自定义Excel界面如图10所示：  
[](http://www.flickr.com/photos/35750775@N02/3550893986/)  
图10：自定义的Excel界面  
看起来代码较多，其实并不复杂，如果大家有疑问，先不要着急，有概念就行，后面我们将慢慢详细地讲解。  
示例文档下载：

[**细品RibbonX(10)：使用记事本自定义功能区**](http://www.excelperfect.com/index.php/2009/06/03/ribbonxstudy10/)

2009年06月3日, 8:20 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**3** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

本文主要介绍如何使用记事本自定义Office 2007功能区，仍然以Excel为例。  
下面的示例创建一个非常简单的自定义功能区：在功能区中添加一个新的选项卡，并在该选项卡中放置两个内置的组。  
步骤1 创建自定义功能区的文件，该文件中包含用于修改功能区的XML代码。  
（1）创建一个名为customUI的文件夹。  
（2）在该文件夹中，创建一个名为customUI.xml的文本文件。注意，文件的后缀名为xml，即该文件是一个XML文件。  
（3）打开该文本文件，并输入下面的代码：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon** startFromScratch="false"**>**

**<tabs>**

**<tab** id="rxtabCustom"

label="我的选项卡"

insertBeforeMso="TabHome"**>**

**<group** idMso="GroupFont"**>**

**</group>**

**<group** idMso="GroupZoom"**>**

**</group>**

**</tab>**

**</tabs>**

**</ribbon>**

**</customUI>**

注意，XML代码是区分大小写的，因此，应输入正确的大小写，否则将导致代码失败。  
（4）保存该文件。  
步骤2 创建希望使用自定义功能区的文件  
（5）创建一个新的Excel 2007工作簿，或者打开一个现有的工作簿。这里，通过创建一个新工作簿来介绍。  
（6）在某文件夹中以MyFirstUIModification.xlsx的文件名保存该工作簿，然后退出Excel。  
步骤3 修改Excel压缩包中的文件，以引用自定义功能区的XML文件，从而实现自定义功能区。  
（7）右击MyFirstUIModification.xlsx文件并选择“重命名”，保存完整的文件名，但是在其后面添加.zip扩展名。当出现警告框时，选择“是”。  
现在，该文件修改为zip压缩文件，代替刚才标准的Office文档文件。  
（8）双击该压缩文件，将其打开。  
（9）拖动customUI文件夹并将其放置在压缩文件夹中。  
（10）拖动压缩文件夹中的\_rels文件夹到桌面，编辑.rels文件指定该文件和customUI文件夹之间的联系，从而链接功能区的修改。  
（11）打开\_rels文件夹并使用记事本编辑其.rels文件。在该文件末尾元素之前，插入新的联系语句。在输入语句时，应非常小心地使用标点符号、空格和大小写：

**<Relationship**

Id="customUIRelID"

Type="http://schemas.microsoft.com/office/2006/relationships/ui/extensibility"

Target="customUI/customUI.xml"**/>**

（12）保存该.rels文件，关闭记事本。  
（13）在压缩文件夹中，删除原来的\_rels文件夹。  
（14）将编辑过的\_rels文件夹拖回到压缩文件夹中。  
（15）关闭压缩文件夹。  
（16）在压缩文件中单击右键并选择“重命名”，去掉名称后面的.zip扩展名，恢复为MyFirstUIModification.xlsx文件名。  
（17）打开MyFirstUIModification.xlsx文件。  
此时，如果出现错误消息，请重新检查.rels文件和customUI.xml文件，可能由于代码输入错误特别是大小写错误或拼写错误而导致错误。  
如果一切正常，那么在“开始”选项卡左侧会出现一个名为“我的选项卡”的新选项卡，包含内置的“字体”组和“显示比例”组，即一个内置组的副本，如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/06/ribbonx10-1.jpg)  
示例文档下载：  
[myFirstUIModification.rar](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/06/myFirstUIModification.rar)  
或

[**细品RibbonX(11)：使用Custom UI Editor自定义功能区**](http://www.excelperfect.com/index.php/2009/06/18/ribbonxstudy11/)

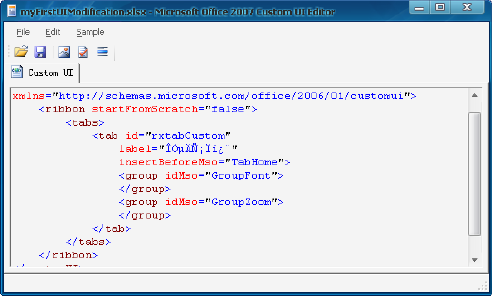
2009年06月18日, 1:24 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**2** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

Custom UI Editor是一款用来编辑OpenXML文件的小型工具，更方便且实用。此外，还提供了一些验证和其它工具来减少开发量。  
在下面的网站中可以免费下载该工具软件：

http://openxmldeveloper.org/articles/customuieditor.aspx

**配置并安装CustomUI Editor**  
步骤1 对于Windows XP用户来说，首先应该安装Microsoft .NET Framework 2.0  
通过“开始”─“控制面板”—“添加或删除程序”，在“添加或删除程序”对话框中检查是否有“Microsoft .NET Framework 2.0”项。如果没有，则可以到Microsoft的网站上下载。  
对于Windows Vista用户来说，系统已安装Microsoft .NET Framework 2.0。  
步骤2 安装Microsoft Office 2007 Custom UI Editor  
下载CustomUI Editor并解压后，双击图标即可安装Microsoft Office 2007 Custom UI Editor。安装完成后，打开Custom UI Editor，然后在该编辑器中打开MyFirstUIModification.xlsx文件，此时如下图1所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/06/ribbonx11-1.png)  
图1：在Custom UI Editor中打开定制后的Excel 2007文件（注：乱码为中文，由于该编辑器不支持中文，因此显示乱码）  
从图1中可以看到，在Custom UI编辑器中文件使用了颜色来区分不同的标记文本，并使用了缩排，版面清新自然。在该编辑器中，颜色能帮助阅读和理解代码；也能容易地嵌入图片、验证代码、存储代码片断，甚至生成响应的回调的代码框架。（回调，即当单击自定义的Ribbon控件时触发的VBA程序，后面将详细介绍。）  
**使用CustomUI Editor来定制功能区**  
下面，将使用CustomUI Editor来定制Excel功能区，使用与使用记事本自定义功能区相同的代码，从而演示CustomUI Editor和记事本处理功能区时的不同。  
步骤1 打开Excel 2007，创建一个新的Excel文档，并将该文档命名为MySecondModification.xlsx。然后，关闭该文档。  
步骤2 打开CustomUI Editor，在该编辑器中打开MySecondUIModification.xlsx文档，然后输入与之前在记事本中定制功能区时相同的代码。

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon** startFromScratch="false"**>**

**<tabs>**

**<tab** id="rxtabCustom"

label="My Tab"

insertBeforeMso="TabHome"**>**

**<group** idMso="GroupFont"**>**

**</group>**

**<group** idMso="GroupZoom"**>**

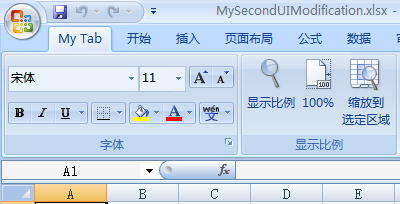
**</group>**

**</tab>**

**</tabs>**

**</ribbon>**

**</customUI>**

步骤3 验证代码  
单击CustomUI编辑器工具栏右侧第二个按钮（Validate）。此时，如果没有错误，则会出现如下图2所示的消息框。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/06/ribbonx11-2.png)  
图2：在CustomUI Editor中验证代码，没有错误  
如果有错误，那么将会指出存在的错误，如下图3所示的消息框。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/06/ribbonx11-3.png)  
图3：在CustomUI Editor中验证代码，有错误  
在大多数情况下，错误消息都指出了错误发生的原因。  
步骤4 一旦验证通过，保存并关闭CustomUI Editor。  
步骤5 打开MySecondModification.xlsx文档，如下图4所示，已在该Excel文档中添加了新的选项卡。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/06/ribbonx11-4.png)  
图4：通过CustomUI Editor定制的Excel界面  
**在CustomUI Editor中存储定制模板**  
有时，您可能想引用以前曾经开发过的自定义界面，而CustomUI Editor提供了这样的功能，将代码存储为模板，方便以后再利用。  
步骤1 打开记事本，输入下面的代码：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon** startFromScratch="false"**>**

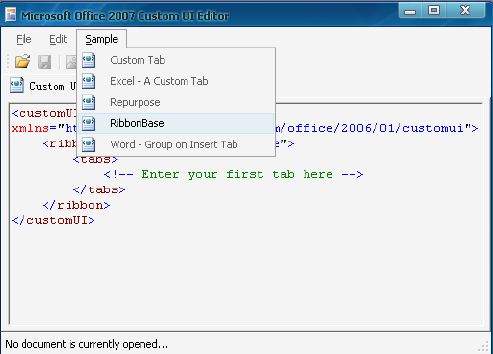
**<tabs>**

*<!-- Enter your first tab here -->*

**</tabs>**

**</ribbon>**

**</customUI>**

步骤2 将该文本文件保存在Program Files\CustomUIEditor\Samples文件夹中，文件名为RibbonBase.xml。  
（注意，Program Files\CustomUIEditor为Microsoft Office 2007 Custom UI Editor缺省的安装目录，如果安装在不同的路径下，只需更改为相应的路径即可）  
步骤3 打开CustomUI Editor，选择菜单“Samples”，在列表中会看到“RibbonBase”条目，如图5所示。单击该条目，将出现模板中的代码，能使用该代码开始新的功能区定制。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/06/ribbonx11-5.png)  
图5：在CustomUI Editor中存储和使用模板  
**关于CustomUI Editor的使用**  
虽然使用CustomUI Editor能够使编辑XML代码更容易，但在使用前还是要对其进一步了解：

* CustomUI Editor不会检查XML标记的形式，因此应确保只使用在XML架构中定义的属性。然而，CustomUI Editor能核查在引号内提供的属性的有效性。
* 在编写和调试RibbonX代码时，不能够同时在应用程序和CustomUI Editor中打开要定制的文件。当文件已经在Office应用程序中打开时，试图在CustomUI Editor中保存该文件将导致错误。此外，即便关闭了正在编辑的Office应用程序文档，然后将其在CustomUI Editor中保存，CustomUI Editor仍将覆盖在应用程序中编辑文档所作的任何修改。在另外的工具中作出修改之前关闭应用程序将是更安全的。
* CustomUI Editor没有查找/替换工具，因此，如果打算在XML代码中作大量的编辑，可以先将其复制到另一个应用程序，编辑后再复制回来。
* 当处理充满屏幕的多行XML文件时，CustomUI Editor习惯刷新屏幕使光标总是在屏幕的最后一行，这使得经常需要移至后面想要编辑的行。因此，与前述方法一样，可以将代码复制到另一个应用程序中，编辑后再复制回来。
* CustomUI Editor不支持中文，因此，如果想定制中文界面，必须在记事本中编写代码并将其以utf-8格式保存。含有中文的代码在CustomUI Editor中显示为乱码。

[**细品RibbonX(12)：使用XML Notepad自定义功能区**](http://www.excelperfect.com/index.php/2009/06/24/ribbonxstudy12/)

2009年06月24日, 1:13 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

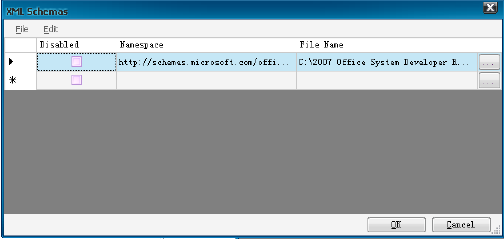
Loading ... Loading ...

XML Notepad是另一款用来编辑或编写XML代码的工具，可以从Microsoft的站点免费下载。XML Notepad能够捕捉XML架构，从而验证代码。  
**安装XML Notepad**  
步骤1 从下列站点下载并安装XML Notepad

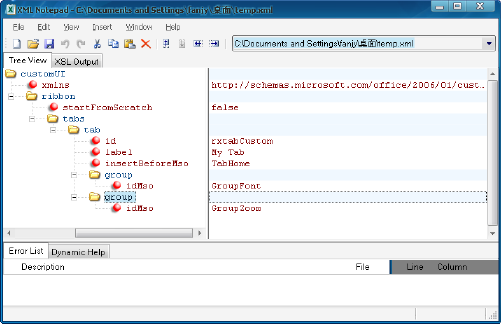
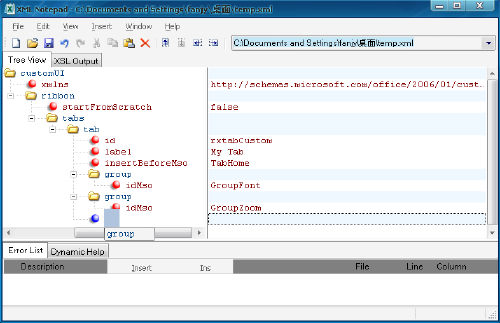
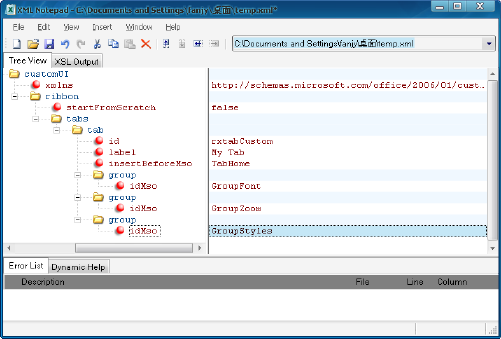
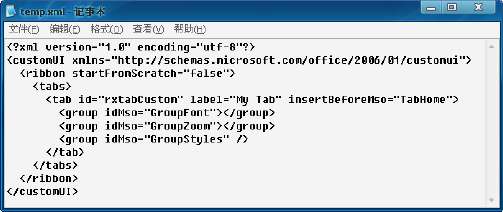
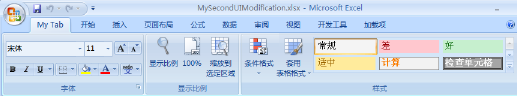
http://www.microsoft.com/downloads/details.aspx?familyid=72d6aa49-787d-4118-ba5f-4f30fe913628&displaylang=en

步骤2 从下列站点下载并提取Office 2007 XML架构

http://www.microsoft.com/downloads/details.aspx?familyid=15805380-F2C0-4B80-9AD1-2CB0C300AEF9&displaylang=en

步骤3 设置XML Notepad将架构链接到程序  
首先打开XML Notepad，选择菜单“View”─“Schemas”，然后在“XML Schemas”屏幕中选择“File”─“Add schemas”，然后导航到需要提取的XML架构所在的文件夹，默认位置为C:\2007 Office System Developer Resources\Office2007XMLSchema\CustomUI.xsd。装载架构后，显示如下图1所示，单击“OK”。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/06/ribbonx12-1.png)  
图1：添加XML架构到XML Notepad  
**使用XML Notepad**  
使用XML Notepad最困难的部分是将文件装载并开始，这并不像在CustomUI Editor中打开Excel或Word文件那样简单，甚至不能从中创建新文件。相反，需要打开一个现有的XML文件。有两个主要的办法：

* 与在记事本中一样，将文档重命名为zip文件，复制其中的CustomUI.xml文件进行处理。完成后，将更新的CustomUI.xml文件复制回zip容器，然后将文件改为原始的.xlsx扩展名。
* 创建一个空的文本文件，从CustomUI Editor中复制现有的XML，然后保存为XML文件，接着在XML Notepad中打开该文件，在这里编辑代码，然后将代码复制回CustomUI Editor并保存。

虽然使用XML Notepad的很繁锁，但是它提供了几个功能，可以容易地创建有效的文件而不会有输入错误的危险。这对开发者来说是非常有利的。  
下面来演示XML Notepad的使用。  
步骤1 在CustomUI Editor中打开MySecondUIModification.xlsx文件。  
步骤2 复制代码后关闭CustomUI Editor。  
步骤3 在桌面上右击，创建一个新的文本文件并命名为temp.xml。  
步骤4 右击该文件并选择编辑（此时不要选择XML Notepad）。  
步骤5 粘贴刚才从CustomUI Editor中复制的代码。  
步骤6 保存该文本文件并关闭。  
步骤7 再在该文件上右击，此时选择“Edit with XML Notepad”。  
单击其中的扩展符号（+），得到如下图2所示的屏幕，一个图形化的显示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/06/ribbonx12-2.png)  
图2：XML Notepad  
步骤8 右击“tab”元素，选择“Element”，然后选择“Child”，将在最后一个组的下面创建一个新元素，如下图3所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/06/ribbonx12-3.png)  
图3：添加元素  
此时，XML Notepad将提供合适的项目列表，可以从中选择，这里是“group”元素，即设置该元素为组。  
步骤9 接下来，在新添加的元素中右击，选择“Atrribute”，然后选择“Child”，此时将出现一个更长的项目列表，从中找到“idMso”后单击选中。  
此时，光标将出现在右侧的屏幕中，与idMso一致，输入文本：GroupStyles，注意大小写。如下图4所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/06/ribbonx12-4.png)  
图4：添加属性  
现在，已经修改了代码，于是想试验新代码。  
此时，单击菜单“View”─“Source”，当提示是否保存代码时，选择“是”。将出现如下图5所示的屏幕，包含了新添加的标记。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/06/ribbonx12-5.png)  
图5：在XML Notepad中查看XML源代码  
注意到代码与前面复制的代码稍微有点不同，出现了前面没有看到过的其它元素。  
现在已经有了代码，那么继续进行下面的操作。  
步骤10 复制整段代码并关闭该窗口。  
步骤11 在CustomUI Editor中打开MySecondUIModification.xlsx文件。  
步骤12 按Ctrl+A选择所有现有的代码。  
步骤13 粘贴从XML Notepad程序中复制的代码。  
步骤14 运行有效验证检查，然后保存并关闭文件。  
最后，打开MySecondUIModification.xlsx文件以检查“My Tab”选项卡的外观，应该如下图6所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/06/ribbonx12-6.png)  
图6：Excel中更新后的My Tab选项卡  
**XML Notepad的优点**  
XML Notepad提供了许多CustomUI Editor或标准的记事本所没有的优点，包括：

* XML Notepad能够使开发更容易。由于安装了相关的架构，因此能够从下拉列表中选择有效的元素和属性，类似于大多数Microsoft编码工具中的智能感知（IntelliSense）功能。
* 与CustomUI Editor不同，XML Notepad有能力进行查找和替换样式搜索，这是其主要的优点。
* XML Notepad包含一种称之为“Nudge”的功能，将一块代码作为一个单元向上或向下移动。
* 在XML Notepad中显示的步进布局使得容易识别嵌套的元素。
* 因为XML Notedpad被链接到XML架构，所以能够实时进行有效性检查并报告所找到的错误。

**XML Notepad的缺点**  
除了不能将文件放入编辑器或者从编辑器中获得文件外，与CustomUI Editor相比，XML Notepad还有下面的缺点：

* XML Notepad和CustomUI Editor的错误检查功能查找不同的情形。虽然能够从下拉列表中选择从而创建格式良好的XML，但是XML Notepad实际上不会验证代码。因此应该先后使用XML Notepad和CustomUI Editor，使得代码得到完美的验证。
* CustomUI Editor能够生成回调标签，但XML Notepad没有这项功能。
* CustomUI Editor能够使得创建自定义用户界面更容易。除了自动创建某些代码部分外，也提供了易于附加图片到文件中的接口。但XML Notepad缺乏这项功能。
* 如果使用XML Notepad而不是使用CustomUI Editor创建customUI.xml文件来开始功能区修改，那么需要人工链接.rels文件。因此，建议使用CustomUI Editor创建初始的联接。仍然在XML Notepad中创建CustomUI代码的构造，然后将其复制到CustomUI Editor中，正如上面的示例所做的。

[**细品RibbonX(13)：初识XML**](http://www.excelperfect.com/index.php/2009/07/08/ribbonxstudy13/)

2009年07月8日, 12:43 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

XML是扩展标记语言（Extensible Markup Language）的缩写，最初由W3C发布。XML不是真正的编程语言，因为它缺乏执行操作的某种机制，相反，XML是一组规则，其意图是简化跨平台的数据共享。

Office 2007文件以Microsoft的OpenXML格式发布，是包含几个XML文件的一个简单的压缩容器。使用XML格式的文件具有下列优点：

（1）XML文件能够明显地节省空间。

（2）通过XML结构化数据后，能够快速地被索引，因而能使其他程序通过索引搜索到特定的字符串或其他项目。

（3）对于开发者来说，能够链接某个应用程序里的XML架构来验证XML而确保代码像意料中的那样工作。例如前面介绍的CustomUI Editor和XML Notepad，如果没有这些架构，CustomUI Editor将不能验证代码，XML Notepad也不能提供智能感知功能。

当然，我们之所以关心XML，是因为XML是功能区的核心。要定制Office 2007用户界面，必须编写XML代码。虽然VBA在定制界面时也能起到关键的作用，但它不是必需的。而XML在定制中几乎总是必需的，但也有例外。例如弹出式菜单和一些出现在加载项选项卡中的VBA定制。

下面，我们来了解如何构成XML代码，以及如何编写XML。

**标签（Tags）**

先看看下面的代码片断：

<group id=”rxgrpTest”>

<button idMso=”Bold”/>

<button idMso=”Italic”/>

<splitButton id=”rxsbtnTest”>

<button idMso=”Underline”/>

</splitButton>

</group>

留意上面代码的结构，注意<、>和/号的使用，这些符号在XML中有特定的用途：指出了标签的开始和结束位置。例如，开始的<group>标签和结束的</group>标签之间的内容包含了在功能区中显示的组。

在组里面列出的splitButton标签使用与组相同的结构，以<splitButton>标签开始，接着包含了所有的项目直至由</splitButton>标签结束。本例中，仅仅下划线按钮出现在拆分按钮中。

现在，看看按钮本身的代码，注意到/号位于标签的结尾，这样就不需要单独的标签来关闭元素。

这里演示了用于开始和结束XML标签的两种方法。您需要理解和使用这两种方法来创建自定义UI的结构。如下所示：

开始和关闭标签分开：<element attribute(s)=”Value”></element>

开始和关闭标签未分开：<element attribute(s)=”Value”/>

通过上面的例子，应该了解到：

（1）在XML中，每个标签至少包含一个元素并且通常至少有一个属性。

（2）在处理代码时，涉及的每个项目都被称作对象（object）。例如，对象包括功能区中的组、按钮、复选框、甚至是菜单。许多对象都有子对象。当某控件有子控件时，该控件称作为父控件。注意，父对象通常（但不总是）有不同类型的子对象。选项卡有一个或多个组对象作为子对象，并且一个组可能有按钮、复选框和动态菜单子对象的组合。其中，dynamicMenu对象除了其他子对象类型外，还有dynamicMenu子对象。某些父对象甚至有完整的子对象群，称之为集合（collection）。

每个子对象同时也是它自已的子对象的父对象。每个子对象控件必须嵌套在其父对象的开始和结束标签中。并且，每个子对象仅可能有一个父对象。

（3）为了创建格式良好（有效的）XML，必须理解：①所有的标签，无论是元素还是属性，都区分大小写，即SPLITBUTTON元素与splitButton元素不相同。②属性值必须被放置在单引号或双引号内。③在父元素中的子元素的嵌套必须准确，每个开始标签必须与其结束标签相匹配，或者使用“/>”结束在相同的标签内，或者在后面使用单独的标签。

**元素（Elements）**

当处理功能区的XML时，可以看到每个元素指定功能区控件（或结构）的特定部分。

对于初学者来说不容易弄清楚标签和元素的区别，然而很容易定义：group是一个元素，而<group>是一个标签。因此，标签实质上是通过<和>符号识别代码块的标志。

每个标签必须包括一个且唯一一个元素。此外，元素总是XML标签的第一部分，告诉编译器希望开始处理或者停止处理的特定项目。

**属性（Attributes）**

标签告诉编译器要处理的对象，而属性告诉编译器该对象的特性，例如对象的名称、在屏幕中显示的标题以及是否对象可见。

与元素不同的是，在所给标签内可以为对象设置多个属性。例如，下面的代码片断显示了带有多个属性的标签示例：

<button id=”rxbtnProtectAll”

size=”normal”

label=”Protect All Sheets”

imageMso=”ReviewProtectWorkbook”

onAction=”rxbtnProtectAll\_click”/>

上面的代码提供给按钮一个唯一的id并指定了大小、标签文本和图像，还提供了一个onAction回调签名，当单击该按钮时启动相应的VBA过程。

**1、id属性**

在创建额外的对象之前，需要探讨如何在代码中识别这些对象。而id属性用于识别在自定义XML代码内特定的对象，为其提供名称，使得后面能够使用该名称引用该对象。这是引用对象的唯一方法。在下列容器中每个对象都需要id属性：

         contextualTabs

         officeMenu

         qat

         tabs

id属性有几种不同的类型并且每一个都有不同的用途。下面列出了id属性的不同类型及其主要用途。

         id属性用于唯一地标识控件。如果动态地装载项目，其属性将被赋值。

         idMso属性用于唯一地标识内置控件、选项卡、命令，等等。使用该属性与内置对象交互。

         idQ属性用于跨共享的命名空间引用对象。

为了识别某对象，只需简单地在开始标签中添加id属性，例如，要通过id引用某选项卡，使用下列XML：

<tab id=”rxtab”

<!—放置其它的选项卡属性! –>

</tab>

同样地，如果试图引用内置的字体（Font）组，则使用下面的XML：

<group idMso=”Font”

<!—放置其它的组属性! –>

</group>

注意，所有的id和idQ属性都是唯一的。如果使用已存在的id，或者由Microsoft保留的id，将会导致错误并阻止装载自定义用户界面。

为了避免与内置的控件相冲突，建议所有自定义控件使用标准的命名约定加上前缀。

**2、label属性**

label属性指定用户能够在屏幕上读取的内容，它不必是唯一的，但应该合乎逻辑、简洁明了且一致。label属性为用户提供了清晰的向导。要为控件添加标签文本，使用下面所示的代码：

<tab id=”rxtab”

label=”My Custom Tab”>

</tab>

**XML代码的排版技巧**

在编写代码时，建议使用硬回车或制表符将代码分成逻辑块，使代码更容易阅读和理解。看看下面的示例代码，非常清晰，更容易阅读：

<group

id=”rxgrp”

label=”My First Group”>

<button

id=”rxbtn1″

imageMso=”Italic”

label=”Large size button”

size=”large”

onAction=”rxbtn1\_Click”/>

<button

id=”rxbtn2″

imageMso=”Bold”

label=”Normal size button”

size=”normal”

onAction=”rxbtn2\_Click”/>

<button

id=”rxbtn3″

imageMso=”WrapText”

label=”Normal size button”

size=”normal”

onAction=”rxbtn3\_Click”/>

<button

id=”rxbtn4″

imageMso=”ConditionalFormatting”

label=”Normal size button”

size=”normal”

onAction=”rxbtn4\_Click”/>

</group>

**在XML代码中创建注释**

注释能够使自已或他人清楚代码的作用。

放置注释的方法是：在开始处放置一个小于号（<），然后输入感叹号（！），随后是两个连字号（–），结束注释需要两个连字号加一个大于号（>），例如：

<!—这是一段注释 –>

如果注释很长，可能需要几行，那么可以按如下所示编写：

<!–

这是一段相当长的注释。因为您想使用长的注释解释这段代码有什么作用，各个符号代码什么意思，以帮助您以后好理解。

–>

在<!–和–>之间的内容被当作连续的注释。

此外，不能够将注释放置在一块开始的XML代码的中间。下面的示例来进行演示：

<!—这是我的按钮 –>

<button id=”rxbtn”

label=”This is my button”

imageMso=”HappyFace”

size=”large”

onAction=”rxbtn\_Click”/>

或：

<button id=”rxbtn”

label=”This is my button”

imageMso=”HappyFace”

size=”large”

onAction=”rxbtn\_Click”>

<!—这是我的按钮 –>

</button>

上述两段代码都是正确的，但下面的代码会产生错误：

<button id=”rxbtn”

<!—这是我的按钮 –>

label=”This is my button”

imageMso=”HappyFace”

size=”large”

onAction=”rxbtn\_Click”/>

上面的问题在于注释被插入到了未关闭的代码块中。

[**细品RibbonX(14)：核心的OpenXML架构元素**](http://www.excelperfect.com/index.php/2009/07/10/ribbonxstudy14/)

2009年07月10日, 1:16 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**3** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

在编写自定义Office 2007界面的XML时，总是以customUI元素开始，紧接着是Ribbon元素。下面我们将详细介绍customUI元素、Ribbon元素，以及tabs、tab和group元素，在自定义界面时几乎总是要使用这些元素。

**customUI元素**

要创建格式良好的XML文档，必须包含一个且唯一一个“最外层的元素”，嵌套所有其他的元素。这个最外层的元素被称作“根元素”。对于功能区定制来说，该元素就是customUI元素。这意味着customUI标签里嵌套着所要使用的所有其它标签。

现在，打开CustomUI Editor，输入下面的代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<!– All other instructions go here –>

</customUI>

然后，单击“validate”按钮验证代码。注意，验证代码是一个好习惯。

**1、customUI元素必需的属性**

每个customUI标签必须指定xmlns属性，如下表1所示。

表1：customUI元素必需的属性

|  |  |
| --- | --- |
| **静态属性** | **允许值** |
| xmlns | http://schemas.microsoft.com/office/2006/01/customui |

**2、带有回调签名的可选的静态属性和动态属性**

也可以在customUI标签中添加下表2列出的部分或全部属性。当然，这些属性不一定非得在XML中使用，随您的需要而定。

表2：customUI元素的可选属性

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| xmlns:Q | (none) | 1至1024个字符 | (none) | (none) |
| (none) | onLoad | 1至1024个字符 | (none) | Sub onLoad(Ribbon as IRibbonUI) |
| (none) | loadImage | 1至1024个字符 | (none) | Sub loadImage(imageID as string,ByRef returnedVal) |

例如，添加onLoad属性的代码如下：

<customUI

xmlns=“http://schemas.microsoft.com/office/2006/01/customui“

onLoad=“rxiRibbonUI\_onLoad“>

<!– All other instructions go here –>

</customUI>

注意，onLoad属性只是被嵌套在与customUI元素相同的<>括号内，对其他可选属性也是如此。并且，可以每行列出一个属性，无须包括连续或行中断的代码。

**3、customUI元素的子对象**

customUI标签是一个容器，包含其他对象。实际上，customUI仅能包含下列元素：

         commands

         ribbon

**ribbon元素**

要修改功能区，则要在customUI标签内嵌套ribbon元素，如下所示：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon>

<!– All other instructions go here –>

</ribbon>

</customUI>

**1、ribbon元素必需的属性**

ribbon元素没有任何必需的属性。

**2、可选的静态属性**

ribbon元素有一个非常特别的属性，如下表3所示。

表3：ribbon元素的可选属性

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| startFromScratch | （none） | true,false,1,0 | false | （none） |

startFromScratch属性能够隐藏整个内置的功能区。因为该属性的默认值为false，所以前面的示例代码中都忽略了该属性。下面的XML能够得到相同的结果：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<!– All other instructions go here –>

</ribbon>

</customUI>

**3、ribbon元素的子对象**

ribbon对象可以包含下列元素：

         contextualTabs

         officeMenu

         qat

         tabs

**tabs元素**

这是除customUI和ribbon之外的另一个容器，必须嵌套在功能区（Ribbon）块中。示例代码如下：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon>

<tabs>

<!– All other instructions go here –>

</tabs>

</ribbon>

</customUI>

tabs标签的目的是集中要引用、创建或修改的每个元素和单个选项卡。

**1、tabs元素必需的属性**

tabs元素是要使用的最容易的元素之一，没有单个的属性。

**2、tabs元素的子对象**

tabs元素是一个容器元素，用于包含特定引用的（或创建的）选项卡控件。

**tab元素**

tab元素包含在tabs容器中，用于创建或引用功能区中单个的选项卡。理解tabs对象和tab对象之间的不同是非常重要的。tabs（复数形式）引用整个选项卡集合，而tab（单数形式）指定特定的选项卡。

**1、tab元素必需的属性**

tab对象是层级中的第一个元素，需要id属性。必须从下表4中选择且只能选择其中一个：

表4：tab元素的id属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的选项卡时 |
| idMso | 当使用现有的Microsoft选项卡时 |
| idQ | 当创建在命名空间之间共享的选项卡时 |

为什么需要id属性的原因很简单：如果选项卡没有id属性，那么如何引用该选项卡呢？

**2、带有回调签名的可选的静态属性和动态属性**

tab元素提供了几个可选的静态属性。如果要设置某选项卡相对于任何其它现有的选项卡的位置，则使用下表5列出的insert属性之一。

表5：tab控件可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **insert属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso选项卡 | 在最近的选项卡之后插入 | 在Microsoft选项卡之后插入 |
| insertBeforeMso | 有效的Mso选项卡 | 在最近的选项卡之后插入 | 在Microsoft选项卡之前插入 |
| insertAfterQ | 有效的选项卡idQ | 在最近的选项卡之后插入 | 在共享的命名空间选项卡之后插入 |
| insertBeforeQ | 有效的选项卡idQ | 在最近的选项卡之后插入 | 在共享的命名空间选项卡之前插入 |

注意，如果没有指定insert属性，那么选项卡将被添加到最后一个选项卡之后，而不管是自定义选项卡还是内置选项卡。

tab控件也接受下表6列出的任一或全部属性：

表6：tab控件的可选属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| keytip | getKeytip | 1至3个字符 | （none） | Sub GetKeytip(control As IRibbonControl,ByRef returnedVal) |
| label | getLabel | 1至1024个字符 | （none） | Sub GetLabel(control As IRibbonControl,  ByRef returnedVal) |
| tag | （none） | 1至1024个字符 | （none） | n/a |
| visible | getVisible | true,false,1,0 | true | Sub GetVisible(control As IRibbonControl, ByRef returnedVal) |

注意，动态回调用于在文件使用时动态修改功能区，使用了VBA代码来运行。

**3、tab元素的子对象**

tab对象仅仅可以包含组元素，例如内置的剪贴板组或者自已创建的自定义组。

**4、内置的选项卡**

当在功能区中处理选项卡时，有两种类型的选项卡：内置的和自定义的。内置选项卡由Microsoft提供，而自定义选项卡由自已创建。

**（1）引用内置选项卡**

每个内置选项卡都有自已唯一的idMso属性，通过调用该属性引用这个选项卡，因此应该弄清楚idMso属性。

下表7列出了Excel、Access和Word的一些最通用的选项卡名称，注意大小写。

表7：最常用的内置选项卡名称

|  |  |  |  |
| --- | --- | --- | --- |
| **选项卡名称** | **idMso（Excel）** | **idMso（Word）** | **idMso（Access）** |
| 开始 | TabHome | TabHome | TabHomeAccess |
| 插入 | TabInsert | TabInsert | （none） |
| 页面布局 | TabPageLayoutExcel | TabPageLayoutWord | （none） |
| 公式 | TabFormulas | （none） | （none） |
| 数据 | TabData | （none） | （none） |
| 视图 | TabReview | TabReviewWord | （none） |
| 创建 | （none） | （none） | TabCreate |
| 外部数据 | （none） | （none） | TabExternalData |
| 数据库工具 | （none） | （none） | TabDatabaseTools |

**（2）修改内置选项卡**

步骤1 打开Excel 2007创建一个新的工作簿。由于不需要任何动态回调，因此将该文件保存为Excel的默认工作簿格式(xlsx)。

步骤2 关闭Excel，然后在CustomUI Editor中打开该工作簿。

步骤3 输入下面的XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab idMso=“TabHome“ visible=“false“>

</tab>

</tabs>

</ribbon>

</customUI>

步骤4 验证代码并保存，关闭CustomUI Editor。

步骤5 在Excel中打开该工作簿文件，此时“开始”选项卡没有了。

此时，关闭该工作簿文件，将使“开始”选项卡重新出现。

**5、自定义选项卡**

**（1）创建自定义选项卡**

创建自定义选项卡时，不需要引用idMso，而是为新选项卡指定一个唯一的id属性。

继续上面的练习，在Excel中关闭工作簿文件，在customUI Editor中重新打开该文件，输入下面的代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“ label=“Demo“>

</tab>

</tabs>

</ribbon>

</customUI>

验证代码并关闭该文件，然后在Excel中重新打开该工作簿，此时在功能区中将增加一个新的自定义选项卡，如下图1所示。

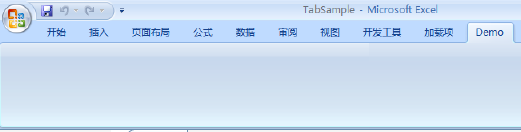
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/07/ribbonx14-1.png)

图1：功能区中新增加的自定义选项卡

**（2）放置自定义选项卡**

现在我们将自定义选项卡放置在“开始”选项卡之后，代码如下：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“

label=“Demo“

insertAfterMso=“TabHome“>

</tab>

</tabs>

</ribbon>

</customUI>

即为insert属性指定有效的idMso，将自定义选项卡放置在想要的位置。此时，工作簿如下图2所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/07/ribbonx14-2.png)

图2：自定义选项卡放置在“开始”和“插入”选项卡之间

注意，当插入多个控件时，将按照它们在XML代码中的顺序插入。

**group元素**

group元素的作用是为实际的按钮、复选框、菜单和其他允许在功能区中配置的命令创建占位符。

**1、group元素必需的属性**

与tab元素一样，每个组需要一个唯一的id属性，如下表8所示。

表8：group元素的id属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的组时 |
| idMso | 当使用现有的Microsoft组时 |
| idQ | 当创建在命名空间之间共享的组时 |

**2、带有回调签名的可选的静态属性和动态属性**

要设置某个组相对于任何其他现有的组的位置，需要指定下表9列出的insert属性之一。

表9：group控件可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **insert属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso组 | 最近的组之后插入 | Microsoft组之后插入 |
| insertBeforeMso | 有效的Mso组 | 最近的组之后插入 | Microsoft组之前插入 |
| insertAfterQ | 有效的组idQ | 最近的组之后插入 | 共享的命名空间组之后插入 |
| insertBeforeQ | 有效的组idQ | 最近的组之后插入 | 共享的命名空间组之前插入 |

group控件也接受下表10列出的任何或全部属性。

表10：group控件可选的属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| image | getImage | 1至1024个字符 | （none） | Sub GetImage(control As IRibbonControl, ByRef returnedVal) |
| imageMso | getImage | 1至1024个字符 | （none） | 同上 |
| keytip | getKeytip | 1至3个字符 | （none） | Sub GetKeytip(control As IRibbonControl, ByRef returnedVal) |
| label | getLabel | 1至1024个字符 | （none） | Sub GetLabel(control As IRibbonControl, ByRef returnedVal) |
| screentip | getScreentip | 1至1024个字符 | （none） | Sub GetScreentip(control As IRibbonControl, ByRef returnedVal) |
| supertip | getSupertip | 1至1024个字符 | （none） | Sub GetSupertip(control As IRibbonControl, ByRef returnedVal) |
| tag | （none） | 1至1024个字符 | （none） | （none） |
| visible | getVisible | true,false,1,0 | true | Sub GetVisible(control As IRibbonControl, ByRef returnedVal) |

**3、group元素的子对象**

group元素能够包含下列对象的任意组合：

         box

         button

         buttonGroup

         checkBox

         comboBox

         control

         dialogBoxLauncher

         dropdown

         editBox

         gallery

         labelControl

         menu

         separator

         splitButton

         toggleButton

下面是在自定义选项卡中创建一个自定义组的XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“

label=“Sample“

insertBeforeMso=“TabHome“>

<group id=“rxgrpDemo“

label=“Demo Group“

imageMso=“HappyFace“

keytip=“D“

screentip=“This is my screentip“

supertip=“This is my supertip“>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

结果如下图3所示：

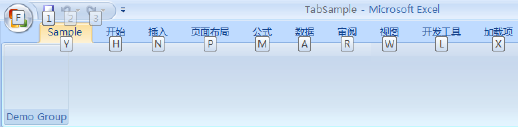
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/07/ribbonx14-3.png)

图3：自定义选项卡中的自定义组

注意，在上面的XML中设置的属性，仅label属性的值出现在功能区中。大多数可选的属性例如imageMso，在组控件中将没有可见的效果。因此，经调试后，可以忽略一些可选的属性。

**4、内置组**

使用内置组，仍然能够提供内置功能。此外，能够将经常使用的控件组合在一起。

**（1）引用内置组**

与选项卡相似，通过引用其id属性识别自定义组，通过引用其idMso属性识别内置组。下表11列出了Excel、Access和Word中的一些内置组。

表11：在Excel、Access和Word中通用的组

|  |  |
| --- | --- |
| **显示的名称** | **idMso名称** |
| 剪贴板 | GroupClipboard |
| 字体 | GroupFont |
| 形状 | GroupShapes |

例如，隐藏Excel中剪贴板组的XML如下：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab idMso=“TabHome“>

<group idMso=“GroupClipboard“

visible=“false“>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

结果如下图4所示：

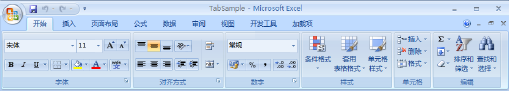
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/07/ribbonx14-4.png)

图4：隐藏了“剪贴板”组的“开始”选项卡

**（2）在自定义选项卡中使用内置组**

能够在自已的选项卡中使用内置组，这是Office 2007的一项关键功能。

Microsoft虽然花费了大量的人力和财力研究用户的使用习惯，但默认的控件顺序不会总是满足特定的需求。因此，利用重复使用Microsoft内置组的能力，可以容易地在单个选项卡中放置最常用的命令，这样更方便于使用且随时可见，避免在功能区选项卡之间来回切换。

在许多场合，创建一个自定义且经过整理的选项卡能够节省无数次的键击并极大地提高生产力。例如，假设要花时间审阅Excel工作簿的一致性，在默认的功能区中，需要经常在功能区选项卡之间来回切换以利用所需要的控件，下表12列出了在Excel中需要使用到的组。

表12：在Excel中常用的审核工具的位置

|  |  |  |
| --- | --- | --- |
| **组名** | **默认的选项卡** | **名称** |
| 剪贴板 | 开始 | GroupClipboard |
| 字体 | 开始 | GroupFont |
| 公式审核 | 公式 | GroupFormulaAuditing |
| 批注 | 审阅 | GroupComments |
| 编辑 | 开始 | GroupEditingExcel |

现在，创建一个包含上述组的自定义选项卡，其XML代码如下：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtabMyTools“

label=“My Tools“

insertBeforeMso=“TabHome“>

<group idMso=“GroupClipboard“/>

<group idMso=“GroupFont“/>

<group idMso=“GroupFormulaAuditing“/>

<group idMso=“GroupComments“/>

<group idMso=“GroupEditingExcel“/>

</tab>

</tabs>

</ribbon>

</customUI>

注意，在上面的代码中，group元素没有使用单独的结束标签，而是在行尾使用/>结束。这样的编写非常简洁。此时，结果如下图5所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/07/ribbonx14-5.png)

图5：填充内置组的自定义选项卡

注意，在自定义组中引用的内置组只是原来组的副本，内置组仍然在原来的位置。

**5、自定义组**

**（1）创建自定义组**

与tab元素一样，通过指定一个唯一的id属性来创建自定义组，而不是引用内置组的idMso。

继续上面的示例，关闭Excel文件，在CustomUI Editor中重新打开该文件，在最后一个group标签之后，添加下面的XML代码：

<group id=“rxgrpMyGroup“

label=“My Group“>

</group>

注意，上述代码并没有在同一行中开始和结束组代码。这样，方便在其中添加其他控件，例如按钮、复选框或动态菜单。

验证代码后保存文件并关闭CustomUI Editor。再打开Excel，此时，在自定义选项卡中添加了一个空组，如下图6所示。

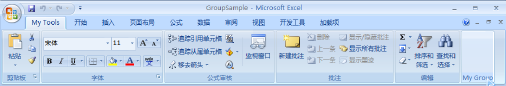
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/07/ribbonx14-6.png)

图6：自定义选项卡中的自定义组

**（2）放置自定义组**

有时，可能想让自定义的组放置在指定的位置或按指定的顺序排列，可以采取两种方法来实现。

方法1：在XML代码中，按想要显示的顺序编写代码。

方法2：使用insertBeforeMso属性或insertAfterMso属性，在某个组之前或之后放置自定义的组。例如，上例中使用下面的代码将自定义组放置在“字体”组之前：

<group id=“rxgrpMyGroup“

label=“My Group“

insertBeforeMso=“GroupFont“>

</group>

结果如下图7所示。

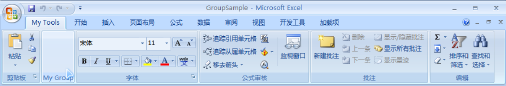
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/07/ribbonx14-7.png)

图7：将自定义选项卡放置在“剪贴板”组和“字体”组之间

**（3）在内置选项卡中的自定义组**

在内置选项卡中插入自定义组也是很容易的，下面仍以Excel 2007为例来演示。

步骤1 创建一个新的Excel文档并保存，然后关闭该文档。

步骤2 在CustomUI Editor中打开该文档，并输入下面的代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab idMso=“TabHome“>

<group id=“rxgrpMyGroup“

label=“My Group“

insertBeforeMso=“GroupFont“>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

步骤3 验证代码并保存代码，然后关闭CustomUI Editor。

步骤4 在Excel中重新打开该文件，现在看到在“开始”选项卡中有一个自定义组，如下图8所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/07/ribbonx14-8.png)

图8：在“开始”选项卡中的自定义组

此时，在内置的“开始”选项卡中添加了自定义组“My Group”，并且将该组放置在第二位，即“字体”组之前。然后，您就可以在自定义组中添加想要的控件了！

[**细品RibbonX(15)：回调(CallBacks)**](http://www.excelperfect.com/index.php/2009/07/18/ribbonxstudy15/)

2009年07月18日, 7:49 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**2** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

回调，即使自定义界面能够工作的代码。如果没有回调，那么自定义界面可能看起来漂亮，但只是一个漂亮的功能区。当然，除内置控件不需要自定义回调外。

下面的内容介绍如何使用回调来提供自定义控件所需的功能。在编写XML代码时，需要指定回调，然后在Excel中编写VBA代码来匹配和处理回调。

**回调概要**

回调是自定义的用户界面使用的子过程和函数，使自定义的用户界面能够工作。回调简单地表现为所提供的指令的动向。例如，当设置某按钮的onAction属性并装载用户界面时，一旦单击该按钮，就产生回调，即在属性中指定的操作。

如果没有找到指定的操作，那么回调将失败，因为代码中发生了例外，即在属性中指定的回调在VBA中不存在，因此失败了。

自定义用户界面一装载就会调用VBA工程，搜寻指定的函数或过程。如果找到，则将其值传回到用户界面。如果没有找到指定的函数或过程，那么将产生错误的结果。

有两种主要的方式来创建回调：

         直接在VBE的标准模块中输入过程或函数

         使用工具生成过程，例如Office 2007 CustomUI Editor

使用诸如Office 2007 CustomUI Editor这类工具的主要优势是能够扫视XML代码并返回回调签名，也称作subprocedure stub，XML代码中每个属性都有一个需要处理的回调。接受回调作为值的属性包括onAction、getVisible和getImage。

除了节省时间外，使用CustomUI Editor还能避免由于手工编写回调签名而导致的拼写错误。上述每个属性都会产生一个不同的回调签名，必须按顺序使用代码处理，使用户界面有合适的功能。

此外，当工程装载时需要访问某些回调并运行，这意味着如果回调处理（即响应回调的VBA代码）在该工程中不存在，将产生错误消息。当然，下面的内容也会介绍如何缓解这类错误，甚至如何避免错误。

**为动态回调建立文件**

为了使自定义控件工作，文件必须启用宏，否则不能够添加或运行VBA代码。

**1、捕获IRibbonUI对象**

IRibbonUI对象引用功能区用户界面，用于控制事物如何响应。

在VBA中使用的关键之一是使整个Ribbon对象无效（因而能够改变功能区的某项特征）或者使功能区中指定的控件无效（因而能够改变该控件的某项特征）。

（1）调整XML以包括onLoad

为了使用IRibbonUI对象，需要在VBA中对其设置。首先，需要对UI的onLoad属性指定值，这能通过为onLoad属性指定回调来实现：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“ onLoad=“rxIRibbonUI\_onLoad“>

因为onLoad属性的值是一个回调，所以需要在VBA代码中对其进行处理。接着，能够在工程中使用IRibbonUI对象。

（2）创建VBA代码处理onLoad事件

可能会注意到，前面将onLoad称作为属性，但现在将其称为事件。这是因为在XML文件中，能为许多属性定义值，例如onAction、getLabel和onLoad。一旦为某属性赋值，如果能以某种方式触发，将导致事件发生。这就是属性和事件的区别。

因为IRibbonUI对象用于整个应用程序，所以需要在标准模块的全局声明部分对其声明，如下面的代码所示。

‘全局的ribbon对象

Dim grxIRibbonUI As IRibbonUI

这里使用标准的格式grxIRibbonUI引用IRibbonUI对象，接着添加回调来设置该对象：

Sub rxIRibbonUI\_onLoad(ribbon As IRibbonUI)

Set grxIRibbonUI = ribbon

End Sub

设置为全局对象之后，该对象可用于整个工程。然而，Ribbon对象对于修改是非常敏感的。这意味着任何时候修改代码，该对象实例将丢失并且任何需要Ribbon对象的操作都将失败。因此，无论何时作出修改，都需要保存、关闭并重新打开工程。

**生成第一个回调**

知道需要获得回调签名后，还需要知道如何使用回调签名。例如，对于切换按钮（toggleButton）有下列签名：

Sub rxtgl\_click(control as IRibbonControl,pressed as boolean)

一个普通按钮的签名如下：

Sub rxbtn\_click(control as IRibbonControl)

**1、从头开始编写回调**

如果知道了回调签名，则不必使用标准的形式声明参数。例如，有一个使用下列XML的切换按钮：

<toggleButton

id=”rxtgl”

label=”Toggle”

size=”large”

onAction=”rxtgl\_click”

imageMso=”FormatPainter”/>

编写rxtgl\_click回调如下：

Sub rxtgl\_click(rxctl As IRibbonControl,toggled As Boolean)

If toggled Then

MsgBox “我已切换 …我的ID是 “ & rxctl.ID,vbInformation

End If

End Sub

注意到回调的参数名与上面介绍的标准的参数名不一样，但是单击切换按钮时仍触发该回调，像平常一样传递参数，因此签名中的修改不会导致过程失败，只要使用了正确类型的正确参数。

也能修改onLoad回调签名来满足需要，如下所示：

Sub rxIRibbonUI\_onLoad(MyRibbon As IRibbonUI)

Set grxIRibbonUI=MyRibbon

End Sub

然而，如果使用标准的签名及其参数名称，那么其他人将更容易理解代码。下面列出了一些回调签名。

属性onLoad，回调签名(ribbon as IRibbonUI)；

属性getLabel,getPressed,getEnabled,getImage,getScreentip,getVisible,等，回调签名(control as IRibbonControl,ByRef returnedVal)；

属性onAction(切换按钮)，回调签名(control as IRibbonControl,pressed as Boolean)；

属性onAction(按钮)，回调签名(control as IRibbonControl)。

**2、使用Office CustomUI Editor生成回调**

一种很容易生成回调的方法是使用CustomUI Editor。这个工具将所有的返回回调的属性搜寻在一起，然后生成需要的回调。因此，无须追踪XML中所有的回调。对于大段的XML代码来说，这是无价的。

在CustomUI Editor中（以Excel为例），可以使用下列步骤自动生成必需的回调：

步骤1 使用CustomUI Editor打开包含XML代码的Excel。

步骤2 单击“Generate Callbacks”按钮。

步骤3 出现一个新的Callbacks选项卡。复制产生的代码并粘贴到VBA工程中。

注意，确保验证代码。

**3、理解文件打开时事件的顺序**

在工程中添加了回调后，该文档打开时将调用某些过程，具体调用的过程取决于工程打开时定制是否获得了焦点。注意，某些过程仅当包含定制的选项卡获取焦点时才调用，而其它过程当鼠标移动到控件之上时调用。

理解过程调用的顺序是复杂的，因为调用顺序受许多可变因素影响。为了帮助预测典型的事件顺序，下表列出了功能区选项卡的事件及相应的顺序。

表：当打开工程后选项卡获取焦点时的事件顺序

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **事件** | **选项卡获取焦点** | **选项卡有焦点** | **按下ALT键** | **鼠标在其上方** |
| onLoad | 最顶级的事件，当装载UI时发生 | | | |
| getVisible | 1 | 1 | N/A | N/A |
| getLabel | 2 | 3 | N/A | N/A |
| getImage | 3 | 4 | N/A | N/A |
| getEnabled | 4 | 2 | N/A | N/A |
| getKeytip | N/A | N/A | 1 | N/A |
| getScreentip | N/A | N/A | N/A | 1 |
| getSupertip | N/A | N/A | N/A | 2 |

上表仅列出了可以使用的通用属性的一些示例，然而顺序可能受其它引入的属性的影响，例如getDescription、getTitle，等等。虽然如此，打算在性能方面以最好的方式解决用户界面问题时，上表仍然可以作为一个通用的向导。

**4、能够有具有相同名称但不同签名的两个回调吗？**

VBA不允许在相同工程中使用相同名称不同签名的两个回调。然而，如果回调在不同的工程中，则相同的回调名称能够有不同的签名。因此，如果同时打开一个以上的Excel文档，那么可能发现回调返回不可预料的结果。因为如果多个操作（签名）与一个回调名称相联系，那么将运行当前文档的回调。

假设打开一个安装了两个加载项的Excel工作簿，三个项目都有一个带有称为rxbtnnsQaShared的控件的用户界面。当单击用户界面中该控件时，所期望的是添加一个新工作簿，然而将出现一个消息框。检查后，发现为所有三个按钮都使用了相同的回调。此时，可以使用断点调试，看看单击操作调用了哪个回调。

上述表明，只要在不同工程中，虽然允许使用带有不同签名相同名称的回调，但这不是一个好主意。

**调用位于不同工作簿中的过程**

与上面所讲的情况相似，如果XML代码运行位于不同工作簿中的VBA，也会碰到类似的问题。

假设有两个工作簿：Book.xlsm和Book2.xlsm，希望在第一个工作簿中添加一个按钮，运行第二个工作簿中的过程。使用下面的XML代码创建用户界面：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon>

<tabs>

<tab id=“rxtabDemo“

label=“My Custom Tab“

insertBeforeMso=“TabHome“>

<group id=“rxgrpDemo“

label=“My Demo Group“>

<button id=“rxbtnDemo“

label=“My Demo Button“

size=“large“

onAction=“Book2.xlsm!rxbtnDemo\_click“

imageMso=“FileStartWorkflow“/>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

在Book2.xlsm的标准模块中，放置下列代码：

Sub rxbtnDemo\_click(control As IRibbonControl)

MsgBox “您调用了位于:” & ThisWorkbook.Name & “中的过程.”

End Sub

注意，必须要打开工作簿才能运行。如果希望onAction属性指向已装载的加载项而不是工作簿，那么简单地使用[add-in name].xlam在onAction代码名称前加上前缀，例如：

onAction=”myAddIn.xlam!rxbtnDemo\_click”

然后，在加载项的标准模块中放置回调VBA代码：

Book3.xlam!rxbtnDemo\_click

此时，必须加上前缀xlam!。注意，如果在用户界面的onAction属性下指定的过程中xlsm和xlam有相同的名称，那么该事件会运行包含在活动工作簿中的代码。你可能已注意到，在前面的介绍的内容中我们必须移除xlsm!，但使用加载项时必须有xlam!。

**组织回调**

随着功能区自定义代码技术的不断熟练，你将会采取不同的方式来组织回调，可以有单个的回调处理，或者有全局的回调处理，一次处理多个控件。

**1、单个的回调处理**

编写XML代码时，可以指定返回回调的多种属性，例如onAction、getLabel、getVisible、getEnabled，等等。这些属性的每一个都必须处理。下面以Excel为例，通过生成组中的3个按钮来演示这个过程。

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon>

<tabs>

<tab id=“rxtabDemo“

label=“My Custom Tab“

insertBeforeMso=“TabHome“>

<group id=“rxgrpDemo“

label=“My Demo Group“>

<button id=“rxbtnPaste“

label=“My Paste Button“

size=“normal“

onAction=“rxbtnPaste\_click“

imageMso=“Paste“

tag=“Custom Paste Button“/>

<button id=“rxbtnCopy“

label=“My Copy Button“

size=“normal“

onAction=“rxbtnCopy\_click“

imageMso=“Copy“

tag=“Custom Copy Button“/>

<button id=“rxbtnCut“

label=“My Cut Button“

size=“normal“

onAction=“rxbtnCut\_click“

imageMso=“Cut“

tag=“Custom Cut Button“/>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

每个按钮都有其onAction属性，因此如果企图对按钮添加功能，则必须处理每个属性：

Sub rxbtnPaste\_click(control As IRibbonControl)

MsgBox “您单击了” & control.Tag, vbInformation

End Sub

Sub rxbtnCopy\_click(control As IRibbonControl)

MsgBox “您单击了” & control.Tag, vbInformation

End Sub

Sub rxbtnCut\_click(control As IRibbonControl)

MsgBox “您单击了” & control.Tag, vbInformation

End Sub

单独处理每个属性非常繁锁且难以维护，而一种可供选择的方法是一次处理多个属性。

**2、使用全局回调处理**

（1）全局回调处理能够用于一次处理几个控件。如果以标准化方式命名全局处理，那么效果将更好，因为可以立即从单个控件中区分全局控件。

（2）使用全局回调的原因是因为某些指令可能重复，甚至当指令不重复时，对某种控件为某项操作共享相同的属性仍有好处，例如onAction属性。此外，如果操作相同，那么过程能组合在单个的回调处理中。

（3）能利用这种优势减少要处理的回调数，即通过通用的回调处理控件本身。

（4）例如，下列对象：tab，group，button，都共享一个通用的属性getLabel。如前所述，当需要对每个控件动态应用值时，不需要在XML代码中为每个控件编写getLabel回调。相反，可以在VBA过程中通过共享任务在单个处理内将它们放在一起，而且因为是回调，所以不需要遍历XML中指定的控件，将自动遍历控件直至所有控件有所需的值。

（5）仍以上节中的示例，在前面的XML代码中修改每个按钮的onAction属性如下：

onAction=”rxshared\_click”

将产生单个的回调，用于处理具有onAction属性和共享该签名的任何控件。

接着，在VBA中为每个按钮处理该回调，使用下面的一小段代码：

Sub rxshared\_click(control As IRibbonControl)

MsgBox “您单击了” & control.Tag, vbInformation

End Sub

也能使用Select Case语句分开每个按钮并赋予其特定的代码：

Sub rxshared\_click(control As IRibbonControl)

Select Case control.ID

Case “rxbtnPaste”

‘为rxbtnPaste按钮的代码

Case “rxbtnCopy”

‘为rxbtnCopy按钮的代码

Case “rxbtnCut”

‘为rxbtnCut按钮的代码

End Select

End Sub

因此，无需为用户界面中的每个自定义控件处理回调，可以使用全局的（共享的）回调，然后处理VBA过程里的每个控件。这里，使用了Select Case语句，因为该语句简洁且易于理解。也可以使用If…Then…ElseIf…End If语句。

**使用户界面组件无效**

Ribbon的一个重要方面是使Ribbon无效或指定的控件无效。

某些操作仅当使整个Ribbon无效才能执行，而另一些操作只需通过使指定控件无效来完成。

**1、无效能处理什么？为什么需要？**

IRibbonUI对象包含两个方法：Invalidate和InvalidateControl。

表：IRibbonUI对象模型

|  |  |
| --- | --- |
| **方法** | **用途** |
| Invalidate() | 为更新而标记整个功能区（顺序标记其中的每个控件） |
| InvalidateControl(strControlID) | 为更新而标记特定的控件，被更新的控件作为字符串被传递到方法的参数中。(strControlID) |

理解Invalidate方法的关键是该方法使用户界面中每个控件都无效，这意味着强制调用在用户界面中定义的所有回调。此外，也将导致更新所有控件，而无论其是否有回调。当使功能区无效时，在功能区中作出调用，并且在用户界面中指定的每个过程都将运行。

注意，“刷新”指在打开工程时简单地更新在用户界面中已经装载的控件，而“重新装载”指卸载并重新装载用户界面中的整个项目。

除非确实需要作用整个功能区，更好的选择是在执行期间的指定时刻使单个控件无效，即仅使必需的控件无效。

要使整个功能区或功能区中指定的控件无效，需要设置代表IRibbonUI对象的全局变量，可通过在onLoad属性中指定回调来实现：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“ onLoad=“rxIRibbonUI\_onLoad“>

</customUI>

接下来，需要编写VBA代码处理在onLoad属性中指定的回调：

Public grxIRibbonUI As IRibbonUI

Sub rxIRibbonUI\_onLoad(ribbon As IRibbonUI)

Set grxIRibbonUI=ribbon

End Sub

注意，代表IRibbonUI对象的变量在标准模块的全局声明区中声明，以便于能被工程的其他部分访问。

**2、使整个Ribbon无效**

步骤1 创建一个新的Excel文件，并保存为启用宏的工作簿。

步骤2 使用CustomUI Editor，在工作簿文件中添加XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“ onLoad=“rxIRibbonUI\_onLoad“>

<ribbon>

<tabs>

<tab id=“rxtabDemo“

label=“My Custom Tab“

insertBeforeMso=“TabHome“>

<group id=“rxgrpDemo“

label=“My Demo Group“>

<button id=“rxbtn“

getLabel=“rxshared\_getLabel“

size=“normal“

onAction=“rxshared\_click“

imageMso=“FillRight“/>

<button id=“rxbtn2“

getLabel=“rxshared\_getLabel“

size=“normal“

onAction=“rxshared\_click“

imageMso=“FillRight“/>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

步骤3 保存并验证代码。

步骤4 单击“Generate Callbacks”按钮生成回调。

步骤5 复制回调代码并关闭该文件。

步骤6 在Excel中打开该工作簿，添加一个标准模块。（不要担心出现的错误消息）

步骤7 粘贴代码到该模块中，并补充代码。

Public grxIRibbonUI As IRibbonUI

Public glngCount1 As Long

Public glngCount2 As Long

Sub rxIRibbonUI\_onLoad(ribbon As IRibbonUI)

Set grxIRibbonUI = ribbon

End Sub

Sub rxshared\_click(control As IRibbonControl)

grxIRibbonUI.Invalidate

End Sub

Sub rxshared\_getLabel(control As IRibbonControl, ByRef returnedVal)

Select Case control.ID

Case “rxbtn”

returnedVal = “功能区无效: ” & glngCount1 & “次.”

glngCount1 = glngCount1 + 1

Case “rxbtn2″

returnedVal = “功能区无效: ” & glngCount2 & “次.”

glngCount2 = glngCount2 + 1

End Select

End Sub

按照代码作用的顺序放置代码，然后保存工作簿，关闭工作簿，接着重新打开该工作簿。

注意，此时单击任一按钮，两个标签均更新，如下图1所示。因为单击按钮时代码使整个功能区无效，所以两个控件都无效，共享的getLabel过程为第一个按钮调用一次，又为第二个按钮所调用。

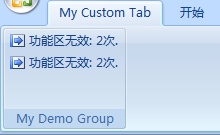
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/07/ribbonx15-1.png)

图1：使IRibbonUI对象无效导致其中的所有控件都被验证

**3、使单个控件无效**

仍然以上面的例子来介绍，现在仅使触发click事件的按钮无效。将共享的click事件和共享的getLabel事件修改如下：

Public grxIRibbonUI As IRibbonUI

Public glngCount1 As Long

Public glngCount2 As Long

Sub rxIRibbonUI\_onLoad(ribbon As IRibbonUI)

Set grxIRibbonUI = ribbon

End Sub

Sub rxshared\_click(control As IRibbonControl)

Select Case control.ID

Case “rxbtn”

glngCount1 = glngCount1 + 1

Case “rxbtn2″

glngCount2 = glngCount2 + 1

End Select

grxIRibbonUI.InvalidateControl (control.ID)

End Sub

Sub rxshared\_getLabel(control As IRibbonControl, ByRef returnedVal)

Select Case control.ID

Case “rxbtn”

returnedVal = “功能区无效: ” & glngCount1 & “次.”

glngCount1 = glngCount1 + 1

Case “rxbtn2″

returnedVal = “功能区无效: ” & glngCount2 & “次.”

glngCount2 = glngCount2 + 1

End Select

End Sub

此时，只会使单击的控件无效并统计其数量，如下图2所示。

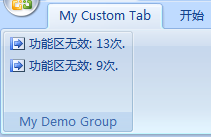
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/07/ribbonx15-2.png)

图2：仅使单个的控件对象无效

综上，本文主要介绍了自定义功能区的一个重要的内容，即通过回调添加功能。讲述了如何捕获和使用Ribbon对象以及如何通过单独的或共享的回调来组织过程。

[**细品RibbonX(16)：RibbonX开发基础问答**](http://www.excelperfect.com/index.php/2009/08/27/ribbonxstudy16/)

2009年08月27日, 5:16 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**2** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

在前面的一系列关于RibbonX的介绍中，我们一起认识了Office 2007的功能区界面，了解了RibbonX开发的基础知识，介绍了一些有用的资源，并重点讲解了RibbonX开发的工具以及步骤，关键的XML知识和回调基础。下面，我们以问答的形式对前面的一些主要知识进行小结。

**问题1 没有任何第三方工具，我如何创建自定义的选项卡？**

如果有第三方工具辅助，例如Custom UI Editor，那么自定义功能区是很方便的。不过没有也不要紧，只是稍微复杂一点罢了。

**步骤1** 打开Excel，输入数据后，使用xlsm扩展名保存该文件。

确保使用xlsm扩展名保存Excel文件，否则Excel会去除包含宏的文件。

**步骤2** 关闭Excel，然后在Windows中打开保存该文件的文件夹。

**步骤3** 在该文件夹中创建一个名为customUI的新文件夹。

**步骤4** 在这个新文件夹中添加一个名为customUI.xml的新文本文件。记住，该文本文件的扩展名为xml。

在customUI.xml文件中添加自定义选项卡的XML代码：

<?xml version=”1.0″ encoding=”UTF-8″ standalone=”yes” ?>

<customUI xmlns=”http://schemas.microsoft.com/office/2006/01/customui”>

<ribbon>

<tabs>

<tab id=”myTab” label=”我的选项卡” keytip=”MT”>

</tab>

</tabs>

</ribbon>

</customUI>

说明：上述代码中有一个keytip属性，代表用户按Alt键时选项卡出现的提示键，即用户按Alt+MT键就可以选择自定义的选项卡。在定义提示键时，选择的字符应避免与相同级别已有的提示键相冲突，例如，每个选项卡必须有唯一的提示键，然而其中的控件可以与选项卡有相同的提示键，只要它不与其它控件的提示键相同。

**步骤5** 将刚创建的Excel文件的扩展名改为ZIP。

**步骤6** 从ZIP文件中提取\_rels文件夹，并修改其中的.rels文件，用于告诉Excel存在customUI.xml文件。在.rels文件中添加下面的代码：

<Relationship Id=”rId4″ Type=”http://schemas.microsoft.com/office/2006/relationships/ui/extensibility” Target=”customUI/customUI.xml” />

必须提供唯一的Id属性值。Type属性必须指向自定义用户界面扩展的架构，因此使用如代码中显示的URL。Target属性指向自定义文件的位置。

**步骤7** 将\_rels文件夹放回到ZIP文件中。

**步骤8** 将customUI文件夹放到ZIP文件中。

**步骤9** 将ZIP文件的扩展名修改回Excel的扩展名。

此时，打开Excel文件后，将看到如下图1所示的自定义选项卡。该选项卡是空的，因为我们还没有在其中添加任何的组。注意，该选项卡显示了MT提示键。

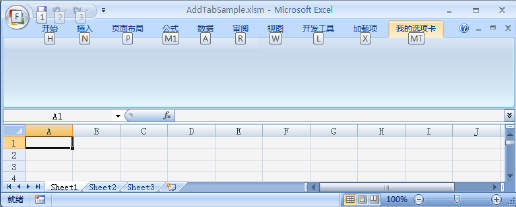
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/08/ribbonx16-1.png)

图1：添加空白选项卡

**问题2 如何在选项卡中添加组？**

在上文中，我们已经创建了一个新的选项卡。那么，在该选项卡中添加组也是相当容易的，只需要修改customUI.xml文件。此时，首先应该设计好布局，然后再添加相应的VBA代码。

下面是修改后在customUI.xml文件中添加的xml代码：

<tab id=”myTab” label=”我的选项卡” keytip=”MT”>

<group id=”Default” label=”默认组“>

</group>

<group id=”SeparatorGroup” label=”使用分隔条的组“>

<separator id=”SeparatorItem” visible=”true” />

</group>

<group id=”DialogBoxLaunch” label=”使用对话框启动器的组“>

<dialogBoxLauncher>

<button id=”LaunchDialog”

screentip=”对话框启动器“

supertip=”单击此按钮通常会显示一个对话框.”

keytip=”LD” />

</dialogBoxLauncher>

</group>

</tab>

上述代码在选项卡中添加了三个组。按上文所述步骤，将修改后的customUI.xml文件替换ZIP文件夹中原来的customUI.xml文件，然后将ZIP文件扩展名修改回xlsm扩展名，打开Excel文件，结果如下图2所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/08/ribbonx16-2.png)

图2：添加了三个组的选项卡

下面我们对上述代码进行简单的说明。

上述代码以标准的组开始，没有包括任何特定的功能。通常，可以在<group>和</group>标签之间放置控件。组能够使用下列属性：

         image（添加图像到组中）

         imageMso（添加内置图像到组中）

         keytip（为组提供加速键）

         screentip（为使用组显示一条简短的提示）

         supertip（为使用组显示一条较详细的提示）

然而，在通常情况下，我们不会使用这些额外的属性。事实上，由于组本身的特点，即使设置了这些属性，相应的特点可能也不会反映到界面中，因为Office忽略了这些属性。

除了在组中放置控件外，还有两个额外的子元素。<separator>元素在组中放置一条分隔条，提供控件之间的分隔，例如“视图”选项卡中的一些组，为我们提供了如何放置分隔条以提供更好的外观的示例。

一些组中还包括有对话框启动器，单击它将提供更多的功能。如上述代码所示，使用<dialogBoxLauncher>元素来提供对话框启动器。<dialogBoxLauncher>元素需要包括<button>控件，否则Office不会显示，并且<dialogBoxLauncher>元素必须包含一个且唯一一个<button>控件，不能使用任何其它的控件。

**问题3 使用记事本定制Ribbon太麻烦且易出错，有没有第三方工具可以简化定制Ribbon的过程？**

有。包括一款名为Office 2007 Custom UI Editor的工具和XML NotePad的工具，此外，还可以使用Visual Studio 2008来定制Ribbon。

**问题4 Office 2007 Custom UI Editor是什么？**

Office 2007 Custom UI Editor是一款小巧的用于自定义Office功能区的工具，使用该工具非常方便。不仅可以直接输入代码，还可以验证代码、添加图像，产生回调。但遗憾的是，不支持中文。

**问题5 如何使用Office 2007 Custom UI Editor自定义功能区？**

使用Office 2007 Custom UI Editor创建自定义功能区非常简单。仍以上述代码为例，简要地介绍其步骤：

**步骤1** 创建需要自定义功能区的Excel文件并保存。

**步骤2** 关闭Excel文件。

注意，在使用Custom UI Editor自定义功能区之前，必须首先关闭相应的Excel文件。

**步骤3** 打开Custom UI Editor。

**步骤4** 在Custom UI Editor中打开Excel文件。

**步骤5** 编写xml代码。

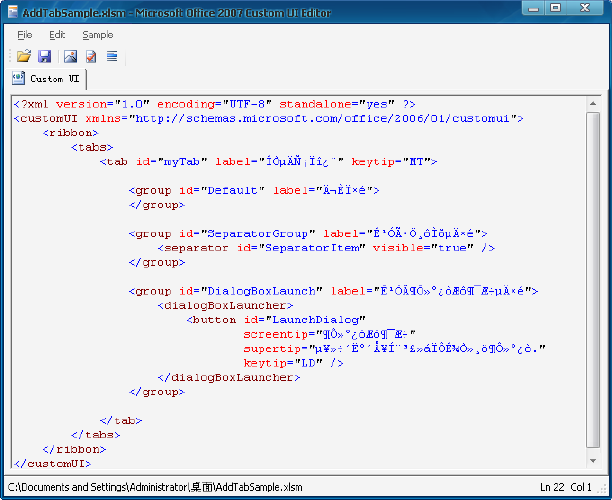
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/08/ribbonx16-3.png)

图3：在CustomUIEditor中输入代码

**步骤6** 保存并关闭Custom UI Editor

**步骤7** 打开Excel文件。此时，将出现自定义的功能区元素。

**问题6 我要在Office 2007中开发中文的选项卡和组，如何进行？**

可以采取使用记事本的方法，只不过在保存xml文件时，选择utf-8格式。

**问题7 在哪里可以获取Office图标？**

您可以从Microsoft的网站上搜索Office2007IconsGallery.EXE，下载该文件后安装。然后找到Office2007IconsGallery.xlsm文件并打开，此时在“开发工具”选项卡中将包含一个Office图标组，可以确定所有内置图标的名字。

您也可以从下列站点获得：

<http://www.excelperfect.com/wordpress/2009/05/06/ribbonxstudy8/>

**问题8 如何获取控件的标识符**

Microsoft提供了一种简单的方法很容易获得控件的标识符名。单击“Office按钮——选项”，调出应用程序选项对话框。在该对话框中，选择“自定义”选项卡。在“从下列位置选择命令”组合框中选择，然后将鼠标悬浮在下方列表框中的某个命名上，即可显示其标识符。然而，这仅对控件有用，如果需要知道选项卡和组的标识符，应在Microsoft的网站上下载专门的文档。

**问题9 如何添加自定义功能？**

在使用XML自定义功能区后，除了使用内置控件或者希望利用内置控件的功能外，还可以添加自定义功能。这需要使用回调，即使自定义的界面工作的VBA代码。

**（1）为含有组和控件的选项卡编写VBA代码**

最常见的任务之一是创建带有组和控件的选项卡，以添加自定义的功能。下面介绍如何处理“My Tab”选项卡的“My Group”组中名为“My Button”的按钮。

**步骤1 在Custom UI Editor中编写XML**

<?xml version=“1.0“ encoding=“UTF-8“ standalone=“yes“?>

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon>

<tabs>

<tab id=“myTab“ label=“My Tab“>

<group id=“myGroup“ label=“My Group“>

<button id=“myButton“

label=“My Button“

image=“Colorblk2“

size=“large“

onAction=“myButton\_ClickHandler“/>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

**步骤2 插入图标**

单击“Insert Icons”按钮，插入名为Colorblk2的图标。这里，使用的是自定义图像，因此使用image属性。

**步骤3 生成回调代码**

单击“Generate Callbacks”按钮，生成回调代码：

‘Callback for myButton onAction

Sub myButton\_ClickHandler(control as IRibbonControl)

End Sub

复制该段代码，并关闭Custom UI Editor。

**步骤4 编写VBA代码**

打开工作簿，切换到VBE，插入一个新模块，并粘贴代码。然后，在其中编写要实现的功能的代码。最后的代码如下：

‘Callback for myButton onAction

Sub myButton\_ClickHandler(control As IRibbonControl)

MsgBox control.ID + “被单击.”

End Sub

此时，保存Excel文件，切换到“My Tab”选项卡，单击“My Button”按钮，会出现包含该按钮标识符信息的消息框，如图4所示。

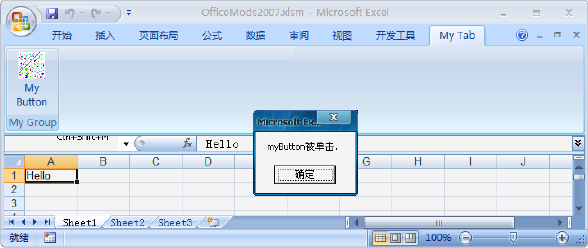
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/08/ribbonx16-4.png)

图4：单击自定义的按钮后执行相应的VBA代码的结果

**（2）修改已有的选项卡、组和控件**

只要知道相应的标识符，我们就能够修改已有的选项卡、组和控件。下面，首先在现有的选项卡中添加一个新组和控件，然后实现该控件的自定义操作。

在现有的选项卡中添加新组和控件的XML如下：

<tab idMso=“TabHome“>

<group id=“BehaviorChange“

label=“Behavior“

insertAfterMso=“GroupFont“>

<toggleButton id=“StopUnderline“

label=“Stop Underlining“

onAction=“StopUnderline\_ClickHandler“

getPressed=“StopUnderline\_GetPressed“

size=“large“

imageMso=“ShapeFillColorPicker“

insertBeforeMso=“UnderlineGallery“/>

</group>

</tab>

从上述代码可知，新组出现在“开始”选项卡中，其名称为“Behavior”，并且将该组插入到“字体”组之后，该组中包含一个大尺寸的切换按钮。必须为该切换按钮提供两个回调：onAction和getPressed。在单击切换按钮时，onAction回调执行与标准按钮相同的任务，而getPressed回调记录切换按钮的状态。该切换按钮使用了Office提供的图标。

界面如下图5所示：

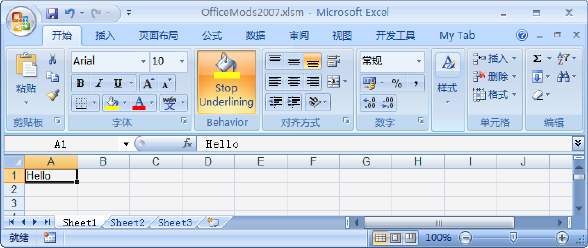
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/08/ribbonx16-5.png)

图5：在“开始”选项卡中添加新组和控件

（注意，在Custom UI Editor生成的回调中，不总是提供正确的参数，例如OnAction回调，其正确的参数为：

Sub OnAction(control as IRibbonControl,pressed as Boolean,ByRef fCancelDefault)

可以参考：

<http://msdn2.microsoft.com/en-us/library/ms406047.aspx>）

下面的XML用于重利用“下划线”按钮：

<commands>

<command idMso=“Underline“

onAction=“myUnderline“/>

</commands>

接着，在Custom UI Editor中生成回调代码并复制到VBE中，添加代码如下：

‘Define a global variable to hold the Ribbon reference.

Dim Rib As IRibbonUI

‘Determines the behavior button state.

Dim lBehavior As Boolean

‘Callback for customUI.onLoad

Sub RibbonLoaded(ribbon As IRibbonUI)

‘Save the ribbon reference.

Set Rib = ribbon

‘Initialize the behavior state.

lBehavior = False

‘Tell the user the Ribbon is loaded.

MsgBox “Ribbon Loaded”

End Sub

‘Callback for myUnderline onAction

Sub myUnderline(control As IRibbonControl, pressed As Boolean, ByRef fCancelDefault)

If (lBehavior) Then

MsgBox “No Underlined Allowed!”

pressed = False

fCancelDefault = True

Else

fCancelDefault = False

End If

End Sub

‘Callback for StopUnderline onAction

Sub StopUnderline\_ClickHandler(control As IRibbonControl, pressed As Boolean)

‘ Change the behavior state.

lBehavior = pressed

‘ Update the control.

Rib.InvalidateControl (control.ID)

End Sub

‘Callback for StopUnderline getPressed

Sub StopUnderline\_GetPressed(control As IRibbonControl, ByRef returnedVal)

‘ Return the current behavior state.

returnedVal = lBehavior

End Sub

上述代码在开始部分定义了一个变量来追踪应用程序的状态，以确保添加的控件反映正确的状态。StopUnderline\_ClickHandler()过程接受当前的控件及其按下的状态，代码在lBehavior中存储该控件的状态，然后使用InvalidateControl()重绘。通过返回lBehavior到功能区，StopUnderline\_GetPressed()过程执行显示当前切换按钮状态的处理。由于首次装载文档时功能区调用该过程，因此必须在RibbonLoaded()中提供lBehavior的默认值。本例中，当lBehavior为False时，myUnderline()执行默认的行为。当用户设置Stop Underlining按钮并且lBehavior为True时，代码显示一条消息告诉用户不允许添加下划线。

**（3）修改或重利用Office菜单**

也可以修改或重利用Office菜单项。下面的XML在Office菜单中添加项目：

<officeMenu>

<menu idMso=“FilePrepareMenu“>

<button id=“NewPrepButton“

label=“My Prepare Button“

description=“Prepare Time“

image=“TIME“

insertBeforeMso=“FileProperties“

onAction=“NewPrepButton\_ClickHandler“/>

</menu>

<splitButton idMso=“FileSaveAsMenu“>

<menu idMso=“FileSaveAsMenu“>

<button id=“SayHello“

label=“Say Hello“

description=“This button says hello!“

image=“Colorblk2“

onAction=“SayHello\_ClickHandler“/>

</menu>

</splitButton>

</officeMenu>

如上所示，Office菜单项出现在<officeMenu>元素中。本例中，代码在“准备”菜单里添加了一个名为“My Prepare Button”的新按钮，放置在菜单列表的顶部并使用了自定义图标。

应该注意到，“另存为”（FileSaveAsMenu）菜单以<splitButton>开始，因为它是一个拆分按钮。

接着，在Custom UI Editor中生成回调代码并复制到VBE中，添加代码如下：

‘ Callback for My Prepare Button on the

‘ Prepare menu of the Office menu.

Sub NewPrepButton\_ClickHandler(control As IRibbonControl)

‘ Display a message.

MsgBox “Are you prepared?”, vbYesNo

End Sub

‘ Provides support for the Say Hello Button on the

‘ Save As menu of the Office Menu.

Sub SayHello\_ClickHandler(control As IRibbonControl)

‘ Display a message.

MsgBox “Hello!”

End Sub

下面的XML用于重利用另存为“Excel 97-2003工作簿”命令：

<commands>

<command idMso=“FileSaveAsExcel97\_2003“

onAction=“FileSaveAs\_ClickHandler“/>

</commands>

下面是相应的VBA代码：

‘ Repurposes callback for the File Save As button.

Sub FileSaveAs\_ClickHandler(control As IRibbonControl, ByRef fCancelDefault)

‘ Holds the user response.

Dim Answer As VbMsgBoxResult

‘ Display a message.

Answer = MsgBox(“Saving as an older version. Are you sure?”, vbYesNo)

‘ Act on the response.

If Answer = vbYes Then

fCancelDefault = False

Else

fCancelDefault = True

End If

End Sub

注意参数fCancelDefault的使用。代码开始显示一条消息框，如果用户单击“是”，那么代码将fCancelDefault设置为False，表示显示“另存为”对话框；否则，代码将fCancelDefault设置为True，表示不做任何操作。

上述所有代码中，最重要的回调之一是功能区装载，即在<customUI>元素中使用onLoad属性：

<customUI onLoad=“RibbonLoaded“

xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

在工作簿打开时，将产生onLoad回调，查找VBA代码中的RibbonLoaded过程。该过程的代码如下：

‘Define a global variable to hold the Ribbon reference.

Dim Rib As IRibbonUI

‘Callback for customUI.onLoad

Sub RibbonLoaded(ribbon As IRibbonUI)

‘Save the ribbon reference.

Set Rib = ribbon

‘Initialize the behavior state.

lBehavior = False

‘Tell the user the Ribbon is loaded.

MsgBox “Ribbon Loaded”

End Sub

使用Invalidate方法告诉功能区进行重绘，显示移除、修改或添加的特征。

**问题10 可以从头开始设计功能区吗？**

可以。Microsoft提供了一个名为startFromScratch的属性，将其值设置为true，将移除现有的功能区界面。从头开始设计功能区可以限制用户访问您不希望其使用的功能。

只需在<ribbon>元素中添加startFromScratch属性并设置其值为true，即可从头开始设置功能区：

<ribbon startFromScratch=”true”>

**问题11 可以从功能区中调用自定义的用户窗体吗？**

可以，其实这也是取代自定义复杂的功能区的一种方式，您可以在用户窗体中实现一系列RibbonX控件的操作。而您只需要在功能区中添加一个按钮，用来调用用户窗体。

下面介绍一个显示和使用简单用户窗体的示例。

首先，在功能区中添加一个用于访问用户窗体的按钮。其XML代码如下：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon>

<tabs>

<tab idMso=“TabHome“>

<group id=“FormList“ label=“Forms“>

<button id=“ShowForm“

label=“Show Form“

imageMso=“HappyFace“

size=“large“

onAction=“ShowMyForm“ />

</group>

</tab>

</tabs>

</ribbon>

</customUI>

在工作簿中添加回调代码：

‘ Callback for ShowMyForm onAction

Sub ShowMyForm(control As IRibbonControl)

‘ Call the common processing routine.

ProcessForm

End Sub

其中，ProcessForm是一个过程的名称，该过程用于处理用户窗体的操作。

## [细品RibbonX(17)：RibbonX控件概述](http://www.excelperfect.com/index.php/2009/08/31/ribbonxstudy17/)

2009年08月31日, 1:20 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**2** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

在Office 2007中，可以使用比以前版本更多的控件，并且在大多数情况下也能改变控件尺寸。各种各样的控件选项意味着可以通过从RibbonX工具箱中选择正确的控件并正确地设置它们来创建更好的界面。下面，让我们先概要了解一下RibbonX控件。

**RibbonX控件概览**

在Office 2007中，控件不仅仅有特定的作用，而且有大小和上下文内容。例如，在大多数编程环境中按钮都是标准形式的按钮，然而在RibbonX中，可以创建大尺寸或者标准尺寸的按钮。此外，按钮的使用（以及在某些情况下的外观）随上下文而不同：

         功能区中的标准按钮

         对话框启动器的一部分

         下拉列表的一部分

         在快速访问工具栏中（QAT）

         拆分按钮的一部分

         带有标题的拆分按钮的一部分

         在控件组内

         菜单的一部分

上下文使得在如何使用控件方面有非常大的不同。例如，在对话框启动器里面必须包括一个按钮。Office限制下拉列表中按钮的数量是16，但不必包括任何任何内容。在拆分按钮里包括的按钮总是可见的，如果试图使用visible属性，Office就会激发错误。

下表提供了RibbonX控件列表及其简要的介绍。

|  |  |
| --- | --- |
| **控件名称** | **描 述** |
| 框（box） | 将控件组合在一个组里。可以在框里放置任何控件，将控件按水平或垂直流向设置。必须提供boxStyle属性：vertical或horizontal。与按钮组控件不同，除了设置控件流向外，框控件不会提供可视化外观。 |
| 按钮（button） | 提供基本的执行功能。单击按钮，会发生某项操作。上文已提及，Office在多种上下文内容中使用按钮控件。 |
| 按钮组（buttonGroup） | 将不同类型的按钮组合在一起。按钮出现在自然的框里面，并且Office将它们放置在一起来表明其以某种方式联系。可以使用其组合下列控件：按钮、切换按钮、库、菜单、动态菜单和拆分按钮控件。 |
| 复选框（checkBox） | 提供基本的选择功能。通过单击该控件，用户启用或禁用某选项。Office为复选框控件提供了两种上下文环境，包括作为单独的控件或者是菜单的一部分。 |
| 组合框（comboBox） | 为用户显示一列选项，使用item控件创建选项列表。每个comboBox控件必须包括至少一个item控件作为子项。当使用comboBox控件时，用户也能够输入一个不在列表中出现的值（dropDown控件需要用户选择列表中的某个项目）。 |
| 下拉控件（dropDown） | 为用户为示一列选项，使用item或button控件创建选项列表。列表至少应包含两类可选的控件之一。用户必须选择提供的列表中的某选项。当用户选择button而不是item时，Office执行所需求的动作，而不是选择期望的选项。 |
| 动态菜单（dynamicMenu） | 定义在运行时而不是设计时所创建的菜单。菜单内容能够修改以满足特定的需要。必须包括getContent回调来使用该控件。dynamicMenu控件能够作为buttonGroup、menu或splitButton控件的一部分出现。 |
| 编辑框（editBox） | 让用户输入纯文本到功能区中。可以使用这项功能执行某任务，例如搜索。使用这个控件来进行任何不能通过使用其它控件来定义的输入。 |
| 库（gallery） | 在下拉结构中显示一组控件以节省功能区空间。Word在“开始”选项卡的“样式”组中使用这样的组。gallery控件不同于其它的组合控件，它提供了可以以不同方式控制的下拉列表。可以使用rows和columns属性修改控件的外观。itemWidth和itemHeight属性用来控制组中每项的大小。在buttonGroup或menu控件里使用Gallery控件，或者作为独立的控件。要在库中显示项目，则需在getItemCount、getItemImage和getItemLabel回调中添加代码。 |
| 标签控件（labelControl） | 在屏幕中创建标签。可以在标签控件组或者不易使其以其它形式识别的元素中使用该控件。用户不能够与提供的标签相交互。 |
| 菜单（menu） | 定义在设计时创建的菜单。菜单可以包含诸如button和checkBox控件之类的控件。可以单独使用菜单，也可以作为splitButton控件的一部分。使用menuSeparator控件在菜单元素之间放置分隔条。与gallery控件不同，menu控件在单列呈现所有的选项（更像老版本Office的菜单系统）。 |
| 菜单分隔条（menuSeparator） | 在任何控件组里提供分隔元素。 |
| 拆分按钮（splitButton） | 创建具有默认操作和一列选择性选项的按钮。拆分按钮最好的示例之一是“开始”选项卡的“剪贴板”组中的“粘贴”按钮。必须包括button或toggleButton控件为默认控件。在menu控件里出现可选的操作，这里可以添加button或toggleButton控件。 |
| 切换按钮（toggleButton） | 提供checkBox和button控件的组合。用户通过单击toggleButton选择某状态和执行某操作。 |

**通用的RibbonX控件属性**

RibbonX提供了许多通用的属性，可以用来控制应用程序的外观。下表简要介绍了大多数通用的属性。

|  |  |
| --- | --- |
| **属性** | **描 述** |
| description | 当菜单的itemSize属性设置为大时指定Office显示的描述文本。 |
| enabled | 确定Office是否启用或禁用控件。可以选择true或false作为其值。 |
| id | 为自定义控件指定标识符。不能与idMso或idQ属性一起使用。 |
| idMso | 为内置控件指定标识符。不能与id或idQ属性一起使用。 |
| idQ | 为限定的控件指定标识符（限定的控件取决于定义的自定义命名空间）。不能与id或idMso属性一起使用。 |
| image | 定义控件使用的自定义图像的名称。 |
| imageMso | 定义控件使用的内置图像的名称。 |
| insertAfterMso | 基于内置控件的位置，标识新控件的位置。 |
| insertAfterQ | 基于限定的控件的位置，标识新控件的位置。 |
| insertBeforeMso | 基于内置控件的位置，标识新控件的位置。 |
| insertBeforeQ | 基于限定的控件的位置，标识新控件的位置。 |
| itemSize | 确定菜单或其它控件项的大小。可以选择large和normal。 |
| keyTip | 对控件添加指定的加速键组合。当用户按下Alt键时显示键提示。可以指定任意键组合，使用1至3个字母。 |
| label | 指定控件显示的文本。 |
| screenTip | 提供简要的提示，帮助用户理解控件的用途。 |
| showImage | 确定Office是否显示与控件项相关的图像。可以选择true或false。 |
| showItemImage | 确定Office是否显示与菜单或其它控件项相关的图像。可以选择true或false。 |
| showItemLabel | 确定Office是否显示与菜单或其它控件项相前的标签。可以选择true或false。 |
| showLabel | 确定Office是否显示与控件相关的标签。可以选择true或false。 |
| size | 确定控件的大小。可以选择large或normal。 |
| sizeString | 设置控件的大小来包含指定宽度的字符串。 |
| superTip | 提供详细的提示，帮助用户理解控件的用途。 |
| tag | 包含用户定义的数据，可以在应用程序中用来与控件或其它元素交互。 |
| title | 为menuSeparator控件指定显示的文本代替水平线。 |
| visible | 确定Office是否显示某控件或其它功能。可以选择true或false。 |

**通用的RibbonX控件回调**

RibbonX提供了许多通用的回调，可用于监控应用程序。下表介绍了创建应用程序时常用的回调。

|  |  |  |
| --- | --- | --- |
| **回调** | **相关的控件** | **描述** |
| getContent | dynamicMenu | 定义控件的内容。 |
| getDescription | 多个控件 | 获取控件的描述。 |
| getEnabled | 多个控件 | 让代码启用或禁用控件。 |
| getImage | 多个控件 | 获取为控件提供的自定义图像。 |
| getImageMso | 多个控件 | 获取为控件定义的标准图像。 |
| getItemCount | comboBox，DropDown，gallery | 获取控件的项目列表中的项目数。 |
| getItemHeight | gallery | 以像素为单位确定项目显示在屏幕中的高度。 |
| getItemID | comboBox，dropDown，gallery | 确定当前项目的ID。 |
| getItemImage | comboBox，dropDown，gallery | 获取与当前项目相关的图像。 |
| getItemLabel | comboBox，dropDown，gallery | 获取与当前项目相关的标签。 |
| getItemScreenTip | comboBox，dropDown，gallery | 获取与当前项目相关的屏幕提示。 |
| getItemSuperTip | comboBox，dropDown，gallery | 获取与当前项目相关的超级提示。 |
| getItemWidth | gallery | 以像素为单位确定项目显示在屏幕中的宽度。 |
| getKeytip | 多个控件 | 获取控件的键提示（加速键）。 |
| getLabel | 多个控件 | 获取控件的标签。 |
| getPressed | checkBox，toggleButton | 确定用户是否以手动的方式单击了控件以激活该控件。 |
| getScreentip | 多个控件 | 获取控件的屏幕提示。 |
| getSelectedItemIndex | dropDown，gallery | 确定用户从列表中选取的项目。 |
| getSelectItemID | gallery | 获取用户已经在列表中选取的项目的ID。 |
| getShowImage | button | 确定控件是否显示图像（允许禁止图像，即便已经为控件定义了图像）。 |
| getShowLabel | button | 确定控件是否显示标签（允许禁止标签，即便已经为控件定义了标签）。 |
| getSize | 多个控件 | 基于应用程序的输出，定义控件的大小。 |
| getSupertip | 多个控件 | 获取控件的超级工具提示。 |
| getText | comboxBox，editBox | 获取与列表中当前所选择的项目相关的文本。 |
| getTitle | menuSeparator | 为指定的控件提供标题。Office显示文本来代替通常显示的水平线。 |
| getVisible | button | 确定是否控件可见。 |
| LoadImage | customUI | 作为整体装载与用户界面相关的图像。 |
| onAction | 多个控件 | 执行控件在应用程序代码中定义的操作。 |
| onChange | comboBox，editBox | 在用户选择或控件内容中发现变化。 |
| onLoad | customUI | 在装载过程中执行指定的操作。 |
|  |  |  |
|  |  |  |

[**细品RibbonX(18)：button控件详解**](http://www.excelperfect.com/index.php/2009/08/31/ribbonxstudy18/)

2009年08月31日, 7:37 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**3** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

在功能区的按钮中，可以使用内置图像或提供自已的图像，可以指定大按钮或者更小的形式，添加少量的代码甚至可以同时提供标签。此外，可以利用大量的回调过程从而使按钮为动态的。按钮提供了几种用于定制外观和响应的属性。某些属性是必需的，另外一些是可选的。

在创建按钮时，下面所列出的表可以提供良好的参考。

**1、button元素必需的属性**

按钮需要下表1所列的id属性之一。

表1：button元素必需的属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的按钮时 |
| idMso | 当使用现有的Microsoft按钮时 |
| idQ | 当在命名空间之间创建共享的按钮时 |

每个按钮也需要表2所列出的onAction回调。

表2：button元素必需的回调

|  |  |  |
| --- | --- | --- |
| **动态属性** | **允许值** | **VBA回调签名** |
| onAction | 1至4096个字符 | Sub OnAction(control As IRibbonControl) |
| onAction | 重新使用（或重利用） | Sub OnAction(control As IRibbonControl,  byRef CancelDefaultcancelDefault) |

**2、带有回调签名的可选的静态属性和动态属性**

使用按钮时，可以使用下表3列出的任一insert属性。

表3：button元素可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **INSERT属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之后插入 |
| insertBeforeMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之前插入 |
| insertAfterQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之后插入 |
| insertBeforeQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之前插入 |

也可以提供下表4所列的任何或所有的属性。

表4：button元素可选的属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| description | getDescription | 1至4096个字符 | （none） | Sub GetDescription(control As IRibbonControl,ByRef returnedVal) |
| enabled | getEnabled | true,false,1,0 | true | Sub GetEnabled(control As IRibbonControl,ByRef returnedVal) |
| image | getImage | 1至1024个字符 | （none） | Sub GetImage(control As IRibbonControl,ByRef returnedVal) |
| imageMso | getImage | 1至1024个字符 | （none） | 同上 |
| keytip | getKeytip | 1至3个字符 | （none） | Sub GetKeytip(control As IRibbonControl,ByRef returnedVal) |
| label | getLabel | 1至1024个字符 | （none） | Sub GetLabel(control As  IRibbonControl,ByRef returnedVal) |
| screentip | getScreentip | 1至1024个字符 | （none） | Sub GetScreentip(control As IRibbonControl,ByRef returnedVal) |
| showImage | getShowImage | true,false,1,0 | true | Sub GetShowImage(control As IRibbonControl,ByRef returnedVal) |
| showLabel | getShowLabel | true,false,1,0 | true | Sub GetShowLabel(control As IRibbonControl,ByRef returnedVal) |
| size | getSize | normal,large | normal | Sub GetSize(control As IRibbonControl,ByRef returnedVal) |
| supertip | getSupertip | 1至1024个字符 | (none) | Sub GetSupertip(control As IRibbonControl,ByRef returnedVal) |
| tag | (none) | 1至1024个字符 | (none) | (none) |
| visible | getVisible | true,false,1,0 | true | Sub GetVisible(control As IRibbonControl,ByRef returnedVal) |

**3、button元素允许的子对象**

button控件不支持任何子对象。

**4、button元素的父对象**

在下列任何控件内都能放置button控件：

         box

         buttonGroup

         dialogBoxLauncher

         documentControls

         dynamicMenu

         gallery

         group

         menu

         splitButton

         officeMenu

**5、使用内置的按钮控件**

（1）创建一个新的.xlsx文件，并将其保存为Excel Built In Button Example.xlsx。

（2）关闭该文件并在CustomUI Editor中打开。

（3）应用RibbonBase模板。

（4）在<tabs>和</tabs>之间插入下列XML：

<tab id=“rxtabCustom1“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“DemoGroup“

label=“Demo Group“>

<button idMso=“AccountingFormat“/>

</group>

</tab>

下图为上述代码创建的组。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/08/builtinbutton1.jpg)

注意，当使用内置控件时，获得完整的包装：图像、标签、预设的回调，等等。因此，如果仅希望有图像和功能，则除了期望的idMso，需要设置showLabel属性为false。此时，简单地修改代码如下：

<tab id=“rxtabCustom1“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“DemoGroup“

label=“Demo Group“>

<button idMso=“AccountingFormat“

showLabel=“false“/>

</group>

</tab>

现在将创建仅显示图像的一个漂亮按钮，没有标签，如下图所示，当然它仍具有该内置控件的所有其他属性。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/08/builtinbutton2.jpg)

**6、按钮的特性：showLabel属性**

假设您希望上文中的按钮尺寸更大一些，则可以修改相应的代码如下：

<button idMso=“AccountingFormat“

showLabel=“false“

size=“large“/>

结果如下图所示：

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/08/builtinbutton3.jpg)

代码中没有让标签显示，结果却出现了标签？这是按钮模型的一个奇怪的特性。在size=”normal”情况下正常工作的按钮，在size=”large”时却不按要求工作。如上所见，当按钮是大尺寸时，默认值覆盖了自定义的设置。

如何达到要求呢？要想在大尺寸时不显示标签，需要将标签文本指定为空格（必须要有一个空格）。如下所示：

<group id=“DemoGroup“

label=“Demo Group“>

<button idMso=“AccountingFormat“

label=“ “

size=“large“/>

</group>

结果如下图：

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/08/builtinbutton4.jpg)

**7、创建自定义按钮控件**

即添加自已的按钮，单击该按钮时，会触发自已编写的宏代码实现自定义的功能。

本例中，假设你是一名会计师，并且已经为支付的费用创建了一份连续的清单，如下图所示，

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/08/builtinbutton5.jpg)

现在需要编写宏代码以每月为基础来准备下月的工作表，特别是要完成下述操作：

（1）复制Ending Balance到Openning Balance列。

（2）清除Additions和Usage区（本月的条目）。

（3）预先填写到下月结束的日期。

（4）指导用户保存文件。

因此，在标准模块中编写下列代码：

Public Sub RollForward()

With ActiveSheet

.Range(“E6:E10″).Copy

.Range(“B6:B10″).PasteSpecial Paste:=xlValues

.Range(“C6:D10″).ClearContents

.Range(“A3″) = .Range(“A3″).Value + 40 – DatePart(“d”, .Range(“A3″).Value + 40)

End With

MsgBox “Rolled forward successfully!” & vbCrLf & \_

“Please save the file under a new name.”, \_

vbOKOnly + vbInformation, “Success!”

End Sub

可以按Alt+F8键，在“宏”对话框中选择RollForward来测试。

下面，我们将该宏附加到按钮中。首先，保存Excel文件并关闭。然后，在CustomUI Editor中打开该文件，输入下面的XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“DemoTab“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“DemoGroup“

label=“Demo Group“>

<button id=“rxbtnRollForward“

label=“Roll Forward“

imageMso=“CreateReportFromWizard“

size=“large“

onAction=“rxbtnRollForward\_Click“/>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

单击“Generate Callbacks”生成回调，并复制生成的代码。保存并关闭CustomUI Editor，然后重新打开Excel。在VBE中粘贴从CustomUI Editor中复制的代码，然后添加代码来调用上面已经测试过的VBA过程。代码如下所示：

‘Callback for rxbtnRollForward onAction

Sub rxbtnRollForward\_Click(control As IRibbonControl)

Select Case control.ID

Case Is = “rxbtnRollForward”

Call RollForward

Case Else

‘do nothing

End Select

End Sub

保存后，回到Excel界面，再测试按钮。

[**细品RibbonX(19)：checkBox控件详解**](http://www.excelperfect.com/index.php/2009/09/28/ribbonxstudy19/)

2009年09月28日, 12:39 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**2** 人投票, 平均: **4.50** out of 5)

Loading ... Loading ...

checkBox控件能够使用户在两种状态之间切换。虽然默认情况下状态是真和假，但是该元素能够指明开/关、上/下、左/右、1/0、或者任何可以想象到的相对状态组合。

您可能想使用checkBox来做下列操作：

        表明数据库中的某字段是否满足特定的条件。在这种情况下，当满足条件时，将自动选中复选框。

        允许用户确定是否显示某对象，例如显示或隐藏网格线，或者子窗体。

**1、checkBox元素必需的属性**

复选框控件需要下表1所列的id属性之一。

表1：checkBox元素必需的属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的复选框时 |
| idMso | 当使用现有的Microsoft复选框时 |
| idQ | 当在命名空间之间创建共享的复选框时 |

每个复选框控件也需要表2所列出的onAction回调。

表2：checkBox元素必需的回调

|  |  |  |
| --- | --- | --- |
| **动态属性** | **允许值** | **VBA回调签名** |
| onAction | 1至4096个字符 | Sub OnAction(control As IRibbonControl,  pressed as Boolean) |

**2、带有回调签名的可选的静态属性和动态属性**

复选框控件可以使用下表3列出的任一insert属性。

表3：checkBox元素可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **INSERT属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之后插入 |
| insertBeforeMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之前插入 |
| insertAfterQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之后插入 |
| insertBeforeQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之前插入 |

也可以提供下表4所列的任何或所有的属性。

表4：checkBox元素可选的属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| description | getDescription | 1至4096个字符 | （none） | Sub GetDescription(control As IRibbonControl,ByRef returnedVal) |
| enabled | getEnabled | true,false,1,0 | true | Sub GetEnabled(control As  IRibbonControl,  ByRef returnedVal) |
| keytip | getKeytip | 1至3个字符 | （none） | Sub GetKeytip(control As IRibbonControl,  ByRef returnedVal) |
| label | getLabel | 1至1024个字符 | （none） | Sub GetLabel(control As  IRibbonControl,  ByRef returnedVal) |
| （none） | getPressed | true,false,1,0 | false | Sub GetPressed(control As IRibbonControl,  ByRef returnedVal) |
| screentip | getScreentip | 1至1024个字符 | （none） | Sub GetScreentip(control As IRibbonControl,  ByRef returnedVal) |
| supertip | getSupertip | 1至1024个字符 | (none) | Sub GetSupertip(control As IRibbonControl,  ByRef returnedVal) |
| tag | (none) | 1至1024个字符 | (none) | (none) |
| visible | getVisible | true,false,1,0 | true | Sub GetVisible(control As IRibbonControl,  ByRef returnedVal) |

**3、checkBox元素允许的子对象**

checkBox控件不支持任何子对象。

**4、checkBox元素的父对象**

在下列任何控件内都能放置checkBox控件：

         box

         dynamicMenu

         group

         menu

         officeMenu

**5、使用内置的复选框控件**

（1）创建一个新的.xlsx文件，并将其保存为Excel Built In CheckBox Example.xlsx。

（2）关闭该文件并在CustomUI Editor中打开该文件。

（3）输入下列XML：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“rxgrpDemo“

label=“Demo Group“>

<checkBox idMso=“GridlinesExcel“/>

</group>

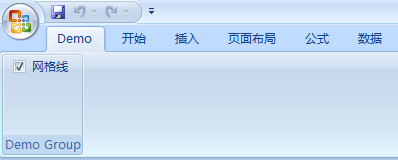
</tab>

</tabs>

</ribbon>

</customUI>

下图为上述代码创建的组：

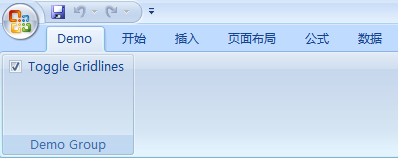
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/09/checkboxsample1.png)

您也可以修改该控件的名字，使用下面的XML代替相应的语句：

<checkBox idMso=“GridlinesExcel“

label=“Toggle Gridlines“/>

此时，显示的界面如下图所示：

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/09/checkboxsample2.png)

**6、创建自定义复选框控件**

使用Excel时，便利的事情之一是迅速在工作簿中切换A1和R1C1样式公式的能力。在一些情形下，使用R1C1引用的公式将更好。下面的示例演示如何容易地快速切换两种引用形式的公式。

本示例在“公式”选项卡中添加一个控件，省去了必须通过Office菜单来找到该设置的复选框的麻烦。如下图所示，添加的新复选框控件在最右侧的“Other Settings”组中。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/09/checkboxsample3.png)

首先，新建一个工作簿，保存后关闭。然后使用CustomUI Editor输入下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“ onLoad=“rxIRibbonUI\_onLoad“>

<ribbon startFromScratch=“false“>

<tabs>

<tab idMso=“TabFormulas“>

<group id=“rxgrpOtherSettings“

label=“Other Setting“

insertBeforeMso=“GroupNamedCells“>

<checkBox id=“rxchkR1C1“

label=“R1C1 Formulas“

getPressed=“rxchkR1C1\_getPressed“

onAction=“rxchkR1C1\_click“/>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

注意，在XML代码的第一行，添加了onLoad属性，以便后面使用时捕获Ribbon对象。

单击“Generate Callbacks”按钮，复制生成的代码。关闭CustomUI Editor。重新打开Excel文件，并打开VBE，将复制的代码粘贴到标准模块中。

现在，该文件中有三个回调签名：rxIRibbonUI\_onLoad，rxchkR1C1\_getPressed，rxchkR1C1\_click，其目的是：

         rxIRibbonUI\_onLoad存储RibbonUI对象，能够使控件无效以便随后强制更新控件。

         当第一次激活“公式”选项卡（或使无效）并适当地设置复选框时触发rxchkR1C1\_getPressed。

         无论何时选中或取消选中复选框都将触发rxchkR1C1\_click，其目的实际上是切换开关设置。

此外，rxIRibbonUI\_onLoad需要自定义工作簿属性来操作，因此应该首先对其设置。在该工程的ThisWorkbook模块中，输入下列代码：

‘保存功能区和功能区控件状态的私有变量

Private pRibbonUI As IRibbonUI

Public Property Let RibbonUI(iRib As IRibbonUI)

‘设置RibbonUI属性以便后面使用

Set pRibbonUI = iRib

End Property

Public Property Get RibbonUI() As IRibbonUI

‘获取RibbonUI属性以供使用

Set RibbonUI = pRibbonUI

End Property

转到标准模块中，添加下列代码：

‘customUI.onLoad回调

Sub rxIRibbonUI\_onLoad(ribbon As IRibbonUI)

‘设置RibbonUI为工作簿属性以供后面使用

ThisWorkbook.RibbonUI = ribbon

End Sub

此时，当装载该工作簿时，将捕获RibbonUI对象，并将可用于使之无效。

接下来，生成切换设置的宏。最容易的方法是先录制实现该功能的宏：打开宏录制器，选择“Office菜单—Excel选项—公式”，选中“R1C1引用样式”前的复选框。停止录制，转到VBE中查看录制的代码：

Application.ReferenceStyle = xlR1C1

这就是宏录制器能够帮助编写代码的一种情形，告诉确切地告诉你需要引用的对象。虽然仍需要对代码作一些修改，但至少知道从哪里开始以及正确的语法。

再在处理getPressed回调。主要是确定应用程序是否是R1C1模式，如果是则返回true。代码如下：

‘rxchkR1C1 getPressed回调

Sub rxchkR1C1\_getPressed(control As IRibbonControl, ByRef returnedVal)

If Application.ReferenceStyle = xlR1C1 Then returnedVal = True

End Sub

下一步，处理单击复选框的回调。pressed参数告诉是否复选框被选中（pressed=true）或没有选中（pressed=false）。代码如下：

‘rxchkR1C1 onAction回调

Sub rxchkR1C1\_click(control As IRibbonControl, pressed As Boolean)

Select Case pressed

Case True

Application.ReferenceStyle = xlR1C1

Case False

Application.ReferenceStyle = xlA1

End Select

End Sub

如果单击“公式”选项卡，将触发getPressed程序，设置复选框来表明当前所处的显示模式。否则，选择复选框来设置其为相反的状态。

但仍存在问题。假设有人通过“Excel选项”修改了复选框的值，此时选项卡中的复选框将不会被更新。

为解决这个问题，可以在无论何时激活新工作表时，强制更新该复选框。此时，使用使特定的控件（即rxchkR1C1 checkBox）无效的能力。

再转到ThisWorkbook代码模块，从代码窗口左侧下拉列表中选择Workbook，然后从右侧下拉列表中选择Sheet\_Activate，添加代码如下：

Private Sub Workbook\_SheetActivate(ByVal Sh As Object)

‘每次激活一个工作表时使该选项卡无效

ThisWorkbook.RibbonUI.InvalidateControl (“rxchkR1C1″)

End Sub

好了！保存并关闭后重新打开工作簿，然后输入公式来测试。

当然，在通过“Excel选项”设置R1C1引用样式后不会立即更新，您需要激活另一工作表后再返回该工作表，即可更新了。

## [细品RibbonX(20)：editBox控件详解](http://www.excelperfect.com/index.php/2009/09/29/ribbonxstudy20/)

2009年09月29日, 2:08 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

editBox控件允许用户输入文本，这为我们提供了一种允许用户输入的方便的方法。例如，可以使用editBox重命名Excel工作表。基本的概念是要求用户在editBox中输入，然后当其失去焦点且下一个控件激活之前发生事件。当然，也可以提供一个按钮，当单击该按钮后处理eidtBox控件中的内容。

**1、editBox元素必需的属性**

编辑框控件需要下表1所列的id属性之一。

表1：editBox元素必需的属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的编辑框时 |
| idMso | 当使用现有的Microsoft编辑框时 |
| idQ | 当在命名空间之间创建共享的编辑框时 |

编辑框控件也需要表2所列出的onAction回调。

表2：editBox元素必需的回调

|  |  |  |
| --- | --- | --- |
| **动态属性** | **允许值** | **VBA回调签名** |
| onChange | 1至4096个字符 | Sub OnChange(control As IRibbonControl,  text as String) |

**2、带有回调签名的可选的静态属性和动态属性**

编辑框控件可以使用下表3列出的任一insert属性。

表3：editBox元素可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **INSERT属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之后插入 |
| insertBeforeMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之前插入 |
| insertAfterQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之后插入 |
| insertBeforeQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之前插入 |

也可以提供下表4所列的任何或所有的属性。

表4：editBox元素可选的属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| enabled | getEnabled | true,false,1,0 | true | Sub GetEnabled(control As IRibbonControl,  ByRef returnedVal) |
| image | getImage | 1至1024个字符 | （none） | Sub GetImage(control As IRibbonControl,  ByRef returnedVal) |
| imageMso | getImage | 1至1024个字符 | （none） | 同上 |
| keytip | getKeytip | 1至3个字符 | （none） | Sub GetKeytip(control As IRibbonControl,  ByRef returnedVal) |
| label | getLabel | 1至1024个字符 | （none） | Sub GetLabel(control As  IRibbonControl,  ByRef returnedVal) |
| maxLength | （none） | 1至1024个字符 | 1024 | （none） |
| screentip | getScreentip | 1至1024个字符 | （none） | Sub GetScreentip(control As IRibbonControl,  ByRef returnedVal) |
| showImage | getShowImage | true,false,1,0 | true | Sub GetShowImage(control As IRibbonControl,  ByRef returnedVal) |
| showLabel | getShowLabel | true,false,1,0 | true | Sub GetShowLabel(control As IRibbonControl,  ByRef returenedVal) |
| sizeString | （none） | 1至1024个字符 | 默认值为12 | （none） |
| supertip | getSupertip | 1至1024个字符 | (none) | Sub GetSupertip(control As IRibbonControl,  ByRef returnedVal) |
| tag | (none) | 1至1024个字符 | (none) | (none) |
| （none） | getText | 1至4096个字符 | （none） | Sub GetText(control As IRibbonControl,  ByRef returnedVal) |
| visible | getVisible | true,false,1,0 | true | Sub GetVisible(control As IRibbonControl,  ByRef returnedVal) |

**3、editBox元素允许的子对象**

editBox控件不支持任何子对象。

**4、editBox元素的父对象**

仅能够在下列两个控件内放置editBox控件：

         box

         group

**5、使用内置的编辑框控件**

很有趣，没有方法自定义任何内置的editBox控件。

**6、创建自定义编辑框控件**

如前所述，编辑框为获取用户输入提供了一种有效的方式。下面的示例介绍一些常见的使用，例如修改文件或工作表的名字。

创建一个新的Excel工作簿并以.xlsm格式保存。打开VBE，插入一个标准模块，在其中创建到将来要对编辑框进行操作的宏代码。代码如下：

Private Function shtRename(sCallSheet As String) As Boolean

On Error Resume Next

ActiveSheet.Name = sCallSheet

If Err.Number = 0 Then shtRename = True

End Function

Public Sub RenameSheet()

Dim sNewSheetName As String

sNewSheetName = InputBox(“请为工作表输入一个新名字.”)

If shtRename(sNewSheetName) = False Then

MsgBox “发生问题了,不能够” & vbCrLf & \_

“重命名工作表.请再试.”, \_

vbOKOnly + vbCritical, “错误!”

End If

End Sub

下面简要解释上述两个过程。shtRename是一个用于重命名工作表的自定义函数。On Error语句在遇到错误时允许程序继续执行。该函数通过测试是否遇到错误，并返回真/假值表明是否重命名成功。

RenameSheet调用shtRename函数并测试是否成功，如果没有成功则通知用户。

要测试代码，运行RenameSheet过程，出现提示时输入名字，此时工作表将被重命名。如果提供了一个无效的参数，例如已存在的工作表字，那么将会返回错误消息。

现在，关闭Excel工作簿，再CustomUI Editor中打开该工作簿，输入下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtabCustom“

label=“My Tools“

insertBeforeMso=“TabHome“>

<group id=“rxgrpCustom“

label=“Worksheet“>

<editBox id=“rxtxtRename“

label=“Rename Sheet to:“

imageMso=“SignatureLineInsert“

keytip=“R“

sizeString=“123456789012345“

onChange=“rxtxtRename\_Click“/>

</group>

</tab>

</tabs>

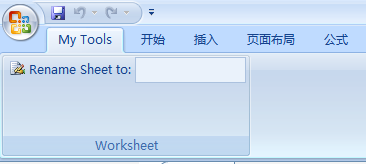
</ribbon>

</customUI>

**技巧：**在使用sizeString属性时，使用增加的数字输入字符数便于迅速判断允许的字符数。本例中，“123456789012345”是15个字符。

接下来，生成回调签名并复制代码，保存后关闭CustomUI Editor。

在Excel中重新打开工作簿，在界面中应该看到一个带有编辑框的自定义选项卡，如下图所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/09/editboxsample1.png)

由于没有实现操作的回调代码，因此该编辑框不做任何操作。打开VBE，在标准模块中粘贴回调签名，并输入下列代码：

‘rxtxtRename onChange回调

Sub rxtxtRename\_Click(control As IRibbonControl, text As String)

If shtRename(text) = False Then

MsgBox “发生问题了,不能够” & vbCrLf & \_

“重命名工作表.请再试.”, \_

vbOKOnly + vbCritical, “错误!”

End If

End Sub

好了！现在可以测试编辑框了。回到Excel界面，在自定义的编辑框中输入新的名字，查看工作表标签是否已被更新。

下面，再演示一个例子：在编辑框中输入数字后，当前单元格所在的列宽将作相应的调整。在CustomUI Editor中打开上例所示的工作簿，在其中添加下面的XML代码：

<group id=“rxgrpCustom1“

label=“Change Column Width“>

<editBox id=“rxtxtWidth“

label=“Width:“

sizeString=“1234567890“

onChange=“rxtxtWidth\_Change“/>

</group>

生成回调签名并复制后，关闭CustomUI Editor。在Excel中打开该工作簿，在VBE中粘贴回调签名，并输入代码：

Dim dblWidth As Double

‘rxtxtWidth onChange回调

Sub rxtxtWidth\_Change(control As IRibbonControl, text As String)

On Error Resume Next

dblWidth = CDbl(text)

If Err.Number <> 0 Then

MsgBox “对不起,必须输入数字值!”

text = “”

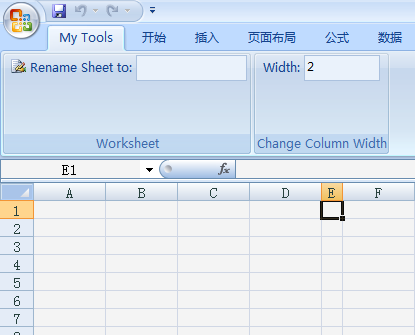
Else

ActiveCell.ColumnWidth = dblWidth

End If

End Sub

如下图所示，在编辑框中输入数值并回车后，当前单元格所在的列宽将根据数值进行调整。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/09/editboxsample2.png)

[**细品RibbonX(21)：toggleButton控件详解**](http://www.excelperfect.com/index.php/2009/09/30/ribbonxstudy21/)

2009年09月30日, 1:53 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

切换按钮用于在两种状态之间选择，例如当按下某切换按钮时开启或关闭某功能。

**1、toggleButton元素必需的属性**

切换按钮需要下表1所列的id属性之一。

表1：toggleButton元素必需的属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的切换按钮时 |
| idMso | 当使用现有的Microsoft切换按钮时 |
| idQ | 当在命名空间之间创建共享的切换按钮时 |

切换按钮也需要表2所列出的onAction回调。

表2：toggleButton元素必需的回调

|  |  |  |
| --- | --- | --- |
| **动态属性** | **允许值** | **VBA回调签名** |
| onAction | 1至4096个字符 | Sub OnAction(control As IRibbonControl,  selectedId As String,  selectedIndex As Integer) |

**2、带有回调签名的可选的静态属性和动态属性**

使用切换按钮时，可以使用下表3列出的任一insert属性。

表3：toggleButton元素可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **INSERT属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之后插入 |
| insertBeforeMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之前插入 |
| insertAfterQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之后插入 |
| insertBeforeQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之前插入 |

也可以提供下表4所列的任何或所有的属性。

表4：toggleButton元素可选的属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| description | getDescription | 1至4096个字符 | （none） | Sub GetDescription(control  As IRibbonControl,  ByRef returnedVal) |
| enabled | getEnabled | true,false,1,0 | true | Sub GetEnabled(control As  IRibbonControl,  ByRef returnedVal) |
| image | getImage | 1至4096个字符 | （none） | Sub GetImage(control As  IRibbonControl,  ByRef returnedVal) |
| imageMso | getImage | 1至4096个字符 | （none） | 同上 |
| keytip | getKeytip | 1至3个字符 | （none） | Sub GetKeytip(control As  IRibbonControl,  ByRef returnedVal) |
| label | getLabel | 1至4096个字符 | （none） | Sub GetLabel(control As  IRibbonControl,  ByRef returnedVal) |
| （none） | getPressed | true,false,1,0 | （none） | Sub GetPressed(control As  IRibbonControl,  ByRef returnedVal) |
| screentip | getScreentip | 1至4096个字符 | （none） | Sub GetScreentip(control As  IRibbonControl,  ByRef returnedVal) |
| showImage | getShowImage | true,false,1,0 | true | Sub GetShowImage(control  As IRibbonControl,  ByRef returnedVal) |
| showLabel | getShowLabel | true,false,1,0 | true | Sub GetShowLabel(control  As IRibbonControl,  ByRef returnedVal) |
| size | getSize | normal,large | normal | Sub GetSize(control As  IRibbonControl,  ByRef returnedVal) |
| supertip | getSupertip | 1至4096个字符 | (none) | Sub GetSupertip(control As  IRibbonControl,  ByRef returnedVal) |
| tag | (none) | 1至4096个字符 | (none) | (none) |
| visible | getVisible | true,false,1,0 | true | Sub GetVisible(control As  IRibbonControl,  ByRef returnedVal) |

**3、toggleButton元素允许的子对象**

toggleButton控件不支持任何子对象。

**4、toggleButton元素的父对象**

在下列任何控件内都能放置toggleButton控件：

         box

         buttonGroup

         dynamicMenu

         group

         menu

         officeMenu

         splitButton

**5、使用内置的切换按钮控件**

在功能区中使用了很多内置的切换按钮控件。下面的示例在自定义选项卡中添加四个内置的切换按钮。

（1）创建一个新的.xlsx文件，并将其保存为Excel Built In toggleButton Example.xlsx。

（2）关闭该文件并在CustomUI Editor中打开。

（3）输入下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtabCustom“

label=“My Tools“

insertBeforeMso=“TabHome“>

<group id=“rxgrpFormats“

label=“Formatting“>

<toggleButton idMso=“Bold“/>

<toggleButton idMso=“Italic“/>

<toggleButton idMso=“Underline“/>

<toggleButton idMso=“UnderlineDouble“/>

</group>

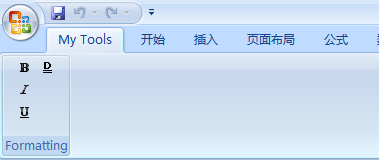
</tab>

</tabs>

</ribbon>

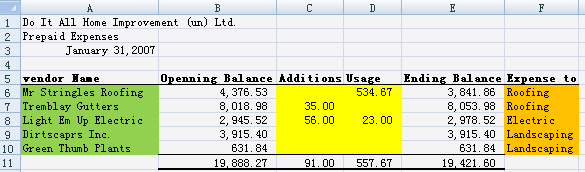
</customUI>

（4）保存并关闭CustomUI Editor。在Excel中打开工作簿，如下图所示。

**[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/09/togglebuttonsample1.png)**

**6、创建自定义切换按钮控件**

下面的示例再次使用在细品RibbonX(18)中使用的预付费用计划示例，我们将添加自定义视图切换功能，这样允许用户显示或隐藏“Expense To”列，如下图所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/09/togglebuttonsample2.png)

 在Excel中，单击“视图—自定义视图”，选择“添加”并命名为“cvw\_Show”。该视图用于返回工作表的全视图，显示所有列。关闭“视图管理器”对话框。

接着，设置隐藏“Expense To”列的第二个视图。隐藏F列，重新打开“视图管理器”对话框，添加一个名为“cvw\_Hide”的新的自定义视图。关闭“视图管理器”对话框。

现在，录制切换视图的宏。单击“录制宏”按钮，开始录制：

（1）单击“视图”选项卡。

（2）单击“自定义视图”。

（3）选择cvw\_Hide视图并选择“显示”。

（4）再次单击“自定义视图”。

（5）选择cvw\_Show视图并选择“显示”。

（6）停止录制。

转到VBE中，查看录制的代码：

ActiveWorkbook.CustomViews(“cvw\_Hide”).Show

ActiveWorkbook.CustomViews(“cvw\_Show”).Show

接下来，让我们设置功能区中所自定义的切换按钮。保存Excel文件，在CustomUI Editor中打开该文件，编写下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“DemoTab“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“DemoGroup“

label=“Demo Group“>

<button id=“rxbtnRollForward“

label=“Roll Forward“

imageMso=“CreateReportFromWizard“

size=“large“

onAction=“rxbtnRollForward\_Click“/>

<toggleButton id=“rxtglHideExpense“

label=“Hide Expenses“

imageMso=“FieldList“

size=“large“

onAction=“rxtglHideExpense\_Click“/>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

生成回调签名并复制，保存并关闭文件。

在Excel中重新打开该文件，转到VBE中，粘贴回调签名代码。

下面，修改回调以满足需要，代码如下：

‘rxtglHideExpense onAction回调

Sub rxtglHideExpense\_Click(control As IRibbonControl, pressed As Boolean)

Select Case pressed

Case True

ActiveWorkbook.CustomViews(“cvw\_Hide”).Show

Case False

ActiveWorkbook.CustomViews(“cvw\_Show”).Show

End Select

End Sub

好了！现在可以转到Excel界面中查看切换按钮的功能了。

然而，如果您隐藏了F列，保存并关闭工作簿，再打开该工作簿时，切换按钮并不会高亮选中，并且需要单击两次才能实现切换功能。此时，可以使用下列方式解决：

         在Workbook\_Open过程中将cvw\_Show视图设置为活动的。

         设置getPressed回调代码测试当工作簿打开时哪个视图为活动视图，然后将其状态返回到切换按钮。

下面再举一个示例。

本示例使用一个切换按钮来切换分页显示。如下面的XML代码所示，在“视图”选项卡中自定义一个组并放置自定义的切换按钮：

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/09/togglebuttonsample3.png)

<customUI onLoad=“rxIRibbonUI\_onLoad“ xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab idMso=“TabView“>

<group id=“rxgrpStyleInsp“

label=“Custom Options“

insertBeforeMso=“GroupZoom“>

<toggleButton id=“rxtglPageBreaks“

label=“Display PageBreaks“

getPressed=“rxtglPageBreaks\_getPressed“

getImage=“rxtglPageBreaks\_getImage“

onAction=“rxtglPageBreaks\_click“/>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

生成回调签名，并将其复制。关闭CustomUI Editor。

在Excel中打开该工作簿，转到VBE，粘贴回调签名到标准模块中，并输写代码：

Private ribbonUI As IRibbonUI

‘customUI.onLoad回调

Sub rxIRibbonUI\_onLoad(ribbon As IRibbonUI)

Set ribbonUI = ribbon

End Sub

‘rxtglPageBreaks getPressed回调

Sub rxtglPageBreaks\_getPressed(control As IRibbonControl, ByRef returnedVal)

returnedVal = ActiveSheet.DisplayPageBreaks

End Sub

‘rxtglPageBreaks getImage回调

Sub rxtglPageBreaks\_getImage(control As IRibbonControl, ByRef returnedVal)

Select Case ActiveSheet.DisplayPageBreaks

Case True

returnedVal = “SignatureInsertMenu”

Case False

returnedVal = “DesignMode”

End Select

End Sub

‘rxtglPageBreaks onAction回调

Sub rxtglPageBreaks\_click(control As IRibbonControl, pressed As Boolean)

ActiveSheet.DisplayPageBreaks = pressed

ribbonUI.InvalidateControl “rxtglPicHold”

End Sub

保存该工作簿并关闭。重新打开该工作簿，此时可以测试该切换按钮的功能了。

[**细品RibbonX(22)：item元素详解**](http://www.excelperfect.com/index.php/2009/10/12/ribbonxstudy22/)

2009年10月12日, 1:06 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**2** 人投票, 平均: **3.00** out of 5)

Loading ... Loading ...

item元素用于在gallery、dropDown、comboBox内创建静态项目，是这些控件必须的一部分。

**1、item元素必需的属性**

每个item需要下表1所列的id属性之一。

表1：item元素必需的属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 使用该属性创建自已的项目 |

item元素仅有一个属性。正如前文所述，item必须与其它控件一起使用，因此所有其它的属性取决于其它元素。

**2、带有回调签名的可选的静态属性和动态属性**

每个item元素可以使用下表2列出的任一或所有的属性。

表2：item元素可选的属性

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| image | （none） | 1至1024个字符 | （none） | （none） |
| imageMso | （none） | 1至1024个字符 | （none） | （none） |
| label | （none） | 1至1024个字符 | （none） | （none） |
| screentip | （none） | 1至1024个字符 | （none） | （none） |
| supertip | （none） | 1至1024个字符 | (none) | （none） |

注意到item元素所有的属性都是静态的，这意味着可以直接提供静态的列表项而无须编写VBA代码。当然，可以通过其父控件的回调签名来动态添加项目。但两者是相互排斥的，即指定静态项目后，不能同时指定动态项目。

**3、item元素允许的子对象**

item元素不支持任何子对象，因此没有任何嵌入的控件。

**4、item元素的父对象**

在下列任何控件内都能使用item：

         comboBox

         dropDown

         gallery

**5、使用内置的控件**

没有机会利用内置的item元素，因为Microsoft没有公开它们为我们所使用。如果希望使用某控件内的一些项目，则可以包含其完整的父元素。

**6、创建自定义控件**

您可以试着在组合框或下拉列表中创建自定义item元素，接下来的两篇将详细介绍这方面的内容。

[**细品RibbonX(23)：comboBox控件详解**](http://www.excelperfect.com/index.php/2009/10/15/ribbonxstudy23/)

2009年10月15日, 2:09 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**2** 人投票, 平均: **4.50** out of 5)

Loading ... Loading ...

组合框控件基于指定的记录源显示数据，它是编辑框和下拉控件的组合。该控件最明显的特征之一是：不仅能够从列表中选取项目，而且能够在其中输入项目。

comboBox控件常用于下列情形：

         列表非常长，希望用户在输入一些关键词后能够迅速跳到合适的地方。（例如fonts控件）

         希望呈现给用户一个预定义列表，同时也希望能够在列表中添加项目。

可以使用静态列表或动态创建列表来填充组合框。

**1、comboBox元素必需的属性**

组合框控件需要下表1所列的id属性之一。

表1：comboBox元素必需的属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的组合框时 |
| idMso | 当使用现有的Microsoft组合框时 |
| idQ | 当在命名空间之间创建共享的组合框时 |

**2、带有回调签名的可选的静态属性和动态属性**

组合框控件可以使用下表2列出的任一insert属性。

表2：comboBox元素可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **INSERT属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之后插入 |
| insertBeforeMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之前插入 |
| insertAfterQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之后插入 |
| insertBeforeQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之前插入 |

也可以为组合框设置下表3所列的任何或所有的属性或回调。

表3：comboBox元素可选的属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| （none） | onChange | 1至4096个字符 | （none） | Sub OnChange(control  As IRibbonControl,  text As String) |
| enabled | getEnabled | true,false,1,0 | true | Sub GetEnabled(control As  IRibbonControl,  ByRef returnedVal) |
| image | getImage | 1至1024个字符 | （none） | Sub GetImage(control As  IRibbonControl,  ByRef returnedVal) |
| imageMso | getImage | 1至1024个字符 | （none） | 同上 |
| （none） | getItemCount | 1至1024个字符 | （none） | Sub GetItemCount(control As  IRibbonControl,  ByRef returnedVal) |
| （none） | getItemID | 1至1024个字符 | （none） | Sub GetItemID(control As  IRibbonControl,  index As Integer,  ByRef id) |
| （none） | getItemImage | 独立的文本字符串 | （none） | Sub GetItemImage(control As  IRibbonControl,  index As Integer,  ByRef returnedVal) |
| （none） | getItemLabel | 1至1024个字符 | （none） | Sub GetItemLabel(control As IRibbonControl,  index As Integer,  ByRef returnedVal) |
| （none） | getItemScreentip | 1至1024个字符 | （none） | Sub GetItemScreenTip(control As IRibbonControl,  index As Integer,  ByRef returnedVal) |
| （none） | getItemSupertip | 1至1024个字符 | （none） | Sub GetItemSuperTip(control As IRibbonControl,  index As Integer,  ByRef returnedVal) |
| keytip | getKeytip | 1至3个字符 | （none） | Sub GetKeytip(control As IRibbonControl,  ByRef returnedVal) |
| label | getLabel | 1至1024个字符 | （none） | Sub GetLabel(control As  IRibbonControl,  ByRef returnedVal) |
| maxLength | （none） | 1至1024个字符 | 1024 | （none） |
| screentip | getScreentip | 1至1024个字符 | （none） | Sub GetScreentip(control As IRibbonControl,  ByRef returnedVal) |
| showImage | getShowImage | true,false,1,0 | true | Sub GetShowImage(control As IRibbonControl,  ByRef returnedVal) |
| showItemAttribute | （none） | true,false,1,0 | true | （none） |
| showItemImage | （none） | true,false,1,0 | true | （none） |
| showLabel | getShowLabel | true,false | true | Sub GetShowLabel(control As  IRibbonControl,  ByRef returnedVal) |
| sizeString | （none） | 1至1024个字符 | 12 | （none） |
| supertip | getSupertip | 1至1024个字符 | (none) | Sub GetSupertip(control  As IRibbonControl,  ByRef returnedVal) |
| tag | (none) | 1至1024个字符 | (none) | (none) |
| （none） | getText | 1至4096个字符 | （none） | Sub GetText(control As  IRibbonControl,  ByRef returnedVal) |
| visible | getVisible | true,false,1,0 | true | Sub GetVisible(control  As  IRibbonControl,  ByRef returnedVal) |

**3、comboBox元素允许的子对象**

comboBox元素可以使用的唯一的子对象是item元素。

**4、comboBox元素的父对象**

在下列任何控件内都能放置comboBox控件：

         box

         group

**5、使用内置的组合框控件**

在Excel的所有控件中，最著名的可能是字体组合框。如果创建组合最常使用的控件的自定义选项卡，那么将会添加字体组合框控件。

（1）创建一个新的.xlsx文件，并将其保存为Excel Built In comboBox Example.xlsx。

（2）关闭该文件并在CustomUI Editor中打开。

（3）输入下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“rxgrpDemo“

label=“Demo“>

<comboBox idMso=“Font“/>

<comboBox idMso=“FontSize“/>

</group>

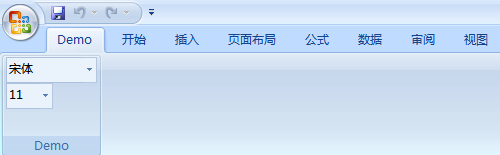
</tab>

</tabs>

</ribbon>

</customUI>

下图为上述代码创建的组。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/comboboxsample1.png)

**6、创建自定义组合框控件**

本例中，假设您已经隐藏了完整的用户界面。此外，也试图使应用程序看起来不太像Excel，因为隐藏了所有的工作表标签。然而，您仍然想用户能够在三个工作表之间移动，因此需要提供一些手段来实现这项操作。在许多情形下，组合框控件是能够用于这类导航的理想的控件：列出应用程序中所有的“页面”，允许用户输入其希望跳转到的页面。

（1）新建一个工作簿，以.xlsm为扩展名保存后关闭。

（2）在CustomUI Editor中打开该工作簿，并输入下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“

label=“Navigation“

insertBeforeMso=“TabHome“>

<group id=“rxgrpNavigate“

label=“Navigate To“>

<comboBox id=“rxcboSelectSheet“

label=“Activate:“

onChange=“rxcboSelectSheet\_Click“>

<item id=“rxitemcboSelectSheet1“

label=“Sheet1“/>

<item id=“rxitemcboSelectSheet2“

label=“Sheet2“/>

<item id=“rxitemcboSelectSheet3“

label=“Sheet3“/>

</comboBox>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

注意到该组合框利用onChange回调来产生当选择项目时的行为。此外，该组合框包含三个项目：Sheet1、Sheet2、Sheet3。这些项目是静态的并且不能从文件中修改，用户也不能添加其它项目，这确保了用户仅能在这三个工作表之间导航。

生成回调签名，并复制代码后关闭CustomUI Editor。

在Excel中打开该工作簿，打开VBE并将代码粘贴到一个标准模块中。下面编辑该回调以响应希望的操作。这时，可以通过考虑事件的顺序来解决如何实现：

（1）用户从工作表列表中选择一个项目。

（2）触发回调。

（3）将组合框中所选择的值（即工作表的名字）传递到程序。

（4）激活工作表。

然而，如果用户在组合框中输入值，会发生什么呢？要处理这些可能发生的事，应该编辑回调签名如下：

‘rxcboSelectSheet onChange回调

Sub rxcboSelectSheet\_Click(control As IRibbonControl, text As String)

On Error Resume Next

Worksheets(text).Activate

If Err.Number <> 0 Then

MsgBox “对不起,不存在这个工作表!”

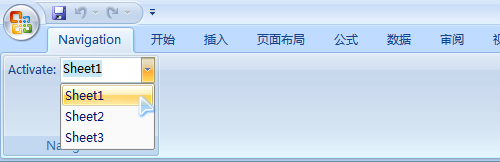
End If

End Sub

第2行代码试图激活传递给回调的名称的工作表。从列表中选择的项目总是有效的名字，而由用户输入的文本可能与列表中的项目不匹配。在开始部分的On Error语句告诉代码即使发生错误也继续运行下一行代码。

如果Err属性不为零，那么一定是发生了错误，因此用户输入的值是无效的。此外，因为希望让用户知道其输入是不可接受的，所以包括了一个消息框。

现在，就可以测试了：可以从组合框中选择一个值（如下图所示），也可以输入一个值。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/comboboxsample2.png)

[**细品RibbonX(24)：dropDown控件详解**](http://www.excelperfect.com/index.php/2009/10/16/ribbonxstudy24/)

2009年10月16日, 12:50 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

与组合框控件相似，dropDown控件提供用户一组预定义列表以供用户从中选择。此外，能够在设计时使用XML提供的静态列表，或者在运行时通过回调动态填充列表。

组合框和下拉控件最大的不同在于组合框可以接受用户输入，而下拉控件则不行，用户只能从预定义的列表并且仅能从该列表中选择项目。

下面是一些使用下拉控件的理由：

         不希望用户输入它们自已的信息。

         列表不长，因此不涉及到使用“自动完成”功能。

         没兴趣编程回调来验证用户输入数据。

**1、dropDown元素必需的属性**

dropDown控件需要下表1所列的id属性之一。

表1：dorpDown元素必需的属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的下拉列表时 |
| idMso | 当使用现有的Microsoft下拉列表时 |
| idQ | 当在命名空间之间创建共享的下拉列表时 |

**2、带有回调签名的可选的静态属性和动态属性**

dropDown元素可以使用下表2列出的任一insert属性。

表2：dropDown元素可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **INSERT属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之后插入 |
| insertBeforeMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之前插入 |
| insertAfterQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之后插入 |
| insertBeforeQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之前插入 |

也可以为下拉列表设置下表3所列的任何或所有的属性或回调。

表3：dropDown元素可选的属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| enabled | getEnabled | true,false,1,0 | true | Sub GetEnabled(control  As IRibbonControl,  ByRef returnedVal) |
| image | getImage | 1至1024个字符 | （none） | Sub GetImage(control  As IRibbonControl,  ByRef returnedVal) |
| imageMso | getImage | 1至1024个字符 | （none） | 同上 |
| （none） | getItemCount | 1至1024个字符 | （none） | Sub GetItemCount(control  As IRibbonControl,  ByRef returnedVal) |
| （none） | getItemID | 独立的文本字符串 | （none） | Sub GetItemID(control  As IRibbonControl,  index As Integer,  ByRef id) |
| （none） | getItemImage | 1至1024个字符 | （none） | Sub GetItemImage(control  As IRibbonControl,  index As Integer,  ByRef returnedVal) |
| （none） | getItemLabel | 1至1024个字符 | （none） | Sub GetItemLabel(control  As IRibbonControl,  index As Integer,  ByRef returnedVal) |
| （none） | getItemScreentip | 1至1024个字符 | （none） | Sub GetItemScreenTip(control  As IRibbonControl,  index As Integer,  ByRef returnedVal) |
| （none） | getItemSupertip | 1至1024个字符 | （none） | Sub GetItemSuperTip(control  As IRibbonControl,  index As Integer,  ByRef returnedVal) |
| keytip | getKeytip | 1至3个字符 | （none） | Sub GetKeytip(control  As IRibbonControl,  ByRef returnedVal) |
| label | getLabel | 1至1024个字符 | （none） | Sub GetLabel(control  As IRibbonControl,  ByRef returnedVal) |
| screentip | getScreentip | 1至1024个字符 | （none） | Sub GetScreentip(control  As IRibbonControl,  ByRef returnedVal) |
| （none） | getSelectedItemID | 独立的文本字符串 | （none） | Sub GetSelectedItemID(control  As IRibbonControl,  ByRef returnedVal) |
| （none） | getSelectedItemIndex | 1至1024个字符 | （none） | Sub  GetSelectedItemIndex(control  As IRibbonControl,  ByRef returnedVal) |
| showImage | getShowImage | true,false,1,0 | true | Sub GetShowImage(control  As IRibbonControl,  ByRef returnedVal) |
| showItemImage | （none） | true,false,1,0 | true | （none） |
| ShowItemLabel | （none） | true,false,1,0 | true | （none） |
| showLabel | getShowLabel | true,false,1,0 | true | Sub GetShowLabel(control  As IRibbonControl,  ByRef returnedVal) |
| sizeString | （none） | 1至1024个字符 | 12 | （none） |
| supertip | getSupertip | 1至1024个字符 | (none) | Sub GetSupertip(control  As IRibbonControl,  ByRef returnedVal) |
| tag | (none) | 1至1024个字符 | (none) | (none) |
| visible | getVisible | true,false,1,0 | true | Sub GetVisible(control  As IRibbonControl,  ByRef returnedVal) |
| （none） | onAction | 1至1024个字符 | （none） | Sub OnAction(control  As IRibbonControl,  selectedId As String,  selectedIndex As Integer) |

**3、dropDown元素允许的子对象**

dropDown元素可以使用的唯一的子对象是item元素。

**4、dropDown元素的父对象**

在下列任何控件内都能放置dropDown控件：

         box

         group

**5、使用内置的下拉控件**

下面引用Excel内置的边框样式控件。

（1）创建一个新的.xlsx文件，并将其保存为Excel Built In dropDown Example.xlsx。

（2）关闭该文件并在CustomUI Editor中打开。

（3）输入下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“rxgrpDemo“

label=“Demo“>

<dropDown idMso=“BorderStyle“/>

</group>

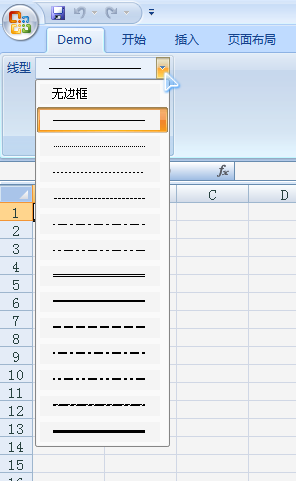
</tab>

</tabs>

</ribbon>

</customUI>

下图为上述代码创建的组。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/dropdownsample1.png)

**6、创建自定义下拉控件**

本示例相当有趣，因为前后使用了两个dropDown元素。与在介绍组合框控件示例时相似，第一个控件中列出了工作簿中的所有工作表。然而，在从工作簿中添加或移除工作表时，我们使用可用的回调来更新下拉列表，

第二个控件允许用户切换所选工作表的可见性：xlSheetVisible、xlSheetHidden和xlSheetVeryHidden。

（1）新建一个工作簿，以.xlsm为扩展名保存后关闭。

（2）在CustomUI Editor中打开该工作簿，并输入下列XML代码：

<customUI onLoad=“rxIRibbonUI\_onLoad“ xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“

label=“Navigation“

insertBeforeMso=“TabHome“>

<group id=“rxgrpNavigate“

label=“Navigate To“>

<dropDown id=“rxddSelectSheet“

label=“Apply To:“

visible=“true“

onAction=“rxddSelectSheet\_click“

getItemID=“rxitemddSelectSheet\_getItemId“

getItemCount=“rxitemddSelectSheet\_getItemCount“

getItemLabel=“rxitemddSelectSheet\_getItemLabel“/>

<dropDown id=“rxddSheetVisible“

label=“Set To:“

onAction=“rxddSheetVisible\_click“>

<item id=“rxitemddSheetVisible1“

label=“Visible“/>

<item id=“rxitemddSheetVisible2“

label=“Hidden“/>

<item id=“rxitemddSheetVisible3“

label=“VeryHidden“/>

</dropDown>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

生成回调签名，并复制代码后关闭CustomUI Editor。

在Excel中打开该工作簿，打开VBE并将回调签名代码粘贴到一个标准模块中。注意到，少生成了一个回调签名：getItemID，需要补充。

首先处理onLoad回调。在代码开始处添加两个变量，一个包含RibbonUI对象，另一个存放所选的工作表名：

Public RibbonUI As IRibbonUI

Dim sSheetName As String

接着，设置onLoad回调确保RibbonUI对象在装载时被捕获：

‘customUI.onLoad回调

Sub rxIRibbonUI\_onLoad(ribbon As IRibbonUI)

Set RibbonUI = ribbon

End Sub

下面的代码获取工作表数：

‘rxddSelectSheet getItemCount回调

Sub rxitemddSelectSheet\_getItemCount(control As IRibbonControl, ByRef returnedVal)

returnedVal = Worksheets.Count

End Sub

下一步，设置getItemLabel回调，返回下拉列表中每个项目的文本：

‘rxddSelectSheet getItemLabel回调

Sub rxitemddSelectSheet\_getItemLabel(control As IRibbonControl, index As Integer, ByRef returnedVal)

returnedVal = Worksheets(index + 1).Name

End Sub

注意到数组参数使用index+1，这是因为VBA默认的数组索引号基于0，而Excel默认的工作表索引是基于1。

接下来，确保为每个下拉项动态生成唯一的ID，回调代码如下：

Sub rxitemddSelectSheet\_getItemID(control As IRibbonControl, index As Integer, ByRef id)

id = “rxitemddSelectSheet” & index+1

End Sub

更新rxddSelectSheet\_click过程如下：

‘rxddSelectSheet onAction回调

Sub rxddSelectSheet\_click(control As IRibbonControl, id As String, index As Integer)

On Error Resume Next

Call rxitemddSelectSheet\_getItemLabel(control, index, sSheetName)

If Err.Number <> 0 Then

MsgBox “Sorry,that worksheet does not exist!”

RibbonUI.InvalidateControl “rxddSelectSheet”

End If

End Sub

已经完成动态dropDown控件的设置后，再来为静态下拉列表设置唯一的回调：

‘rxddSheetVisible onAction回调

Sub rxddSheetVisible\_click(control As IRibbonControl, id As String, index As Integer)

‘检查已选择的工作表

On Error Resume Next

sSheetName = Worksheets(sSheetName).Name

If Err.Number <> 0 Then

MsgBox “Sorry,but you need to select a valid sheet first!”

Exit Sub

End If

‘改变工作表的可见性

Select Case id

Case “rxitemddSheetVisible1″

Worksheets(sSheetName).Visible = xlSheetVisible

Case “rxitemddSheetVisible2″

Worksheets(sSheetName).Visible = xlSheetHidden

Case “rxitemddSheetVisible3″

Worksheets(sSheetName).Visible = xlSheetVeryHidden

End Select

If Err.Number <> 0 Then

MsgBox “Sorry,this is the only visible sheet.” & vbCrLf & “You cann’t hide the all!”

End If

On Error GoTo 0

End Sub

至此，全部代码已完成。可以测试了，如下图所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/dropdownsample2.png)

[**细品RibbonX(25)：使用自定义图片和库**](http://www.excelperfect.com/index.php/2009/10/17/ribbonxstudy25/)

2009年10月17日, 12:37 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

有时，您可能发现内置的图标库是不够的，可能想引入自已的图像（图标或图片）。下面介绍如何利用自定义图片的功能及如何使用库，用户界面中最酷的新功能之一。

**1、自定义图片**

（1）支持的图片格式

如果打算定制用户界面，那么也可能会考虑使用自已的图像来增添额外的感觉。然而，知道一些文件格式，以便于知道在特定的情形下应该使用那种类型。

一种用于自定义图片的好的格式是PNG格式。PNG文件允许全透明（其它可能的格式例如BMP、GIF和JPEG不支持），这意味着不会因为界面背景色的变化而导致自定义的图片变得与背景不匹配。然而，在某些情形下不允许使用PNG文件。例如，如果需要使用VBA的LoadPicture函数实时装载图像，就不能够使用PNG格式。因此，需要寻找一种替代的方法：或者不同的格式或者不同的方法来装载图像。

下表1显示了能够在自定义用户界面中使用的一些文件格式。

表1：用于自定义UI的一些可能的图片格式

|  |  |
| --- | --- |
| **图片扩展名** | **格式** |
| PNG | 如果计划附加图片到工程中，那么Portable Network Graphics是一种优秀的用于自定义UI的格式。不能够在LoadPicture函数中使用这种文件格式。 |
| BMP | 指位图图片格式。可以使用LoadPicture函数装载这种格式的图片。 |
| ICO | 指图标格式。 |
| WMF | 指Windows图元文件，可以使用LoadPicture函数装载这种格式的图片。 |
| JPG、JPEG | JPEG（Joint Photographic Experts Group）格式提供较低品质的图像。 |
| GIF | Graphic Interchange Format，与JPEG一样，提供较低品质的图像。 |

如果用户界面使用自定义图片，那么应该选择PNG图片格式。PNG格式是提供完全支持透明度的唯一格式，并且PNG图片是高可压缩的而不会丢失质量。相反，因为其低的图像质量，通常不推荐JPG和GIF格式。

（2）适合的图片尺寸和比例

尺寸和比例对在用户界面中显示清楚且明晰的图片是重要的。作为一条规则，图形应该是具有96dpi的16×16和32×32的尺寸。

（3）图像编辑资源

Paint.NET软件提供了一个极好的图像编辑器，支持JPEG、PNG和BMP图像，有很强大的功能。您可以在下面的站点中下载：

<http://www.getpaint.net/index2.html>

**2、在工程中添加自定义图片**

（1）使用CustomUI Editor

要添加自定义图像，必须使用该控件的image属性。假设希望对按钮添加自定义图像，其XML代码如下：

<button id=“rxbtn“

label=“Custom Button“

image=“test\_img“

size=“large“/>

注意到，图像的名字是test\_img，并且没有指定文件扩展名。在XML压缩组里，相关联的文件指向该图像的正确位置。在本例中，如果查看相关联的XML文件（CustomUI.xml.rels），那么会发现下列代码段：

<Relationships xmlns=http://schemas.openxmlformats.org/package/2006/relationships>

<Relationships Type=http://schemas.openxmlformats.org/officeDocument/2006/relationships/image Target=”images/test\_img.bmp” Id=”test\_img” />

</Relationships>

相关的文件及其XML代码由CustomUI Editor自动生成。接下来，装载图像。

要使用CustomUI Editor来装载自定义图像，按照下列步骤：

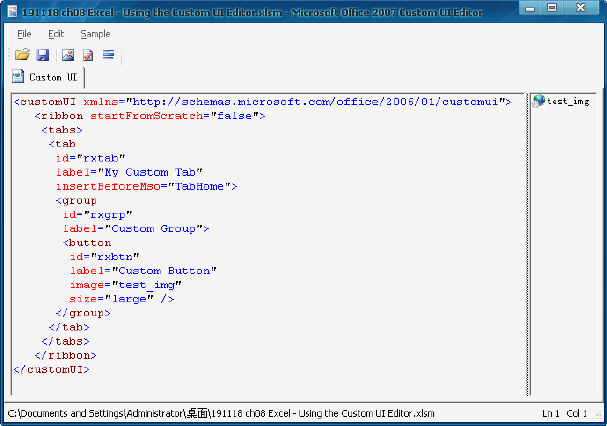
（1）使用CustomUI Editor打开文档。

（2）添加自定义用户界面的XML代码。无论希望在哪里添加图像，都使用image属性并指定文件名，例如image=”test\_img”。不需要担心文件扩展名。

（3）单击CustomUI Editor工具栏中的“Insert Icon”按钮。

（4）浏览包含图像的文件夹，打开该图像文件。

下图显示通过CustomUI Editor添加test\_img文件，注意CustomUI Editor现在有一个额外的列列出了附加到工程中的所有图像。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/custompicsample1.png)

这是装载自定义图片到UI中的一种非常容易的方法，但也可能是非常费时的，特别是要装载大量图片时。另一种方法是在运行时装载图片。

（2）在运行时装载自定义图片

如果需要在运行时切换图像，例如在运行时作出变化，那么需要使用该控件的getImage属性装载图像。

然而，这也带给我们另一个问题。此时，通常通过VBA的LoadPicture函数装载图像的方法。然而，这个函数不能处理PNG文件，而这种格式是功能区最好的图片格式。

下面的示例装载普通的BMP图片到切换按钮里。注意，用于装载位图文件的方法也能够用于图标文件和Windows图元文件，例如FileName.ico、FileName.wmf，等等。

<toggleButton

id=“rxtgl“

getImage=“rxtgl\_getImage“

getLabel=“rxtgl\_getLabel“

size=“large“

onAction=“rxtgl\_Click“ />

使用getImage属性定义处理图片装载的回调。本例中，回调被命名为rxtgl\_getImage并有下列签名：

Sub rxtgl\_getImage(control as IRibbonControl, ByRef returnedVal)

因为本示例使用切换按钮，所以需要追踪切换状态，即按钮切换与否？这里使用全局的布尔变量来追踪状态。同时，需要onLoad事件来使该控件无效，确保根据切换按钮的单击装载图像。

在标准模块中输入下面的VBA代码：

Dim grxIRibbonUI As IRibbonUI

Dim gblnPressed As Boolean

Sub rxIRibbonUI\_onLoad(ribbon As IRibbonUI)

Set grxIRibbonUI = ribbon

End Sub

Sub rxtgl\_getImage(control As IRibbonControl, ByRef returnedVal)

Set returnedVal = LoadPicture(ThisWorkbook.Path & “\mex.bmp”)

If gblnPressed Then Set returnedVal = LoadPicture(ThisWorkbook.Path & “\usa.bmp”)

End Sub

Sub rxtgl\_Click(control As IRibbonControl, pressed As Boolean)

gblnPressed = pressed

grxIRibbonUI.InvalidateControl (“rxtgl”)

End Sub

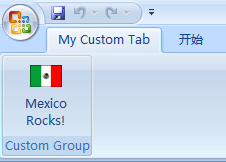
Sub rxtgl\_getLabel(control As IRibbonControl, ByRef returnedVal)

returnedVal = “Mexico Rocks!”

If gblnPressed Then returnedVal = “The US Rocks!”

End Sub

下图显示了自定义的切换按钮。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/custompicsample2.png)

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/custompicsample3.png)

**3、使用GDI+装载PNG文件**

下面介绍如何使用GDI+（Graphics Device Interface Plus）APIs装载PNG文件。

GDI+是Windows核心子系统之一，当需要发送这些对象到诸如显示器和/或打印机等设备时用作表现图形对象的接口。GDI+处理不同绘图对象的渲染，这里我们使用它来渲染PNG对象以便使其能被装载到UI。这需要一些其它的技术，即使用Windows APIs来实现，下面是代码模块，可以直接导入VBA工程中使用：

basGDIPlus.bas

该模块中包含了一个名为LoadImage的函数，可用来直接装载PNG文件。因此，在上例中，如果想使用LoadImage函数代替LoadPicture函数来装载PNG文件，那么需要将其与getImage属性相关的代码中修改为LoadImage函数：

Sub rxtgl\_getImage(control As IRibbonControl, ByRef returnedVal)

Set returnedVal = LoadImage(ThisWorkbook.Path & “\mex.png”)

If gblnPressed Then Set returnedVal = LoadImage(ThisWorkbook.Path & “\usa.png”)

End Sub

[**细品RibbonX(26)：库(Gallery)控件详解**](http://www.excelperfect.com/index.php/2009/10/20/ribbonxstudy26/)

2009年10月20日, 1:16 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(目前还没有人投票)

Loading ... Loading ...

下面介绍如何使用库，Office 2007用户界面中最酷的新功能之一。

库被设计作为图形化显示用户选项的一种方式，例如样式库，能够通过查看该样式图形化的呈现使用户选择一种样式。可以使用库来组织照片、访问样式、或者存储调色板。在需要并提供可视效果的定制中，库是完美的控件。

此外，在库的底部也可以添加按钮。

库的XML标记如下：

<gallery>

<!—放置XML代码>

</gallery>

库由静态的、动态的和可选的子属性组成，需要下表1所示的静态属性。

表1：库的静态属性

|  |  |  |
| --- | --- | --- |
| **静态属性** | **允许值** | **VBA回调签名** |
| columns | 1至1024 | N/A |
| itemHeight | 1至4096 | Sub getItemHeight(control As IRibbonControl,  ByRef height) |
| itemWidth | 1至4096 | Sub getItemWidth(control As IRibbonControl,  ByRef width) |
| rows | 1至1024 | N/A |
| sizeString | 1至1024个字符 | N/A |
| showItemImage | true,false,1,0 | N/A |
| showItemLabel | true,false,1,0 | N/A |

库也可以包含下表2所示的动态属性。

表2：库的动态属性

|  |  |  |
| --- | --- | --- |
| **动态属性** | **允许值** | **VBA回调签名** |
| getItemCount | 0至1000 | Sub GetItemCount(control As IRibbonControl,  ByRef count) |
| getItemID |  | Sub GetItemID(control As IRibbonControl,  index As Integer,  ByRef id) |
| getItemImage |  | Sub GetItemImage(control As IRibbonControl,  index As Integer,  ByRef image) |
| getItemLabel | 1024 | Sub GetItemLabel(control As IRibbonControl,  index As Integer,  ByRef label) |
| getItemScreentip | 1024 | Sub GetItemScreenTip(control As IRibbonControl,  index As Integer,  ByRef screentip) |
| getItemSupertip | 1024 | Sub GetItemSuperTip(control As IRibbonControl,  index As Integer,  ByRef supertip) |
| getSelectedItemID | 1至1024个字符 | Sub GetSelectedItemID(control As IRibbonControl,  ByRef index) |
| getSelectedItemIndex | 1至1024 | Sub GetSelectedItemIndex(control As IRibbonControl,  ByRef index) |
| onAction |  | Sub OnAction(control As IRibbonControl,  selectedId As String,  selectedIndex As Integer) |

库还能够接受下表3所列的一个或全部对象作为其子元素，也可以具有每个对象的多个实例。

|  |  |
| --- | --- |
| **对象** | **作用** |
| button | 在库中添加一个可单击的按钮 |
| item | 在库中添加一个项目。该项目可以图形化地呈现希望执行的操作，例如图表布局的图形化呈现 |

下面，介绍一个示例，创建下图所示的库。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/gallerysample1.png)

（1）创建一个新的Excel工作簿，保存后关闭。

（2）在CustomUI Editor中打开该文件，输入下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab

id=“rxtab“

insertBeforeMso=“TabHome“

label=“My Custom Tab“>

<group

id=“rxgrp“

label=“My Photo Gallery“>

<!– Starts the definition of our gallery–>

<gallery

id=“rxgal“

label=“My Photo Gallery“

image=“img4“

columns=“2“

rows=“2“

itemWidth=“200“

itemHeight=“150“

showItemLabel=“false“

size=“large“>

<!– Insert the photo gallery–>

<!– Import the photos you want to use first–>

<item id=“rxitem0“ label=“London 1“ image=“img0“/>

<item id=“rxitem1“ label=“London 2“ image=“img1“ />

<item id=“rxitem2“ label=“London 3“ image=“img2“ />

<item id=“rxitem3“ label=“London 4“ image=“img3“ />

<item id=“rxitem4“ label=“London 5“ image=“img4“ />

<item id=“rxitem5“ label=“London 6“ image=“img5“ />

<!– Insert a button at the end of the gallery–>

<button id=“rxbtn“

imageMso=“RefreshStatus“

label=“Visit excelperfect…“

onAction=“rxbtn\_Click“/>

</gallery>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

（3）在CustomUI Editor中装载名为img0至img5的图像。

（4）保存并关闭CustomUI Editor，然后重新打开该工作簿，即可看到上图所示的自定义库。

**使用内置库控件**

有时，可能想将开始、插入、页面布局选项卡中的库放置在自定义选项卡中以方便使用，而无须在选项卡之间来回移动。下面的例子将三个内置库控件集中放置在一个自定义选项卡中。XML代码为：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtab“

insertBeforeMso=“TabHome“

label=“Galleries Tab“>

<group

id=“rxgrp“

label=“Gallery Group“>

<box id=“rxbox1“ boxStyle=“horizontal“>

<gallery idMso=“FontColorPicker“ label=“AAA“/>

<labelControl id=“rxlbl1“ label=“Font Color“/>

</box>

<box id=“rxbox2“ boxStyle=“horizontal“>

<gallery idMso=“CellFillColorPicker“ />

<labelControl id=“rxlbl2“ label=“Cell Color“/>

</box>

<box id=“rxbox3“ boxStyle=“horizontal“>

<gallery idMso=“ChartTypeColumnInsertGallery“ size=“normal“ />

</box>

</group>

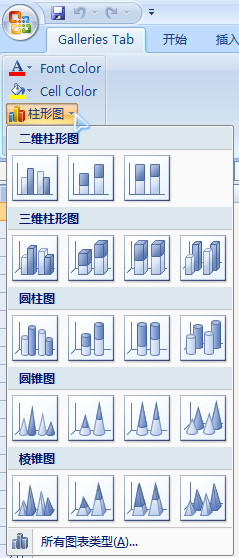
</tab>

</tabs>

</ribbon>

</customUI>

结果如下图所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/gallerysample2.png)

**在运行时创建图像库**

如果使用XML代码装载图像的话，那么随着库中图像数量的增加，需要输入更多的XML代码。显然，这不仅浪费时间，而且使维护变得更复杂。

下面的示例演示如何在库中动态添加图像，以避免额外的输入。在代码中：

         getImage：用于装载库控件的图片（不是真正的库里面的图片）。

         getItemCount：用于返回库中的项目数。在VBA代码中使用常量来决定这个值，这样如果需要更多的图片项目，只需简单地修改其数值。

         getItemImage：用于装载项目图像，也就是说，每一个显现在库中的图像。

注意，不会遍历每一项，相反，Ribbon将回调getItemImage属性直至遍历由getItemCount属性决定的所有项目。此时，返回代表每个项目的图像的索引值。接下来，使用这个索引来捕获并装载相应的图片。

定制Excel界面库控件的XML代码如下：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtab“

insertBeforeMso=“TabHome“

label=“My Custom Tab“>

<group id=“rxgrp“

label=“My Photo Gallery“>

<!– Starts the definition of our gallery –>

<gallery id=“rxgal“

label=“My Photo Gallery“

columns=“2“

rows=“2“

itemWidth=“200“

itemHeight=“150“

getImage=“rxgal\_getImage“

getItemCount=“rxgal\_getItemCount“

getItemImage=“rxgal\_getItemImage“

onAction=“rxgal\_Click“

showItemLabel=“false“

size=“large“>

<!– Inserts a button at the bottom of the gallery –>

<button id=“rxbtn“

imageMso=“RefreshStatus“

label=“Visit excelperfect …“

onAction=“rxbtn\_Click“/>

</gallery>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

在Excel VBE中的代码如下：

Public Const gcItemCount = 6

‘rxgal getImage回调

Sub rxgal\_getImage(control As IRibbonControl, ByRef returnedVal)

Set returnedVal = LoadPicture(ActiveWorkbook.Path & “\MUN.bmp”)

End Sub

‘rxgal getItemCount回调

Sub rxgal\_getItemCount(control As IRibbonControl, ByRef returnedVal)

returnedVal = gcItemCount

End Sub

‘rxgal getItemImage回调

Sub rxgal\_getItemImage(control As IRibbonControl, index As Integer, ByRef returnedVal)

Set returnedVal = LoadPicture(ActiveWorkbook.Path & “\img” & index & “.bmp”)

End Sub

‘rxgal onAction回调

Sub rxgal\_Click(control As IRibbonControl, id As String, index As Integer)

MsgBox “您单击的图像索引号为 ” & index

End Sub

‘rxbtn onAction回调

Sub rxbtn\_Click(control As IRibbonControl)

MsgBox “更多技术请访问:http://www.excelperfect.com”

End Sub

当然，如果BMP格式效果不好，可以使用上篇文章中介绍的LoadImage函数装载PNG格式。

[**细品RibbonX(27)：menu控件详解**](http://www.excelperfect.com/index.php/2009/10/23/ribbonxstudy27/)

2009年10月23日, 12:43 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(目前还没有人投票)

Loading ... Loading ...

菜单提供给用户可供从中选取的预定义选项列表，与dropDown控件一样，能够组合图像和文本。

表面上，菜单的用途与下拉控件的用途非常相似。然而，下拉控件的一个主要局限是仅能包含“项目”，而菜单控件可以包含各种各样的其它控件，包括按钮、复选框、库、甚至是另一个菜单。此外，下拉控件的默认显示是一个空框，而菜单可以设置一张“脸”，而不会依赖出现在其列表中的实际项目，因此菜单控件能够给人良好的印象。

**1、menu元素必需的属性**

菜单控件需要下表1所列的id属性之一。

表1：menu元素必需的属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的菜单时 |
| idMso | 当使用现有的Microsoft菜单时 |
| idQ | 当在命名空间之间创建共享的菜单时 |

**2、带有回调签名的可选的静态属性和动态属性**

菜单控件可以使用下表2列出的任一insert属性。

表2：menu元素可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **INSERT属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之后插入 |
| insertBeforeMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之前插入 |
| insertAfterQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之后插入 |
| insertBeforeQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之前插入 |

也可以为菜单控件设置下表3所列的任何静态属性或等价的动态属性。

表3：menu元素可选的属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| description | getDescription | 1至1024个字符 | （none） | Sub GetDescription(control As  IRibbonControl,  ByRef returnedVal) |
| enabled | getEnabled | true,false,1,0 | true | Sub GetEnabled(control  As IRibbonControl,  ByRef returnedVal) |
| image | getImage | 1至1024个字符 | （none） | Sub GetImage(control As  IRibbonControl,  ByRef returnedVal) |
| imageMso | getImage | 1至1024个字符 | （none） | 同上 |
| itemSize | （none） | normal,large | normal | （none） |
| keytip | getKeytip | 1至3个字符 | （none） | Sub GetKeytip(control As  IRibbonControl,  ByRef returnedVal) |
| label | getLabel | 1至1024个字符 | （none） | Sub GetLabel(control As  IRibbonControl,  ByRef returnedVal) |
| screentip | getScreentip | 1至1024个字符 | （none） | Sub GetScreentip(control  As IRibbonControl,  ByRef returnedVal) |
| showImage | getShowImage | true,false,1,0 | true | Sub GetShowImage(control  As IRibbonControl,  ByRef returnedVal) |
| showLabel | getShowLabel | true,false,1,0 | true | Sub GetShowLabel(control As  IRibbonControl,  ByRef returnedVal) |
| size | getSize | normal,large | normal | Sub GetSize(control As  IRibbonControl,  ByRef returnedVal) |
| supertip | getSupertip | 1至1024个字符 | (none) | Sub GetSupertip(control  As IRibbonControl,  ByRef returnedVal) |
| tag | (none) | 1至1024个字符 | (none) | (none) |
| visible | getVisible | true,false,1,0 | true | Sub GetVisible(control As  IRibbonControl,  ByRef returnedVal) |

**3、menu元素允许的子对象**

menu元素接受下列子对象的任意组合：

         button

         checkbox

         control

         dynamicMenu

         gallery

         menu

         menuSeparator

         splitButton

         toggleButton

**4、menu元素的父对象**

在下列任何控件内都能放置menu控件：

         box

         buttonGroup

         dynamicMenu

         group

         menu

         officeMenu

         splitButton

**5、使用内置的菜单控件**

（1）创建一个新的.xlsx文件，并将其保存为Excel Built In menu Example.xlsx。

（2）关闭该文件并在CustomUI Editor中打开。

（3）输入下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“txtabDemo“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“rxgrpDemo“

label=“Demo“>

<menu idMso=“FilePrepareMenu“

size=“large“/>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

下图为上述代码创建的组。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/menusample1.png)

**6、创建自定义菜单控件**

本例中，创建一个菜单，包含一些有用的网站链接，并且将在名为“Ribbon Help”且放置在“开发工具”选项卡末尾的自定义组中包含该菜单。

（1）新建一个工作簿，以.xlsm为扩展名保存后关闭。

（2）在CustomUI Editor中打开该工作簿，并输入下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab idMso=“TabDeveloper“>

<group id=“rxgrpRibbonHelp“

label=“Ribbon Help“>

<menu id=“munResources“

imageMso=“HyperlinkInsert“

size=“large“

label=“Useful Links“>

<menuSeparator id=“rxmSepRibbon“

title=“RibbonX Resources“/>

<button id=“rxbtnMSDN“

label=“MSDN Ribbon Developer Centre“

onAction=“rxsharedLinks\_click“

tag=“http://msdn2.microsoft.com/en-us/office/aa905530.aspx“/>

<button id=“rxbtnKenPuls“

label=“The Ken Puls blog (Excel MVP)“

onAction=“rxsharedLinks\_click“

tag=“http://www.excelguru.ca/blog/2006/12/01/ribbon-example-table-of-contents“/>

<button id=“rxbtnRondeBruin“

label=“Ron deBruin’s Site (Excel MVP)“

onAction=“rxsharedLinks\_click“

tag=“http://www.rondebruin.nl/ribbon.htm“/>

<button id=“rxbtnAccessFreak“

label=“Access Freak (Access MVP)“

onAction=“rxsharedLinks\_click“

tag=“http://www.access-freak.com“/>

<button id=“rxbtnPatrickSchmid“

label=“Patrick Schmid’s RibbonX Forum“

onAction=“rxsharedLinks\_click“

tag=“http://pschmid.net/office2007/forums“/>

<menuSeparator id=“rxmSepAuthors“

title=“Authors Sites“/>

<button id=“rxbtnAuthorRobertMartin“

label=“Robert Martin“

onAction=“rxsharedLinks\_click“

tag=“http://www.msofficegurus.com“/>

<button id=“rxbtnAuthorKenPuls“

label=“Ken Puls“

onAction=“rxsharedLinks\_click“

tag=“http://www.excelguru.ca/“/>

<button id=“rxbtnAuthorTeresaHenning“

label=“Teresa Henning“

onAction=“rxsharedLinks\_click“

tag=“http://www.DataDyanmicsNW.com/“/>

<button id=“rxbtnExcelPerfect“

label=“Excel Perfect“

onAction=“rxsharedLinks\_click“

tag=“http://www.excelperfect.com/“/>

</menu>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

代码创建了一个菜单，列出了九个网站。menuSeparator有效地将这些网站分成了两组。

注意，我们为每个menu控件使用了相同的回调，这使得VBA代码非常简洁。同时，我们将实际的URLs存储在tag属性中。

（3）生成回调签名，并复制代码后关闭CustomUI Editor。

（4）在Excel中打开该工作簿，打开VBE并将回调签名代码粘贴到一个标准模块中。下面编辑该回调以响应希望的操作。

‘rxbtnMSDN onAction回调

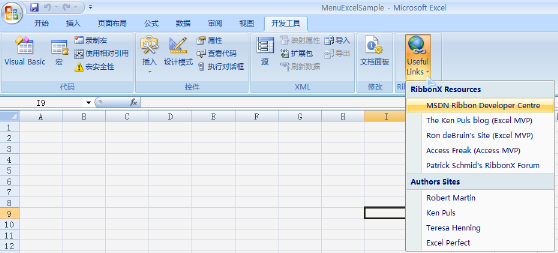
Sub rxsharedLinks\_click(control As IRibbonControl)

ActiveWorkbook.FollowHyperlink Address:=control.Tag, NewWindow:=True

End Sub

代码在新的Web窗口中启动链接。当触发回调时，control对象被传递到过程。该对象具有三个属性：context,id,tag。回调通常查询该控件的ID并相应地作出响应，但在本例中，查询标签，因为标签中存储了URL。接着链接被提供给单击的控件，启动合适的链接。

现在，就可以测试了。切换到Excel界面，单击“开发工具”选项卡，在末尾有一个“Useful Links”的菜单，试着单击其中的菜单项。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/menusample2.png)

下面，再看一个示例。本示例在自定义的菜单中组合一些内置控件以方便使用，其XML代码如下：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“rxgrpViews“

label=“Document Views“>

<menu id=“rxmnuViewsMenu“

itemSize=“normal“

imageMso=“FilePrintPreview“

label=“Views“

size=“large“>

<toggleButton idMso=“ViewPageBreakPreviewView“/>

<toggleButton idMso=“ViewFullScreenView“/>

<checkBox idMso=“ViewHeadings“/>

<checkBox idMso=“ViewFormulaBar“/>

<checkBox idMso=“GridlinesExcel“/>

</menu>

</group>

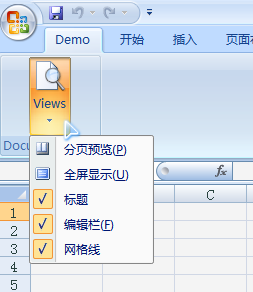
</tab>

</tabs>

</ribbon>

</customUI>

Excel中的自定义界面如下图：

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/menusample3.png)

[**细品RibbonX(28)：splitButton控件详解**](http://www.excelperfect.com/index.php/2009/10/28/ribbonxstudy28/)

2009年10月28日, 1:34 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(目前还没有人投票)

Loading ... Loading ...

咋看起来，splitButton实质上与menu元素相同。与使用菜单一样，当用户单击拆分按钮时，将执行一项命令或者显示额外选项的层级列表。

与菜单在不同的项目之间有线条分隔不一样，splitButton没有这样的格式属性。因此，splitButton通常将相似的命令组合在一起，而将不同的命令组织到逻辑组和子组中。

splitButton胜过menu之处是能够将toggleButton控件作为界面中显示的控件的能力（“脸”控件）。当然，也可以使用按钮控件作为splitButton控件的“脸”控件。

**1、splitButton元素必需的属性**

拆分按钮控件需要下表1所列的id属性之一。

表1：splitButton元素必需的属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的拆分按钮时 |
| idMso | 当使用现有的Microsoft拆分按钮时 |
| idQ | 当在命名空间之间创建共享的拆分按钮时 |

**2、带有回调签名的可选的静态属性和动态属性**

拆分按钮控件可以使用下表2列出的任一insert属性。

表2：splitButton元素可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **INSERT属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之后插入 |
| insertBeforeMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之前插入 |
| insertAfterQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之后插入 |
| insertBeforeQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之前插入 |

也可以为菜单控件设置下表3所列的任何静态属性或等价的动态属性。

表3：splitButton元素可选的属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| enabled | getEnabled | true,false,1,0 | true | Sub GetEnabled(control As IRibbonControl,  ByRef returnedVal) |
| keytip | getKeytip | 1至3个字符 | （none） | Sub GetKeytip(control As  IRibbonControl,  ByRef returnedVal) |
| showLabel | getShowLabel | true,false,1,0 | true | Sub GetShowLabel(control  As IRibbonControl,  ByRef returnedVal) |
| tag | (none) | 1至1024个字符 | (none) | (none) |
| visible | getVisible | true,false,1,0 | true | Sub GetVisible(control As  IRibbonControl,  ByRef returnedVal) |

**3、splitButton元素允许的子对象**

splitButton元素必须包含一个按钮或一个切换按钮（作为该控件的外观），也可以是一个menu元素，但必须在按钮或切换按钮之后定义。

**4、splitButton元素的父对象**

在下列任何控件内都能放置splitButton控件：

         box

         buttonGroup

         dynamicMenu

         group

         menu

         officeMenu

**5、使用内置的splitButton控件**

（1）创建一个新的.xlsx文件，并将其保存为Excel Built In splitButton Example.xlsx。

（2）关闭该文件并在CustomUI Editor中打开。

（3）输入下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon

startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“rxgrpDemo“

label=“Demo“>

<splitButton idMso=“FileSaveAsMenu“

size=“large“/>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

下图为上述代码创建的组。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/splitbuttonsample1.png)

**6、创建自定义splitButton控件**

本例中，添加能够让我们在工作簿中插入新工作表的拆分按钮控件，而不是在工作表标签中右击，选择插入，然后选择工作表类型。即在插入选项卡中使用拆分按钮使用户插入一个新工作表。此外，因为插入工作表可能是一项频繁的任务，所以将新控件放置在功能区的开始。

（1）新建一个工作簿，以.xlsm为扩展名保存后关闭。

（2）在CustomUI Editor中打开该工作簿，并输入下列XML代码：

<customUI

xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon

startFromScratch=“false“>

<tabs>

<!– Enter your first tab here –>

<tab idMso=“TabInsert“>

<group id=“rxgrpInsertSheet“

label=“Sheets“

insertBeforeMso=“GroupInsertTablesExcel“>

<splitButton id=“rxsbtnInsertSheet“

size=“large“>

<button id=“rxbtnSplitFace“

label=“Sheets“

imageMso=“CreateReportFromWizard“/>

<menu id=“mnuInsertSheet“>

<button id=“rxbtnWorksheet“

label=“Insert Worksheet“

imageMso=“GetExternalDataFromText“

onAction=“rxbtnInsertSheet\_click“/>

<button id=“rxbtnChartsheet“

label=“Insert Chart Sheet“

imageMso=“PivotChartType“

onAction=“rxbtnInsertSheet\_click“/>

</menu>

</splitButton>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

注意，splitButton包含一个没有onAction回调的按钮，该按钮的唯一目的是作为拆分按钮的外观图像（“脸”）。

此外，该按钮带有一个菜单，包含两个按钮：Insert Worksheet和Insert Chart Sheet，为方便起见，共享一个回调签名。

（3）生成回调签名，并复制代码后关闭CustomUI Editor。

（4）在Excel中打开该工作簿，打开VBE并将回调签名代码粘贴到一个标准模块中。下面编辑该回调以响应希望的操作。

‘rxbtnWorkSheet onAction回调

Sub rxbtnInsertSheet\_click(control As IRibbonControl)

Select Case control.ID

Case Is = “rxbtnWorksheet”

ActiveWorkbook.Worksheets.Add

Case Is = “rxbtnChartsheet”

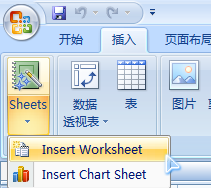
ActiveWorkbook.Charts.Add

End Select

End Sub

如代码所示，设置回调来立即查询已经触发的控件的ID。如果是rxbtnWorksheet控件，那么将插入新工作表；如果是rxbtnChartsheet控件，那么将插入新图表工作表。

现在，该工作簿的“插入”选项卡如下图。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/splitbuttonsample2.png)

下面，我们再看一个示例。该示例沿用细品RibbonX(27)中的菜单示例，看看menu与splitButton控件有什么不同。其XML代码如下：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“rxgrpViews“

label=“Document Views“>

<menu id=“rxmnuViewsMenu“

itemSize=“normal“

imageMso=“FilePrintPreview“

label=“Views Menu“

size=“large“>

<toggleButton idMso=“ViewPageBreakPreviewView“/>

<toggleButton idMso=“ViewFullScreenView“/>

<checkBox idMso=“ViewHeadings“/>

<checkBox idMso=“ViewFormulaBar“/>

<checkBox idMso=“GridlinesExcel“/>

</menu>

<splitButton id=“rxsbtnSplitVersion“

size=“large“

showLabel=“false“>

<button id=“rxbtnSplitVersionFace“

imageMso=“FilePrintPreview“

label=“Split Button Version“/>

<menu id=“rxmnuSplitVersionMenu“

itemSize=“normal“>

<toggleButton idMso=“ViewPageBreakPreviewView“/>

<toggleButton idMso=“ViewFullScreenView“/>

<checkBox idMso=“ViewHeadings“/>

<checkBox idMso=“ViewFormulaBar“/>

<checkBox idMso=“GridlinesExcel“/>

</menu>

</splitButton>

</group>

</tab>

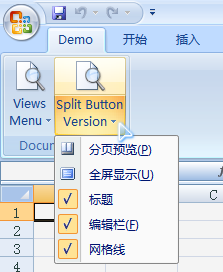
</tabs>

</ribbon>

</customUI>

上述XML代码使用splitButton元素创建了一个基本的菜单。然而，menu元素中的一些良好特征例如menuSeparator不可用于splitButton元素。

Excel中的自定义界面如下图：

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/splitbuttonsample3.png)

能注意到它们之间的不同吗？在splitButton控件中有一条细的水平线将图像和文本分隔开，这条线在menu控件中不明显。

[**细品RibbonX(29)：dynamicMenu元素详解**](http://www.excelperfect.com/index.php/2009/10/30/ribbonxstudy29/)

2009年10月30日, 12:34 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

dynamicMenu元素的主要目的是在运行时创建菜单，通过VBA回调反馈给控件XML代码。dynamicMenu通过使用getContent回调请求需要创建功能区菜单的XML代码来工作。为ID、图像、回调签名和所有其它属性的XML能够由VBA代码编译，并传递给dynamicMenu。然后执行这段代码，就好像是通常已经使用创建功能区定制编码到XML结构里的菜单一样。

dynamicMenu控件能够创建系统中打开的所有工作簿和工作表的最新列表，提供给网络驱动器里所有文件可更新的菜单层级。

在使用dynamicMenu元素时，必须编写VBA代码。即使您只是想动态创建使用内置控件的菜单，仍然需要编译XML，并通过使用VBA回调反馈到dynamicMenu控件。

**1、dynamicMenu元素必需的属性**

dynamicMenu元素需要下表1所列的id属性之一。

表1：dynamicMenu元素必需的属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的动态菜单时 |
| idMso | 当使用现有的Microsoft动态菜单时 |
| idQ | 当在命名空间之间创建共享的动态菜单时 |

此外，每个dynamicMenu必须同时具有下表2所示的回调。

表2：dynamicMenu元素必需的回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **VBA回调签名** |
| （none） | getContent | 1至1096个字符 | （none） | Sub GetContent(control As  IRibbonControl,  ByRef returnedVal) |

**2、带有回调签名的可选的静态属性和动态属性**

dynamicMenu元素可以使用下表3列出的任一insert属性。

表3：dynamicMenu元素可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **INSERT属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之后插入 |
| insertBeforeMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之前插入 |
| insertAfterQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之后插入 |
| insertBeforeQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之前插入 |

也可以为dynamicMenu元素设置下表4所列的任何静态属性或等价的动态属性。

表4：dynamicMenu元素可选的属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| description | getDescription | 1至1024个字符 | （none） | Sub GetDescription(control  As IRibbonControl,  ByRef returnedVal) |
| enabled | getEnabled | true,false,1,0 | true | Sub GetEnabled(control  As IRibbonControl,  ByRef returnedVal) |
| image | getImage | 1至1024个字符 | （none） | Sub GetImage(control  As IRibbonControl,  ByRef returnedVal) |
| imageMso | getImage | 1至1024个字符 | （none） | 同上 |
| keytip | getKeytip | 1至3个字符 | （none） | Sub GetKeytip(control  As IRibbonControl,  ByRef returnedVal) |
| label | getLabel | 1至1024个字符 | （none） | Sub GetLabel(control  As IRibbonControl,  ByRef returnedVal) |
| screentip | getScreentip | 1至1024个字符 | （none） | Sub GetScreentip(control  As IRibbonControl,  ByRef returnedVal) |
| showImage | getShowImage | true,false,1,0 | true | Sub GetShowImage(control  As IRibbonControl,  ByRef returnedVal) |
| showLabel | getShowLabel | true,false,1,0 | true | Sub GetShowLabel(control  As IRibbonControl,  ByRef returnedVal) |
| size | getSize | normal,large | normal | Sub GetSize(control  As IRibbonControl,  ByRef returnedVal) |
| supertip | getSupertip | 1至1024个字符 | (none) | Sub GetSupertip(control  As IRibbonControl,  ByRef returnedVal) |
| tag | (none) | 1至1024个字符 | (none) | (none) |
| visible | getVisible | true,false,1,0 | true | Sub GetVisible(control  As IRibbonControl,  ByRef returnedVal) |

**3、dynamicMenu元素允许的子对象**

dynamicMenu控件接受下列子对象：

         button

         checkbox

         control

         dynamicMenu

         gallery

         menu

         menuSeparator

         splitButton

         toggleButton

**4、dynamicMenu元素的父对象**

在下列任何控件内都能放置dymamicMenu控件：

         box

         buttonGroup

         dynamicMenu

         group

         menu

         officeMenu

**5、创建自定义控件**

前文已提及，创建一个成功的动态菜单的关键是使用VBA代码来为getContent回调提供适合的XML代码。

（1）新建一个工作簿，以.xlsm为扩展名保存后关闭。

（2）在CustomUI Editor中打开该工作簿，并输入下列XML代码：

<customUI

onLoad=“rxIRibbonUI\_onLoad“

xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon

startFromScratch=“false“>

<tabs>

<!– Enter your first tab here –>

<tab idMso=“TabHome“>

<group id=“rxgrpNewFile“

label=“New File“

insertBeforeMso=“GroupClipboard“>

<dynamicMenu id=“rxdmnuTemplates“

label=“Create from…“

imageMso=“CreateReportFromWizard“

size=“large“

getContent=“rxdmnuTemplates\_getContent“/>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

如上所示，用于创建动态菜单的XML代码实际上相当简短。提供了标签和大图像作为菜单的外观，同时需要getContent回调来在运行时创建实际的菜单项。

同时，要为用户提供刷新该菜单的功能，需要使功能区无效，因此需要对customUI元素添加onLoad属性。

（3）生成回调签名，复制代码后关闭CustomUI Editor。

（4）在Excel中打开该工作簿，打开VBE并将回调签名代码粘贴到一个标准模块中。下面编辑回调代码以响应期望的操作。

onLoad回调捕获RibbonUI对象：

Dim rxIRibbonUI As IRibbonUI

Sub rxIRibbonUI\_onLoad(ribbon As IRibbonUI)

‘Callback for onLoad to capture RibbonUI

Set rxIRibbonUI = ribbon

End Sub

下一步需要为动态菜单创建getContent回调。

Sub rxdmnuTemplates\_GetContent(control As IRibbonControl, ByRef content)

‘Callback for GetContent to return XML used to create dynamicMenu

Dim objFSO As Object

Dim objTemplateFolder As Object

Dim file As Object

Dim sXML As String

Dim lBtnCount As Long

‘Create FSO object

Set objFSO = CreateObject(“Scripting.FileSystemObject”)

‘Set FSO object to templates folder

Set objTemplateFolder = objFSO.getfolder(Application.TemplatesPath)

‘Open the XML string

sXML = “<menu xmlns=”"http://schemas.microsoft.com/office/2006/01/customui”">”

‘Add template files

If objTemplateFolder.Files.Count > 0 Then

For Each file In objTemplateFolder.Files

If Not Left(file.Name, 2) = “~$” Then

Select Case LCase(Right(file.Name, 4))

Case “.xlt”, “xltx”, “xltm”

‘Excel template.

sXML = sXML & \_

“<button id=”"rxbtnDyna” & lBtnCount & “”" ” & \_

“label=”"” & file.Name & “”" ” & \_

“imageMso=”"FileSaveAsExcel97\_2003″” ” & \_

“tag=”"” & file.Path & “”" ” & \_

“onAction=”"rxbtnDyna\_onAction”"/>” & vbCrLf

lBtnCount = lBtnCount + 1

Case Else

‘Unknown format. Ignore.

End Select

End If

Next file

End If

‘Release the FSO objects

Set file = Nothing

Set objTemplateFolder = Nothing

Set objFSO = Nothing

‘Check if any items buttons were created and create a “No Templates” button if not

If lBtnCount = 0 Then \_

sXML = sXML & “<button id=”"rxbtnDyna0″” label=”"No Templates Found”"/>”

‘Add Refresh button & close the menu tags

sXML = sXML & \_

“<button id=”"rxbtnRefresh”" ” & \_

“label=”"Refresh List”" ” & \_

“imageMso=”"RecurrenceEdit”" ” & \_

“onAction=”"rxbtnDyna\_onAction”"/>” & \_

“</menu>”

‘Feed the XML back to the Ribbon

content = sXML

End Sub

代码看起来很复杂！代码的开头创建FSO对象，设置文件夹对象引用模板文件夹，以便后面的代码查看文件夹中的每个文件。此外，代码中检查以“~$”开始的文件以排除临时文件，使用LCase()函数确保不会遇到区分大小写的问题。

下面主要看看传回给dynamicMenu元素的XML字符串（包含在sXML变量中）。XML字符串包含完整的菜单层次，必须以特定的XML命名空间开始。像customUI元素，每个动态菜单回调必须包括这行以便Ribbon知道如何解释代码并编译。接着是在菜单中添加按钮的XML代码，变量lBtnCount代码添加的按钮数量。

在VBA字符串里使用双引号（”）需要两个双引号。因此，第一个双引号实际上表明接下来的双引号应视为文本。例如，为了得到sXML值id=”rxbtnHello”，需要下面的VBA代码字符串：sXML=”id=””rxbtnHello”””。因此，应注意双引号的规范编写，否则在编写XML和VBA代码时会导致错误。

代码也评估添加到XML字符串中的按钮数。如果lBtnCount变量的值为0，则不需要添加任何按钮，因此创建一个按钮让用户知道没有找到模板。

您还希望用户能够刷新按钮，例如可能创建了一个新模板并想在列表中看到该模板，因此添加相应的XML代码。

从根本上说，dynamicMenu的getContent回调的真实目的是对dynamicMenu对象返回格式良好的XML代码，以便于创建菜单。

此外，还需要创建一个回调处理所创建的按钮：

Sub rxbtnDyna\_onAction(control As IRibbonControl)

‘Callback for button onAction

If control.ID = “rxbtnRefresh” Then

rxIRibbonUI.InvalidateControl (“rxdmnuTemplates”)

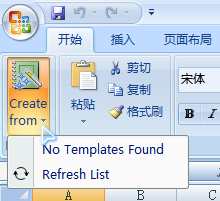
Else

Workbooks.Add (control.Tag)

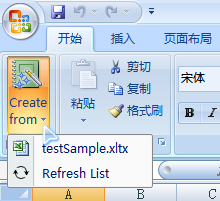
End If

End Sub

该过程检查是否调用Refresh按钮，从而更新列表。如果不是，则基于存储在控件的tag属性中的模板创建新的工作簿。结果如下图所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/dynamicmenusample1.png)

没有模板文件！因为模板文件夹中没有模板。此时，创建一个新模板，例如testSample.xlst。然后回到自定义动态菜单的文件，单击“Refresh List”按钮，此时将看到新模板已出现在列表中，如下图所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/10/dynamicmenusample2.png)

[**细品RibbonX(30)：box元素详解**](http://www.excelperfect.com/index.php/2009/11/02/ribbonxstudy30/)

2009年11月2日, 1:48 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

box元素用来在组里指定的控件周围放置一个可视的框，其主要目的是将控件作为一个单元组合在一起。

通常情况下，分配到组中的每个控件都被放置在先前的控件下面直到该列被填满，然后下一个控件被放置在其右侧列的顶行。然而，通过在框里面组合命令，可以将几个控件视作一个整体，并且一次将整个组放置在功能区。这有助于我们方便地组织显示的顺序，不会使界面混乱，或者留下一些空白。

**1、box元素必需的属性**

box元素需要下表1所列的id属性之一。

表1：box元素必需的属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的框时 |
| idQ | 当在命名空间之间创建共享的框时 |

**2、带有回调签名的可选的静态属性和动态属性**

虽然指定位置是可选的，但是如果选择相对于另一个元素决定框的位置，那么必须使用下表2列出的任一insert属性。

表2：box元素可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **INSERT属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之后插入 |
| insertBeforeMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之前插入 |
| insertAfterQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之后插入 |
| insertBeforeQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之前插入 |

也可以为box元素设置下表3所列的可选的属性和回调。

表3：box元素可选的属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| boxStyle | (none) | horizontal,vertical | horizontal | (none) |
| visible | getVisible | true,false,1,0 | true | Sub GetVisible(control  As IRibbonControl,  ByRef returnedVal) |

**3、box元素允许的子对象**

box控件能够包含各种各样的RibbonX控件：

         box

         button

         buttonGroup

         checkbox

         comboBox

         control

         dropDown

         dynamicMenu

         editBox

         gallery

         labelControl

         menu

         splitButton

         toggleButton

注意，box控件也可以包含其它box控件，称之为嵌套。

**4、box元素的父对象**

在下列任何控件内都能放置box控件：

         box

         group

**5、创建自定义的box元素**

**示例1**

本示例演示使用水平的box控件的效果。

（1）创建一个新的.xlsx文件，并将其保存为Excel box Example.xlsx。

（2）关闭该文件并在CustomUI Editor中打开。

（3）输入下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon>

<tabs>

<tab id = “rxtab\_Demo“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“rxgrp\_Demo“

label=“Demo Group“>

<box id=“rxboxFormat1“

boxStyle=“horizontal“

visible=“true“>

<toggleButton idMso=“Bold“/>

<toggleButton idMso=“Italic“/>

<toggleButton idMso=“Underline“/>

<toggleButton idMso=“UnderlineDouble“/>

</box>

<button id=“rxbtnHappy1“

imageMso=“HappyFace“

label=“Button 1“/>

<button id=“rxbtnHappy2“

imageMso=“HappyFace“

label=“Button 2“/>

<button id=“rxbtnHappy3“

imageMso=“HappyFace“

label=“Button 3“/>

<button id=“rxbtnHappy4“

imageMso=“HappyFace“

label=“Button 4“/>

</group>

</tab>

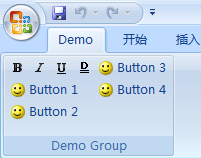
</tabs>

</ribbon>

</customUI>

（4）保存后关闭CustomUI Editor。

（5）在Excel中重新打开该文件，下图即为上述XML代码创建的组。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/boxsample1.png)

创建了一个水平框将四个格式元素组合在一起，允许其作为一个单元格移动。除了允许按钮跨越一列外，水平组也在Button1和Button2的右侧创建了“空白”，而Button3和Button4则没有。

**示例2**

（6）关闭Excel box Example.xlsx。

（7）重新在CustomUI Editor中打开该文件。

（8）修改XML代码如下：

将

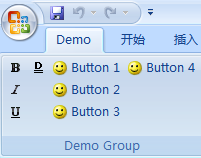
boxStyle=“horizontal“

修改为：

boxStyle=“vertical“

（9）保存后关闭CustomUI Editor。

（10）在Excel中重新打开该文件，下图即为上述XML代码创建的组。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/boxsample2.png)

**示例3**

本示例除了演示嵌套的框控件外，还将展示使用getVisible回调控制框的可视性的效果。

示例中包含两个box控件，每一个包含两个前面示例中使用的格式控件。此外，每个框都宿于其父框里，允许用户在功能区中预留垂直空间以便没有其它按钮在这些控件下面。同时，使用三个复选框来选择哪个框可见。

（1）新建一个工作簿，以.xlsm为扩展名保存后关闭。

（2）在CustomUI Editor中打开该工作簿，并输入下列XML代码：

<customUI onLoad=“rxIRibbonUI\_onLoad“ xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“

insertBeforeMso=“TabHome“

label=“Demo“>

<group id=“rxgrpDemo“

label=“Demo Group“>

<box id=“rxbox1“

boxStyle=“vertical“

getVisible=“rxboxshared\_getVisible“>

<box id=“rxbox11“

boxStyle=“horizontal“

getVisible=“rxboxshared\_getVisible“>

<toggleButton idMso=“Bold“/>

<toggleButton idMso=“Italic“/>

</box>

<box id=“rxbox12“

boxStyle=“horizontal“

getVisible=“rxboxshared\_getVisible“>

<toggleButton idMso=“Underline“/>

<toggleButton idMso=“UnderlineDouble“/>

</box>

</box>

<checkBox id=“rxchkVisibleBox1“

label=“Box 1 Visible?“

getPressed=“rxchkShared\_pressed“

onAction=“rxchkShared\_click“/>

<checkBox id=“rxchkVisibleBox11“

label=“Box 1-1 Visible?“

getPressed=“rxchkShared\_pressed“

onAction=“rxchkShared\_click“/>

<checkBox id=“rxchkVisibleBox12“

label=“Box 1-2 Visible?“

getPressed=“rxchkShared\_pressed“

onAction=“rxchkShared\_click“/>

<button id=“rxbtnReset“

imageMso=“HappyFace“

label=“Reset All“

onAction=“rxbtnReset\_click“/>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

注意，本示例实际上共有三个box控件，其它两个嵌套在第一个中。三个控件共享回调。当然，需要能够使功能区无效以迫使box控件隐藏或显示，因此需要捕获RibbonUI对象。

（3）生成回调签名并复制后，关闭CustomUI Editor。

（4）重新打开该工作簿，打开VBE，将回调签名代码粘贴到标准模块，然后开始完善代码。

首先，在模块顶部声明公共变量：

Dim rxIRibbonUI As IRibbonUI

Dim bBox1\_Visible As Boolean

Dim bBox11\_Visible As Boolean

Dim bBox12\_Visible As Boolean

第1行代码中的变量用于存储RibbonUI对象，能够在后面使功能区无效。其它三个变量包含不同组的可视状态。由于通过复选框控件控制，因此使用这些变量存储与每个框相关的复选框的状态。

除了在装载时捕获RibbonUI外，需要确保每个框控件是可见的。因此，onLoad回调代码如下：

‘customUI.onLoad回调

Sub rxIRibbonUI\_onLoad(ribbon As IRibbonUI)

Set rxIRibbonUI = ribbon

bBox1\_Visible = True

bBox11\_Visible = True

bBox12\_Visible = True

End Sub

接下来，设置box控件的共享的getVisible回调：

‘rxbox1,rxbox11,rxbox12 getVisible共享回调

Sub rxboxshared\_getVisible(control As IRibbonControl, ByRef returnedVal)

Select Case control.ID

Case “rxbox1″

returnedVal = bBox1\_Visible

Case “rxbox11″

returnedVal = bBox11\_Visible

Case “rxbox12″

returnedVal = bBox12\_Visible

End Select

End Sub

case语句评估触发回调的控件，从变量中获取合适的值并返回给RibbonUI。getPressed回调基于该变量处理复选框控件。

‘rxchkVisibleBox1,-Box11,-Box12 getPressed共享的回调

Sub rxchkShared\_pressed(control As IRibbonControl, ByRef returnedVal)

Select Case control.ID

Case “rxchkVisibleBox1″

returnedVal = bBox1\_Visible

Case “rxchkVisibleBox11″

returnedVal = bBox11\_Visible

Case “rxchkVisibleBox12″

returnedVal = bBox12\_Visible

End Select

End Sub

接下来，创建当单击复选框时触发的程序。修改可见的变量的值为当前“按下”状态（true或false），然后触发功能区的无效。这种无效不仅仅触发每一个box控件的getVisible回调，也确保所有的复选框控件更新：

‘rxchkVisibleBox1,-Box11,-Box12 onAction共享的回调

Sub rxchkShared\_click(control As IRibbonControl, pressed As Boolean)

Select Case control.ID

Case “rxchkVisibleBox1″

bBox1\_Visible = pressed

Case “rxchkVisibleBox11″

bBox11\_Visible = pressed

Case “rxchkVisibleBox12″

bBox12\_Visible = pressed

End Select

rxIRibbonUI.Invalidate

End Sub

最后是添加的按钮的onAction回调，便于设置每个控件为默认的可见状态：

‘rxbtnReset onAction共享回调使按钮可见

Sub rxbtnReset\_click(control As IRibbonControl)

bBox1\_Visible = True

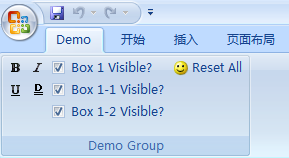
bBox11\_Visible = True

bBox12\_Visible = True

rxIRibbonUI.Invalidate

End Sub

好了！该测试用户界面效果了。保存代码并关闭工作簿，再重新打开该工作簿，如下图所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/boxsample3.png)

试着清除复选框，相应的控件将消失。单击“Reset All”将使所有控件恢复可见。

注意，试图使用getVisible回调设置父框为visible=”false”，将不会像您想像的那样隐藏父框和所有的子控件。相反，父和子box控件将保持当前状态，控件本身是可见的并且工作，而任何试图通过回调对嵌套控件状态的修改都将忽略。这种影响将保留到父框控件的getVisible回调重新被设置为True。

隐藏父box控件的唯一办法是隐藏所有的子项。此时，父框控件将折叠并且不可见，即使其visible属性被设置为True。

[**细品RibbonX(31)：buttonGroup元素详解**](http://www.excelperfect.com/index.php/2009/11/03/ribbonxstudy31/)

2009年11月3日, 2:05 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(目前还没有人投票)

Loading ... Loading ...

buttonGroup元素与box元素相似，但它也有一些独特的特征。buttonGroup控件和box控件之间在外观上最大的不同是box控件不会在组周围放置可视的边框，而buttonGroup元素实际上显示边框。buttonGroup元素不会垂直排列，并且与box不同，buttonGroup不会接受box或buttonGroup作为其子元素。

box元素和buttonGroup元素之间主要的不同归纳于下表1。

表1：box元素和buttonGroup元素之间的不同

|  |  |  |
| --- | --- | --- |
| **功能** | **Box** | **ButtonGroup** |
| 水平排列控件 | √ | √ |
| 垂直排列控件 | √ |  |
| 接受嵌套的box控件 | √ |  |
| 接受嵌套的buttonGroup控件 | √ |  |
| 在控件周围放置可见的轮廓线 |  | √ |

buttonGroup是唯一允许在控件集合周边放置可见的边框的控件。

**1、buttonGroup元素必需的属性**

buttonGroup元素需要下表2所列的id属性之一。

表2：buttonGroup元素必需的属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的buttonGroup时 |
| idQ | 当在命名空间之间创建共享的buttonGroup时 |

**2、带有回调签名的可选的静态属性和动态属性**

为了相对于一个已存在的控件决定buttonGroup的位置，那么必须使用下表3列出的任一insert属性。

表3：buttonGroup元素可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **INSERT属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之后插入 |
| insertBeforeMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之前插入 |
| insertAfterQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之后插入 |
| insertBeforeQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之前插入 |

也可以为buttonGroup元素设置下表4所列的可选的属性和回调。

表4：buttonGroup元素可选的属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| visible | getVisible | true,false,1,0 | true | Sub GetVisible(control  As IRibbonControl,  ByRef returnedVal) |

当buttonGroup具有getVisible回调可用时，有一个bug。如果其值设置为false，该bug完全妨碍该回调，然而当其值设置为true时，该回调会正确地工作。

**3、buttonGroup元素允许的子对象**

buttonGroup元素能够包含下列任一或所有的控件：

         button

         control

         dynamicMenu

         gallery

         menu

         splitButton

         toggleButton

**4、buttonGroup元素的父对象**

在下列任何控件内都能放置buttonGroup元素：

         box

         group

**5、创建自定义的buttonGroup元素**

（1）创建一个新的.xlsx文件，并将其保存为Excel buttonGroup Example.xlsx。

（2）关闭该文件并在CustomUI Editor中打开。

（3）输入下列XML代码：

<customUI

xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon>

<tabs>

<tab id = “rxtab\_Demo“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“rxgrp\_Demo“

label=“Demo Group“>

<box id=“rxboxCustom“

boxStyle=“vertical“> <buttonGroup id=“rxbgrpMsoControls“>

<toggleButton idMso=“Bold“/>

<toggleButton idMso=“Italic“/>

<toggleButton idMso=“Underline“/>

<toggleButton idMso=“UnderlineDouble“

showLabel=“false“/>

</buttonGroup>

</box>

<button id=“rxbtnHappy1“

imageMso=“HappyFace“

label=“Button 1“/>

<button id=“rxbtnHappy2“

imageMso=“HappyFace“

label=“Button 2“/>

<button id=“rxbtnHappy3“

imageMso=“HappyFace“

label=“Button 3“/>

</group>

</tab>

</tabs>

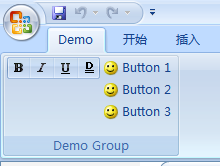
</ribbon>

</customUI>

注意，buttonGroup控件被封装在垂直的box控件里。正如在介绍box控件中所提到的，使用box控件能够迫使所有的按钮控件到下一列，从而使您的控件下保留空白。

（4）保存后关闭CustomUI Editor。

（5）在Excel中重新打开该文件，下图即为上述XML代码创建的组。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/buttongroupsample1.png)

注意到这些在水平按钮组中的字体控件，成一条线地显示在垂直的box控件顶部。垂直控件为其内容保留了完整的区间，因此三个笑脸按钮显示在其右侧的一列。

[**细品RibbonX(32)：labelControl元素详解**](http://www.excelperfect.com/index.php/2009/11/06/ribbonxstudy32/)

2009年11月6日, 12:55 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(目前还没有人投票)

Loading ... Loading ...

labelControl为开发者提供了一种在功能区中显示文本标签的方式。这类控件没有行为，通常用作其它控件的标题或描述。常常用来为在一列中排列的按钮提供上下文。

**1、labelControl元素必需的属性**

labelControl元素需要下表1所列的id属性之一。

表1：labelControl元素必需的属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的labelControl时 |
| idMso | 当使用已有的Microsoft labelControl时 |
| idQ | 当在命名空间之间创建共享的labelControl时 |

**2、带有回调签名的可选的静态属性和动态属性**

为了相对于另一个控件决定labelControl的位置，使用下表2列出的任一insert属性。

表2：labelControl元素可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **INSERT属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之后插入 |
| insertBeforeMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之前插入 |
| insertAfterQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之后插入 |
| insertBeforeQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之前插入 |

也可以为labelControl元素设置下表3所列任意属性或等价的回调的组合。

表3：labelControl元素可选的属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| enabled | getEnabled | true,false,1,0 | true | Sub GetEnabled(control  As IRibbonControl,  ByRef returnedVal) |
| label | getLabel | 1至1024个字符 | （none） | Sub GetLabel(control As  IRibbonControl,  ByRef returnedVal) |
| screentip | getScreentip | 1至1024个字符 | （none） | Sub GetScreentip(control  As IRibbonControl,  ByRef returnedVal) |
| showLabel | getShowLabel | true,false,1,0 | true | Sub GetShowLabel(control  As IRibbonControl,  ByRef returnedVal) |
| supertip | getSupertip | 1至1024个字符 | （none） | Sub GetSupertip(control  As IRibbonControl,  ByRef returnedVal) |
| tag | （none） | 1至1024个字符 | （none） | （none） |
| visible | getVisible | true,false | true | Sub GetVisible(control  As IRibbonControl,  ByRef returnedVal) |

**3、labelControl元素允许的子对象**

labelControl元素不支持任何类型的子控件。

**4、labelControl元素的父对象**

在下列任何控件内都能放置labelControl元素：

         box

         group

**5、使用内置的labelControl元素**

Microsoft实际上在Excel中公开了6个labelControl元素。然而，创建自定义labelControl更快且更直接。

**6、创建自定义的labelControl元素**

labelControl的一个相当酷的使用是将其作为标记。

（1）创建一个新的.xlsm文件，并将其保存为Excel labelControl Example.xlsm。

（2）关闭该文件并在CustomUI Editor中打开。

（3）输入下列XML代码：

<customUI onLoad=“rxIRibbonUI\_onLoad“ xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“

insertBeforeMso=“TabHome“

label=“Demo“>

<group id=“rxgrpDemo“

label=“Demo Group“>

<labelControl id=“rxlblFeedback“

getLabel=“rxlblFeedback\_getLabel“/>

<button id=“rxbtnProcess“

getImage=“rxbtnProcess\_getImage“

onAction=“rxbtnProcess\_click“/>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

（4）生成回调签名，保存后关闭CustomUI Editor。

（5）在Excel中重新打开该文件，在VBE的标准模块中粘贴回调签名。

在编辑回调代码之前，让我们先考虑一个完整的操作过程。我们希望在单击按钮之前标签去读取Process Accounts，随后是Accounts Complete。我们需要一个变量，存储表明按钮是否被单击，因此在模块的开始声明该变量：

Dim bButtonClicked As Boolean

当然，还需要包括一个存储RibbonUI的全局变量。现在，创建onLoad回调代码：

Dim rxIRibbonUI As IRibbonUI

‘customUI.onLoad回调

Sub rxIRibbonUI\_onLoad(ribbon As IRibbonUI)

Set rxIRibbonUI = ribbon

End Sub

接下来，为labelControl创建getLabel回调。评估是否单击按钮的最容易的方法是检查bButtonClicked变量。使用Select Case语句，如下所示：

‘rxlblFeedback getLabel回调

Sub rxlblFeedback\_getLabel(control As IRibbonControl, ByRef returnedVal)

Select Case bButtonClicked

Case False

returnedVal = “Process Accounts”

Case True

returnedVal = “Accounts Complete”

End Select

End Sub

同样，通过Select语句的评估决定显示图像或标签：

‘rxbtnProcess getImage回调

Sub rxbtnProcess\_getImage(control As IRibbonControl, ByRef returnedVal)

Select Case bButtonClicked

Case False

returnedVal = “CreateReportFromWizard”

Case True

returnedVal = “DeclineInvitation”

End Select

End Sub

最后，来处理按钮。同样，检查bButtonClicked的值，如果为False，那么修改bButtonClicked的值为true，运行程序，然后使功能区无效。当然，无效将触发重建，控件将会更新以显示用户已经实施的操作。如果bButtonClicked的值已经是true，那么告诉用户他们已经完成了程序。代码如下：

‘rxbtnProcess onAction回调

Sub rxbtnProcess\_click(control As IRibbonControl)

Select Case bButtonClicked

Case False

bButtonClicked = True

‘Code to process accounts goes here

rxIRibbonUI.Invalidate

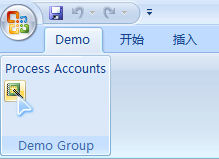
Case True

MsgBox “已经运行了这个程序!”

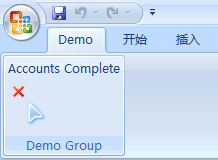
End Select

End Sub

（6）保存并关闭该工作簿，然后重新打开该工作簿，结果如下图所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/labelcontrolsample1.png)

单击该按钮，该组将更新为下图所示的结果。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/labelcontrolsample2.png)

再次单击该按钮，将会弹出一条消息框。

正如上述代码所示，我们使用了标签和图形来唤起用户对状态的注意，或者状态的改变。这是一种技巧，即使用labelControl来警告用户对象或事件的状态。

[**细品RibbonX(33)：separator元素详解**](http://www.excelperfect.com/index.php/2009/11/10/ribbonxstudy33/)

2009年11月10日, 12:31 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(目前还没有人投票)

Loading ... Loading ...

separator控件作为一条垂直线出现，分开组中的项目，从组的顶部跨越到组的底部，并且不能够设置任何其它的方向或尺寸。然而，它能够在功能区中提供空白，与使用垂直的box控件创建的空白相似。

**1、separator元素必需的属性**

separator元素需要下表1所列的id属性之一。

表1：separator元素必需的属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的分隔条时 |
| idQ | 当在命名空间之间创建共享的分隔条时 |

**2、带有回调签名的可选的静态属性和动态属性**

separator控件相对于已存在的控件的放置需要使用下表2列出的insert属性之一。

表2：separator元素可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **INSERT属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之后插入 |
| insertBeforeMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之前插入 |
| insertAfterQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之后插入 |
| insertBeforeQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之前插入 |

separator元素接受下表3所列的静态可见的属性或等价的回调。

表3：separator元素可选的属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| visible | getVisible | true,false,1,0 | true | Sub GetVisible(control  As IRibbonControl,  ByRef returnedVal) |

**3、separator元素允许的子对象**

separator元素不支持任何类型的子控件。

**4、separator元素的父对象**

在下列任何控件内都能放置separator控件：

         documentControl

         group

         sharedControl

**5、创建自定义的separator元素**

下面的示例演示如何使用分隔条来在功能区中创建空白。

（1）创建一个新的.xlsx文件，并将其保存为Excel separator Example.xlsx。

（2）关闭该文件并在CustomUI Editor中打开。

（3）输入下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon>

<tabs>

<tab id = “rxtab\_Demo“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“rxgrp\_Demo“

label=“Demo Group“>

<box id=“rxboxFormat1“

boxStyle=“horizontal“

visible=“true“>

<toggleButton idMso=“Bold“/>

<toggleButton idMso=“Italic“/>

<toggleButton idMso=“Underline“/>

<toggleButton idMso=“UnderlineDouble“/>

</box>

<separator id=“rxsep1“/>

<button id=“rxbtnHappy1“

imageMso=“HappyFace“

label=“Button 1“/>

<button id=“rxbtnHappy2“

imageMso=“HappyFace“

label=“Button 2“/>

<button id=“rxbtnHappy3“

imageMso=“HappyFace“

label=“Button 3“/>

<button id=“rxbtnHappy4“

imageMso=“HappyFace“

label=“Button 4“/>

</group>

</tab>

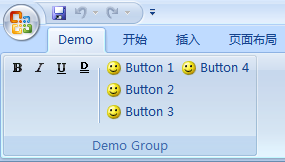
</tabs>

</ribbon>

</customUI>

（4）保存后关闭CustomUI Editor。

（5）在Excel中重新打开该文件，结果如下图所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/separatorsample.png)

[**细品RibbonX(34)：menuSeparator元素详解**](http://www.excelperfect.com/index.php/2009/11/12/ribbonxstudy34/)

2009年11月12日, 1:19 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(目前还没有人投票)

Loading ... Loading ...

menuSeparator元素有些像separator元素和labelControl元素的混合，可以为特定类型的菜单控件创建一条水平线，并且能够提供文本。这使得能够在菜单项之间添加标题。

**1、menuSeparator元素必需的属性**

menuSeparator元素需要下表1所列的id属性之一。

表1：menuSeparator元素必需的属性

|  |  |
| --- | --- |
| **属性** | **何时使用** |
| id | 当创建自已的菜单分隔条时 |
| idQ | 当在命名空间之间创建共享的菜单分隔条时 |

**2、带有回调签名的可选的静态属性和动态属性**

menuSeparator控件相对于已存在的控件的放置需要使用下表2列出的insert属性之一。

表2：menuSeparator元素可选的insert属性

|  |  |  |  |
| --- | --- | --- | --- |
| **INSERT属性** | **允许值** | **默认值** | **何时使用** |
| insertAfterMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之后插入 |
| insertBeforeMso | 有效的Mso组 | 在组末尾插入 | 在Microsoft控件之前插入 |
| insertAfterQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之后插入 |
| insertBeforeQ | 有效的组idQ | 在组末尾插入 | 在共享的命名空间控件之前插入 |

menuSeparator元素接受下表3所列的title元素或等价的回调。

表3：menuSeparator元素可选的属性和回调

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **静态属性** | **动态属性** | **允许值** | **默认值** | **动态属性的VBA回调签名** |
| title | getTitle | 1至1024个字符 | line | Sub GetTitle(control As  IRibbonControl,  ByRef returnedVal) |

**3、menuSeparator元素允许的子对象**

menuSeparator元素不支持任何类型的子控件。

**4、menuSeparator元素的父对象**

在下列任何控件内都能放置menuSeparator控件：

         menu

         officeMenu

         dynamicMenu

**5、创建自定义的menuSeparator元素**

（1）创建一个新的.xlsx文件，并将其保存为Excel menuSeparator Example.xlsx。

（2）关闭该文件并在CustomUI Editor中打开。

（3）输入下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon startFromScratch=“false“>

<tabs>

<tab id=“tabDemo“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“grpSaveFiles“

label=“Save File“>

<menu id=“mnuNewFile“

label=“Save As File Type“

size=“large“

imageMso=“FileCompatibilityChecker“>

<menuSeparator id=“rxmSep01“

title=“Save in 2007 format“/>

<button idMso=“FileSaveAsExcelXlsx“/>

<button idMso=“FileSaveAsExcelXlsxMacro“/>

<menuSeparator id=“rxmSep02“/>

<button idMso=“FileSaveAsExcelXlsb“/>

<menuSeparator id=“rxmSep03“

title=“Save in 97-2003 format“/>

<button idMso=“FileSaveAsExcel97\_2003“/>

<menuSeparator id=“rxmSep04“

title=“Save in other format“/>

<button idMso=“FileSaveAsOtherFormats“/>

</menu>

</group>

</tab>

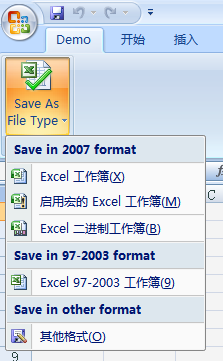
</tabs>

</ribbon>

</customUI>

注意，rxmSep02没有指定title属性，这将创建一条细线来分隔菜单项，而不是带有线条的标题栏。

（4）保存后关闭CustomUI Editor。

（5）在Excel中重新打开该文件，结果如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/menuseparatorsample1.png)

[**细品RibbonX(35)：dialogBoxLauncher元素详解**](http://www.excelperfect.com/index.php/2009/11/13/ribbonxstudy35/)

2009年11月13日, 1:44 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(目前还没有人投票)

Loading ... Loading ...

dialogBoxLauncher是一个小的、几乎看不见的、能够在组中使用的“小玩意”。在界面上是一个带有箭头的小正方形，位于组的右下角。

在大多数情形下，dialogBoxLauncher通过启动内置对话框，提供给用户更多的选项。当然，您也可以创建自已的用户窗体，提供给用户额外的选项或者帮助文档。

**1、dialogBoxLauncher元素必需和可选的属性**

dialogBoxLauncher元素没有任何必需的和可选的属性，其依赖于拥有属性的子对象。

**2、dialogBoxLauncher元素允许的子对象**

dialogBoxLauncher是一个容器，因此不仅接受子对象，而且需要子对象实现功能。dialogBoxLauncher必须具有且仅有一个按钮作为其子对象。实际上，按钮的作用是启动应用程序对话框。

**3、dialogBoxLauncher元素的父对象**

使用dialogBoxLauncher的唯一的地方是组。

**4、dialogBoxLauncher元素使用示例**

在UI中，有三种不同的方式实现dialogBoxLauncher：

         使用Microsoft已经使其可用的缺省的对话框启动器之一

         创建自定义启动器来使用Microsoft的其它对话框之一

         创建自定义启动器来启动自定义用户窗体

**内置的对话框启动器**

在自定义组中添加对话框启动器最简单的方式是使用Microsoft内置的对话框启动器按钮作为其子对象。

（1）创建一个新的.xlsx文件，并将其保存为Excel dialogBoxLauncher Example.xlsx。

（2）关闭该文件并在CustomUI Editor中打开。

（3）输入下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon

startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“rxgrpTest“

label=“Test“>

<box id=“rxboxFormat“>

<comboBox idMso=“Font“ />

<comboBox idMso=“FontSize“ />

</box>

<dialogBoxLauncher>

<button idMso=“FormatCellsFontDialog“/>

</dialogBoxLauncher>

</group>

</tab>

</tabs>

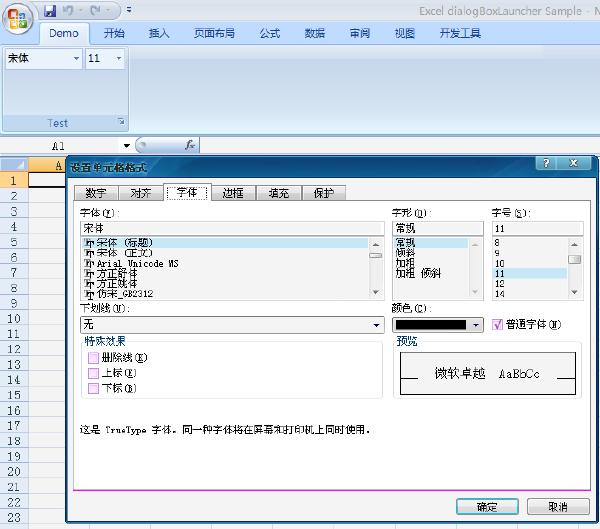
</ribbon>

</customUI>

注意，dialogLauncher仅仅是包含一个按钮的壳。使用dialogBoxLauncher的技巧是确保使用了正确的内置对话框按钮。

（4）保存后关闭CustomUI Editor。

（5）在Excel中重新打开该文件，单击“Test”组右下角的对话框启动器，结果如下图所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/dialogboxlaunchersample1.png)

**使用内置对话框的自定义对话框启动器**

（1）创建一个新的.xlsm文件，并将其保存为Excel dialogBoxLauncher Example1.xlsm。

（2）关闭该文件并在CustomUI Editor中打开。

（3）输入下列XML代码：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon

startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“rxgrpTest“

label=“Test“>

<box id=“rxboxFormat“>

<comboBox idMso=“Font“ />

<comboBox idMso=“FontSize“ />

</box>

<dialogBoxLauncher>

<button id=“rxbtnDialog“

onAction=“rxbtnDialog\_click“

screentip=“Launch Dialog“/>

</dialogBoxLauncher>

</group>

</tab>

</tabs>

</ribbon>

</customUI>

注意，没有与按钮相关的图像。即使您提供了按钮图像，它也会被对话框启动器图像覆盖。

（4）生成回调签名并复制，保存后关闭CustomUI Editor。

（5）在Excel中重新打开该文件，打开VBE，插入一个新模块，粘贴回调签名代码。

（6）编写回调代码：

‘rxbtnDialog onAction回调

Sub rxbtnDialog\_click(control As IRibbonControl)

Application.Dialogs(xlDialogFormatFont).Show

End Sub

编写代码较困难的是找到合适的对话框常量（本例中为xlDialogFormatFont），您可以使用对象浏览器或者智能感知功能来查找。

（7）关闭工作簿后重新打开。单击“Test”组右下角的对话框启动器，将出现“字体”对话框。

当然，您也可以修改回调代码中的对话框常量，调用其它对话框。

**使用自定义用户窗体的自定义对话框**

（1）创建一个新的.xlsm文件，并将其保存为Excel dialogBoxLauncher Example2.xlsm。

（2）关闭该文件并在CustomUI Editor中打开。

（3）输入下列XML代码（即沿用前一示例）：

<customUI xmlns=“http://schemas.microsoft.com/office/2006/01/customui“>

<ribbon

startFromScratch=“false“>

<tabs>

<tab id=“rxtabDemo“

label=“Demo“

insertBeforeMso=“TabHome“>

<group id=“rxgrpTest“

label=“Test“>

<box id=“rxboxFormat“>

<comboBox idMso=“Font“ />

<comboBox idMso=“FontSize“ />

</box>

<dialogBoxLauncher>

<button id=“rxbtnDialog“

onAction=“rxbtnDialog\_click“

screentip=“Launch Dialog“/>

</dialogBoxLauncher>

</group>

</tab>

</tabs>

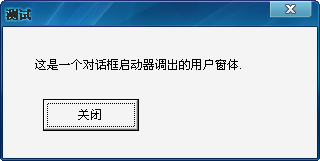
</ribbon>

</customUI>

（4）生成回调签名并复制，保存后关闭CustomUI Editor。

（5）在Excel中重新打开该文件，打开VBE，插入一个新模块，粘贴回调签名代码。

（6）插入一个新的用户窗体，在其中添加标签和按钮。修改标签和按钮，如下图所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/dialogboxlaunchersample2.png)

添加按钮单击事件：

Private Sub CommandButton1\_Click()

Unload Me

End Sub

表明当单击按钮后卸载用户窗体。

（7）编写回调代码，使用户单击对话框启动器时启动自定义的用户窗体：

‘rxbtnDialog onAction回调

Sub rxbtnDialog\_click(control As IRibbonControl)

UserForm1.Show

End Sub

（8）关闭工作簿后重新打开。单击“Test”组右下角的对话框启动器，将显示刚才自定义的用户窗体。

[**细品RibbonX(36)：keytip属性详解**](http://www.excelperfect.com/index.php/2009/11/17/ribbonxstudy36/)

2009年11月17日, 12:38 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**3** 人投票, 平均: **4.33** out of 5)

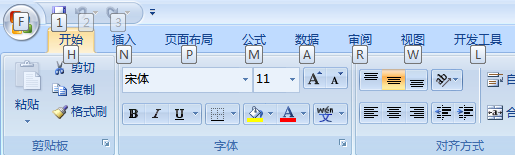
Loading ... Loading ...

keytip能够使用户通过键盘导航功能区。这是不喜欢使用鼠标的用户非常喜欢的功能，也有助于使用户界面更易访问。

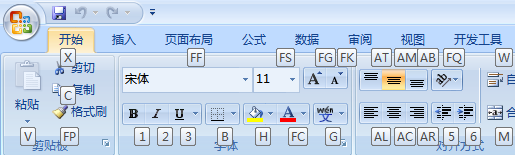
按Alt键将进入键击导航模式，显示可用的键击名，能使用户导航选项卡、组和控件层次。简单地按下代表某命令的字符，将激活该命令。

keytip属性接受至多三个字符。由于其目的是使UI通过键盘比使用鼠标访问更有效，因此应该使其尽可能简短。并且，键击名与其要激活的控件名越不相关，用户要记住也就越难，因此就越不会使用。

如下图所示，按下Alt键，进入键击导航模式。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/keytipSample1.png)

如果再按下代表开始选项卡的H键，将呈现该选项卡每个命令的快捷键，如下图所示。

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/keytipSample2.png)

然后，按下相应的字母键或数字键，就会执行相应的命令。

虽然记住所有的命令是不可能的，但是Microsoft已经为我们提供了一种容易的显示键击的方式。

**创建keytip**

要创建自已的keytip属性，只需在XML结构的父元素中添加适当的keytip=”abc”标签。

（1）创建一个新的.xlsx文件，并将其保存为Excel keytip Example.xlsx。

（2）关闭该文件并在CustomUI Editor中打开。

（3）输入下列XML代码：

<customUI xmlns=”http://schemas.microsoft.com/office/2006/01/customui”>

    <ribbon startFromScratch=”false”>

        <tabs>

            <tab id=”rxtabDemo”

               label=”Demo”

               keytip=”cD”

               insertBeforeMso=”TabHome”>

               <group id=”rxgrpDemo”

                  label=”Demo Group”>

                  <button id=”rxbtnDemo”

                  label=”Testing”

                  keytip=”B”

                  imageMso=”HappyFace”/>

               </group>

            </tab>

        </tabs>

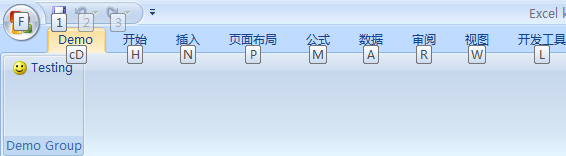
    </ribbon>

</customUI>

上述代码简单地创建了一个包含笑脸按钮的新组。

（4）保存后关闭CustomUI Editor。

（5）在Excel中重新打开该文件，按下Alt键之后，结果如下图所示。

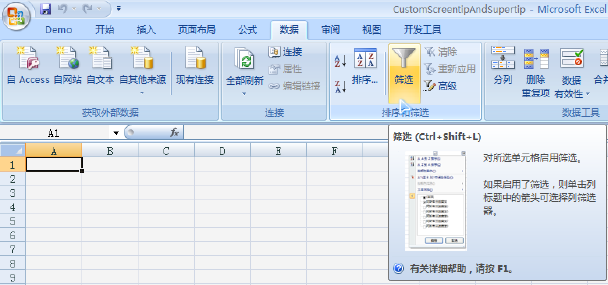
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/keytipSample3.png)

[**细品RibbonX(37)：screentip属性和supertip属性详解**](http://www.excelperfect.com/index.php/2009/11/24/ribbonxstudy37/)

2009年11月24日, 1:09 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(目前还没有人投票)

Loading ... Loading ...

当用户悬浮鼠标在控件上时，screentip属性和supertip属性能够为控件提供有帮助的文本，与用户共享信息，指出该控件的目的。使用screentip告诉用户控件是什么，使用supertip提供稍微更长一点信息，解释该控件做什么。下图为一个示例。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/screentipandsupertipSample1.png)   
screentip部分是框顶部给出键盘快捷键的粗线，而supertip是其剩下的信息。  
然而，Microsoft不会让我们在supertip中添加自已的图像  
**创建screentip属性和supertip属性**  
（1）创建一个新的.xlsx文件，并将其保存为CustomKeytip.xlsx。  
（2）关闭该文件并在CustomUI Editor中打开。  
（3）输入下列XML代码：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon** startFromScratch="false"**>**

**<tabs>**

**<tab** id="rxtabDemo"

label="Demo"

keytip="cD"

insertBeforeMso="TabHome"**>**

**<group** id="rxgrpDemo"

label="Demo Group"**>**

**<button** id="rxbtnDemo"

label="Testing"

screentip="This is very accurate information!"

supertip="It tells you nothing... which is exactly what this button does!"

imageMso="HappyFace"**/>**

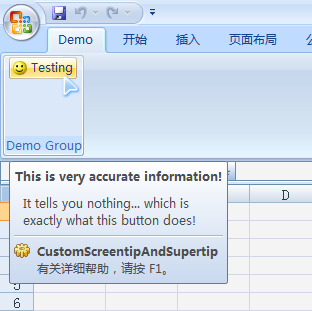
**</group>**

**</tab>**

**</tabs>**

**</ribbon>**

**</customUI>**

（4）保存后关闭CustomUI Editor。  
（5）在Excel中重新打开该文件，将鼠标悬浮在自定义按钮上，结果如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/screentipandsupertipSample2.png)  
当在XML中使用跨多行的文本时，tab字符能够用于排列XML代码而不会影响用户看到的文字。然而，输入额外的硬回车或空格将在属性显示的文本中插入空格。  
如果希望在输出的文本中强迫换行，则需要在XML中添加字符。这五个字符强迫在输出中硬回车。

[**细品RibbonX(38)：覆盖内置控件的属性**](http://www.excelperfect.com/index.php/2009/11/25/ribbonxstudy38/)

2009年11月25日, 2:04 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(目前还没有人投票)

Loading ... Loading ...

很多时候，我们都希望使用Microsoft的内置控件，但有时其标签并不一定反映出我们的目的，因此需要重命名它们。  
例如，在Excel中，“开始”选项卡的键击键是“H”，我们想将其修改为“S”。XML代码如下：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon** startFromScratch="false"**>**

**<tabs>**

**<tab** idMso="TabHome"

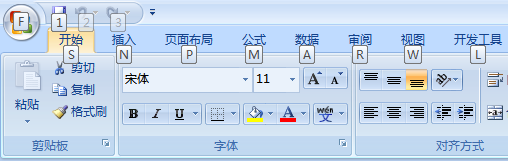
keytip="S"**>**

**</tab>**

**</tabs>**

**</ribbon>**

**</customUI>**

结果如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/overwritingbuiltincontrolSample1.png)  
也能够修改其它控件的一些内置属性，最常用的是重命名按钮或者隐藏其标签。  
然而，Microsoft不允许修改内置组的任何属性。例如，下面的XML代码：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon** startFromScratch="false"**>**

**<tabs>**

*<!-- Enter your first tab here -->*

**<tab** id="rxTabHome"**>**

**<group** idMso="GroupFont"

visible="false"**/>**

**</tab>**

**</tabs>**

**</ribbon>**

**</customUI>**

原本想使“开始”选项卡中的“字体”组隐藏，但结果“字体”组仍然存在于界面中。如果能够使组隐藏，那么我们就可以将选项卡中不常使用的组隐藏起来了。

[**细品RibbonX(39)：技巧—使用VBA自定义功能区元素属性**](http://www.excelperfect.com/index.php/2009/11/27/ribbonxstudy39/)

2009年11月27日, 12:31 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

除了使用类创建自定义属性外，也可以为许多对象（例如工作簿、工作表，等）创建自定义属性。此外，在给对象定义自定义属性之后，属性作为对象的成员被公开，例如：

ThisWorkbook.MyRibbon

正如所见，从父对象直接引用UI对象非常容易，当引用回Ribbon对象时可以省去很多麻烦。考虑到用户经常同时打开多个应用程序，每个程序都会有一个名为MyRibbon的对象，因此让代码可以清楚地指定引用的确切对象是至关重要的。  
**设置自定义属性**  
设置内置对象的自定义属性与使用标准的类模块创建自定义属性相同。唯一不同的是，属性必须驻留在其容器对象里，以便属性能够作为该指定对象的成员公开。  
可以使用自定义属性模拟上下文选项卡来容易地访问可视的属性、确定标签值，等等。下面的示例中，我们开发自定义属性来控制可见性。该示例实现下列功能：

* 显示/隐藏与工作表相关的选项卡的自定义属性
* 两个复选框，控制两个内置组的可见性（字体和表）

本例中，关键属性是getVisible属性，我们将使之变成自定义属性以便通过该属性修改其值。注意，通过这些自定义属性将显示/隐藏位于不同选项卡里的组。

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"

onLoad="rxIRibbonUI\_onLoad"**>**

**<ribbon** startFromScratch="false"**>**

**<tabs>**

**<tab** id="rxtabSheet"

label="Sheet Tab"

insertBeforeMso="TabHome"

getVisible="rxtabSheet\_getVisible"**>**

**<group** id="rxgrp1"

label="My Custom Group"**>**

**</group>**

**</tab>**

**<tab** idMso="TabHome"

label="Modified Home"**>**

**<group** id="rxgrp2"

insertBeforeMso="GroupClipboard"

label="My CheckBox"**>**

*<!-- Add a checkBox -->*

**<checkBox** id="rxchkHideFontGroup"

getLabel="rxchkHideFontGroup\_getLabel"

onAction="rxchkHideFontGroup\_Click"

screentip="Hide the Font Group"

supertip="Click here to hide/unhide the Font Group"**/>**

*<!-- Add a checkBox -->*

**<checkBox** id="rxchkHideTableGroup"

getLabel="rxchkHideTableGroup\_getLabel"

onAction="rxchkHideTableGroup\_Click"

screentip="Hide the Tables Group"

supertip="Click here to hide the Tables Groups"**/>**

**</group>**

**</tab>**

**<tab** idMso="TabHome"**>**

**<group** idMso="GroupFont"

getVisible="GroupFont\_getVisible"**/>**

**</tab>**

**<tab** idMso="TabInsert"**>**

**<group** idMso="GroupInsertTablesExcel"

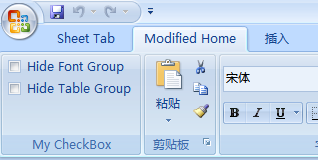
getVisible="GroupInsertTablesExcel\_getVisible"**/>**

**</tab>**

**</tabs>**

**</ribbon>**

**</customUI>**

上述代码创建了“Modified Home”选项卡，在剪贴板前添加了“My CheckBox”组，并且添加了一个名为“Sheet Tab”的空选项卡，界面如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/CustomPropertiesSample1.png)  
接下来，定义属性。在ThisWorkbook对象中定义下列属性：

**Private** pRibbonUI **As** IRibbonUI

**Private** pblnGrpTblVisible **As** **Boolean**

**Private** pblnGrpFontVisible **As** **Boolean**

'设置Ribbon对象,使之能够作为ThisWorkbook的成员访问

**Public** **Property** **Let** rxIRibbonUI(iRib **As** IRibbonUI)

**Set** pRibbonUI = iRib

**End** **Property**

**Public** **Property** **Get** rxIRibbonUI() **As** IRibbonUI

**Set** rxIRibbonUI = pRibbonUI

**End** **Property**

'设置可见性属性,使之能够作为ThisWorkbook的成员访问

**Public** **Property** **Let** rxIRibbonUIGroupTableVisible( \_

**ByVal** blnVisible **As** **Boolean**)

pblnGrpTblVisible = blnVisible

**End** **Property**

**Public** **Property** **Get** rxIRibbonUIGroupTableVisible() **As** **Boolean**

rxIRibbonUIGroupTableVisible = pblnGrpTblVisible

**End** **Property**

**Public** **Property** **Let** rxIRibbonUIGroupFontVisible( \_

**ByVal** blnVisible **As** **Boolean**)

pblnGrpFontVisible = blnVisible

**End** **Property**

**Public** **Property** **Get** rxIRibbonUIGroupFontVisible() **As** **Boolean**

rxIRibbonUIGroupFontVisible = pblnGrpFontVisible

**End** **Property**

注意，虽然用于决定可见性的这些变量在ThisWorkbook模块，但由于ThisWorkbook对象可以在本工程的任何地方访问，因此可以在该模块之外设置或获取这些属性的值。  
在编写回调代码之前，还需要为工作表创建自定义属性，然后使用这些属性来决定与该工作表相关的选项卡是否显示，这里使用工作表Sheet1，当然您可以根据需要修改。  
使用下列代码为该工作表设置自定义属性：

**Private** pglnTabVisible **As** **Boolean**

**Property** **Let** rxIRibbonUISheetTabVisible(**ByVal** blnVisible **As** **Boolean**)

pglnTabVisible = blnVisible

**End** **Property**

**Property** **Get** rxIRibbonUISheetTabVisible() **As** **Boolean**

rxIRibbonUISheetTabVisible = pglnTabVisible

**End** **Property**

**Private** **Sub** Worksheet\_Activate()

Sheet1.rxIRibbonUISheetTabVisible = **True**

ThisWorkbook.rxIRibbonUI.Invalidate

**End** **Sub**

**Private** **Sub** Worksheet\_Deactivate()

Sheet1.rxIRibbonUISheetTabVisible = **False**

ThisWorkbook.rxIRibbonUI.Invalidate

**End** **Sub**

注意，现在可以利用我们为ThisWorkbook创建的自定义属性来使Ribbon无效。我们不再将该属性作为普通的对象，相反我们将其作为属性即ThisWorkbook对象的一个成员。因此，该UI是ThisWorkbook的一部分。  
最后，编写回调代码：

**Sub** rxIRibbonUI\_onLoad(ribbon **As** IRibbonUI)

ThisWorkbook.rxIRibbonUI = ribbon

**End** **Sub**

**Sub** rxtabSheet\_getVisible(**control** **As** IRibbonControl, **ByRef** returnedVal)

returnedVal = Sheet1.rxIRibbonUISheetTabVisible

**End** **Sub**

**Sub** rxchkHideFontGroup\_Click(**control** **As** IRibbonControl, pressed **As** **Boolean**)

ThisWorkbook.rxIRibbonUIGroupFontVisible = pressed

ThisWorkbook.rxIRibbonUI.Invalidate

**End** **Sub**

**Sub** rxchkHideTableGroup\_Click(**control** **As** IRibbonControl, pressed **As** **Boolean**)

ThisWorkbook.rxIRibbonUIGroupTableVisible = pressed

ThisWorkbook.rxIRibbonUI.Invalidate

**End** **Sub**

**Sub** GroupFont\_getVisible(**control** **As** IRibbonControl, **ByRef** returnedVal)

returnedVal = **True**

**If** **control**.ID = "GroupFont" **Then**

returnedVal = **Not** (ThisWorkbook.rxIRibbonUIGroupFontVisible)

**Else**

ThisWorkbook.rxIRibbonUI.Invalidate

**End** **If**

**End** **Sub**

**Sub** GroupInsertTablesExcel\_getVisible(**control** **As** IRibbonControl, **ByRef** returnedVal)

returnedVal = **True**

**If** **control**.ID = "GroupInsertTablesExcel" **Then**

returnedVal = **Not** (ThisWorkbook.rxIRibbonUIGroupTableVisible)

**Else**

ThisWorkbook.rxIRibbonUI.Invalidate

**End** **If**

**End** **Sub**

**Sub** rxchkHideFontGroup\_getLabel(**control** **As** IRibbonControl, **ByRef** returnedVal)

**Select** **Case** ThisWorkbook.rxIRibbonUIGroupFontVisible

**Case** **True**

returnedVal = "Show Font Group"

**Case** **False**

returnedVal = "Hide Font Group"

**End** **Select**

**End** **Sub**

**Sub** rxchkHideTableGroup\_getLabel(**control** **As** IRibbonControl, **ByRef** returnedVal)

**Select** **Case** ThisWorkbook.rxIRibbonUIGroupTableVisible

**Case** **True**

returnedVal = "Show Table Group"

**Case** **False**

returnedVal = "Hide Table Group"

**End** **Select**

**End** **Sub**

现在，你可以通过自定义属性来定义与UI对象相关的不同的属性。

## [细品RibbonX(40)：技巧—在注册表中保存值和从注册表中获取值](http://www.excelperfect.com/index.php/2009/11/30/ribbonxstudy40/)

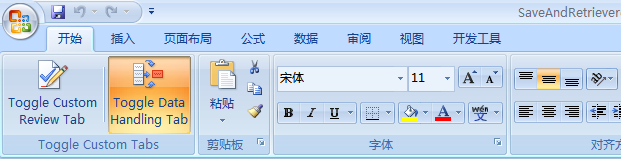
2009年11月30日, 12:48 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**2** 人投票, 平均: **4.50** out of 5)

Loading ... Loading ...

Windows注册表是一个数据库，用于存储与计算机不同方面相关的设置，例如用户设置、应用程序设备、硬件设置，等等。  
VBA提供了与注册表直接交互的方式，这不仅允许我们获取其它程序和硬件的信息，而且也能够使我们选择应用程序中的重要信息并将其存储在注册表中。  
本文中，我们的目的是存储和获取关于UI的信息，我们将使用专门为VBA（和VB）配置而留出的部分。并且，我们不需要使用Windows APIs。  
VBA提供了用于处理注册表的两个函数：GetSetting和SaveSetting，这两个函数仅能访问下面的注册表句柄键：

HKEY\_CURRENT\_USER\Software\VB **and** VBA Program Settings

如下图所示，当切换按钮时，即使关闭了该工程，其状态值仍保存，因此，在重新打开该工程时，按钮的状态为关闭前的状态，不会因为关闭工程而使按钮的状态值丢失。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/11/SaveAndRetrieveregistry1.png)  
定义上图所示界面的XML代码如下：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"

onLoad="rxIRibbonUI\_onLoad"**>**

**<ribbon** startFromScratch="false"**>**

**<tabs>**

**<tab** idMso="TabHome"**>**

**<group** id="rxgrp1"

insertBeforeMso="GroupClipboard"

label="Toggle Custom Tabs"**>**

**<toggleButton** id="rxtglCustomReview"

label="Toggle Custom Review Tab"

imageMso="ReviewAcceptChangeMenu"

size="large"

getPressed="rxtglCustomReview\_getPressed"

onAction="rxtglShared\_Click"**/>**

**<separator** id="rxsep1"**/>**

**<toggleButton** id="rxtglDataHandling"

label="Toggle Data Handling Tab"

imageMso="Consolidate"

size="large"

getPressed="rxtglDataHandling\_getPressed"

onAction="rxtglShared\_Click"**/>**

**</group>**

**</tab>**

*<!-- THE CUSTOM REVIEW TAB UI STARTS HERE -->*

**<tab** id="rxtabCustomReview"

getVisible="rxtabCustomReview\_getVisible"

label="Custom Review"

keytip="K"**>**

**<group** idMso="GroupClipboard"**/>**

**<group** idMso="GroupFont"**/>**

**<group** id="rxgrpProofingComments"

label="Proofing and Comments"**>**

**<box** id="rxbox1" boxStyle="vertical"**>**

**<button** idMso="Spelling"**/>**

**<button** idMso="Thesaurus"**/>**

**<button** idMso="TranslationPane"**/>**

**</box>**

**<separator** id="rxsep2"**/>**

**<box** id="rxbox2" boxStyle="vertical"**>**

**<button** idMso="ReviewNewComment"**/>**

**<button** idMso="ReviewDeleteComment"

label="Delete Comment"**/>**

**<toggleButton** idMso="ReviewShowAllComments"**/>**

**</box>**

**</group>**

**<group** id="rxgrpChanges"

label="Worksheet related changes"**>**

**<box** id="rxbox" boxStyle="vertical"**>**

**<button** idMso="ReviewHighlightChanges"**/>**

**<button** idMso="ReviewProtectAndShareWorkbook"**/>**

**<button** idMso="ReviewAllowUsersToEditRanges"**/>**

**<button** idMso="SheetProtect"**/>**

**</box>**

**</group>**

**</tab>**

*<!-- END OF CUSTOM REVIEW TAB -->*

**<tab** id="rxtabDataHandling"

getVisible="rxtabDataHandling\_getVisible"

label="Data Handling"

keytip="Z"**>**

**</tab>**

**</tabs>**

**</ribbon>**

**</customUI>**

代码包含了一些必须由VBA处理的关键回调。设置了getPressed属性，因为它返回从注册表中获取toggleButton的按下状态的回调，状态是true或false。  
接下来，处理Click事件（该事件通过OnAction属性定义为共享的回调）。每次单击时，都会将按钮当前状态保存到注册表。  
最后，getVisible属性根据toggleButton的状态决定是否选项卡可见。  
本例中，将下面的两个VBA函数封装到自已的函数中，这样可以传递参数到自定义函数中，并且在一定程度上能够控制程序：

**Function** getRegistry(**ByVal** strKey **As** **String**) **As** **Boolean**

**On** **Error** **Resume** **Next**

getRegistry = GetSetting("AddInProject", "TabVisibility", strKey)

**If** Err > 0 **Then** getRegistry = **False**

**End** **Function**

**Function** saveRegistry(**ByVal** strKey **As** **String**, \_

**ByVal** blnSetting **As** **Boolean**)

SaveSetting "AddInProject", "TabVisibility", strKey, blnSetting

**End** **Function**

因为如果键不存在的话，getRegistry函数将返回错误，所以需要添加错误处理以捕获错误。如果发生错误，那么知道该键不存在，getRegistry函数应该返回False。  
回调代码如下：

**Public** gblnShowCustomReviewTab **As** **Boolean**

**Public** gblnShowCustomDataHandlingTab **As** **Boolean**

**Public** gblnCalledOnOpen **As** **Boolean**

**Public** grxIRibbonUI **As** IRibbonUI

**Sub** rxIRibbonUI\_onLoad(ribbon **As** IRibbonUI)

**Set** grxIRibbonUI = ribbon

**End** **Sub**

**Sub** rxtglCustomReview\_getPressed(**control** **As** IRibbonControl, **ByRef** returnedVal)

returnedVal = getRegistry("CustomReviewTab")

gblnShowCustomReviewTab = getRegistry("CustomReviewTab")

grxIRibbonUI.InvalidateControl ("rxtabCustomReview")

**End** **Sub**

**Sub** rxtglDataHandling\_getPressed(**control** **As** IRibbonControl, **ByRef** returnedVal)

returnedVal = getRegistry("CustomDataHandlingTab")

gblnShowCustomDataHandlingTab = getRegistry("CustomDataHandlingTab")

grxIRibbonUI.InvalidateControl ("rxtabDataHandling")

**End** **Sub**

**Sub** rxtglShared\_Click(**control** **As** IRibbonControl, pressed **As** **Boolean**)

**Select** **Case** **control**.ID

**Case** "rxtglCustomReview"

gblnShowCustomReviewTab = pressed

saveRegistry "CustomReviewTab", pressed

gblnCalledOnOpen = **False**

**Case** "rxtglDataHandling"

gblnShowCustomDataHandlingTab = pressed

gblnCalledOnOpen = **False**

saveRegistry "CustomDataHandlingTab", pressed

**End** **Select**

grxIRibbonUI.Invalidate

**End** **Sub**

**Sub** rxtabCustomReview\_getVisible(**control** **As** IRibbonControl, **ByRef** returnedVal)

returnedVal = gblnShowCustomReviewTab

**If** **Not** gblnCalledOnOpen **Then**

**If** gblnShowCustomReviewTab = **True** **Then** Application.SendKeys ("%K{RETURN}")

**End** **If**

**End** **Sub**

**Sub** rxtabDataHandling\_getVisible(**control** **As** IRibbonControl, **ByRef** returnedVal)

returnedVal = gblnShowCustomDataHandlingTab

**If** **Not** gblnCalledOnOpen **Then**

**If** gblnShowCustomDataHandlingTab = **True** **Then** Application.SendKeys ("%Z{RETURN}")

**End** **If**

**End** **Sub**

现在，就可以试试了！关闭工作簿后再打开，处理控件的设置将根据记录在注册表键中的值更新。

## [细品RibbonX(41)：从头开始创建功能区用户界面](http://www.excelperfect.com/index.php/2009/12/01/ribbonxstudy41/)

2009年12月1日, 1:46 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

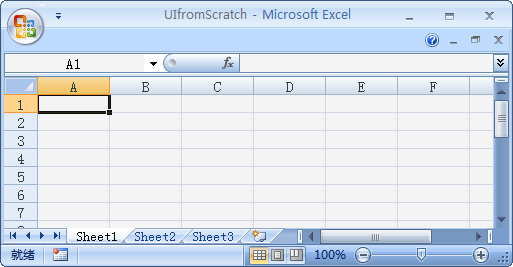
从头开始创建用户界面将取消用户界面已经提供的一些易于访问的功能，因此应该谨慎使用。  
**设置startFromScratch属性**  
从头开始设计项目的用户界面只需要使用startFromScratch属性，将其值设置为True，XML代码如下：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon** startFromScratch="true"**>**

**</ribbon>**

**</customUI>**

结果如下图所示：  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/UIfromScratch1.png)  
然而，在界面中仍然有一些可供使用的命令。下列命令在Excel中仍可使用：新建、打开、保存、Excel选项、退出Excel。  
此外，还可以使用快速访问工具栏（QAT）中的两个选项：在功能区上方/下方显示快速访问工具栏、功能区最小化。  
除了上述已经列出的命令外，虽然从头开始设计用户界面时移除了所有的控件，但没有禁用任何内置的命令，即便这些命令没有在功能区中显示，例如可以使用快捷键Ctrl+c复制区域，使用Ctrl+w关闭文档。  
同样，可以使用Office 2003中的加速键访问Excel的“设置单元格格式”对话框，即按下“Alt+O+E”键。

## [细品RibbonX(42)：启动时激活功能区中某选项卡](http://www.excelperfect.com/index.php/2009/12/02/ribbonxstudy42/)

2009年12月2日, 1:12 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

您可能认为所需要做的就是添加一个选项卡，并将其指定放置在“开始”选项卡前面，这样，因为它是功能区中的第一个选项卡，因此会成为默认的选项卡并在打开文档时会自动选择该选项卡。听起来比较合乎逻辑，但事实呢？缺省情况下会选择自定义选项卡，因此只是在“开始”选项卡之前添加自定义选项卡并不意味着因为它是功能区中的第一个选项卡就选择该选项卡，至少在Excel中不是这样。  
在启动文档时选择某选项卡需要两步。第一步是在XML代码中定义键击键。第二步是编写一个过程，使用SendKeys方法发送键击键到应用程序，执行在文档启动时选择该选项卡的功能。  
第一步：添加XML代码。

**<tab** id="rxtab"

label="Active Tab"

insertBeforeMso="TabHome"

keytip="UN"**>**

第二步：在文档打开时首先发生的onLoad事件中指定选择选项卡的语句。要在装载UI时选择指定的选项卡，使用SendKeys方法，代码如下：

**Sub** rxIRibbonUI\_onLoad(ribbon **As** IRibbonUI)

Application.SendKeys "%UN{RETURN}"

**End** **Sub**

百分号代表Alt键，以触发键击键，UN是我们定义的键击键，同时添加了RETURN键以便当选择选项卡时在选项卡里的键击键失去焦点。这样做是因为我们希望该选项卡有焦点，但不希望显示键击键。  
下表列出了SendKeys方法中可以使用的组合键。

|  |  |
| --- | --- |
| **组合键** | **符号** |
| ALT | % |
| CTRL | ^ |
| SHIFT | + |

可以使用上表列出的键与其它任何键形成组合。下表列出了一些其它的键，在使用SendKeys方法和需要键组合时，可能会发现这些键会有用。虽然不区分大小写，但建议使用大写。  
表：一些特别的键及其相应的代码

|  |  |
| --- | --- |
| **键** | **键代码** |
| BACKSPACE | {BACKSPACE}或{BS} |
| BREAK | {BREAK} |
| CAPS LOCK | {CAPSLOCK} |
| DELETE或DEL | {DELETE}或{DEL} |
| DOWN ARROW | {DOWN} |
| END | {END} |
| ENTER | ~ |
| ENTER(数字键盘) | {ENTER} |
| ESC | {ESCAPE}或{ESC} |
| F1至F15 | {F1}至{F15} |
| HOME | {HOME} |
| RETURN | {RETURN} |
| TAB | {TAB} |

## [细品RibbonX(43)：禁用和重利用功能区中的命令](http://www.excelperfect.com/index.php/2009/12/04/ribbonxstudy43/)

2009年12月4日, 1:39 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(目前还没有人投票)

Loading ... Loading ...

与以前的版本不一样，Office 2007允许全局地禁用和重利用命令。在过去，禁用命令必需禁用UI中执行该命令的控件的每一单个的实例，这显然是一项巨大的工作。  
**禁用命令**  
禁用Office 2007中的命令是非常简单的，最重要的是知道打算禁用的命令的控件名以及这些命令在哪类控件之下。从commands集合中完全禁用某命令，而不是从Ribbon对象和该对象的父容器。试图从父容器里禁用某命令将产生一条错误消息。  
例如，假设希望禁用“加粗”命令，可以使用下列代码：

**<commands>**

**<command** idMso="Bold"

enabled="false"**/>**

**</commands>**

就这么简单！关键是确保正确使用了你希望禁用的命令的idMso。  
**禁用与应用程序选项和退出应用程序相关的命令**  
虽然应用程序选项和退出应用程序控件属于officeMenu元素，然而不能从officeMenu容器里禁用这两个控件。相反，必须使用command元素，如下所示：

**<commands>**

**<command** idMso="FileExit"

enabled="false"**/>**

**<command** idMso="ApplicationOptionsDialog"

enabled="false"**/>**

**</commands>**

当要禁用许多控件时，一个更好的选择是共享回调，这样不需要指定每个命令的enabled属性。这种情况下，代码如下所示：

**<commands>**

**<command** idMso="FileExit"

getEnabled="rxshared\_getEnabled"**/>**

**<command** idMso="ApplicationOptionsDialog"

getEnabled="rxshared\_getEnabled"**/>**

**<command** idMso="Bold"

getEnabled="rxshared\_getEnabled"**/>**

**</commands>**

下面来处理回调，禁用控件，代码如下：

**Sub** rxshared\_getEnabled(**control** **As** IRibbonControl, **ByRef** returnedVal)

returnedVal = **False**

**End** **Sub**

**重利用与通用控件相关的命令**  
重利用命令与禁用命令所采取的方式相同，也是在commands集合中进行处理。  
同样，重利用命令成功的关键在于知道希望重利用的命令所在的控件。此外，虽然某命令是不可见的，但是如果通过快捷键调用的话，仍然具有所设计的功能。这也带来了一些问题，例如事件的重复触发。  
下面，我们来重利用应用程序选项按钮：

**<command** idMso="ApplicationOptionsDialog"

onAction="rxApplicationOptionsDialog\_repurpose"**/>**

回调代码如下：

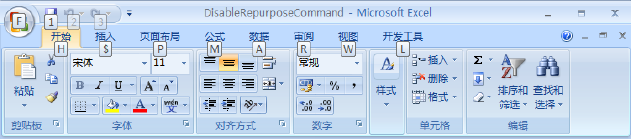
**Sub** rxApplicationOptionsDialog\_repurpose(**control** **As** IRibbonControl, **ByRef** cancelDefault)

MsgBox "对不起,Excel选项目前已经被禁用.", vbCritical

**End** **Sub**

与前面的示例一样，如果多个控件共享相同的回调签名，那么可以使用同一过程。  
**影响键盘快捷键和键击键**  
有时，可能希望覆盖掉内置快捷键和键击键。重载键击键是非常简单的，只需引用该控件，然后给它赋新的键击键，如下面的XML代码所示：

**<tab** idMso="TabInsert" keytip="$"**>**

通过将“$”赋值给“插入”选项卡作为键击键，从而将其键击键从“I”修改为“$”。如下图所示，尽管使用了特殊字符$，然而仍然显示自定义的结果。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/DisableRepurposeCommand1.png)  
提示：键击键通过“键，然后键”的方式访问，即按下并释放第一个键，然后按第二个键。而快捷键则通过“键加键”的方式访问，即同时按下所有的键。  
下面，我们演示重载内置快捷键，例如Ctrl+c。这里介绍使用OnKey方法来重载内置快捷键。OnKey方法用于运行指定的过程，与SendKeys方法类似，因此可以使用与之相同的键代码。但是，OnKey方法稍微有点不同，使用下表所列的参数。

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **必需/可选** | **数据类型** | **描述** |
| Key | 必需 | 字符串 | 代表键或按键组合的字符串 |
| Procedure | 可选 | Variant | 代表要运行的过程名称的字符串。如果值为””（空文本），当按下键或键组合时没有任何操作，这种形式的OnKey修改了键击的正常结果。如果忽略参数，那么键或键组合将恢复到正常行为（即OnKey方法清除的任何键或键组合以前被赋予的功能） |

OnKey方法的通用语法如下：

Application.OnKey Key,ProcedureName

正如上表如所，参数Key指定计划捕获的键或键组合，而参数ProcedureName指定当按下键或键组合时必须执行的过程名。假设希望捕获打印快捷键（Ctrl+p），可以使用下列代码：

**Sub** print\_override()

Application.OnKey "^p", "myPrintMsg"

**End** **Sub**

为了恢复快捷键的正常功能，必须忽略OnKey方法参数中的过程名，如下代码所示：

**Sub** print\_override()

Application.OnKey "^p"

**End** **Sub**

该快捷键命名将恢复其默认的功能。

## [细品RibbonX(44)：在Office菜单中添加项目](http://www.excelperfect.com/index.php/2009/12/07/ribbonxstudy44/)

2009年12月7日, 1:28 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(目前还没有人投票)

Loading ... Loading ...

Office菜单就是应用程序窗口左上角的一个小的应用程序按钮，这个按钮被称作Office按钮，包含一些通用的操作或命令，例如打印、保存和发布。定制Office菜单时，其下的命令将影响整个文档，而不是文档的特定部分，例如段落或字体格式。  
Office菜单使用下面的XML标记：

**<officeMenu>**

*<!--*

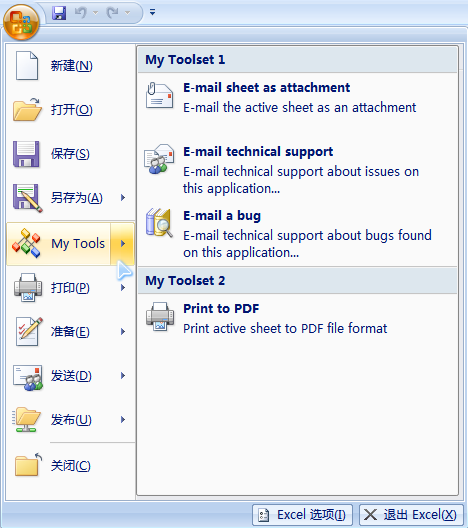
*Everything else goes here*

*-->*

**</officeMenu>**

下表列出了Office菜单的子元素。

|  |  |
| --- | --- |
| **对象** | **用来做什么** |
| control | 引用通用控件对象，能够表现其它对象例如按钮、拆分按钮、组，等 |
| button | 引用通过正常的单击就能执行某类操作的按钮控件 checkbox 引用复选框控件 |
| gallery | 引用库控件 |
| toggleButton | 引用切换按钮，在True/False值之间切换 |
| menuSeparator | 引用菜单分隔条项 |
| splitButton | 引用能够用于容纳其它控件例如按钮控件的拆分按钮 |
| menu | 引用能够用于容纳其它控件例如按钮控件的菜单控件 |
| dynamicMenu | 引用动态菜单，能够在运行时接受动态的XML内容 |

下图为定制Excel中Office菜单的一个示例。在Office菜单中添加了一个拆分按钮My Tools，包含两个组My Toolset1和My Toolset2。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/OfficeMenu1.png)  
在这个示例中，我们创建了包含菜单（根据特定任务组织的几个按钮）的拆分按钮，XML代码如下：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon** startFromScratch="false"**>**

**<officeMenu>**

**<splitButton**

id="rxsbtn"

insertBeforeMso="FilePrintMenu"**>**

**<button**

id="rxbtnSplitMain"

label="My Tools"

imageMso="CreateModule"**/>**

**<menu**

id="rxmnu"

itemSize="large"**>**

**<menuSeparator**

id="rxsep1"

title="My Toolset 1"**/>**

**<button**

id="rxbtnEmailSheet"

imageMso="FileSendAsAttachment"

label="E-mail sheet as attachment"

description="E-mail the active sheet as an attachment"

onAction="rxshared\_click"**/>**

**<button**

id="rxbtnEmailSupport"

imageMso="MessageToAttendeesMenu"

label="E-mail technical support"

description="E-mail technical support about issues on this application..."

onAction="rxshared\_click"**/>**

**<button**

id="rxbtnEmailBug"

imageMso="ResearchPane"

label="E-mail a bug"

description="E-mail technical support about bugs found on this application..."

onAction="rxshared\_click"**/>**

**<menuSeparator**

id="rxsep2"

title="My Toolset 2"**/>**

**<button**

id="rxbtnPrintPDF"

imageMso="FilePrint"

label="Print to PDF"

description="Print active sheet to PDF file format"

onAction="rxshared\_click"**/>**

**</menu>**

**</splitButton>**

**</officeMenu>**

**</ribbon>**

**</customUI>**

实际上，菜单控件也可以提供相同的定制而不须像拆分按钮那样的中间步骤，例如上例可以修改如下：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon** startFromScratch="false"**>**

**<officeMenu>**

**<menu**

id="rxmnu"

insertBeforeMso="FilePrintMenu"

label="My Tools"

imageMso="CreateModule"

itemSize="large"**>**

**<menuSeparator**

id="rxsep1"

title="My Toolset 1"**/>**

**<button**

id="rxbtnEmailSheet"

imageMso="FileSendAsAttachment"

label="E-mail sheet as attachment"

description="E-mail the active sheet as an attachment"

onAction="rxshared\_click"**/>**

**<button**

id="rxbtnEmailSupport"

imageMso="MessageToAttendeesMenu"

label="E-mail technical support"

description="E-mail technical support about issues on this application..."

onAction="rxshared\_click"**/>**

**<button**

id="rxbtnEmailBug"

imageMso="ResearchPane"

label="E-mail a bug"

description="E-mail technical support about bugs found on this application..."

onAction="rxshared\_click"**/>**

**<menuSeparator**

id="rxsep2"

title="My Toolset 2"**/>**

**<button**

id="rxbtnPrintPDF"

imageMso="FilePrint"

label="Print to PDF"

description="Print active sheet to PDF file format"

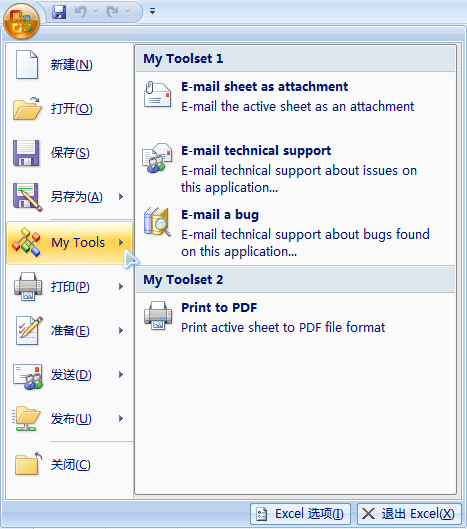
onAction="rxshared\_click"**/>**

**</menu>**

**</officeMenu>**

**</ribbon>**

**</customUI>**

结果如下图所示：  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/OfficeMenu2.png)  
可以看出，实现的效果并没有太大的差异。如果认真观察，就会发现拆分按钮在箭头和按扭之间有一条分隔线，而菜单则没有。  
上面的示例是在Office菜单中添加自已自定义的项目，当然也可以在Office菜单里的内置元素中添加自定义项目或内置项目，此时您需要知道在什么元素中放置项目，下表列出了Office菜单元素。  
表：Office菜单元素

|  |  |  |  |
| --- | --- | --- | --- |
| **元素** | **类型** | **IDMSO** | **应用于** |
| 新建 | 按钮 | FileMenu | Excel/Access/Word |
| 打开 | 按钮 | FileOpen | Excel/Word |
| 打开 | 按钮 | FileOpenDatabase | Access |
| 保存 | 按钮 | FileSave | Excel/Word/Access |
| 另存为 | 拆分按钮 | FileSaveAsMenu | Excel/Word |
| 另存为 | 拆分按钮 | FileSaveAsMenuAccess | Access |
| 打印 | 拆分按钮 | FilePrintMenu | Excel/Word/Access |
| 准备 | 菜单 | FilePrepareMenu | Excel/Word |
| 管理 | 菜单 | FileManageMenu | Access |
| 发送 | 菜单 | FileSendMenu | Excel/Word |
| 电子邮件 | 按钮 | FileSendAsAttachment | Access |
| 发布 | 菜单 | MenuPublish | Excel/Word/Access |
| 关闭 | 按钮 | FileClose | Excel/Word |
| 关闭数据库 | 按钮 | FileCloseDatabase | Access |

## [细品RibbonX(45)：在快速访问工具栏（QAT）中添加项目](http://www.excelperfect.com/index.php/2009/12/10/ribbonxstudy45/)

2009年12月10日, 9:47 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(目前还没有人投票)

Loading ... Loading ...

快速访问工具栏（QAT）是Office 2007新用户界面的一部分，可以使用老的Office工具栏定制方法进行定制。下面，我们主要介绍如何使用XML定制QAT。  
**概述**  
QAT可以包含共享控件和特定文档控件，也可以包含整个控件组（内置的和自定义的）以方便地在单一位置存储多个控件。  
要定制QAT，必须从头开始设计用户界面，这意味着必须将startFromScratch属性设置为true：

**<ribbon** startFromScratch="true"**>**

在处理QAT时，你会注意到有两类图标，一种在其周围有边框而另一种则没有，这种区别表明哪种控件是共享控件，哪种控件是文档控件。  
QAT文档控件的XML代码如下：

**<qat>**

**<documentControls>**

**<control/>**

**</documentControls>**

**</qat>**

QAT共享控件的XML代码如下：

**<qat>**

**<sharedControls>**

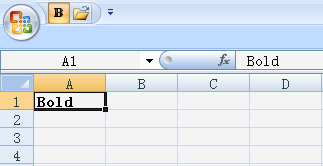
**<control/>**

**</sharedControls>**

**</qat>**

**在QAT中添加自定义和内置命令**  
快速访问工具栏共享或文档控件的子元素如下表。  
表：QAT的子元素

|  |  |
| --- | --- |
| **对象** | **用来做什么** |
| control | 引用可以表现其它对象例如按钮、拆分按钮、组等的普通控件对象 |
| button | 引用按钮控件 |
| separator | 引用分隔条控件 |

如下图所示，在QAT中添加内置控件和自定义按钮。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/QAT1.PNG)  
XML代码如下：

**<qat>**

**<documentControls>**

**<control**

idMso="Bold"

screentip="Make it Bold"

supertip="Click here to make the selected text bold."**/>**

**<button**

id="rxbtnOpen"

imageMso="FileOpen"

screentip="This is Happy"

supertip="Click here for a happy message"

onAction="rxbtnOpen\_click"**/>**

**</documentControls>**

**</qat>**

可以使用control对象引用内置的按钮（本例中为加粗按钮），接着使用按钮创建自已的定制按钮（可以使用control对象引用其它控件例如按钮或拆分按钮）。  
下面的示例创建一个splitButton控件，然后将其添加到QAT。因为QAT没有splitButton子元素，我们必须在QAT之外创建splitButton，然后将其引用到QAT。可以通过以普通的方式添加splitButton来实现——也就是说，通过将其添加到组中开始：

**<group**

id="rxgrp"

label="My Custom Group"**>**

**<splitButton**

id="rxsbtn"

size="large"**>**

**<button**

id="rxbtn2"

imageMso="HappyFace"

label="My Happy Split"**/>**

**<menu**

id="rxmnu"**>**

**<button**

id="rxbtn3"

label="My Happy Menu"

imageMso="HappyFace"

onAction="rxbtn3\_click"**/>**

**</menu>**

**</splitButton>**

**</group>**

上面已经有splitButton并带有一个菜单，菜单中包含一个按钮。由于QAT使用现有的控件，因此现在可以在QAT中引用已存在的拆分按钮了，XML代码如下：

**<qat>**

**<documentControls>**

**<control**

id="rxsbtn"

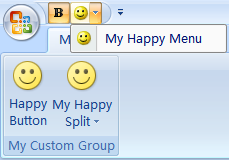
imageMso="HappyFace"

screentip="This is Happy"

supertip="Click here for a happy message"**/>**

**</documentControls>**

**</qat>**

结果如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/QAT2.PNG)  
**在QAT中添加自定义和内置组**  
必须首先创建组，然后从QAT中引用指定该组的id。可以添加组到选项卡中并使其在选项卡中不可见而在QAT中可见。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/QAT3.PNG)  
如上图所示，My QAT Custom Group属于自定义的“Home”选项卡，然而我们将其可见属性设置为False，使其在选项卡上隐藏而在QAT中可见。XML代码如下：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon** startFromScratch="true"**>**

**<tabs>**

**<tab**

id="rxtabHome"

label="Home"**>**

**<group**

idMso="GroupClipboard"**/>**

**<group**

idMso="GroupFont"**/>**

**<group**

idMso="GroupAlignmentExcel"**/>**

**<group**

idMso="GroupNumber"**/>**

**<group**

idMso="GroupStyles"**/>**

**<group**

idMso="GroupCells"**/>**

**<group**

idMso="GroupEditingExcel"**/>**

**<group**

id="rxgrp"

label="My QAT Custom Group"

getVisible="rxshared\_getVisible"**>**

**<button**

id="rxbtnHappy"

label="Mr. Happy Face"

imageMso="HappyFace"

size="large"

onAction="rxshared\_click" **/>**

**<button**

id="rxbtnHappy2"

label="Mr. Happy Face 2"

imageMso="HappyFace"

size="large"

onAction="rxshared\_click" **/>**

**</group>**

**</tab>**

**</tabs>**

**<qat>**

**<documentControls>**

**<control**

idMso="GroupInsertChartsExcel"**/>**

**<control**

idMso="GroupFunctionLibrary"**/>**

**<control**

id="rxgrp"

imageMso="FormatCellsDialog"**/>**

**</documentControls>**

**</qat>**

**</ribbon>**

**</customUI>**

这里，因为要处理QAT，所以我们从头开始定制用户界面，然后以常规方式设置自定义选项卡和组并在自定义组中添加了两个按钮，将自定义组的getVisible属性设置为False，使其不会在选项卡中显示。最后，在QAT标签中，添加了两个自定义组，然后使用通用的control对象来引用想在QAT中显示的自定义组，同时为组赋予了内置的图像。  
注意，有时虽然我们在QAT中定制了组，但打开Excel时不会出现，这是QAT中的一个“小问题”，后文将给出解决方法。  
**重利用QAT控件**  
当重利用QAT中的控件时，实际上重利用与之相关的命令，然后作为控件在QAT中添加相同的命令。  
重利用的一个主要优势是会对该控件产生全局影响。  
下面的XML代码重利用Excel中的两个控件——打开和保存：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"

onLoad="rxIRibbonUI\_onLoad"**>**

**<commands>**

**<command**

idMso="FileSave"

onAction="rxFileSave\_repurpose"**/>**

**<command**

idMso="FileOpen"

onAction="rxFileOpen\_repurpose"**/>**

**</commands>**

**<ribbon** startFromScratch="true"**>**

**<qat>**

**<documentControls>**

**<control**

idMso="FileSave"

screentip="Repurposed Save"

supertip="This is a repurposed command"**/>**

**<control**

idMso="FileOpen"

screentip="Repurposed File Open"

supertip="This is a repurposed command"**/>**

**</documentControls>**

**</qat>**

**</ribbon>**

**</customUI>**

首先，声明希望重利用的命令并赋宏给每个控件，接着在ribbon标记里定义希望在QAT中出现的命令。  
注意，这将产生全局影响，也就是说，如果在该命令出现的任一位置单击该命令或者使用指向该命令的快捷键（这里是Ctrl+o和Ctrl+s），该命令将指向赋值给onAction属性的回调。  
在Excel中还可以使用不同的方式。Excel有一个名为OnKey的便捷的方法，当按下指定的键或键组合时触发。这是一个应用程序级的方法，因此一旦在某工作簿中禁用了某命令，所有在相同会话中打开的其它工作簿都将禁用该命令。  
因此，在Excel中，如果仅仅需要取消包含UI的工作簿中的快捷键，那么在移动到另一个工作簿中时或者当打开工作簿时需要撤销该快捷键的取消。因为这是一个应用程序级的事件，所以需要使用类模块来监控并响应在工作簿间的转换。  
在Excel项目中添加一个类模块并命名，本例中命名为clsAppExcelEvents，输入下面的代码：

**Public** **WithEvents** appXL **As** Excel.Application

**Private** **Sub** ShortcutsEnabled(**ByVal** blnEnabled **As** **Boolean**)

**Select** **Case** blnEnabled

**Case** **Is** = **True**

Application.OnKey "^o"

Application.OnKey "^s"

**Case** **Is** = **False**

Application.OnKey "^o", "commandDisabled"

Application.OnKey "^s", "commandDisabled"

**End** **Select**

**End** **Sub**

**Private** **Sub** setEnabled(**ByVal** Wb **As** Workbook)

**Select** **Case** Wb.Name

**Case** **Is** = ThisWorkbook.Name

ShortcutsEnabled **False**

**Case** **Else**

ShortcutsEnabled **True**

**End** **Select**

**End** **Sub**

注意，在类模块的声明部分声明Excel应用程序。有两个程序来实现这项任务：一个程序检查哪个工作簿是活动工作簿，另一个程序指定OnKey方法。OnKey方法的键组合字符之后，是程序名commandDisabled，该程序必须放置在标准模块中。  
在类模块中，可以指定监控的事件。例如，可以监控某工作簿的激活或失活，决定是否取消快捷键：

**Private** **Sub** appXL\_WorkbookActivate(**ByVal** Wb **As** Workbook)

setEnabled Wb

**End** **Sub**

**Private** **Sub** appXL\_WorkbookDeactivate(**ByVal** Wb **As** Workbook)

setEnabled Wb

**End** **Sub**

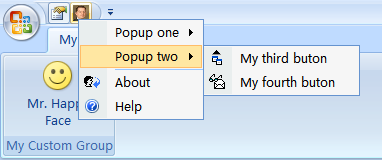
最后，需要在工程打开时设置类，这由包含该工程的工作簿的Open事件来实现：

**Dim** XL **As** **New** clsAppExcelEvents

**Private** **Sub** Workbook\_Open()

**Set** XL.appXL = Application

**End** **Sub**

**使用表驱动（Table-Driven）方式定制QAT**  
下图是一个自定义QAT的示例，使用表装载详细信息到QAT中。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/QAT4.PNG)  
首先，编写包含UI和QAT菜单按钮的XML代码，这里创建的是文档控件按钮：

**<documentControls>**

**<control**

id="rxgrp"

imageMso="AdvancedFileProperties"**/>**

**<button**

id="rxbtnShowPopup"

image="rob"

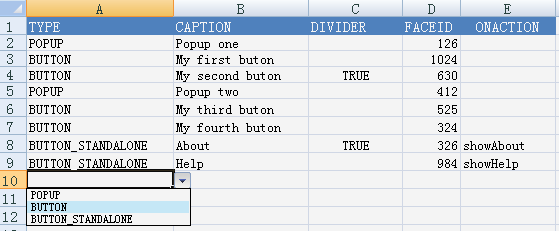
screentip="This is Robert's QAT"

supertip="You can only customize the QAT by starting from scratch. If you do not do that you will not be able to make any changes..."

onAction="rxbtnShowPopup\_Click"

**/>**

**</documentControls>**

上述XML代码将产生上图所示的两个QAT按钮，这里的关键是赋给onAction属性的回调，单击该按钮后将显示菜单。  
接着，创建包含菜单信息的表，如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/QAT5.PNG)  
上图所示的表只是一个建议，因为您可以在其中添加更多的选项。现在，使用VBA阅读该表并创建菜单：

**Public** Const POPNAME **As** **String** = "MY POPUP"

**Sub** loadPopup()

**Dim** mnuWs **As** Worksheet

**Dim** cmdbar **As** CommandBar

**Dim** cmdbarPopup **As** CommandBarPopup

**Dim** cmdbarBtn **As** CommandBarButton

**Dim** nRowCount **As** **Long**

**Call** unloadPopup

**Set** mnuWs = ThisWorkbook.Sheets("MenuItems")

**Set** cmdbar = Application.CommandBars.Add(POPNAME, msoBarPopup)

nRowCount = 2

**With** mnuWs

**Do** **Until** IsEmpty(.Cells(nRowCount, 1))

**Select** **Case** UCase(.Cells(nRowCount, 1))

**Case** "POPUP"

**Set** cmdbarPopup = cmdbar.Controls.Add(msoControlPopup)

cmdbarPopup.Caption = .Cells(nRowCount, 2)

**If** .Cells(nRowCount, 3) <> "" **Then**

cmdbarPopup.BeginGroup = **True**

**End** **If**

**Case** "BUTTON"

**Set** cmdbarBtn = cmdbarPopup.Controls.Add(msoControlButton)

cmdbarBtn.Caption = .Cells(nRowCount, 2)

**If** .Cells(nRowCount, 3) <> "" **Then**

cmdbarBtn.BeginGroup = **True**

**End** **If**

cmdbarBtn.FaceId = .Cells(nRowCount, 4)

cmdbarBtn.OnAction = .Cells(nRowCount, 5)

**Case** "BUTTON\_STANDALONE"

**Set** cmdbarBtn = cmdbar.Controls.Add(msoControlButton)

cmdbarBtn.Caption = .Cells(nRowCount, 2)

**If** .Cells(nRowCount, 3) <> "" **Then**

cmdbarBtn.BeginGroup = **True**

**End** **If**

cmdbarBtn.FaceId = .Cells(nRowCount, 4)

cmdbarBtn.OnAction = .Cells(nRowCount, 5)

**End** **Select**

nRowCount = nRowCount + 1

**Loop**

**End** **With**

**End** **Sub**

**Sub** unloadPopup()

**On** **Error** **Resume** **Next**

Application.CommandBars(POPNAME).Delete

**End** **Sub**

**Sub** showAbout()

MsgBox "This is a sample on how to customize the QAT on the fly!!", vbInformation

**End** **Sub**

**Sub** showHelp()

**On** **Error** **GoTo** Err\_Handler

ThisWorkbook.FollowHyperlink "http://www.msofficegurus.com", , **True**, **True**

**Exit** **Sub**

Err\_Handler:

MsgBox Err.Description, vbCritical, Err.Number

**End** **Sub**

最后，需要编写回调的代码。使用onLoad事件调用loadPopup过程，以便创建弹出菜单，并准备当在QAT中单击该按钮时使用，也包含当发生单击时显示弹出菜单的单击事件代码：

**Dim** grxIRibbonUI **As** IRibbonUI

**Sub** rxIRibbonUI\_onLoad(ribbon **As** IRibbonUI)

**On** **Error** **Resume** **Next**

**Set** grxIRibbonUI = ribbon

Application.Workbooks.Add

**If** ActiveWorkbook.Name <> ThisWorkbook.Name **Then**

**With** ActiveWorkbook

.Saved = **True**

.**Close**

**End** **With**

**End** **If**

' 可以在这个事件或者ThisWorkbook的Open事件中装载弹出菜单

**Call** loadPopup

**End** **Sub**

**Sub** rxbtnShowPopup\_Click(**control** **As** IRibbonControl)

**On** **Error** **Resume** **Next**

Application.CommandBars(POPNAME).ShowPopup

**End** **Sub**

**Sub** rxbtnHappy\_Click(**control** **As** IRibbonControl)

MsgBox "This is Mr. Happy Face... hurray!!", vbExclamation

**End** **Sub**

**定制QAT时的一些注意事项**  
虽然在QAT中可以方便地实现自定义，但也有一些缺陷。  
（1）无法装载控件  
上文中曾经谈到，在定制好后，例如按钮和组，打开工作簿时，却发现定制的控件没有出现。这种情况在使用sharedControls时非常普遍。  
一种解决方法是先最小化工作簿，然后再最大化，通过刷新来使定制的控件出现；或者再打开一个工作簿后，将其关闭，看看定制的控件是否出现。  
（2）无法为控件装载自定义图像  
共享控件的表现通常无法预料，并且不能提供可信赖且一致的界面，因此建议在共享控件中尽量不要使用自定义图像。  
至于文档控件，可以使用下面的过程刷新包含UI的窗口来解决此类问题：

**Sub** rxIRibbonUI\_onLoad(ribbon **As** IRibbonUI)

**Set** grxIRibbonUI = ribbon

**On** **Error** **Resume** **Next**

Application.Workbooks.Add

**If** ActiveWorkbook.Name <> ThisWorkbook.Name **Then**

**With** ActiveWorkbook

.Saved = **True**

.**Close**

**End** **With**

**End** **If**

**End** **Sub**

上述技巧也能用于无法装载自定义控件中。  
（3）复制控件  
在QAT中控件的复制通常发生在工作簿或文档之间切换时。假设有一个包含定制的QAT的工作簿，当按Alt+Tab移动到另一个文档，然后返回定制的工作簿时，在QAT中的控件被复制、三次复制、四次复制……这种复制能够被传播到没有包含任何XML定制的其它工作簿和文档。  
此时，需要关闭后重新打开文档才能消除这种不应有的复制。

## [细品RibbonX(46)：在Excel 2007的QAT中以表驱动的方式构建自定义菜单](http://www.excelperfect.com/index.php/2009/12/11/ribbonxstudy46/)

2009年12月11日, 1:22 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(目前还没有人投票)

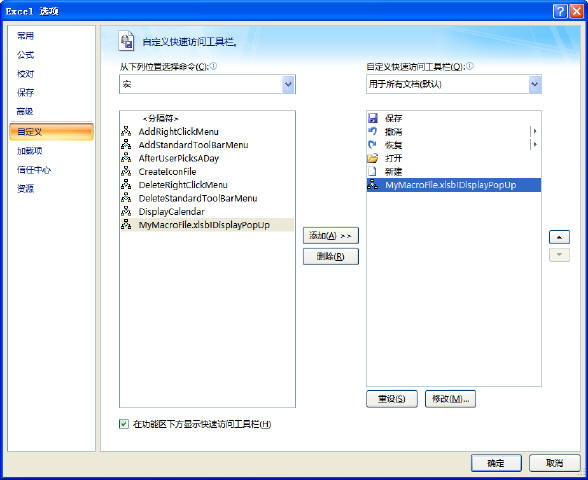
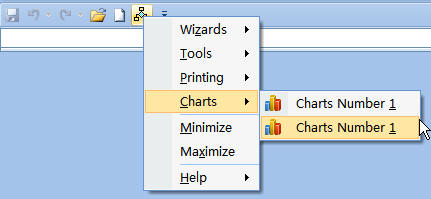
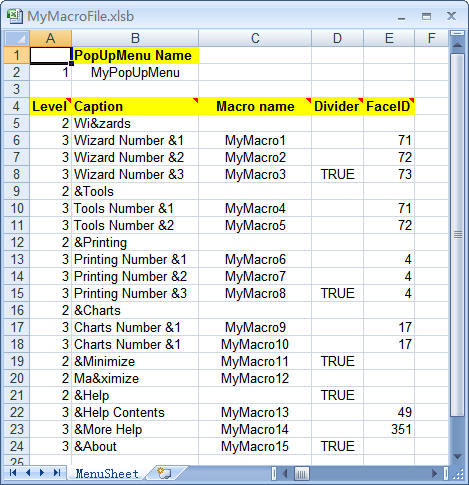
Loading ... Loading ...

**一、创建在所有工作簿中都能使用的自定义菜单**  
如果想在所有工作簿中都可以使用自已喜欢的宏，那么可以将这些宏复制到Personal.xlsb工作簿中，或者在XLStart文件夹中使用另一个隐藏的xlsb工作簿（在Excel启动时会打开该文件夹中的每一个文件），也可以创建加载项。  
**XLSTART文件夹的位置**

C:\Documents and Settings\(username)\Application Data\Microsoft\Excel\XLSTART

如果找不到指定的文件或文件夹，则可能是Windows设置将其隐藏了，此时需要在文件夹选项中启动“显示所有文件和文件夹”选项。  
**如何在Excel2007中创建菜单**  
在Excel 97-2003中，在已存在的菜单栏中创建一个新菜单或者创建自定义菜单栏一点也不困难。但是在Excel 2007中，定制功能区并不容易。  
1) 在下面的地址中下载文件MyMacroFile.zip：

http://www.rondebruin.nl/files/My%20Add-in.zip

2) 解压并复制该文件到XLSTART文件夹中，然后打开Excel（不能看到该文件，因为它是隐藏的）。  
3) 在快速访问工具栏(QAT)中单击鼠标右键，选择“自定义快速访问工具栏”。  
在“从下列位置选择命令”下拉框中选择“宏”，然后在“自定义快速访问工具栏”下拉框中选择“用于所有文档(默认)”。  
选择“DisplayPopUp”宏，按下“添加” ，然后单击“确定”按扭，如图1所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/customqatsample1.JPG)  
图1  
在图1中，可以使用“修改”按钮命令来改变图标。  
注 : 仅需执行操作一次，因为该按钮被保存在Excel QAT定制文件中。  
如果不想再使用该菜单，则从XLSTART文件夹中移除该xlsb文件后，还需手工从QAT中删除该菜单按钮。  
4) 如果在QAT中单击该图标，则将弹出自定义的菜单，如图2所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/customqatsample2.JPG)  
图2  
编辑该菜单：  
在功能区“视图”选项卡中单击“取消隐藏”命令，在弹出的对话框中选择MyMacroFile.xlsb 文件并单击“确定”按钮。  
此时，将显示如图3所示隐藏的工作表“MenuSheet”：  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/customqatsample3.JPG)  
图3  
Level: 指定菜单项的层级，有效值为2和3。2级代表菜单项，3级代表子菜单项。  
Caption: 显现在菜单、菜单项或子菜单里的文本，使用符号（&）来指定加下划线(热键)的字符。  
Macro name: 对于2级或3级项目，在选择该项时要执行的宏。如果2级项目有一个或多个3级项，则2级项目可能没有与之相关联的宏。使用Alt+F11键打开VBE编辑器，可以在MacroModule模块中添加或修改宏程序。  
Divider: 值为True时，则在菜单项或子菜单项前放置一个分隔条。  
FaceID: 可选的。代表显示在项目旁边的内置图形图像的代号数字。  
您可以编辑该表中的信息，从而创建自已的菜单。单击“Refresh Menu”按钮来查看是否作出了正确的修改。如果正确，则单击“Hide Save”按钮。  
**二、创建只在一个工作簿中可用的自定义菜单**  
本节的内容与上节内容大致相同，主要的区别在于工作簿文件为xlsm工作簿，直接打开该工作簿，并不需要将其放置在特定的文件夹中。  
在下面的地址中下载MyWorkbook.xlsm工作簿文件。

http://www.rondebruin.nl/files/MyWorkbook.zip

下载该工作簿后，直接在Excel中打开该工作簿，然后按照上节3）以后的内容进行操作即可。  
**三、在加载项中存储自定义菜单**  
可以将带有自定义菜单的工作簿保存为Excel加载项(xlam)，然后再在工作簿中启用该加载项。这样，QAT中的按钮将保存在加载项中，并且可以在所有打开的工作簿中使用。  
对于上面介绍的示例工作簿，只需将下面的过程中的两行代码删除或注释掉，然后将其保存为Excel加载项。

**Sub** WBDisplayPopUp()

' If ActiveWorkbook.Name = ThisWorkbook.Name Then

**On** **Error** **Resume** **Next**

Application.CommandBars(ThisWorkbook.Sheets("MenuSheet").Range("B2").Value).ShowPopup

**On** **Error** **GoTo** 0

' End If

**End** **Sub**

创建带有菜单的加载项，而这些菜单中是您想要分发的宏程序。这是一种很好的方式。

注：本文参考了Ron de Bruin的一系列文章，有兴趣的朋友可以直接参考其网站的文章。  
同时，参见：[表驱动的方式构建自定义菜单](http://www.excelperfect.com/2009/10/23/buildcommandbarswithtabledrive/)

## [细品RibbonX(47)：更改Excel 2007快速访问工具栏中按钮的图标](http://www.excelperfect.com/index.php/2009/12/14/ribbonxstudy47/)

2009年12月14日, 1:20 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **4.00** out of 5)

Loading ... Loading ...

下面介绍修改QAT中宏按钮或内置按钮图标的方法。  
如下图所示，在QAT中添加一个宏。  
[ChangeAQTImageSample1](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/ChangeAQTImageSample1.png)  
Excel将QAT的设置保存在名为Excel.qat的文件中，该文件位于：

C:\Documents and Settings\\Local Settings\Application Data\Microsoft\Office\Excel.qat

注意，如果没有自定义QAT，那么该文件不存在。  
现在，要修改该宏的图标，虽然可以在自定义选项中进行修改，但只是修改宏按钮的图像，并且只能修改为内置的图标。使用下面的方法可以修改内置按钮或宏按钮的图标。  
在记事本或任何XML编辑器中打开Excel.qat文件，将看到下列内容：

**<mso:customUI** xmlns:x1="http://schemas.microsoft.com/office/2006/01/customui/macro"

xmlns:mso="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<mso:ribbon>**

**<mso:qat>**

**<mso:sharedControls>**

**<mso:control** idQ="mso:FileNewDefault" visible="false"**/>**

**<mso:control** idQ="mso:FileOpen" visible="false"**/>**

**<mso:control** idQ="mso:FileSave" visible="true"**/>**

**<mso:control** idQ="mso:FileSendAsAttachment" visible="false"**/>**

**<mso:control** idQ="mso:FilePrintQuick" visible="false"**/>**

**<mso:control** idQ="mso:FilePrintPreview" visible="false"**/>**

**<mso:control** idQ="mso:Spelling" visible="false"**/>**

**<mso:control** idQ="mso:Undo" visible="true"**/>**

**<mso:control** idQ="mso:Redo" visible="true"**/>**

**<mso:control** idQ="mso:SortAscendingExcel" visible="false"**/>**

**<mso:control** idQ="mso:SortDescendingExcel" visible="false"**/>**

**<mso:button** idQ="x1:C:\_Documents\_and\_Settings\_Administrator\_桌面\_ChangeAQTImageSample.xlsm\_MyMacro\_1" visible="true" label="MyMacro" onAction="C:\Documents and Settings\Administrator\桌面\ChangeAQTImageSample.xlsm!MyMacro" imageMso="ListMacros"**/>**

**</mso:sharedControls>**

**</mso:qat>**

**</mso:ribbon>**

**</mso:customUI>**

其中，中间部分的前11行是QAT中默认的按钮，可以通过下拉QAT右侧的箭头看到，而最后一行（即下面所示的）是我们添加的按钮。

**<mso:button** idQ="x1:C:\_Documents\_and\_Settings\_Administrator\_桌面\_ChangeAQTImageSample.xlsm\_MyMacro\_1" visible="true" label="MyMacro" onAction="C:\Documents and Settings\Administrator\桌面\ChangeAQTImageSample.xlsm!MyMacro" imageMso="ListMacros"**/>**

将imageMso=”ListMacros”修改为imageMso=”M”，即可以修改该按钮的图像。

**<mso:button** idQ="x1:C:\_Documents\_and\_Settings\_Administrator\_桌面\_ChangeAQTImageSample.xlsm\_MyMacro\_1" visible="true" label="MyMacro" onAction="C:\Documents and Settings\Administrator\桌面\ChangeAQTImageSample.xlsm!MyMacro" imageMso="M"**/>**

修改后如下图所示。

[ChangeAQTImageSample2](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/ChangeAQTImageSample2.png)  
若要修改内置按钮的图像，同样修改为imageMso=”M”或者其他图像。  
当然，你也可以将文件名更改为压缩文件后缀，然后解压缩，修改相应的qat代码来修改按钮的图像。

## [细品RibbonX(48)：在Visual Studio中自定义Office 2007用户界面的快速访问工具栏(QAT)](http://www.excelperfect.com/index.php/2009/12/16/ribbonxstudy48/)

2009年12月16日, 1:12 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

在Microsoft Office 2007中自定义快速访问工具栏，仅需要几行XML和编程代码。  
**概述**  
2007 Microsoft Office Fluent用户界面（UI）使用更简单的设计代替了当前系统的层级菜单、工具栏和任务窗格，使得操作更有效率。Office Fluent Ribbon、快捷菜单、快速访问工具栏和Offce菜单是Office Fluent UI的所有组成部分。可以在Office Fluent Ribbon中添加很多的自定义控件和内置控件，例如按钮、复选框和组合框。  
通过使用XML标记元素，在Office Fluent Ribbon中添加组件，并且通过属性来设置这些组件的属性，通过使用任何Microsoft Visual Studio支持的编程语言，例如Visual Basic和Visual C#、以及VBA、Visual C++、Visual Basic 6.0来赋予这些组件功能。  
**使用XML添加控件**  
XML提供了Office Fluent用户界面层级的、公开的模型。通过使用XML元素指定组件的类型，可以在功能区中添加控件（例如按钮），例如可以使用button元素添加一个按钮，通过使用属性例如label属性给控件赋属性值。  
下面是用于自定义Office Fluent UI的XML的示例，在Excel 2007中通过添加提供内置复制功能的按钮和内置粘贴功能的按钮来自定义QAT。

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon** startFromScratch="true"**>**

**<qat>**

**<sharedControls>**

**<button** idMso="Copy"**/>**

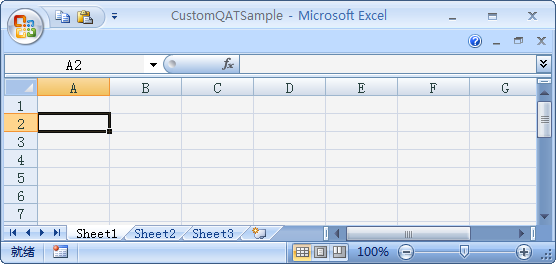
**<button** idMso="Paste"**/>**

**</sharedControls>**

**</qat>**

**</ribbon>**

**</customUI>**

该示例首先设置ribbon元素的startFromScratch属性为true，设置该属性使Office Fluent Ribbon在“start from scratch”模式下，隐藏了所有内置选项卡。接着，通过在代码中添加qat元素，创建对QAT的引用。然后，添加sharedControls元素封装QAT中任何自定义控件。最后，添加两个按钮，一个是内置的复制功能，一个是内置的粘贴功能。  
结果如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/CustomQATSample1.png)  
注：通过使用idMso属性，可以告诉控件引用内置于Microsoft Office里的功能，包括Mso后缀的属性名称总是指向包含在Microsoft Office里的功能和资源。相反，使用id属性代表自定义控件。  
有两种方式来自定义Office Fluent UI：

* 修改支持Office Fluent UI的Office应用程序所创建的Office Open XML格式文件。
* 使用加载项。

可以根据定制用户界面的需要来选择技术。例如，修改基于Office Open XML格式文件是文档级定制，所定制的Office Fluent Ribbon仅与特定的文档相关，而不是整个应用程序；通过使用加载项，将获取应用程序级的定制，这意味着定制的Ribbon将应用到整个应用程序中，而不管打开哪个文档。  
**使用Office Open XML格式文件创建定制的Ribbon**  
1、通过改变文件的扩展名，使之成为一个压缩文件(.zip)，然后打开该文件。  
2、添加一个包含XML定制代码的文件夹。  
3、修改该文档相联系的文件来指向自定义文件夹。  
4、改变该文档的扩展名。  
5、在Office应用程序里打开该文档。  
6、添加代码到文档中，提供自定义用户界面的功能。  
**使用加载项**  
使用加载项定制用户界面也简单。在创建加载项工程后，实现IRibbonExtensibility接口，包括Microsoft.Office.Core命名空间。该接口包含一个名为GetCustomUI方法，使用该方法返回XML定制代码。然后添加编程过程，提供自定义用户界面的功能。  
**自定义QAT**  
创建在功能区中添加自定义按钮的加载项解决方案。  
1、开启Visual Studio 2008。  
2、在“文件”菜单中，单击“新建项目”。  
3、在“新建项目”对话框中，在“项目类型”窗格中，展开“Office”节点，并选择“2007”，然后选择“Excel 2007外接程序”。  
4、在名称框中，输入CustomQATSample，然后单击“确定”创建项目。  
5、在“解决方案资源管理器”中的项目名上单击右键，选择“添加—新建项”。在“添加新项”对话框中选择“功能区(XML)”，使用默认的名称，单击“确定”。  
6、在“解决方案资源管理器”中的项目名上单击右键，选择“属性”，打开项目属性窗口，并选择“资源”选项卡。将“解决方案资源管理器”中的Ribbon1.xml拖到资源窗口中。  
7、编辑Ribbon1.xml的默认代码为自定义代码：

**<ribbon** startFromScratch="true"**>**

**<qat>**

**<sharedControls>**

**<button** idMso="Copy"**/>**

**<button** idMso="Paste"**/>**

**</sharedControls>**

**</qat>**

**</ribbon>**

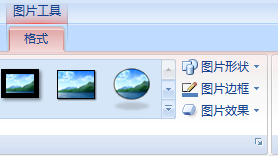
8、将Ribbon1.vb中顶部声明行中的代码复制到ThisAddIn.vb中，并取消注释。  
9、保存并按F5启动调试，结果如上图所示。

## [细品RibbonX(49)：在功能区中使用上下文控件](http://www.excelperfect.com/index.php/2009/12/17/ribbonxstudy49/)

2009年12月17日, 8:28 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

上下文选项卡对新UI的功能提供了极大的推进作用。当用户对某对象执行特定的任务时就会出现特定的选项卡。例如，在Excel中处理图表时，一个上下文选项卡提供用于图表处理的额外选项。本文将介绍创建和执行这些特定的选项卡，以及如何修改内置的上下文选项卡，也介绍如何定制或替换内置的弹出菜单，如何创建自已的上下文弹出菜单。最后，探讨创建一个使用多种语言的UI。  
**使项目上下文**  
使项目上下文意味着必须响应所做的内容，例如操控表或图片。根据对上下文控件的定义，需要在新Office UI中使用上下文选项卡集合来执行任务。然而，也不总是这样。  
下面介绍上下文控件的概念，例如选项卡、组和通用控件。  
选项卡  
当执行上下文敏感的命令时，立即想到的是上下文选项卡。作为使Ribbon的功能完整的一部分，这些特定的选项卡将根据所选择的对象或这些对象是否获得焦点而从功能区中出现和消失，例如图表、数据透视表或图片。  
下图显示了名为“图片工具”的tabSet，包含一个名为“格式”的选项卡。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/contextualtab1.png)  
要实现这样的解决方案，需要使用上下文选项卡集合和tabSet元素。下面的XML标记作为上下文选项卡集和相应的选项卡的容器：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon>**

**<contextualTabs>**

**<tabSet>**

*<!-- Your contextual tab code goes here -->*

**</tabSet>**

**</contextualTabs>**

**</ribbon>**

**</customUI>**

contextualTabs集合是每个tabSet的父对象，同样选项卡的collection对象是选项卡的父对象。可以使用上述代码来创建自已的上下文敏感功能，访问内置的tabSets来修改它们。不巧的是，这样的tabSets扩展不能应用到Excel或Word，仅能对Access的窗体和报表提供这样的扩展。当在Access中使用自定义选项卡tabSets时，需要按下列方式引用扩展的tabSet：

**<tabSet** idMso=”TabSetFormReportExtensibility”**>**

在这个tabSet里，创建上下文选项卡UI。  
对于Excel和Word，需要通过完全控制内置的上下文选项卡或者通过联合getVisible属性使用事件来使选项卡的行为像上下文选项卡。  
组  
例如，当用户处理Excel工作簿中的工作表Sheet1时，希望仅使字体组和插入图表组可见。XML代码：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"

onLoad="rxIRibbonUI\_onLoad"**>**

**<ribbon** startFromScratch="false"**>**

**<tabs>**

**<tab** idMso="TabHome"**>**

**<group** idMso="GroupFont"

getVisible="rxShared\_getVisible"**/>**

**</tab>**

**<tab** idMso="TabInsert"**>**

**<group** idMso="GroupInsertChartsExcel"

getVisible="rxShared\_getVisible"**/>**

**</tab>**

**</tabs>**

**</ribbon>**

**</customUI>**

使用在XML中定义的共享回调，现在能够通过使用想控制的特定工作表的自定义属性来决定属性的返回值。使用下面的代码：

**Sub** rxShared\_getVisible(**control** **As** IRibbonControl, **ByRef** returnedVal)

**Select** **Case** **control**.ID

**Case** "GroupFont"

returnedVal = Sheet1.rxGroupFontVisible

**Case** "GroupInsertChartsExcel"

returnedVal = Sheet1.rxGroupInsertChartsExcel

**End** **Select**

**End** **Sub**

根据自定义的属性值设置该属性的值。首先，设置Ribbon对象，这样当文档打开时以便设置该Ribbon为ThisWorkbook对象的属性。下面的代码放置在ThisWorkbook模块中：

**Private** pRibbonUI **As** IRibbonUI

**Public** **Property** **Let** rxIRibbonUI(iRib **As** IRibbonUI)

**Set** pRibbonUI = iRib

**End** **Property**

**Public** **Property** **Get** rxIRibbonUI() **As** IRibbonUI

**Set** rxIRibbonUI = pRibbonUI

**End** **Property**

其次，指定想控制的工作表的自定义属性，代码如下所示。自定义属性可读写，以便执行期间的任何时候都能重新赋值。

**Private** pblnGroupFontVisible **As** **Boolean**

**Private** pblnGroupChartVisible **As** **Boolean**

**Property** **Let** rxGroupFontVisible(**ByVal** blnVisible **As** **Boolean**)

pblnGroupFontVisible = blnVisible

**End** **Property**

**Property** **Get** rxGroupFontVisible() **As** **Boolean**

rxGroupFontVisible = pblnGroupFontVisible

**End** **Property**

**Property** **Let** rxGroupInsertChartsExcel(**ByVal** blnVisible **As** **Boolean**)

pblnGroupChartVisible = blnVisible

**End** **Property**

**Property** **Get** rxGroupInsertChartsExcel() **As** **Boolean**

rxGroupInsertChartsExcel = pblnGroupChartVisible

**End** **Property**

最后，根据需要修改自定义属性。这里，我们希望当指定工作表是活动工作表时显示这些组，因此使用Worksheet\_Activate和Worksheet\_Deactivate事件来修改属性值，代码如下：

**Private** **Sub** Worksheet\_Activate()

Sheet1.rxGroupFontVisible = **True**

Sheet1.rxGroupInsertChartsExcel = **True**

ThisWorkbook.rxIRibbonUI.Invalidate

**End** **Sub**

**Private** **Sub** Worksheet\_Deactivate()

Sheet1.rxGroupFontVisible = **False**

Sheet1.rxGroupInsertChartsExcel = **False**

ThisWorkbook.rxIRibbonUI.Invalidate

**End** **Sub**

当激活/取消激活该工作表时，显示或隐藏相应的组。  
**使用不明显的方法**  
使用可见性创建上下文敏感的选项卡和组可能有点不正规，但是确实是达到目标的有效的方式。然而，这种方式不是在所有情形下都可用，特别是在处理内置组中的内置控件时。  
启用和禁用控件  
可以使用enabled属性决定控件是否启用。  
下面通过使用getEnabled属性提供上下文敏感的控件。要禁用命令，必须引用该命令本身和按钮或通用控件，例如禁用复制和剪切命令的XML代码如下：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"

onLoad="rxIRibbonUI\_onLoad"**>**

**<commands>**

**<command** idMso="Copy" getEnabled="rxShared\_getEnabled"**/>**

**<command** idMso="Cut" getEnabled="rxShared\_getEnabled"**/>**

**</commands>**

**</customUI>**

下一步，添加共享回调来处理要执行的程序：

**Sub** rxShared\_getEnabled(**control** **As** IRibbonControl, **ByRef** returnedVal)

**Select** **Case** **control**.ID

**Case** "Copy"

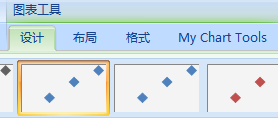
returnedVal = Sheet1.rxCopyEnabled

**Case** "Cut"

returnedVal = Sheet1.rxCutEnabled

**End** **Select**

**End** **Sub**

同样，使用自定义属性指定回调的返回值。  
**修改内置的选项卡集**  
下图展示了对内置的图表工具上下文选项卡集中添加的自定义选项卡。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/contextualtab2.png)  
正确修改内置选项卡集的关键是知道想要定制的tabSet的idMso，本例中，我们想修改TabSetChartTools选项卡集。  
因此，在内置选项卡集中添加自定义选项卡的XML代码如下：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon>**

**<contextualTabs>**

**<tabSet**

idMso="TabSetChartTools"**>**

**<tab**

id="rxtab"

label="My Chart Tools"**>**

**</tab>**

**</tabSet>**

**</contextualTabs>**

**</ribbon>**

**</customUI>**

然而，有时我们想移除上下文图表选项卡中的所有的内置选项卡，仅在该选项卡集中添加一些非常特别的编辑工具。  
首先，我们看看图表工具选项卡集中的选项卡，如下表所示。  
表：图表工具选项卡集中默认的选项卡

|  |  |
| --- | --- |
| **选项卡名** | **描述** |
| TabChartToolsDesign | 提供用户设计所选择的图表的工具，例如内置样式选项、数据选择、图表类型，等等。 |
| TabChartToolsLayout | 指供用户布局所选择的图表的工具，例如标签、趋势线，等等。 |
| TabChartToolsFormat | 提供用户格式化所选择的图表的工具，例如形状样式、文本填充、文本效果、排列，等等。 |

选项卡不会像按钮一样提供enabled属性，因而使用其visible属性控制其出现。通过使选项卡不可见，其中的所有对象都将从UI中消失，但不会禁用其中包含的控件。  
接着完成上面的示例，其XML代码如下：

**<contextualTabs>**

**<tabSet**

idMso="TabSetChartTools"**>**

**<tab**

id="rxtab"

label="My Chart Tools"**>**

**<group** idMso="GroupFont"**/>**

**<group** idMso="GroupInsertChartsExcel"**/>**

**<group** idMso="GroupNumber"**/>**

**</tab>**

**<tab**

idMso="TabChartToolsDesign"

visible="false" **/>**

**<tab**

idMso="TabChartToolsLayout"

visible="false" **/>**

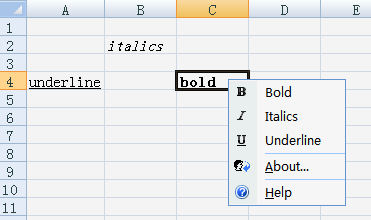
**<tab**

idMso="TabChartToolsFormat"

visible="false" **/>**

**</tabSet>**

**</contextualTabs>**

修改后的图表工具选项卡如下图所示。仅当选择图表时才出现。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/contextualtab3.png)  
**处理上下文弹出菜单**  
当在某对象上右击或者在有弹出菜单的工作环境中右击时，会出现弹出菜单。与上下文选项卡相似，弹出菜单显示与活动对象相关的选项。  
弹出菜单的设置基于VBA，不需要XML代码，它们仍然基于命令栏对象。大多数弹出菜单都能够被定制，然而基于新的OfficeArt不能够被定制。下图显示了Excel中内置弹出菜单的示例。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/contextualtab4.png)  
上图显示的弹出菜单称为“Cell”，索引值等于36。  
不能够定制“Mini工具栏”，然而可以在应用程序选项中控制其是否显现。  
**完全取代内置的弹出菜单**  
当在Excel工作表单元格中右击时，将得到单元格弹出菜单，如上图所示。然而，可能有一个区域需要合适的自定义弹出菜单，如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/contextualtab5.png)  
在编写代码前，让我们先看看该自定义的弹出菜单需要做什么：

* 必须在单元格弹出菜单的位置显示。
* 必须仅显示在预先确定的区域。本例中为单元格区域A1:O32。

首先，添加下面的代码到想要取代内置单元格弹出菜单的工作表代码模块中：

**Private** **Sub** Worksheet\_BeforeRightClick(**ByVal** Target **As** Excel.Range, \_

Cancel **As** **Boolean**)

**If** Union(Target.Range("A1"), Range("A1:O32")).Address = \_

Range("A1:O32").Address **Then**

CommandBars(MYPOPUP).ShowPopup

Cancel = **True**

**End** **If**

**End** **Sub**

下一步，需要在工作簿代码窗口添加两个过程。一个过程指工作簿的Open事件，用来创建弹出菜单；另一个过程指工作簿的Close事件，用来删除弹出菜单。

**Private** **Sub** Workbook\_Open()

**Call** mnuPopup

**End** **Sub**

**Private** **Sub** Workbook\_BeforeClose(Cancel **As** **Boolean**)

**Call** delPopup

**End** **Sub**

然后，在标准模块中包含产生弹出菜单以及删除该菜单的代码，即编写mnuPopup过程和delPopup过程，通过声明公共常量MYPOPUP开始，以便在工程中的其它过程中可用：

**Public** Const MYPOPUP **As** **String** = "MY POPUP"

**Sub** mnuPopup()

**Dim** cmdBar **As** CommandBar

**Dim** mnu **As** CommandBarButton

delPopup

**Set** cmdBar = CommandBars.Add \_

(Name:=MYPOPUP, Position:=msoBarPopup, Temporary:=**True**)

**Set** mnu = cmdBar.Controls.Add(**Type**:=msoControlButton)

**With** mnu

.Caption = "Bold"

.OnAction = "bold"

.FaceId = 113

**End** **With**

**Set** mnu = cmdBar.Controls.Add(**Type**:=msoControlButton)

**With** mnu

.Caption = "Italics"

.OnAction = "italics"

.FaceId = 114

**End** **With**

**Set** mnu = cmdBar.Controls.Add(**Type**:=msoControlButton)

**With** mnu

.Caption = "Underline"

.OnAction = "underline"

.FaceId = 115

**End** **With**

**Set** mnu = cmdBar.Controls.Add(**Type**:=msoControlButton)

**With** mnu

.Caption = "&About..."

.OnAction = "about"

.FaceId = 326

.BeginGroup = **True**

**End** **With**

**Set** mnu = cmdBar.Controls.Add(**Type**:=msoControlButton)

**With** mnu

.Caption = "&Help"

.OnAction = "help"

.FaceId = 984

.BeginGroup = **True**

**End** **With**

**End** **Sub**

**Sub** delPopup()

**On** **Error** **Resume** **Next**

CommandBars(MYPOPUP).Delete

**End** **Sub**

当打开工作簿时，执行第一个过程；当关闭工作簿时，执行第二个过程。  
最后，添加弹出菜单中按钮的功能，代码如下：

**Sub** bold()

Selection.Font.bold = **Not** Selection.Font.bold

**End** **Sub**

**Sub** italics()

Selection.Font.Italic = **Not** Selection.Font.Italic

**End** **Sub**

**Sub** underline()

**If** Selection.Font.underline = xlUnderlineStyleSingle **Then**

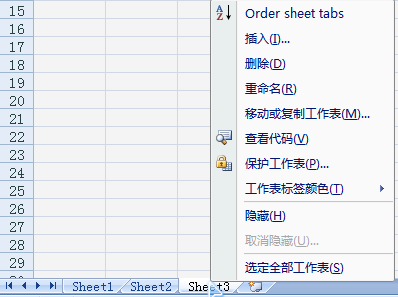
Selection.Font.underline = xlUnderlineStyleNone

**Else**

Selection.Font.underline = xlUnderlineStyleSingle

**End** **If**

**End** **Sub**

**在弹出菜单中添加单独的项目**  
自定义内置弹出菜单的另一种有用的方式是添加新功能。如下图所示，我们在工作表标签右击后出现的弹出菜单中添加新的功能，只需单击该功能应能按字母顺序对工作表排序。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/contextualtab6.png)  
在标准模块中输入下面的代码：

**Sub** addButton()

**Dim** cmdbar **As** CommandBar

**Dim** btn **As** CommandBarButton

resetPopup

**On** **Error** **GoTo** Err\_Handler

**Set** cmdbar = Application.CommandBars("Ply")

**Set** btn = cmdbar.Controls.Add(**Type**:=msoControlButton, Before:=1)

**With** btn

.Style = msoButtonIconAndCaption

.Caption = "Order sheet tabs"

.FaceId = 210

.OnAction = "orderTabs"

**End** **With**

**Exit** **Sub**

Err\_Handler:

MsgBox Err.Description, vbCritical, Err.Number

**End** **Sub**

与上例一样，如果想在打开工作簿时自动实现该功能，则在工作簿的打开事件中添加调用该过程，此外，在工作簿的关闭事件中添加调用恢复原菜单的功能。恢复菜单的代码如下：

**Sub** resetPopup()

Application.CommandBars("Ply").Reset

**End** **Sub**

下面，编写代码实现当单击添加的按钮时所实现的功能：

**Sub** orderTabs()

**Dim** i **As** **Long**

**Dim** j **As** **Long**

**Dim** n **As** **Long**

n = ActiveWorkbook.Sheets.Count

**If** n = 1 **Then**

MsgBox "这个工作簿中仅有一个工作表!", \_

vbInformation

**Exit** **Sub**

**End** **If**

**For** i = 1 **To** n - 1

**For** j = i + 1 **To** n

**If** Sheets(j).Name < Sheets(i).Name **Then**

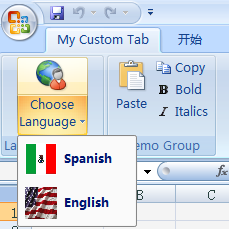
Sheets(j).Move Before:=Sheets(i)

**End** **If**

**Next**

**Next**

**End** **Sub**

**使用多种语言的UI**  
本例的目的是让用户能够选择他们喜欢的语言。示例中将提供两种语言，下图显示了我们将在Excel中创建的示例。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/contextualtab7.png)  
实现上述目的需要完成四项任务：

* 每个控件的标签必须在运行时修改，因此使用getLabel属性。
* 因为这是在所有控件中共享的属性，所以我们使用一个共享的回调来处理变化。这也避免了许多代码量。
* 使用工作表保持控件的名字和翻译。可以隐藏该工作表，以便用户不容易访问它。
* 使用VLookup函数搜索列表中的控件，返回标签值。

控件名和翻译是实现该解决方案的关键元素，一个工作表示例如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/contextualtab8.png)  
可以为包含翻译的区域定义动态区域，也可以将区域转换为表，但是在里只是提供一个示例，所以使用固定区域。下面开始编写VBA代码。  
首先，定义一些需要使用的变量。在标准模块声明部分声明下列全局变量：

**Public** grxIRibbonUI **As** IRibbonUI

**Public** giColControls **As** **Integer**

**Public** gWS **As** Worksheet

第一个变量是Ribbon对象，第二个变量指向语言版本所在的列；第三个变量指向包含翻译的工作表对象。  
接着，使用onLoad事件设置这些变量以便后面使用：

**Sub** rxIRibbonUI\_onLoad(ribbon **As** IRibbonUI)

**Set** grxIRibbonUI = ribbon

giColControls = 2

**Set** gWS = ThisWorkbook.Sheets("Languages")

Application.SendKeys "%UN{RETURN}"

**End** **Sub**

这段代码执行下列任务：

* 设置ribbon对象
* 决定翻译位于的初始列
* 设置包含翻译的工作表
* 向应用程序发送快捷键Alt+UN，以便当打开该工作簿时选择自定义选项卡

当打开文档时将装载初始标签，因此需要处理共享的getLabel回调，代码如下：

**Sub** rxshared\_getLabel(**control** **As** IRibbonControl, **ByRef** returnedVal)

**On** **Error** **Resume** **Next**

returnedVal = Application.WorksheetFunction.VLookup( \_

**control**.ID, gWS.Range("A1:C200"), giColControls, 0)

**End** **Sub**

当用户从UI的splitButton/menu中选择不同语言时调用代码。当用户单击所选择的语言命令时，需要使功能区无效以便重新装载值到UI中，代码如下：

**Sub** rxbtnshared\_Click(**control** **As** IRibbonControl)

**Select** **Case** **control**.ID

**Case** "rxbtnSpanish"

giColControls = 3

**Case** "rxbtnEnglish"

giColControls = 2

**End** **Select**

grxIRibbonUI.Invalidate

**End** **Sub**

使用Select语句选择两种可用的语言。如果选择的是Spanish，查找的列被设置为3；如果选择English，该列被设置为2（缺省值）。

## [细品RibbonX(50)：如何共享Ribbon定制之Excel部署技术](http://www.excelperfect.com/index.php/2009/12/18/ribbonxstudy50/)

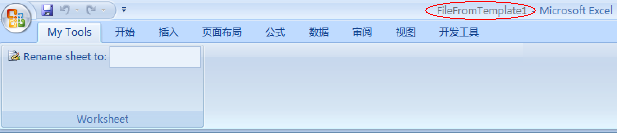
2009年12月18日, 1:33 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **3.00** out of 5)

Loading ... Loading ...

下面介绍Excel中可用于部署定制的三种方法：工作簿、模板、加载项。  
**分发工作簿**  
到目前为止，我们所创建的Excel解决方案都可以称之为“工作簿级的部署”，每个创建的工作簿都以xlsx或xlsm（启用宏）的文件格式保存，这是最常用的格式。所有必需的xml和VBA代码都保存在这样的文件中。  
与其它部署方法相比，工作簿级部署有很多优势，只需要简单地将工作簿发送给用户。最终用户只需要使用安全设置许可运行任何代码，并且VBA代码能够正确地随环境而变化调整，例如可变的文件路径。但这种部署方法也存在缺点，最大的问题是文件的维护，因为它违反了最佳的实践原则：逻辑与数据分离。还有一个缺点是会导致大量相同定制工作簿的副本，这样很难再更新代码。  
**使用模板**  
将文件存储为模板，这样可以在指定的内在结构上快速创建新文件。  
模板的目的是在调用模板文件时，提供“新鲜干净的”工作簿，简单到只需开发者正确地设置工作簿，然后将其保存为模板。一旦将文件安装到最终用户的模板（templates）文件夹中，只需简单地选择“Office菜单——新建——我的模板”，然后双击合适的文件，就会按所选模板创建新工作簿。应该为新文件提供新名称，以免覆盖模板。  
创建模板比想象的更容易。下面来创建模板文件，使用以前的示例editBox – Rename Worksheet.xlsm。打开该文件，选择“Office菜单——另存为”，从“保存类型”下拉列表中选择“Excel启用宏的模板”并提供文件名，将其保存到默认的目录下。在Windows XP中，保存模板的缺省目录为：

C:\Documents And Settings\username\Application Data\Microsoft\Templates

就这么简单！关闭该文件。  
接着，单击“Office菜单——新建”，在“我的模板”中选择并打开刚保存的文件。文件打开后，看起来像保存时的样子，但注意到应用程序标题栏中文件的名字，在模板文件名后自动添加了数字1，并且没有扩展名。如果此时再以该模板新建文字，添加的数字将递增。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/DeploymentTech1.png)  
实际上，因为该文件还没有保存，所以缺少扩展名。一旦单击“保存”按钮，就会提示输入文件名和选择类型。该文件不会覆盖模板文件。  
由于能够将模板存储在用户自已的模板目录里或者网络目录里，所以可以容易地发布和更新工作簿模板，确保都从模板中创建所有的新文件。  
然而，如果模板使用了VBA，那么要保证在基于该模板创建的文件中VBA仍然起作用，必须保存为启用宏的工作簿，否则会删除VBA代码。  
**创建和部署加载项**  
加载项将数据与代码分离，并且加载项能够使定制对所有的文件都可见并可访问。  
下面，以前面文章中的示例dynamicMenu-CreateFromTemplate.xlsm来介绍Excel中加载项的构建。  
当准备转换某文件为加载项时，确保所有的VBA代码无缝地转移。如果文件中含有VBA代码，那么需要检查代码中ThisWorkbook的引用，以及子过程和函数前面没有加上关键字Public或Private。  
因为加载项主要是xlsm文件，这些文件在代码中大都引用ThisWorkbook对象，这对于代码和数据都在相同的工作簿中没有任何问题，但是一旦该文件被转换成加载项，将要处理的数据几乎不会存储在实际的加载项中。因此，最好的方法是加载项仅仅作为代码的容器。这样，就不会担心在更新加载项时会覆盖掉用户设置和数据。  
现在，打开dynamicMenu-CreateFromTemplate.xlsm文件，并打开VBE，搜索代码中的ThisWorkbook，然后决定是否应该将其替换为ActiveWorkbook。注意，大多数工作簿中的代码的目标是处理在活动工作簿（ActiveWorkbook）中的数据，而不是代码所在的工作簿（ThisWorkbook）。  
然而，还需要注意代码中潜在的对ThisWorkbook对象的引用。  
其次，需要注意区分Public、Private和不确定的过程和函数。默认情况下，所有没有加上Public或Private关键字的过程和函数都被视为Public。这意味着，按下Alt+F8键后出现的“宏”对话框中可以看到它们。  
使程序尽可能Private，是一个良好的编程实践。创建公共的过程，以便它们能够被用户运行，然而，如果过程仅从其它过程或函数中调用，那么它们应该被声明为Private。  
代码编写并修改完成后，就可以将其转换为加载项了。  
转换工作簿成为加载项是非常简单的。单击“Office菜单—另存为—其他格式”，在出现的对话框的保存类型中，选择“Excel加载宏”格式（\*.xlam）。此时，文件夹位置改变至“AddIns”文件夹。对于本例来说，将其命名为“createFromTemplate.xlam”，单击“保存”按钮。  
下面，来安装加载项。  
前面已经在系统中创建了加载项，它被存储在本地的AddIns目录中。如果发送该加载项给其他用户，并且想他们能够尽可能容易地安装加载项，那么应该直接将该加载项保存到用户的AddIns文件夹中。在Windows XP中，该文件夹位于：

C:\Documents And Settings\username\Application Data\Microsoft\AddIns

打开Excel，单击“Office菜单—Excel选项—加载项—转到”，出现如下图所示的对话框。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/DeploymentTech2.png)  
如果将加载项保存到本地的AddIns文件夹，那么该加载项就会出现在这个对话框中。如果在其他位置存储加载项，那么需要单击“浏览”按钮来找到该文件。为了完成加载项的装载，只需简单地选中加载项名称前的复选框，并单击“确定”。  
正如所看到的，创建加载项能够共享定制，而不管打开了哪个文件。  
那么，如何卸载和移除加载项呢？  
当装载加载项时，需要内存，因此在需要时装载加载项，不需要时卸载是一项经常使用的操作。卸载加载项非常容易，但是移除加载项则需要更进一步的操作。  
要卸载加载项，但仍需要在以后能够使用该加载项，则调出上图所示的对话框后，取消加载项名称前复选框的选择。此时，该加载项并没有从系统中移除，仍然存在于加载项列表中以便需要时装载。  
要彻底从系统中移除加载项，首先按上述方法将其卸载。然后，关闭Excel，找到实际的加载项文件，将其移动到其他位置或者删除。接着，打开Excel并返回加载项对话框界面，你会注意到该加载项仍然在列表中，单击该加载项前面的复选框，将出会出一条消息，提供你将会从列表中删除该加载项。选择“是”，然后关闭Excel。这是重要是，因为当关闭Excel会在维护加载项列表的注册表中设置修改。  
IsAddin属性  
如果需要在加载项的工作表中添加一些数据以便于使用，那该怎么办呢？此时，可以通过设置加载项的IsAddin属性为False，临时打开该工作表。  
在VBE中，选择加载项工程的ThisWorkbook模块，在工程属性窗口中，将IsAddin属性设置为False。  
将IsAddin属性修改为False，告诉Excel将该文件视作常规工作簿。此时所有的工作表都将可见，并且能够被修改。要再次将该文件转换为加载项，只须按上面的操作将IsAddin属性设置为True。  
注意，在定制功能区时使用Personal.xlsb工作簿往往会带来一些不可预见的问题，因此推荐使用单独的加载项来存储跨Excel工作簿共享的定制。

## [细品RibbonX(51)：如何共享Ribbon定制之跨文件共享Ribbon项目](http://www.excelperfect.com/index.php/2009/12/20/ribbonxstudy51/)

2009年12月20日, 7:53 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(目前还没有人投票)

Loading ... Loading ...

下面主要介绍创建包含共享选项卡和组的宿主文件，以及添加自已的命令到宿主文件的选项卡中的单独的附属文件。所演示的技术主要用于创建加载项时，操作如下：

* 在宿主UI文件中为共享选项卡创建功能区定制。
* 在此文件中添加所期望的全局的组或命令。
* 另外的上下文定制或命令被包含在其它文件或加载项格式中。
* 在需要时装载其他文件，附加其命令到宿主文件的定制中。

要创建共享的选项卡和组，必须先看看如何在XML代码中创建共享的命名空间。  
**创建共享的命名空间**  
每个UI定制都需要与一个命名空间相联系。先看看标准的customUI标记：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

已经给customUI元素提供了xmlns属性，其值为一个URL。URL是希望使用的XML命名空间的名称。这个特殊的URL只是Microsoft决定使用的文本值，没有对应的网页页面，意味着功能区定制不需要Internet访问。  
Microsoft原先为预定义的命名空间使用名称“CustomRibbon”或任何其他单词，但是他们最终选择使用URL。对于我们来说，重要的是如果希望处理Microsoft的选项卡、组和控件，那么必须指向这个命名空间，包含所有内置的RibbonX元素。  
Microsoft也为我们提供了创建自已的命名空间的能力，能够使我们以同样的方式共享选项卡和组。此时，需要在customUI标记中添加另一个xmlns属性：

**<customUI**

xmlns="http://schemas.microsoft.com/office/2006/01/customui"

xmlns:Q="Custom Namespace"**>**

注意到，Microsoft的默认命名空间仍保留在customUI标记中。同时，这确保仍然能够访问所有Microsoft的默认选项卡、组和其他控件。除了默认的命名空间外，我们也添加了xmlns:Q属性。  
在上述代码中，Q是为命名空间声明的本地名称，当然可以使用其他你喜欢的名称，建议在本地名称前加上前缀“ns”，以符合命名规范。同时，建议命名应该易识别和简短，以方便使用。  
假设决定使用本地名称“Custom”，那么自定义xmlns:Q应该如下：

xmlns:nsCustom="Custom Namespace"

双引号之间的部分用于跨文件引用定制，完全是纯文本。例如，如果希望定制看起来像官方的一样，那么可以放置一个指向自已站点的URL。这是一个文本引用，而不管该引用页是否存在。  
关键是在每个文件开始的引用应附加到自定义的命名空间。创建xmlns:Q属性是跨多个文件共享选项卡和组的基础。  
**在Excel中共享选项卡和组**  
本示例被分成两部分：宿主Excel加载项文件的创建，附加到宿主的自定义选项卡的文件的创建。  
首先，打开Excel并以加载项（xlam）文件格式保存新工作簿，将其命名为UIHost.xlam并保存在默认的加载项目录。创建完毕后，关闭Excel。  
在CustomUI Editor中打开UIHost.xlam文件，输入下列代码：

**<customUI**

xmlns="http://schemas.microsoft.com/office/2006/01/customui"

xmlns:nsHost="My Shared Ribbon"**>**

**<ribbon** startFromScratch="false"**>**

**<tabs>**

**<tab** idQ="nsHost:rxTabUI"

label="UI Test"

insertBeforeMso="TabHome"**>**

**<group** idQ="nsHost:rxGrpUI"

label="UI Host"**>**

**<button** id="rxHost\_Btn1"

label="Host UI Button"

onAction="rxHost\_Buttons"

imageMso="HappyFace"**/>**

**</group>**

**</tab>**

**</tabs>**

**</ribbon>**

**</customUI>**

注意到，我们为共享的功能区控件创建了一个名为nsHost的自定义命名空间。再看看tab元素，使用idQ属性代替id属性，确保选项卡被创建在共享的命名空间中，并允许其他文件在其中放置组。  
idQ属性总是采用下面的形式：

idQ="localname:id"

对于本示例来说，本地名为nsHost。本地名后面是“:”表示名称结束，然后是控件的唯一标识符。  
接下来的代码中，group元素也使用idQ属性声明，允许组被共享，因此能够使开发人员在其中添加单独的控件。  
最后代码创建了一个按钮，注意到没有使用idQ属性声明，而是使用了标准的id属性，因为：

* 该按钮被存储在加载项中，因此它将总是可见的。
* 任何期望的动态功能都能够通过与该按钮元素相关的回调来设置。
* 没有在按钮中嵌套控件的方法，因此不需要共享。

验证代码并保存后，生成回调签名并复制，关闭CustomUI Editor，然后重新打开Excel。按照前面文章所介绍的方法安装加载项，打开VBE，导航到UIHost.xlam工程，添加一个新标准模块，粘贴回调签名，并输入代码：

**Private** **Sub** rxHost\_Buttons(**Control** **As** IRibbonControl)

'管理按钮事件

**Select** **Case** **Control**.ID

**Case** **Is** = "rxHost\_Btn1"

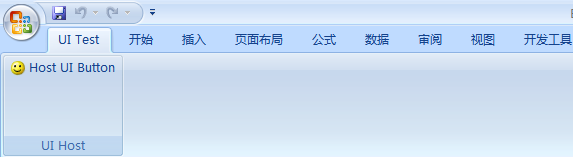
MsgBox "调用至" & ThisWorkbook.Name

**Case** **Else**

'放置其它宏

**End** **Select**

**End** **Sub**

该过程的目的是反馈调用它的工作簿的名称。  
回到Excel界面，测试按钮，如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/sharedUICustome1.png)  
下一步需要创建一个新文件，附加该文件自已的自定义组到UIHost的共享选项卡。  
打开Excel，创建一个新工作簿，将其保存为启用宏的文件，并命名为Leech.xlsm。在CustomUI Editor中打开该文件，输入下列XML代码：

**<customUI**

xmlns="http://schemas.microsoft.com/office/2006/01/customui"

xmlns:nsLeech="My Shared Ribbon"**>**

**<ribbon** startFromScratch="false"**>**

**<tabs>**

*<!-- Shared tab in UI Host file -->*

**<tab** idQ="nsLeech:rxTabUI"**>**

*<!-- Shared group in UI Host file -->*

**<group** idQ="nsLeech:rxGrpLeech"

label="Leech"

insertBeforeQ="nsLeech:rxGrpUI"**>**

*<!-- Controls built using a shared namespace in current file -->*

**<labelControl** idQ="nsLeech:rxLbl01"

label="Defined Namespace"**/>**

**<button** idQ="nsLeech:rxButton01"

label="Button 1"

onAction="rxSharedCallControl\_Click"

imageMso="HappyFace"**/>**

**<button** idQ="nsLeech:rxButton02"

label="Button 2"

onAction="rxSharedCallControl\_Click"

imageMso="HappyFace"**/>**

*<!--Controls build without using shared namespace(alse in current file) -->*

**<separator** id="rxSeparator01"**/>**

**<labelControl** id="rxLbl\_02"

label="No Namespace"**/>**

**<button** id="rxButton03"

label="Button 3"

onAction="rxSharedCallControl\_Click"

imageMso="HappyFace"**/>**

**<button** id="rxButton04"

label="Button 4"

onAction="rxSharedCallControl\_Click"

imageMso="HappyFace"**/>**

**</group>**

**</tab>**

**</tabs>**

**</ribbon>**

**</customUI>**

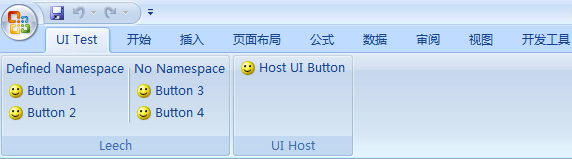
你应该注意到上述代码中，定义共享的命名空间与在UIHost.xlam文件中一样。只是将命名空间的名字修改为nsLeech，而文本标识符仍然是“My Shared Ribbon”。当然，命名空间的名称可以是任意的，修改的目的只是为了更清晰。相反，自定义命名空间“My Shared Ribbon”必须与UIHost.xlam文件中指定的相同，这是允许我们将文件链接在一起的工具。  
完成代码输入并验证后，保存代码，复制回调签名，关闭CustomUI Editor。  
在Excel中重新打开Leech.xlsm文件，打开VBE，插入一个新的标准模块，粘贴回调签名，并修改代码如下：

**Sub** rxSharedCallControl\_Click(**control** **As** IRibbonControl)

'响应按钮单击并通知用户在哪里调用的过程

MsgBox "单击的 " & **control**.ID & " 来自 " & ThisWorkbook.Name

**End** **Sub**

现在，打开Leech.xlsm文件后的界面如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/sharedUICustome2.png)  
在“UI Test”选项卡中添加了Leech组，以及我们创建的所有控件。  
现在，创建一个新工作簿，UI将会更新，如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/sharedUICustome3.png)  
我们创建的带有id属性的控件消失了，仅剩下带有idQ声明的控件。  
然而，当单击按钮Button1和Button2时，什么也没有发生，为什么呢？  
使用idQ属性声明的控件，不会触发任何回调，其原因与存在和不存在命名空间之间的差别相关，没有办法允许对创建在不存在的命名空间中带有idQ属性的控件触发回调。这限制了共享的选项卡和组的使用。

## [细品RibbonX(52)：如何共享Ribbon定制之在正在使用的多个Office版本中部署Excel解决方案](http://www.excelperfect.com/index.php/2009/12/20/ribbonxstudy52/)

2009年12月20日, 8:04 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

除非每个人都在使用Office 2007，因而知道在仍使用老的Office版本中如何部署加载项文件是非常有帮助的。  
**旧的命令栏定制仍然工作吗？**  
考虑安装Office 2007的Office开发者面临的最大的问题之一是“我的定制在全新的功能区中仍然工作吗？”回答是肯定的。定制仍然工作，只是已经在不显眼的地方。即这些定制将会自动出现在Office 2007应用程序的“加载项”选项卡中。  
下面，介绍几种在不同环境中部署Excel解决方案的方法。  
**方法1：创建单独的版本**  
在带有多个Office版本的环境中部署解决方案的第一种方法是最明显的：为应用程序的每种文件格式创建一个版本。  
**方法2：从新的加载项中调用以前的版本**  
在带有多个Office版本的环境中工作的另一种方法是：创建传统的加载宏或全局模板来包含2007文件的RibbonX界面。在2007文件中使用Application.Run方法，可以调用存储在传统文件中的过程。采用这种方法，传统的文件实际上包含了所有的功能代码，而2007文件只是担当启动器。  
使用2003 Excel加载项作为2007 加载项的装载器  
迁移2003文件处理2003环境以及担当2007文件的后台的过程最好按照下列三个步骤：

* 创建基本的2007功能区定制加载项。
* 对2003加载宏做必需的修改。
* 链接到2007文件确保2003文件首先被打开。

事件的顺序是重要的，以便不会在继续时引起崩溃。最后一点是，确保用户在没有2003版时决不要装载该文件的2007版，2003版包含着2007版运行需要的所有宏代码。  
首先，创建基本的2007加载项部分。打开Excel，创建新工作簿，将在默认的AddIns文件夹中以ForumLauncher\_2007.xlam来保存。关闭Excel，在CustomUI Editor中打开该文件，输入下面的XML代码来创建功能区定制：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon** startFromScratch="false"**>**

**<tabs>**

**<tab** idMso="TabDeveloper"**>**

**<group** id="rxgrpForums"

label="Forums"**>**

**<button** id="rxbtnRibbonX"

label="Patrick Schmid's RibbonX Forum"

onAction="rxsharedLinks\_click"

imageMso="HyperlinkInsert"

tag="RibbonX"**/>**

**<button** id="rxbtnVBAX"

label="VBA Express"

onAction="rxsharedLinks\_click"

imageMso="HyperlinkInsert"

tag="VBAX"**/>**

**</group>**

**</tab>**

**</tabs>**

**</ribbon>**

**</customUI>**

上述代码在“加载项”选项卡的末尾的自定义组中添加了两个按钮。注意，按钮使用了共享的回调rxsharedLinks\_click，使用了tag属性。  
验证代码并保存，复制回调签名，关闭CustomUI Editor。打开Excel并安装该加载项。转到VBE，导航到ForumLauncher工程，插入一个新模块，粘贴回调签名，并作如下修改：

**Sub** rxsharedLinks\_click(**control** **As** IRibbonControl)

Application.Run "LaunchFrom2007", **control**.Tag

**End** **Sub**

这个过程使用Application.Run方法从另一个打开的文件中启动宏，指定了要启动的宏的名字LaunchFrom2007，还有一个参数：控件的标签。  
保存后卸载该加载项。  
下一步，对2003加载宏作必需的修改，即ForumLauncher\_v2003.xla。  
首先，修改开始代码确保菜单不会被创建在“加载项”选项卡中，然而在Excel 2003中仍然是菜单。使用下列步骤测试版本并装载正确的菜单。打开VBE，导航到ForumLauncher\_v2003.xla工程，展开ThisWorkbook模块，修改Workbook\_Open过程如下：

**Private** **Sub** Workbook\_Open()

**Dim** wbAddin **As** AddIn

**Dim** bInstalled **As** **Boolean**

**Dim** s2007Filename **As** **String**

s2007Filename = Application.WorksheetFunction.Substitute(ThisWorkbook.Name, "2003", "2007") & "m"

**If** Val(Application.Version) < 12 **Then**

**Call** CreateMenu

**Else**

'检查是否安装了加载项

**For** **Each** wbAddin **In** Application.AddIns

**If** wbAddin.Name = s2007Filename **Then**

'加载项已安装,打开它

Workbooks.**Open** ThisWorkbook.Path & Application.PathSeparator & s2007Filename

bInstalled = **True**

**Exit** **For**

**End** **If**

**Next** wbAddin

'如果需要安装加载项

**If** **Not** bInstalled **Then** Application.AddIns.Add \_

ThisWorkbook.Path & Application.PathSeparator & s2007Filename

**End** **If**

**End** **Sub**

Workbook\_Open事件现在完成如下操作：

* 通过将2003加载项文件名中的2003替换为2007，然后添加m到结果字符串中，从而捕获2007加载项的名字并赋值给变量。
* 检查应用程序的版本，如果版本小于12，则创建菜单。
* 如果版本不小于12，检查2007加载项是否已经安装。
* 如果2007版加载项已安装，则装载文件并记录2007版本已安装。
* 评估bInstalled属性。如果为假，那么安装加载项。

因为我们已经在2003文件被启动时安装并打开了2007文件，所以结束时明确地卸载2007文件。为此，需要修改Workbook\_BeforeClose过程来检查版本并且卸载2007加载项：

**Private** **Sub** Workbook\_BeforeClose(Cancel **As** **Boolean**)

**If** Val(Application.Version) > 11 **Then**

Workbooks(Application.WorksheetFunction.Substitute(ThisWorkbook.Name, "2003", "2007") & "m").**Close**

**Else**

**Call** DeleteMenu

**End** **If**

**End** **Sub**

同理，如果文件版本小于12，那么已经装载了2003菜单修改，因此需要触发DeleteMenu过程。如果应用程序的版本不小于12，那么需要代码关闭2007加载项文件。  
最后，对2003加载项需要提供输入点以便2007版本的文件能够调用过程。当在Excel 2003或更早的版本中单击菜单按钮时触发下列两个过程：

**Private** **Sub** Launch\_VBAX()

'启动VBAX网站

ActiveWorkbook.FollowHyperlink (sVBAXURL)

**End** **Sub**

**Private** **Sub** Launch\_RibbonX()

'启动RibbonX论坛

ActiveWorkbook.FollowHyperlink (sRibbonXURL)

**End** **Sub**

接着，在标准模块中添加下面的过程：

**Public** **Sub** LaunchFrom2007(sSiteToLaunch)

'从2007加载项中担当装载器

**Select** **Case** UCase(sSiteToLaunch)

**Case** **Is** = "VBAX"

**Call** Launch\_VBAX

**Case** **Is** = "RIBBONX"

**Call** Launch\_RibbonX

**End** **Select**

**End** **Sub**

Case语句评估单击的控件，并从2003文件中启动合适的宏。  
现在，我们已经完成了需要的转换，保存2003 xla加载宏并卸载。回到2007加载项，强迫用户首先打开2003版。  
再次安装2007版的加载项。打开VBE，导航到ForumLauncher\_v2007.xlam工程，在ThisWorkbook模块中插入下列代码：

**Private** Const sReqdAddin = "Forum\_Launcher\_v2003.xla"

**Private** **Sub** Workbook\_Open()

**Dim** wbTest **As** Workbook

**On** **Error** **Resume** **Next**

**Set** wbTest = Workbooks(sReqdAddin)

**If** Err.Number = 0 **Then**

'加载项打开

**On** **Error** **GoTo** 0

**Exit** **Sub**

**End** **If**

'加载项必须被打开

**On** **Error** **GoTo** 0

MsgBox "You must load " & sReqdAddin & " to use " & ThisWorkbook.Name

ThisWorkbook.**Close** savechanges:=**False**

**End** **Sub**

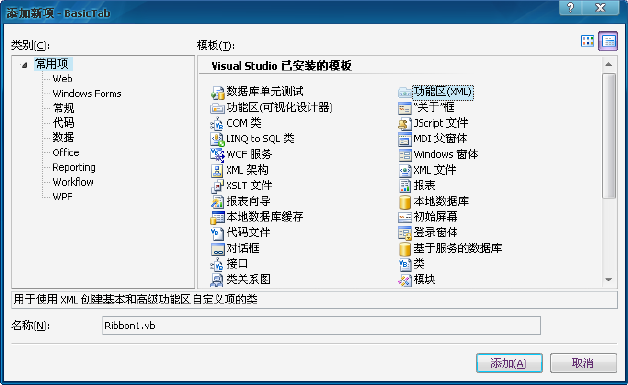
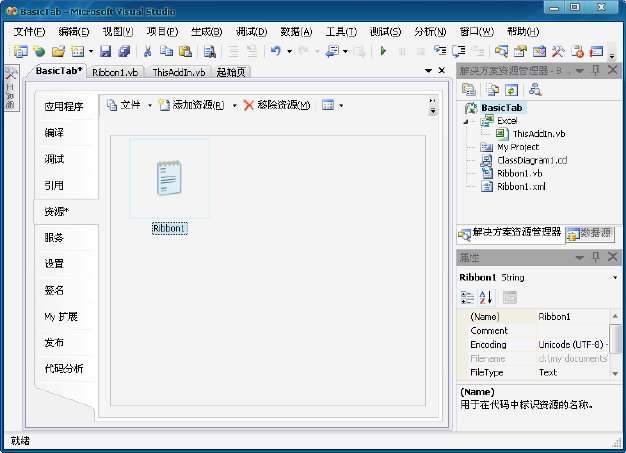
好了！转换完成。保存文件并再次卸载该加载项。  
现在来测试。打开加载项管理器，安装2003版本的加载项，可以在开发工具选项卡的右侧看到完整的功能组，如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/multiversionshared1.png)  
返回到加载项管理器，卸载2003版，安装2007版，单击“确定”返回用户界面，此时会显示如下图所示的错误消息。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/multiversionshared2.png)  
转到VBE，你将看到2003版和2007版的加载项文件都没有被装载。此时，装载2007 xlam加载项的唯一方法是首先装载2003的xla文件。这样，避免用户不小心选择了错误的加载项。

## [细品RibbonX(53)：RibbonX和Visual Studio(1)](http://www.excelperfect.com/index.php/2009/12/21/ribbonxstudy53/)

2009年12月21日, 1:27 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

RibbonX和Visual Studio互为补充，可以非常方便地使用它们创建许多令人惊叹的应用程序。Microsoft已经为Visual Studio和RibbonX协同工作做出了很多努力，不仅仅有用于创建每类RibbonX项目的特定模板，而且利用Visual Studio可以做更多的事情。  
**使用Visual Studio的优势**  
使用Visual Studio的最大的一个特点是将XML代码和资源注入到Office中的能力。如果你决定创建额外的选项卡、组和相关控件，以满足特定的用户需求，那么你能够这样做。使用这项技术，也能够定制满足特定的角色需求的显示，管理员可以比其他员工看到更多的控件。使用XML注入同时意味着能够在标准的XML文件中存储Ribbon变化，并且在运行期在需要时简单地添加功能。  
资源——例如用于定义界面的XML——容易被注入到Office环境。例如，可以容易地在运行时为需要的按钮添加图标或位图，而不是必须包含它们作为文件的一部分。这种添加的灵活性意味着能够更新图标或位图，而无须接触到创建的加载项、模板或文档。成功执行的唯一的条件是，新资源要与原先使用的应用程序具有相同的文件名。  
另一方面，Visual Studio提供了安全的环境，大大减少了受到安全威胁的情形。  
最后，托管的代码有很多好处，可以排除内存泄漏，提供强大的错误处理，也更稳定。并且，使用托管的代码可以很容易地执行许多任务。例如，在Visual Studio中，处理Web服务只需添加一个简单的Web引用，然后访问Web服务功能就像任何其它类一样。  
**RibbonX基础**  
可以创建许多Office应用程序类型。显示在Visual Basic\Office文件夹中的文档和模板如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/ribbonxandvisualstudio1.png)  
在希望使用加载项时，选择Visual Basic\Office\2007文件夹。加载项将会创建跨文档和模板工作的应用程序，在用户删除或禁用之前，加载项总是可用的。当然，在需要时很容易添加和删除加载项。  
**创建基本的选项卡**  
与VBA不同，Visual Studio为处理Ribbon提供了一种完整的解决方案。处理Ribbon最容易的方法是以加载项工程模板开始。  
**步骤1** 打开Visual Studio。  
**步骤2** 选择“文件——新建项目”，将看到如上图所示的“新建项目”对话框。  
**步骤3** 使用喜欢的语言，打开2007文件夹。  
**步骤4** 选择应用程序的加载项模板，输入名称，单击“确定”。这里的名称为BasicTab。  
为了处理Ribbon，需要添加其它项目。Ribbon文件包括代码文件，包含创建的控件的代码，还包括XML文件，告诉Office如何配置功能区。  
**步骤5** 在“解决方案资源管理器”中的项目名称（BasicTab）中单击右键，选择“添加——新建项”，出现“添加新项——BasicTab”对话框，如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/ribbonxandvisualstudio2.png)  
**步骤6** 选择“功能区(XML)”，你可以输入名称或者使用默认的名称（这里使用默认的名称），单击“添加”。  
Visual Studio添加一个新的代码文件和XML文件。Ribbon1.XML文件已经包含一个简单的选项卡、组和按钮，因此本例中保留默认项目。但是，在能够真正看到功能区修改之前，要添加XML文件到应用程序资源中。  
**步骤7** 在解决方案资源管理器中打开项目属性窗口，拖动Ribbon1.xml到“资源”窗口中，如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/ribbonxandvisualstudio3.png)  
你可能已经注意到，使示例完整所必需的大多数代码已经作为创建的工程的一部分出现。当然，你需要作已一些修改以完善示例。  
**步骤8** 按照提示，将Ribbon1.vb中的注释代码复制到ThisAddIn.vb中并取消注释。代码如下：

Protected Overrides **Function** CreateRibbonExtensibilityObject() **As** Microsoft.Office.Core.IRibbonExtensibility

Return **New** Ribbon1()

**End** **Function**

**步骤9** 在新建的功能区中添加一些元素，即修改Ribbon1.xml代码，添加组：

**<group** id="MyGroup"

label="我的组"**>**

**<toggleButton** id="toggleButton1"

size="large"

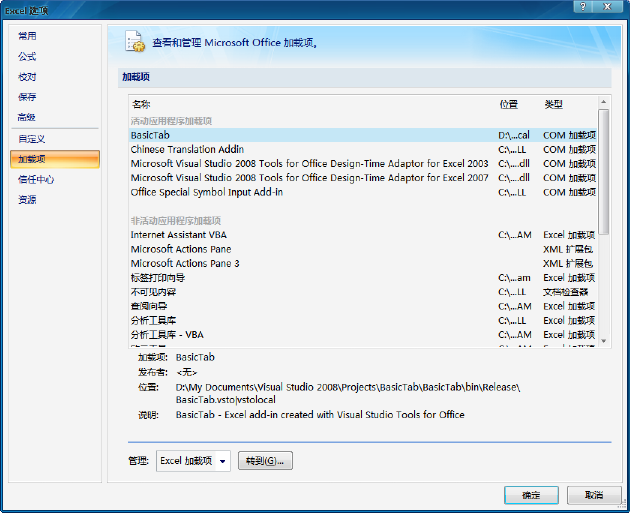
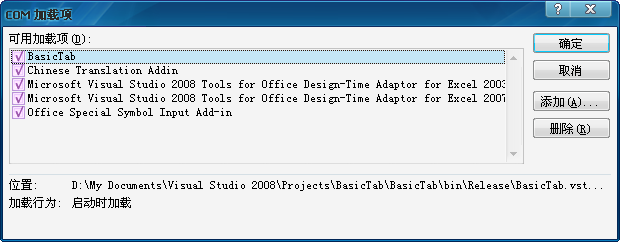
label="我的按钮"

screentip="My Button Screentip"

onAction="OnToggleButton1"

imageMso="HappyFace" **/>**

**</group>**

**步骤10** 按F5键，调试代码。此时，会启动Excel 2007，界面如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/ribbonxandvisualstudio4.png)  
可以看到，与CustomUI Editor不同，Visual Studio可以创建中文的选项卡和组。  
当发布该项目后，加载项会自动添加到Excel，以后打开Excel时都将启动该加载项。如何卸载呢？  
**步骤1** 选择“Office菜单——Excel选项”，出现“Excel选项”对话框。  
**步骤2** 选择“加载项”。看到“BasicTab”加载项列于“添动应用程序加载项”列表中，如下图所示。注意到，“类型”列表明这是一个COM加载项。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/ribbonxandvisualstudio5.png)  
**步骤3** 在“管理”下拉列表中，选择“COM加载项”，单击“转到”。出现下图所示的“COM加载项”对话框。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/ribbonxandvisualstudio6.png)  
**步骤4** 取消BasicTab加载项前的选择，单击“删除”，从列表中移除该加载项应用程序。  
至此，你不会再看到该加载项对应用程序界面的修改。  
然而，上述操作并没有从注册表中移除该加载项。你需要使用特别的实用程序或工具来移除注册表项。  
**步骤1** 找到该加载项的完整文件路径。  
**步骤2** 选择“开始—所有程序—Microsoft Visual Studio 2008—Visual Studio Tools—Visual Studio 2008命令提示”。  
**步骤3** 在命令提示中，输入RegAsm/Unregister <完整的DLL名称>，按回车键。  
RegAsm实用程序将告诉你成功注销了类型。

## [细品RibbonX(54)：RibbonX和Visual Studio(2)](http://www.excelperfect.com/index.php/2009/12/22/ribbonxstudy54/)

2009年12月22日, 1:48 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

下面，让我们继续使用Visual Studio为功能区添加更多的功能。

* 创建带有组和控件的新选项卡
* 在现有的选项卡中添加新控件
* 在Office菜单中添加新控件
* 重利用现有选项卡中的控件
* 重利用Office菜单中的控件

重利用是指将现有的控件实现新功能或者增强现有的功能。  
**在Visual Studio中处理图形**  
在使用Visual Studio时，必须提供图形作为加载项的一部分，这意味着要创建额外的代码。满足这项特定需求的XML作为customUI元素的一部分：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"

onLoad="Ribbon\_Load"

loadImage="GetImage"**>**

在代码中，loadImage属性提供对GetImage()方法的调用，但GetImage()方法不知道要求向Office应用程序发送哪个图标或图像，因此必须以某种方式创建使它能处理任何需求。下面是GetImage()方法的代码：

**Public** **Function** GetImage(**ByVal** ImageName **As** **String**) **As** Bitmap

' 包含传递到Office的位图

**Dim** ThisBitmap **As** Bitmap = **New** Bitmap(20, 20)

' 发现图像名和相应的资源

**Select** **Case** ImageName

**Case** "Colorblk2"

ThisBitmap = **New** Bitmap(My.Resources.Colorblk2)

**Case** "TIME"

ThisBitmap = **New** Bitmap(My.Resources.TIME.ToBitmap())

**End** **Select**

' 返回简单的位图

Return ThisBitmap

**End** **Function**

本例使用两类不同的图像。第一个是标准的位图，Colorblk2，因此可以直接使用；第二个是一个图标，使用前必须调用ToBitmap()方法将它转换为位图。  
注意，代码需要从“资源”中请求图像，因此必须通过“添加资源”将图像添加到资源窗口。  
**当装载功能区时执行任务**  
使用Visual Studio创建的默认项目包括OnLoad()方法，当Office装载功能区时自动执行该方法。onLoad属性决定初始化时Office调用的方法，可以修改方法以满足需要，默认的方法名为onLoad()。本例中相应的回调代码为：

**Public** **Sub** Ribbon\_Load(**ByVal** ribbonUI **As** Office.IRibbonUI)

'保存Ribbon引用

Me.ribbon = ribbonUI

'初始化下划线状态

UnderlineState = **False**

'显示装载消息

MessageBox.Show("功能区装载")

**End** **Sub**

本例提供了一个以某种方式工作的控件示例，该控件取决于用户对另一个控件的选择。本例中，用户能够控制是否应用程序允许下划线功能。因为Excel需要确定在Ribbon装载时该控件的状态，所以代码包含了一个名为UnderlineState的初始化变量。  
在调试时，在代码中包括状态信息是有帮助的。这里，只是简单地告诉功能区已经装载。如果没有看到这个消息框，则表明加载项出现了错误。  
**创建新的选项卡、组和控件**  
最经常执行的任务之一是为应用程序创建新的选项卡、组和控件。为达到此目的，开始仍然需要添加XML。本例中，添加的XML代码如下：

**<tab** id="myTab"

label="我的选项卡"**>**

**<group** id="MyGroup"

label="我的组"**>**

**<button** id="myButton"

label="我的按钮"

image="Colorblk2"

size="large"

onAction="MyButton\_ClickHandler"**/>**

**</group>**

**</tab>**

创建了一个名为“我的选项卡”的选项卡，包含一个名为“我的组”的组，其中含有一个名为“我的按钮”的按钮。使用了自定义图像，因此在image属性中添加了相应的信息。当用户单击“我的按钮”时，调用myButton\_ClickHandler()方法：

**Public** **Sub** myButton\_ClickHandler(**ByVal** **Control** **As** Office.IRibbonControl)

' 显示一条简单的消息

MessageBox.Show("我的按钮被单击")

**End** **Sub**

**修改或重利用已有的选项卡、组和控件**  
有时，希望修改或重利用已有的选项卡、组或控件，而不是创建新的。例如，可能想改变“格式刷”的功能，满足内在的格式需求，或者完全隐藏某些选项卡、组或控件。  
下面的XML代码在“开始”选项卡中添加一个控件：

**<tab** idMso="TabHome"**>**

**<group** id="BehaviorChange"

label="操作"

insertAfterMso="GroupFont"**>**

**<toggleButton** id="StopUnderline"

label="中止下划线功能"

onAction="StopUnderline\_ClickHandler"

getPressed="StopUnderline\_GetPressed"

size="large"

imageMso="ShapeFillColorPicker"

insertBeforeMso="UnderlineGallery"**/>**

**</group>**

**</tab>**

在修改Office已有的元素时，使用idMso属性。新创建的“操作”组使用insertAfterMso属性指定该组在“开始”选项卡中的位置。  
本例中，toggleButton控件除了需要通常的onAction属性外，还需要getPressed属性来获取按钮的状态。这两个属性对应的回调代码为：

**Public** **Sub** StopUnderline\_ClickHandler(**ByVal** **Control** **As** Office.IRibbonControl, **ByVal** Pressed **As** **Boolean**)

' 存储当前按钮的状态

UnderlineState = Pressed

' 在屏幕上显示正确的状态

ribbon.InvalidateControl(**Control**.Id)

**End** **Sub**

**Public** **Function** StopUnderline\_GetPressed(**ByVal** **Control** **As** Office.IRibbonControl) **As** **Boolean**

' 返回当前的按下状态

Return UnderlineState

**End** **Function**

重利用现有的控件的处理过程与修改现有的控件不同。本例中，必须添加command元素：

**<commands>**

**<command** idMso="Underline"

onAction="myUnderline"**/>**

**<command** idMso="FileSaveAsExcel97\_2003"

onAction="FileSaveAs\_ClickHandler"**/>**

**</commands>**

本示例重利用了两个控件：“开始”选项卡中的“下划线”按钮，以及“Office菜单”中“另存为”菜单的“Excel 97-2003工作簿”项目。本示例重载onAction属性，这是最常执行的重载。然而，应该看看其它需要的回调，例如处理切换按钮时的getPressed。

**Public** **Sub** myUnderline(**ByVal** **Control** **As** Office.IRibbonControl, **ByVal** Pressed **As** **Boolean**, **ByRef** CancelDefault **As** **Boolean**)

' 检查下划线控件的状态

**If** (UnderlineState) **Then**

' 显示错误消息

MessageBox.Show("没有允许操作的下划线")

' 设置控件使之不能被按下

Pressed = **False**

' 在屏幕上显示正确的状态

ribbon.InvalidateControl(**Control**.Id)

' 告诉Office不要执行默认的操作

CancelDefault = **True**

**Else**

' 否则,告诉Office执行默认的操作

CancelDefault = **False**

**End** **If**

**End** **Sub**

**Public** **Sub** FileSaveAs\_ClickHandler(**ByVal** **Control** **As** Office.IRibbonControl, **ByRef** CancelDefault **As** **Boolean**)

' 包含用户的响应

**Dim** Response **As** DialogResult

' 要求用户保存文件

Response = \_

MessageBox.Show("保存为老版本. 确定吗?", \_

"老文件版本警告", \_

MessageBoxButtons.YesNo)

' 检查响应

**If** Response = DialogResult.Yes **Then**

CancelDefault = **False**

**Else**

CancelDefault = **True**

**End** **If**

**End** **Sub**

myUnderline()方法控制“下划线”控件是否按正常状态工作，还是基于用户是否按下“中止下划线功能”按钮而提供其它功能。正常状态工作时，该方法简单地设置CancelDefault为false，告诉Office执行缺省的操作。  
此外，当用户按下“中止下划线功能”按钮时，该方法显示一条消息，说明下划线控件不工作了。然后设置Pressed为false，意味着下划线控件不会显示像它正常状态下按下时的操作。接着，重设控件为期望的设置。最后，设置CancelDefault为true。  
**修改或重利用Office菜单**  
Office菜单包含许多配置和文件项，不直接与具体操作相关。当希望创建与具体操作无关的项目时，将其添加到OfficeMenu项中。

**<officeMenu>**

**<menu** idMso="FilePrepareMenu"**>**

**<button** id="NewPrepButton"

label="我的准备按钮"

description="Prepare Time"

image="TIME"

insertBeforeMso="FileProperties"

onAction="NewPrepButton\_ClickHandler"**/>**

**</menu>**

**<splitButton** idMso="FileSaveAsMenu"**>**

**<menu** idMso="FileSaveAsMenu"**>**

**<button** id="SayHello"

label="问好"

description="This button says hello!"

image="Colorblk2"

onAction="SayHello\_ClickHandler"**/>**

**</menu>**

**</splitButton>**

**</officeMenu>**

上述代码演示了如何处理不同的控件，例如本例中的菜单和拆分按钮。相应的回调代码如下：

**Public** **Sub** NewPrepButton\_ClickHandler(**ByVal** **Control** **As** Office.IRibbonControl)

' 显示一条简单的消息

MessageBox.Show("准备好了吗")

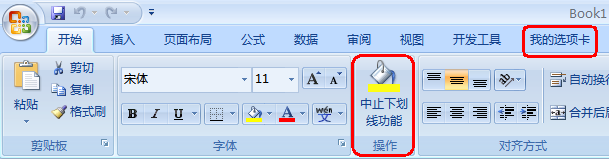
**End** **Sub**

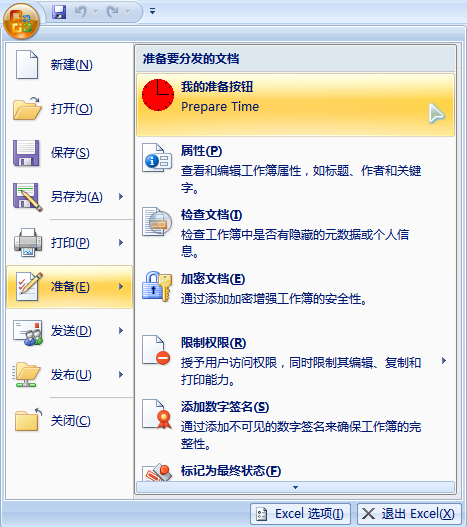
**Public** **Sub** SayHello\_ClickHandler(**ByVal** **Control** **As** Office.IRibbonControl)

' 显示一条简单的消息

MessageBox.Show("Hello")

**End** **Sub**

本示例最终效果的部分截图如下：  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/RibbonxAndVisualStudio21.png)

[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/RibbonxAndVisualStudio22.png)  
**使用startFromScratch模式创建功能区**  
要从头开始创建功能区，只需在ribbon元素中使用startFromScratch属性：

**<ribbon** startFromScratch="true"**>**

## [细品RibbonX(55)：在Visual Studio中开发Excel商务应用程序(1)](http://www.excelperfect.com/index.php/2009/12/23/ribbonxstudy55/)

2009年12月23日, 1:20 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

大多数人使用Excel进行各种各样的计算，从普通的财务计算到复杂的科学计算。而有些人使用Excel创建演示图形，还有些人甚至使用Excel进行基本的文字处理或数据库需要。简而言之，Excel是一种相当灵活的应用程序，能够帮助你执行广泛的任务。并且，你会发现许多Excel定制是非常特别的。  
下面，我们介绍一些定制Excel的应用示例。  
许多商务活动使用非标准等式，例如保险公司使用专门的等式来确定费率。另一类非标准等式是一个行业的通用公式不一定在另一个行业使用，例如电路分析是电子工业的一项通用需求，而银行业不需要任何用于此分析的等式。还有一类非标准的等式是提供某种公共利益，例如，可能需要Excel工作表来帮助比较汽车贷款的利益，或者帮助决定家庭的抵押贷款。一些等式能够帮助执行下列任务，决定信用卡的利息或者为当前的信用卡选择最好的还款计划。一些特定的工作表可以帮助你转换度量单位，或者确定身体特征例如肥胖指标。  
因此，当你遇到要使用这些非标准等式时，就强以考虑在Excel中创建一个非标准等式选项卡，来执行专门的任务。  
下面的示例演示如何在同一选项卡中使用多个版本来完成多项非标准等式的计算任务。  
下图展示了示例应用程序的一个界面。大多数人首先会注意到最显眼的“开始”组没有包含任何控件。许多开发者习惯使应用程序中的每项功能必须执行一项任务。本例中，“开始”组只是作为起点，使用三个标签提供信息，从而起到提示作用。XML代码如下：

**<group** id="Start" label="开始"**>**

**<labelControl** id="StartLabel1"

label="选择一个"**/>**

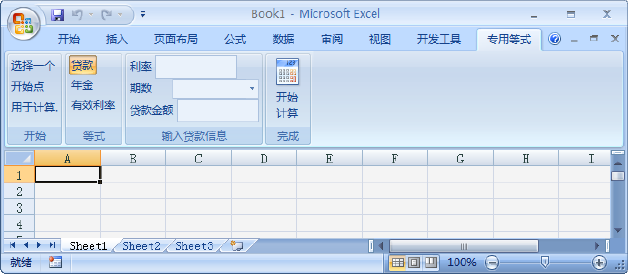
**<labelControl** id="StartLabel2"

label="开始点"**/>**

**<labelControl** id="StartLabel3"

label="用于计算."**/>**

**</group>**

这个选项卡按照工作流设计，确保用户从左至右，按顺序执行完整个工作过程。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/RibbonXApplication1.png)  
**选择正确的等式**  
如上图所示，默认选择的是“贷款”切换按钮，当选择“年金”或“有效利率”时，其右侧的组将呈现不同的项目供输入计算数据。当然，这里的“等式”组也可以使用拆分按钮或库的方式。无论使用哪种方式，都必须提供给用户选择等式。本例中，使用的切换按钮在许多情形下都工作得非常好。三个切换按钮的XML代码如下：

**<group** id="Equations"

label="等式"**>**

**<toggleButton** id="Loan"

label="贷款"

onAction="SetupLoan"

getPressed="SelectedEquation"**/>**

**<toggleButton** id="Annuity"

label="年金"

onAction="SetupAnnuity"

getPressed="SelectedEquation"**/>**

**<toggleButton** id="EffectiveRate"

label="有效利率"

onAction="SetupEF"

getPressed="SelectedEquation"**/>**

**</group>**

当用户单击某个切换按钮时，应用onAction属性提供的方法

**Public** **Sub** SetupLoan(**ByRef** **control** **As** Office.IRibbonControl, **ByVal** pressed **As** **Boolean**)

' 设置计算类型

CalcType = "Loan"

' 设置按下状态

pressed = **True**

' 使整个功能区无效

ribbon.Invalidate()

**End** **Sub**

**Public** **Sub** SetupAnnuity(**ByRef** **control** **As** Office.IRibbonControl, **ByVal** pressed **As** **Boolean**)

' 设置计算类型

CalcType = "Annuity"

' 设置按下状态

pressed = **True**

' 使整个功能区无效

ribbon.Invalidate()

**End** **Sub**

**Public** **Sub** SetupEF(**ByRef** **control** **As** Office.IRibbonControl, **ByVal** pressed **As** **Boolean**)

' 设置计算类型

CalcType = "Effective Rate"

' 设置按下状态

pressed = **True**

' 使整个功能区无效

ribbon.Invalidate()

**End** **Sub**

变量CalcType是一个字符串型的全局变量，用来追踪所使用的等式。这个变量决定应用程序的许多操作，甚至选项卡的最终外观。  
设置pressed为true，改变目标控件的状态，但不会影响选项卡中的其他控件，而该应用程序在用户选择不同的等式时会更改许多控件，因此需要调用ribbon.Invalidate()来使整个功能区无效，从而重绘功能区。  
然后，Excel调用getPressed属性指向的方法。本例中，所有三个切换按钮都使用相同的方法，因为它们实质上都执行相同的任务。SelectedEquation()方法的代码如下：

**Public** **Function** SelectedEquation(**ByRef** **control** **As** Office.IRibbonControl) **As** **Boolean**

' 基于当前的等式确定按下状态

**Select** **Case** CalcType

**Case** "Loan"

**If** **control**.Id = "Loan" **Then**

Return **True**

**Else**

Return **False**

**End** **If**

**Case** "Annuity"

**If** **control**.Id = "Annuity" **Then**

Return **True**

**Else**

Return **False**

**End** **If**

**Case** "Effective Rate"

**If** **control**.Id = "EffectiveRate" **Then**

Return **True**

**Else**

Return **False**

**End** **If**

**Case** **Else**

Return **False**

**End** **Select**

**End** **Function**

基于变量CalcType的值，代码使用Case语句来选择计算类型，并给调用者返回True或False。返回值确定是否按下了该控件。  
**定义多个功能区元素**  
本例虽然只是自定义设计了单个选项卡，但是多重设计给应用程序在单个选项卡中提供多个功能的外观界面。当用户选择特定的等式时，选项卡内容改变以反映该等式的需求。跨功能区的控件让用户从左移到右以解决特定的问题，而工作表显示用户输入的项目的结果，下图显示了选择一个等式并输入数据后的结果。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/RibbonXApplication2.png)  
注：三个等式都是相当简单的财务等式，Excel已经提供了这些工作表函数：PMT、FV、EFFECT。由于三个函数所需要的输入项不同，因此功能区中设计的控件也不同。例如，计算FV函数的功能区界面如下（在“等式”组中选择“年金”）：  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/RibbonXApplication3.png)  
下面列出了选项卡中呈现不同视图所需要的XML：

**<group** id="DataEntry" getLabel="GetDataEntryLabel"**>**

**<editBox** id="Rate"

label="利率"

onChange="GetRateText"**/>**

**<dropDown** id="Term"

label="期数"

getVisible="TermVisible"

getItemCount="TermCount"

getItemID="TermItemID"

getItemLabel="TermItemLabel"

onAction="GetSelectedTerm"**/>**

**<editBox** id="Payment"

label="期初付款"

getVisible="PaymentVisible"

onChange="GetPaymentText"**/>**

**<editBox** id="Amount"

getLabel="AmountLabel"

getVisible="AmountVisible"

onChange="GetAmountText"**/>**

**</group>**

注意，本应用程序如何使用不同的属性来在不同的情形下看到每个所需要的控件。“利率”控件出现在每个应用程序中，因此不需要使用getVisible属性。其他的控件都有getVisible属性，根据当前等式选择决定该控件是否出现在功能区。各控件的getVisible属性对应的回调代码如下：

**Public** **Function** TermVisible(**ByRef** **control** **As** Office.IRibbonControl) **As** **Boolean**

' 应用程序不会使用该字段进行有效利率计算

**If** CalcType = "Effective Rate" **Then**

Return **False**

**Else**

Return **True**

**End** **If**

**End** **Function**

**Public** **Function** PaymentVisible(**ByRef** **control** **As** Office.IRibbonControl) **As** **Boolean**

' 当处理年金计算时应用程序仅使用该字段

**If** CalcType = "Annuity" **Then**

Return **True**

**Else**

Return **False**

**End** **If**

**End** **Function**

**Public** **Function** AmountVisible(**ByRef** **control** **As** Office.IRibbonControl) **As** **Boolean**

' 应用程序不会使用该字段进行有效利率计算

**If** CalcType = "Effective Rate" **Then**

Return **False**

**Else**

Return **True**

**End** **If**

**End** **Function**

取决于变量CalcType，在功能区中呈现相应的控件。这是一项编程技巧，在代码中仅使用一个变量来控制应用程序的外观。  
比较上面的两个图，你会发现在选择不同的切换按钮后，其右侧组中的标签不同，这是由getLabel属性来实现的：

**Public** **Function** GetDataEntryLabel(**ByRef** **control** **As** Office.IRibbonControl) **As** **String**

' 通过计算类型的选择决定组标签

**Select** **Case** CalcType

**Case** "Loan"

Return "输入贷款信息"

**Case** "Annuity"

Return "输入年金信息"

**Case** "Effective Rate"

Return "输入有效利率信息"

**Case** **Else**

Return "没有实现!"

**End** **Select**

**End** **Function**

**Public** **Function** AmountLabel(**ByRef** **control** **As** Office.IRibbonControl) **As** **String**

' 通过计算类型的选择决定数量标签

' 由于有效利率计算不使用该控件,因此应用程序不给它提供标签

**Select** **Case** CalcType

**Case** "Loan"

Return "贷款金额"

**Case** "Annuity"

Return "每月年金付款"

**Case** **Else**

Return "没有实现!"

**End** **Select**

**End** **Function**

应用程序根据选择调整控件及显示。注意，应该小心使用ribbon.Invalidate()，最好仅在应用程序中包含一次对该方法的调用，否则可能会引发很多问题。  
**获取在功能区中输入的数据**  
功能区不允许任何直接的交互，因此不能直接获取用户输入到功能区控件中的信息。但如何获取这些信息呢？仍然要使用回调。大多数控件提供了onChange属性，能够用于发现控件数据的变化，但下拉列表例外，需要使用onAction属性发现选项的变化。下面的代码显示了onChange属性的实现及Term控件的onAction属性的实现。

**Public** **Sub** GetRateText(**ByRef** **control** **As** Office.IRibbonControl, **ByVal** text **As** **String**)

' 保存文本的输入值

Rate = Int32.Parse(text)

**End** **Sub**

**Public** **Sub** GetSelectedTerm(**ByRef** **control** **As** Office.IRibbonControl, **ByVal** selectedId **As** **String**, **ByVal** selectedIndex **As** Int32)

' 存储默认值

Term = 0

' 保存贷款项

**If** CalcType = "Loan" **Then**

**Select** **Case** selectedIndex

**Case** 0

Term = 10

**Case** 1

Term = 15

**Case** 2

Term = 20

**Case** 3

Term = 30

**End** **Select**

**End** **If**

' 保存年金项

**If** CalcType = "Annuity" **Then**

**Select** **Case** selectedIndex

**Case** 0

Term = 5

**Case** 1

Term = 7

**Case** 2

Term = 10

**Case** 3

Term = 15

**Case** 4

Term = 20

**End** **Select**

**End** **If**

**End** **Sub**

**Public** **Sub** GetPaymentText(**ByRef** **control** **As** Office.IRibbonControl, **ByVal** text **As** **String**)

' 保存文本的输入值

Payment = Int32.Parse(text)

**End** **Sub**

**Public** **Sub** GetAmountText(**ByRef** **control** **As** Office.IRibbonControl, **ByVal** text **As** **String**)

' 保存文本的输入值

Amount = Int32.Parse(text)

**End** **Sub**

使用Int32.Parse()将文本框中输入的数据值转换为数字，如果用户在文本框中输入的不是数字，那么Int32.Parse()方法将输出0。  
**执行计算**  
需要两段代码。第一段代码响应功能区中单击“开始计算”按钮的操作；第二段代码转换所有数据为字符串，然后在执行计算之前将它们放置到Excel中。

**Public** **Sub** Calculate(**ByRef** **control** **As** Office.IRibbonControl)

' 选择计算并调用

**Select** **Case** CalcType

**Case** "Loan"

Globals.ThisAddIn.CalculatePMT(Rate, Term, Amount)

**Case** "Annuity"

Globals.ThisAddIn.CalculateFV(Rate, Term, Payment, Amount)

**Case** "Effective Rate"

Globals.ThisAddIn.CalculateEFFECT(Rate)

**End** **Select**

**End** **Sub**

因为应用程序已经在全局变量中放置了所需的数据，因此所有需要做的就是调用合适的加载项函数并提供必需的输入。实际上，计算需要构建与Excel公式命令相同的字符串。下面的代码显示了这些计算函数：

' 计算贷款数量

**Public** **Sub** CalculatePMT(**ByVal** Rate **As** **Double**, **ByVal** NPer **As** Int32, **ByVal** PV **As** Int32)

' 计算利率

**Dim** PeriodicRate **As** **Double** = (Rate / 100) / 12

' 计算期数

**Dim** Periods **As** Int32 = NPer \* 12

' 执行计算

Application.ActiveWindow.ActiveCell.Cells(1, 1) = \_

"=PMT(" + PeriodicRate.ToString() + "," + \_

Periods.ToString() + "," + PV.ToString() + ",0,0)"

Application.ActiveWindow.ActiveCell.Calculate()

**End** **Sub**

' 计算年金数量

**Public** **Sub** CalculateFV(**ByVal** Rate **As** **Double**, **ByVal** NPer **As** Int32, **ByVal** PMT **As** Int32, **ByVal** PV **As** Int32)

' 计算利率

**Dim** PeriodicRate **As** **Double** = (Rate / 100) / 12

' 计算期数

**Dim** Periods **As** Int32 = NPer \* 12

' 执行计算

Application.ActiveWindow.ActiveCell.Cells(1, 1) = \_

"=FV(" + PeriodicRate.ToString() + "," + \_

Periods.ToString() + "," + PMT.ToString() + "," + \_

PV.ToString() + ",0)"

Application.ActiveWindow.ActiveCell.Calculate()

**End** **Sub**

' 计算有效利率

**Public** **Sub** CalculateEFFECT(**ByVal** Rate **As** **Double**)

' 计算利率

**Dim** PeriodicRate **As** **Double** = Rate / 100

' 执行计算

Application.ActiveWindow.ActiveCell.Cells(1, 1) = \_

"=EFFECT(" + PeriodicRate.ToString() + ",12)"

Application.ActiveWindow.ActiveCell.Calculate()

**End** **Sub**

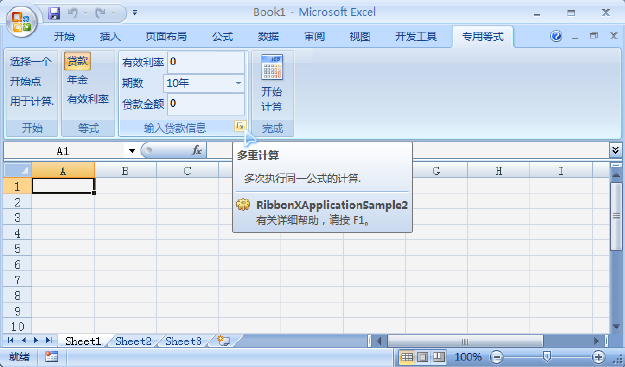
在工作表中简单地添加文本，不能确保发生计算，因此代码的末尾调用Application.ActiveWindow.ActiveCell.Calculate()来执行计算。

## [细品RibbonX(56)：使用Visual Studio开发Excel商务应用程序(2)](http://www.excelperfect.com/index.php/2009/12/24/ribbonxstudy56/)

2009年12月24日, 12:55 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**3** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

让我们接着上文介绍，继续丰富已开发的应用程序。  
下面，让应用程序执行更丰富的计算。例如，可能想计算一定范围内的抵押值，而不是单个值，以便为客户提供多种选择。在上例中，我们能够指导用户输入不同的值，执行相同的计算多次，然而这种方式往往会花费很多时间。因此，需要使用不同的方法使应用程序更好的执行任务。  
较好的解决方案是使用对话框来输入计算范围，而使用对话框启动器来访问对话框。添加了对话框启动器的新界面如下图。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/RibbonXApplicationSample21.png)  
相应的XML代码如下：

**<dialogBoxLauncher>**

**<button** id="RedundantCalcsLaunch"

screentip="多重计算"

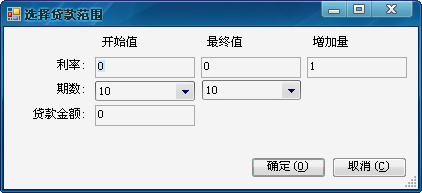
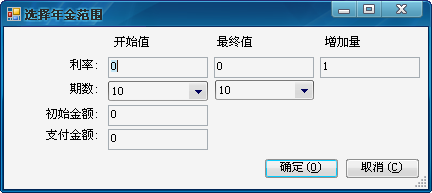
supertip="多次执行同一公式的计算."

onAction="DisplayRedundantCalc"**/>**

**</dialogBoxLauncher>**

要实现本例的功能，需要：

* 为每个等式创建不同的对话框，需要创建3个对话框。
* 代码必须区分不同的等式。
* 任何用于与用户交互的对话框需要与功能区中的信息相链接，功能区控件同样需要从对话框中更新。
* 代码对所需的ThisAddIn方法作多次调用，填充单元格区域中的单元格。
* 取决于你的需求，可以提供额外的信息，例如列标题和行标题，使输出更易理解。

**设计对话框**  
本例限定用户选择两个范围，这样可以在表中显示输出。设计的“选择贷款范围”窗体如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/RibbonXApplicationSample22.png)  
“选择年金范围”窗体设计如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/RibbonXApplicationSample23.png)  
“选择有效利率范围”窗体设计如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/RibbonXApplicationSample24.png)  
注：设计功能区的目的是减少复杂性。如果由于一些多余的设计选择而增加了更多的复杂性，那么用户将不会感受到使用功能区的任何好处。  
**创建计算代码**  
执行一次或多次计算实质上是相同的，但稍有不同。代码如下：

' 计算贷款金额并包括位置数据

**Public** **Sub** CalculatePMT(**ByVal** Rate **As** **Double**, **ByVal** NPer **As** Int32, **ByVal** PV **As** Int32, **ByVal** X **As** Int32, **ByVal** Y **As** Int32)

' 计算利率.

**Dim** PeriodicRate **As** **Double** = (Rate / 100) / 12

' 计算期数.

**Dim** Periods **As** Int32 = NPer \* 12

' 执行计算.

Application.ActiveWindow.ActiveCell.Cells(X, Y) = \_

"=PMT(" + PeriodicRate.ToString() + "," + \_

Periods.ToString() + "," + PV.ToString() + ",0,0)"

Application.ActiveWindow.ActiveCell.Calculate()

**End** **Sub**

' 计算年金金额并包括位置数据.

**Public** **Sub** CalculateFV(**ByVal** Rate **As** **Double**, **ByVal** NPer **As** Int32, **ByVal** PMT **As** Int32, **ByVal** PV **As** Int32, **ByVal** X **As** Int32, **ByVal** Y **As** Int32)

' 计算利率.

**Dim** PeriodicRate **As** **Double** = (Rate / 100) / 12

' 计算期数.

**Dim** Periods **As** Int32 = NPer \* 12

' 执行计算.

Application.ActiveWindow.ActiveCell.Cells(X, Y) = \_

"=FV(" + PeriodicRate.ToString() + "," + \_

Periods.ToString() + "," + PMT.ToString() + "," + \_

PV.ToString() + ",0)"

Application.ActiveWindow.ActiveCell.Calculate()

**End** **Sub**

' 计算有效利率并包括数据位置.

**Public** **Sub** CalculateEFFECT(**ByVal** Rate **As** **Double**, **ByVal** X **As** Int32, **ByVal** Y **As** Int32)

' 计算利率.

**Dim** PeriodicRate **As** **Double** = (Rate / 100) / 12

' 执行计算.

Application.ActiveWindow.ActiveCell.Cells(X, Y) = \_

"=EFFECT(" + PeriodicRate.ToString() + ",12)"

Application.ActiveWindow.ActiveCell.Calculate()

' 格式单元格.

**Dim** ThisRange **As** Excel.Range

ThisRange = Application.Cells(X, Y)

ThisRange.NumberFormat = "0.0000%"

**End** **Sub**

可以将上述代码与前文中相应的示例代码比较。最大的不同在于，添加了两个额外的位置参数，使得可以在任何位置放置输出的值。  
**定义对现有数据的链接**  
为功能区应用程序创建对话框时，需要创建对话框与合适的功能区控件的链接，否则两个元素将不同步并且应用程序会显示错误的数据。要创建链接，需要：

* 从功能区中获取信息
* 在功能区中设置信息
* 在XML中定义需要的回调

创建链接的XML代码如下：

**<editBox** id="Rate"

label="有效利率"

onChange="GetRateText"

getText="SetRateText"**/>**

**<dropDown** id="Term"

label="期数"

getVisible="TermVisible"

getItemCount="TermCount"

getItemID="TermItemID"

getItemLabel="TermItemLabel"

onAction="GetSelectedTerm"

getSelectedItemIndex="SetSelectedTerm"**/>**

**<editBox** id="Payment"

label="期初支付"

getVisible="PaymentVisible"

onChange="GetPaymentText"

getText="SetPaymentText"**/>**

**<editBox** id="Amount"

getLabel="AmountLabel"

getVisible="AmountVisible"

onChange="GetAmountText"

getText="SetAmountText"**/>**

用于实现链接的属性取决于控件类型。大多数控件使用getText属性，从应用程序中提取信息并在控件中显示（例如，本例中的文本框）；一些控件需要使用其它属性，例如本例中的getSelectedItemIndex属性。回调代码如下：

**Public** **Function** SetRateText(**ByVal** **control** **As** Office.IRibbonControl) **As** **String**

' 返回Rate变量的当前值.

Return Rate.ToString()

**End** **Function**

**Public** **Function** SetSelectedTerm(**ByVal** **control** **As** Office.IRibbonControl) **As** Int32

' 设置贷款项.

**If** CalcType = "Loan" **Then**

**Select** **Case** Term

**Case** 10

Return 0

**Case** 15

Return 1

**Case** 20

Return 2

**Case** 30

Return 3

**End** **Select**

**End** **If**

' 设置年金项.

**If** CalcType = "Annuity" **Then**

**Select** **Case** Term

**Case** 5

Return 0

**Case** 7

Return 1

**Case** 10

Return 2

**Case** 15

Return 3

**Case** 20

Return 4

**End** **Select**

**End** **If**

' 提供缺省的返回值.

Return 0

**End** **Function**

**Public** **Function** SetPaymentText(**ByVal** **control** **As** Office.IRibbonControl) **As** **String**

' 返回Payment变量的当前值.

Return Payment.ToString()

**End** **Function**

**Public** **Function** SetAmountText(**ByVal** **control** **As** Office.IRibbonControl) **As** **String**

' 返回Amount变量的当前值.

Return Amount.ToString()

**End** **Function**

**执行更丰富的计算**  
对话框启动器dialogBoxLauncher仅有一个onAction属性，因此任何计算的开始点都是该属性指向的方法DisplayRedundantCalc。当然，由于每个等式都不相同，因此都需要采用某种方式单独实现调用。

**Public** **Sub** DisplayRedundantCalc(**ByVal** **control** **As** Office.IRibbonControl)

' 选择正确的过程.

**Select** **Case** CalcType

**Case** "Loan"

PerformLoanRangeCalc()

**Case** "Annuity"

PerformAnnuityRangeCalc()

**Case** "Effective Rate"

PerformEffectiveRateRangeCalc()

**End** **Select**

**End** **Sub**

代码根据所选择的项目不同，调用不同的对话框计算程序。

**Private** **Sub** PerformLoanRangeCalc()

' 创建对话框.

**Dim** ThisSelection **As** LoanRangeSelection = **New** LoanRangeSelection()

' 在对话框中添加已存在的变量.

ThisSelection.txtIntBeg.Text = Rate.ToString()

ThisSelection.txtIntEnd.Text = Rate.ToString()

ThisSelection.txtIntInc.Text = "1"

ThisSelection.cbTermBeg.Text = Term.ToString()

ThisSelection.cbTermEnd.Text = Term.ToString()

ThisSelection.txtLoanAmt.Text = Amount.ToString()

' 显示对话框并且如果用户单击确定则处理数据.

**If** ThisSelection.ShowDialog() = DialogResult.OK **Then**

' 转换数据值为Int32egers.

Rate = Int32.Parse(ThisSelection.txtIntBeg.Text)

Term = Int32.Parse(ThisSelection.cbTermBeg.Text)

Amount = Int32.Parse(ThisSelection.txtLoanAmt.Text)

' 创建本地变量,包括计算数据.

**Dim** EndRate **As** Int32 = Int32.Parse(ThisSelection.txtIntEnd.Text)

**Dim** IncRate **As** Int32 = Int32.Parse(ThisSelection.txtIntInc.Text)

**Dim** EndTerm **As** Int32 = Int32.Parse(ThisSelection.cbTermEnd.Text)

' 更新功能区中的值.

ribbon.InvalidateControl("Rate")

ribbon.InvalidateControl("Term")

ribbon.InvalidateControl("Amount")

' 添加初始标题.

Globals.ThisAddIn.SetHeading("利息", 1, 1)

' 执行计算.

**Dim** i **As** Int32

**For** i = Rate **To** EndRate

' 计算X和Y的位置值.

**Dim** X **As** Int32 = i + 2 - Rate

**Dim** Y **As** Int32 = 2

' 打印Int32erest利率.

Globals.ThisAddIn.SetHeading(i.ToString() + "%", X, 1)

' 使用一系列if语句确定年设置.

**If** Term = 10 **And** EndTerm >= 10 **Then**

' 执行计算.

Globals.ThisAddIn.CalculatePMT(i, 10, Amount, X, Y)

' 打印标题.

Globals.ThisAddIn.SetHeading("10年", 1, Y)

' 如果已经使用则增加Y.

Y = Y + 1

**End** **If**

**If** Term <= 15 **And** EndTerm >= 15 **Then**

' 执行计算.

Globals.ThisAddIn.CalculatePMT(i, 15, Amount, X, Y)

' 打印标题.

Globals.ThisAddIn.SetHeading("15年", 1, Y)

' 如果已经使用则增加Y.

Y = Y + 1

**End** **If**

**If** Term <= 20 **And** EndTerm >= 20 **Then**

' 执行计算.

Globals.ThisAddIn.CalculatePMT(i, 20, Amount, X, Y)

' 打印标题.

Globals.ThisAddIn.SetHeading("20年", 1, Y)

' 如果已经使用则增加Y.

Y = Y + 1

**End** **If**

**If** Term <= 30 **And** EndTerm >= 30 **Then**

' 执行计算.

Globals.ThisAddIn.CalculatePMT(i, 30, Amount, X, Y)

' 打印标题.

Globals.ThisAddIn.SetHeading("30年", 1, Y)

' 如果已经使用则增加Y.

Y = Y + 1

**End** **If**

**Next**

**End** **If**

**End** **Sub**

**Private** **Sub** PerformAnnuityRangeCalc()

' 创建对话框.

**Dim** ThisSelection **As** AnnuityRangeSelection = **New** AnnuityRangeSelection()

' 在对话框中添加已存在的变量.

ThisSelection.txtIntBeg.Text = Rate.ToString()

ThisSelection.txtIntEnd.Text = Rate.ToString()

ThisSelection.txtIntInc.Text = "1"

ThisSelection.cbTermBeg.Text = Term.ToString()

ThisSelection.cbTermEnd.Text = Term.ToString()

ThisSelection.txtLoanAmt.Text = Amount.ToString()

ThisSelection.txtPayment.Text = Payment.ToString()

' 显示对话框并且如果用户单击确定则处理数据.

**If** ThisSelection.ShowDialog() = DialogResult.OK **Then**

' 转换数据值为Int32egers.

Rate = Int32.Parse(ThisSelection.txtIntBeg.Text)

Term = Int32.Parse(ThisSelection.cbTermBeg.Text)

Amount = Int32.Parse(ThisSelection.txtLoanAmt.Text)

Payment = Int32.Parse(ThisSelection.txtPayment.Text)

' 创建本地变量以包含计算数据.

**Dim** EndRate **As** Int32 = Int32.Parse(ThisSelection.txtIntEnd.Text)

**Dim** IncRate **As** Int32 = Int32.Parse(ThisSelection.txtIntInc.Text)

**Dim** EndTerm **As** Int32 = Int32.Parse(ThisSelection.cbTermEnd.Text)

' 更新功能区中的值.

ribbon.InvalidateControl("Rate")

ribbon.InvalidateControl("Term")

ribbon.InvalidateControl("Amount")

ribbon.InvalidateControl("Payment")

' 添加初始标题.

Globals.ThisAddIn.SetHeading("利息", 1, 1)

' 执行计算.

**Dim** i **As** Int32

**For** i = Rate **To** EndRate

' 计算X和Y的位置值.

**Dim** X **As** Int32 = i + 2 - Rate

**Dim** Y **As** Int32 = 2

' 打印Int32erest利率.

Globals.ThisAddIn.SetHeading(i.ToString() + "%", X, 1)

' 使用一系列if语句决定年设置.

**If** Term = 5 **And** EndTerm >= 5 **Then**

' 执行计算.

Globals.ThisAddIn.CalculateFV(i, 5, Amount, Payment, X, Y)

' 打印标题.

Globals.ThisAddIn.SetHeading("5年", 1, Y)

' 如果已经使用则增加Y.

Y = Y + 1

**End** **If**

**If** Term <= 7 **And** EndTerm >= 7 **Then**

' 执行计算.

Globals.ThisAddIn.CalculateFV(i, 7, Amount, Payment, X, Y)

' 打印标题.

Globals.ThisAddIn.SetHeading("7年", 1, Y)

' 如果已经使用则增加Y.

Y = Y + 1

**End** **If**

**If** Term <= 10 **And** EndTerm >= 10 **Then**

' 执行计算.

Globals.ThisAddIn.CalculateFV(i, 10, Amount, Payment, X, Y)

' 打印标题.

Globals.ThisAddIn.SetHeading("10年", 1, Y)

' 如果已经使用则增加Y.

Y = Y + 1

**End** **If**

**If** Term <= 15 **And** EndTerm >= 15 **Then**

' 执行计算.

Globals.ThisAddIn.CalculateFV(i, 15, Amount, Payment, X, Y)

' 打印标题.

Globals.ThisAddIn.SetHeading("15年", 1, Y)

' 如果已经使用则增加Y.

Y = Y + 1

**End** **If**

**If** Term <= 20 **And** EndTerm >= 20 **Then**

' 执行计算.

Globals.ThisAddIn.CalculateFV(i, 20, Amount, Payment, X, Y)

' 打印标题.

Globals.ThisAddIn.SetHeading("20年", 1, Y)

' 如果已经使用则增加Y.

Y = Y + 1

**End** **If**

**Next**

**End** **If**

**End** **Sub**

**Private** **Sub** PerformEffectiveRateRangeCalc()

' 创建对话框.

**Dim** ThisSelection **As** EffectiveRateRangeSelection = **New** EffectiveRateRangeSelection()

' 在对话框中添加已存在的变量.

ThisSelection.txtIntBeg.Text = Rate.ToString()

ThisSelection.txtIntEnd.Text = Rate.ToString()

ThisSelection.txtIntInc.Text = "1"

' 显示对话框并且如果用户单击确定则处理数据.

**If** ThisSelection.ShowDialog() = DialogResult.OK **Then**

' 转换数据值为Int32egers.

Rate = Int32.Parse(ThisSelection.txtIntBeg.Text)

' 创建本地变量以包含计算数据.

**Dim** EndRate **As** Int32 = Int32.Parse(ThisSelection.txtIntEnd.Text)

**Dim** IncRate **As** Int32 = Int32.Parse(ThisSelection.txtIntInc.Text)

' 更新功能区中的值.

ribbon.InvalidateControl("Rate")

' 添加初始标题.

Globals.ThisAddIn.SetHeading("利息", 1, 1)

Globals.ThisAddIn.SetHeading("有效利率", 1, 2)

' 执行计算.

**Dim** i **As** Int32

**For** i = Rate **To** EndRate

' 计算X和Y位置值.

**Dim** X **As** Int32 = i + 2 - Rate

' 打印Int32erest利率.

Globals.ThisAddIn.SetHeading(i.ToString() + "%", X, 1)

' 执行计算.

Globals.ThisAddIn.CalculateEFFECT(i, X, 2)

**Next**

**End** **If**

**End** **Sub**

**考虑数据识别需求**  
对输出结果添加有意义的标题。当执行多重计算时，为使数据意义明确，必须提供标题。代码如下：

**Public** **Sub** SetHeading(**ByVal** Heading **As** **String**, **ByVal** X **As** Int32, **ByVal** Y **As** Int32)

' 添加所需要的标题.

Application.ActiveWindow.ActiveCell.Cells(X, Y) = Heading

**End** **Sub**

现在，一切准备就绪。运行代码，在窗体中选择区域，输入相应数据，得到的输出结果如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/RibbonXApplicationSample25.png)

## [细品RibbonX(57)：示例—在Excel 2007的功能区中添加中文选项卡和控件](http://www.excelperfect.com/index.php/2009/12/24/ribbonxstudy57/)

2009年12月24日, 9:04 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**2** 人投票, 平均: **4.50** out of 5)

Loading ... Loading ...

由于Office 2007 CustomUI Editor并不支持中文，因此不能使用这个方便的工具来定制Ribbon，只能使用修改Office 2007文件格式的方法来解决了。此外，还可以使用Visual Studio来自定义中文选项卡。  
**步骤1** 在桌面上创建一个名为customUI的文件夹。  
**步骤2** 打开记事本，并在其中输入下面的XML代码：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui"**>**

**<ribbon>**

**<tabs>**

**<tab** id="rxtabCustom"

label="我自已的选项卡"

insertBeforeMso="TabHome"**>**

**<group** idMso="GroupFont"**/>**

**<group** idMso="GroupZoom"**/>**

**<group** id="myGroup" label="我的组"**>**

**<button** id="b1"

imageMso="HyperlinkInsert"

size="large"

label="启动网站"

onAction="surf"**/>**

**<button** id="b2"

imageMso="HappyFace"

label="微笑图标"

onAction="smile"**/>**

**<button** id="b3"

imageMso="FormatPainter"

label="格式刷图标"

onAction="paint"**/>**

**<button** id="b4"

imageMso="AutoFilterClassic"

label="筛选图标"

onAction="filter"**/>**

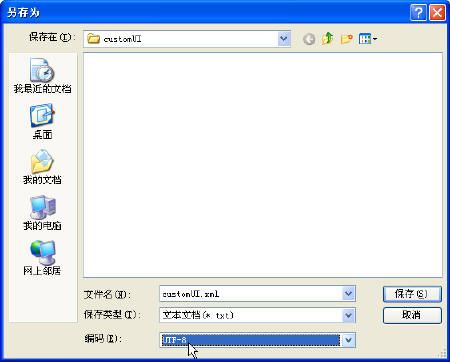
**</group>**

**</tab>**

**</tabs>**

**</ribbon>**

**</customUI>**

该XML文件在Excel的“开始”选项卡前创建了一个名为“我自已的选项卡”的选项卡，并添加了三个组，前两个为Excel内置的组“字体”和“显示比例”，第三个为名为“我的组”的自定义组，其中包括四个自定义的按钮“启动网站”、“微笑图标”、“格式刷图标”和“筛选图标”，并分别定义了onAction属性。  
**步骤3** 将该文件命名为customUI并以.xml为扩展名保存在customUI文件夹中，并关闭该文件。在保存时注意选择“编码”下拉框中的编码为UTF-8，如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/SaveAsUTF8Format.jpg)  
**步骤4** 在桌面上新建一个启用宏的Excel文件，并命名为MyCustomUI.xlsm。因为在自定义的XML中，包含了产生回调的onAction属性，所以创建的Excel文件需要启用宏。  
**步骤5** 在MyCustomUI.xlsm中，按Alt+F11组合键打开VBE，并插入一个标准模块，添加下面的代码供回调使用：

'Callback for b1 onAction

**Sub** surf(**control** **As** IRibbonControl)

ActiveWorkbook.FollowHyperlink \_

Address:="http://www.excelperfect.com", \_

NewWindow:=**True**

**End** **Sub**

'Callback for b2 onAction

**Sub** smile(**control** **As** IRibbonControl)

MsgBox "您单击了微笑图标!呵呵..."

**End** **Sub**

'Callback for b3 onAction

**Sub** paint(**control** **As** IRibbonControl)

MsgBox "您单击了格式刷图标!"

**End** **Sub**

'Callback for b4 onAction

**Sub** filter(**control** **As** IRibbonControl)

MsgBox "您单击了筛选图标!"

**End** **Sub**

**步骤6** 关闭工作簿MyCustomUI.xlsm。  
**步骤7** 在MyCustomUI.xlsm图标上单击右键，选择“重命名”，在文件名后添加“.zip”扩展名，使其变为一个压缩文件。如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/changeextendnameforzip.jpg)  
**步骤8** 双击该压缩文件，打开压缩包，将customUI文件夹拖到该压缩包中，如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/customuitozip.jpg)  
**步骤9** 将压缩包中的\_rels文件夹拖至桌面。  
**步骤10** 打开桌面中的\_rels文件夹，然后用记事本打开其中的.rels文件，如下图所示。  
[relsfile](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/relsfile.jpg)  
**步骤11** 在.rels文件的最后一个之前，添加下面的XML：

**<Relationship** Id="customUIRelID" Type="http://schemas.microsoft.com/office/2006/relationships/ui/extensibility" Target="customUI/customUI.xml"**/>**

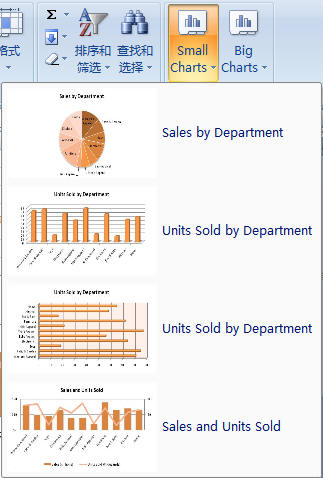
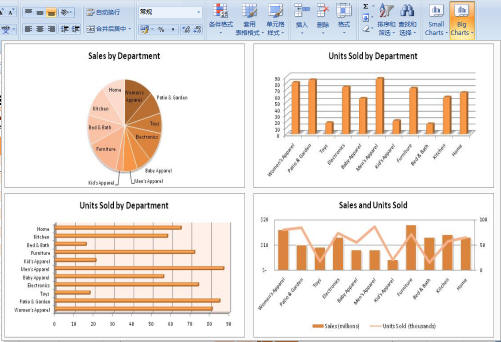
**步骤12** 保存该.rels文件并关闭。  
**步骤13** 将修改后的\_rels文件夹拖回MyCustomUI压缩包中。如果出现提示，则选择是，覆盖原文件。关闭压缩包。  
**步骤14** 将MyCustomUI.xlsm.zip的.zip扩展名删除。  
**步骤15** 双击MyCustomUI.xlsm文件，打开工作簿，此时在Excel内置选项卡“开始”的前面添加了一个自定义选项卡，在该选项卡中添加了两个内置组和一个自定义组，如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/MyCustomUIResult1.jpg)  
看看吧，都是中文的，成功！

## [细品RibbonX(58)：示例—创建动态的Ribbon库](http://www.excelperfect.com/index.php/2009/12/25/ribbonxstudy58/)

2009年12月25日, 1:01 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**2** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

Sam Radakovitz曾在Excel团队博客中发表过一篇文章，介绍了如何创建动态的Ribbon库，即如何通过RibbonX和VBA放置动态的图形图像到功能区库中，在该文中，作者创建了两个库：一个是带有标签的小图表图像，一个是大的图像。如图1和图2所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/CreateDynamicGallery1.JPG)  
图1：小图形库  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/CreateDynamicGallery2.JPG)  
图2：大图形库  
带有完整的RibbonX和VBA代码的示例文档见：  
  
下面对其进行简单的介绍。  
**RibbonX**  
先从RibbonX代码开始。RibbonX简单的解释就是定义Ribbon外观的XML。对于本例，RibbonX将定义所放置的两个库的位置、库按钮以及在库中的项目，RibbonX已存在于文档中，必须使用工具来获取文档中的RibbonX代码，您可以使用Office 2007 Custom UI Editor。打开“Office 2007 Custom UI Editor”，然后打开“Charts\_In\_Ribbon.xlsm”文档，其中的XML代码如下：

**<customUI** xmlns="http://schemas.microsoft.com/office/2006/01/customui" onLoad="ribbonLoaded"**>**

**<ribbon>**

**<tabs>**

**<tab** idMso="TabHome"**>**

**<group** id="grpChartsInRibbon" label="Charts"**>**

**<gallery** id="galSmallCharts"

label="Small Charts"

size="large"

showLabel="true"

imageMso="ChartPlacement"

columns="1"

rows="5"

itemHeight="90"

itemWidth="150"

supertip="Small pictures of your charts!"

getItemCount="getItemCount"

getItemID="getItemID"

getItemImage="getItemImage"

getItemSupertip="getItemSupertip"

getItemLabel="getItemLabel"

onAction="galRefreshAction"**/>**

**<gallery** id="galBigCharts"

label="Big Charts"

size="large"

showLabel="true"

imageMso="ChartPlacement"

columns="2"

rows="2"

itemHeight="291"

itemWidth="483"

supertip="Large pictures of your charts!"

getItemCount="getItemCount"

getItemID="getItemID"

getItemImage="getItemImage"

getItemSupertip="getItemSupertip"

onAction="galRefreshAction"**/>**

**</group>**

**</tab>**

**</tabs>**

**</ribbon>**

**</customUI>**

如果创建一个新文档并在Office 2007 Custom UI Edtior中打开该文档，则不会看到任何内容，因为该文档中没有RibbonX。下面，我们来介绍一个这个Ribbon XML的关键点：  
**回调**  
这里，告诉功能区调用VBA函数来获取：库中的项目数、项目的标签、悬浮在项目上时显示的工具提示。例如在RibbonX中：

getItemCount="getItemCount"

红色的getItemCount是RibbonX属性，告诉功能区去调用VBA函数来获取项目数，蓝色的getItemCount是要调用的VBA函数名称。如果想指定固定的项目数，则应使用：

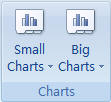
ItemCount="4"

**Ribbon对象**  
在customUI标签中添加

onLoad="ribbonLoaded"

将允许VBA代码获取ribbon对象的一个句柄，允许我们使ribbon无效。使该ribbon无效将强制ribbon在下次用户进入库时调用库程序（回调），为我们提供了更新库中项目和图像的一种方式。在下面的VBA部分将介绍如何使ribbon无效。  
**库的位置**  
在本示例中，我们在“开始”选项卡中创建了一个新组：

idMso="TabHome"

。因为我们没有指定“insertbefore”属性，因此将新按钮排在“开始”选项卡的右侧。  
**在库按钮中的图像**  
这里，为库按钮图像选取了一个已存在的ribbon图标：  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/CreateDynamicGallery4.JPG)  
这种结果是通过

imageMso="ChartPlacement"

来定义的。可以为每个按钮使用不同的图标，或者通过回调或定义图像属性装载自已的图标。  
**小库或大库**  
两个库对代码有相同的回调，因为它们的项目数和图表图片是相同的，它们在RibbonX里的不同是：

* 对于小库，itemWidth和itemHeight属性被设置得比大库更小。
* 小库有另一个回调，来定义库中每个图表的标签。由于在小库中的图表更小，所以添加标签能帮助识别图表。
* 对小库有更少的列，但有更多的行。设置小库为单列来帮助创建小库和大库之间不同的视图。

**VBA代码**  
本例中，oRibbon对象是Ribbon，将使用其来使Ribbon无效，以便可以强制更新库。

**Public** oRibbon **As** IRibbonUI

当ribbon装载RibbonX时，调用接下来的sub过程，传递Ribbon对象，这是获取Ribbon对象的唯一方式。该代码设置公共的ribbon对象为后面使用：

**Public** **Sub** ribbonLoaded(Ribbon **As** IRibbonUI)

**Set** oRibbon = Ribbon

**End** **Sub**

当用户在库中单击时，调用“getItemCount”，ribbon需要清楚其显示什么。因为我们在RibbonX中定义了回调，指定在库中有多少项。  
注意本例中的OnTime方法，如何使ribbon无效。默认情况下，ribbon将调用所有回调一次来获取项目和图像，并隐藏这些值直至使它们无效。这里，每次用户在库中单击时，使图像失效。处理这种操作有更顺畅的方法，但本示例将保持简单的方式。

**Sub** getItemCount(**control** **As** IRibbonControl, **ByRef** count)

count = ActiveSheet.ChartObjects.count

Application.OnTime DateAdd("s", 1, Now), "InvalidateRibbon"

**End** **Sub**

下一个程序为库中每个项目调用，将设置项目在库中显示的图像。本例中，想使所有的图像都在这里，因此Ribbon带有一个索引值，只是告诉索引的图表，然后告诉ribbon使用该图像。

**Sub** getItemImage(**control** **As** IRibbonControl, index **As** **Integer**, **ByRef** image)

ActiveSheet.ChartObjects(index + 1).Chart.Export ThisWorkbook.Path & "\Chart\_" & index + 1 & ".jpg", "jpg"

**Set** image = LoadPicture(ThisWorkbook.Path & "\chart\_" & index + 1 & ".jpg")

**End** **Sub**

“getItemID”程序在通过ribbon获取库项目的id时调用。在本示例中我们不使用。

**Sub** getItemID(**control** **As** IRibbonControl, index **As** **Integer**, **ByRef** id)

id = "Chart\_" & index

**End** **Sub**

下一个回调定义每个库项目的工具提示，或者超级工具提示。在本示例中，工具提示是series名称和公式，提供用户其原理或来源于哪里。这虽然不是特别有帮助的，但有趣。

**Sub** getItemSupertip(**control** **As** IRibbonControl, index **As** **Integer**, **ByRef** supertip)

**Dim** oSeries **As** Series

**Dim** sTooltip **As** **String**

**For** **Each** oSeries **In** ActiveSheet.ChartObjects(index + 1).Chart.SeriesCollection

sTooltip = sTooltip & vbCrLf & oSeries.Name & vbCrLf & oSeries.Formula & vbCrLf

**Next** oSeries

supertip = sTooltip

**End** **Sub**

“getItemLabel”回调仅被小库调用，通过RibbonX来定义，将设置库项目的标签。本例中，将使用图表标题作为标签，如果不存在则使用图表名称。

**Sub** getItemLabel(**control** **As** IRibbonControl, index **As** **Integer**, **ByRef** label)

**If** ActiveSheet.ChartObjects(index + 1).Chart.HasTitle **Then**

label = ActiveSheet.ChartObjects(index + 1).Chart.ChartTitle.Caption

**Else**

label = ActiveSheet.ChartObjects(index + 1).Name

**End** **If**

**End** **Sub**

下一个程序当用户在库项目中单击后调用。例如，用户单击某图表，代码将滚动到该图表并激活该图表。

**Sub** galRefreshAction(**control** **As** IRibbonControl, selectedId **As** **String**, selectedIndex **As** **Integer**)

ActiveWindow.ScrollIntoView ActiveSheet.ChartObjects(selectedIndex + 1).Left, ActiveSheet.ChartObjects(selectedIndex + 1).Top, ActiveSheet.ChartObjects(selectedIndex + 1).Width, ActiveSheet.ChartObjects(selectedIndex + 1).Height

ActiveSheet.ChartObjects(selectedIndex + 1).Activate

**End** **Sub**

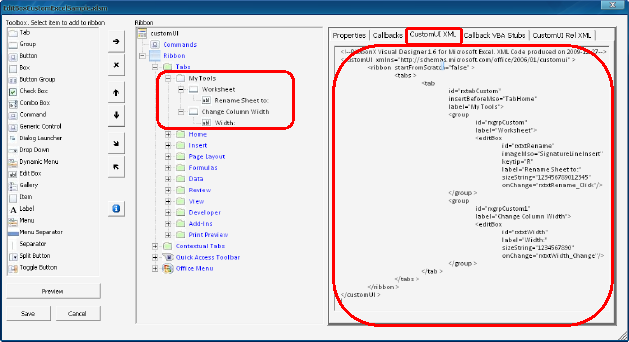
**最终的结果**  
在定义了RibbonX和VBA代码之后，打开该文档将显示两个新库，点击每个库的下拉箭头后，将显示工作表中最近图表的图像，单击库中的某图表将到达该图表。  
您可以修改本示例，以达到更丰富的效果。  
如果将本示例作为一个加载项，将可以在所有工作簿中使用。  
目前，本示例只是遍历活动工作表的图表，可以修改以显示整个工作簿中的图表，甚至是所有打开的工作簿中的图表。

## [细品RibbonX(59)：RibbonX可视化设计器](http://www.excelperfect.com/index.php/2009/12/28/ribbonxstudy59/)

2009年12月28日, 12:53 下午

1 颗星2 颗星3 颗星4 颗星5 颗星(**1** 人投票, 平均: **5.00** out of 5)

Loading ... Loading ...

让我们来看看[Andy Pope](http://andypope.info/)的作品：RibbonX可视化设计器（RibbonX Visual Designer），作为系品RibbonX细列的结尾。  
首先，在下列地址下载RibbonX可视化设计器加载项：  
[RibbonX Visual Designer](http://andypope.info/vba/ribboneditor.htm)  
或者：  
  
然后，将该加载项装载到Excel中。此时，在“开发工具”选项卡中出现名为“Ribbon Designer”的组，带有两个命令按钮：Load RibbonX和Designer，如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/visualribbonxdesign1.png)  
LoadRibbonX用来处理已存在的文件以维护RibbonX信息，如下图所示。中间的树形目录框列出了通用的选项卡和添加的组件，右侧显地了自定义的XML代码（当然，根据顶部的选项卡不同可以显示不同的代码）。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/visualribbonxdesign2.png)  
Designer用来设计功能区，如下图所示。  
[](http://www.excelperfect.com/wordpress/wp-content/uploads/2009/12/visualribbonxdesign3.png)  
这是一款针对RibbonX定制而开发的图形化设计器，版本1.6，如果获得最新版本的信息，可以访问[Andy Pope](http://andypope.info/)的网站。