  之前有一篇博文提到过node-webkit和有道的hex，今天就node-webkit开发做一个详细介绍吧。 顾名思义，node-webkit就是nodejs+webkit。这样做的好处显而易见，核心奥义在于，用nodejs来进行本地化调用，用webkit来解析和执行HTML+JS。

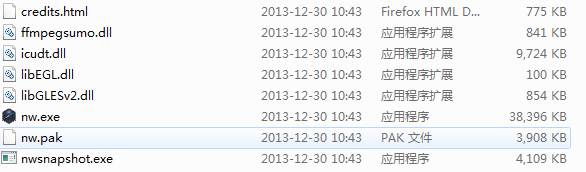
**快速上手**

**下载node-webkit**

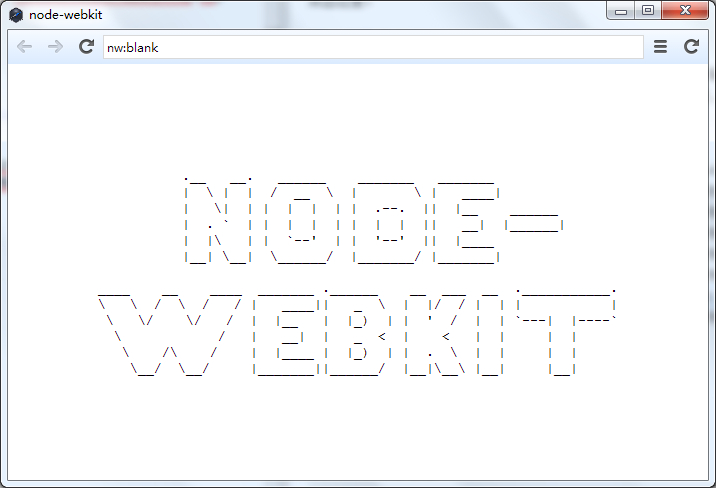
        点击这里：[https://github.com/rogerwang/node-webkit](https://github.com/rogerwang/node-webkit" \t "_blank)

        找到Downloads这一小节，然后下载对应平台的node-webkit预编译包。（为了介绍起来方便，本文后续内容全部针对windows平台，OSX和**[Linux](http://lib.csdn.net/base/linux" \o "Linux知识库" \t "_blank)**平台上的操作类似，就不做特意说明了。）

        下载完之后解压，可以看到如下内容：



        双击nw.exe直接运行，看到如下界面说明一切都ok，在你的机器上可以运行：



**Hello nw**

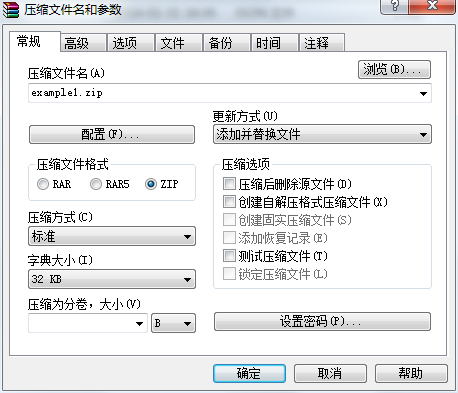
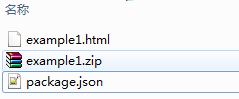
        老规矩，先来Hello World!

        建一个example1.html，内容如下：

**[html]** [view plain](http://blog.csdn.net/xiebaochun/article/details/39078041) [copy](http://blog.csdn.net/xiebaochun/article/details/39078041)

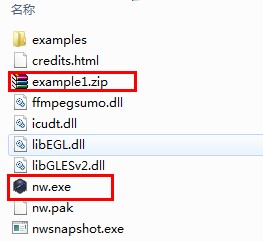
1. <!DOCTYPE html**>**
2. **<html>**
3. **<head>**
4. **<title>**Hello World!**</title>**
5. **</head>**
6. **<body>**
7. **<h1>**Hello node-webkit!**</h1>**
8. We are using node.js **<script>**document.write(process.version)**</script>**.
9. **</body>**
10. **</html>**
12. 在同一级目录下再建一个package.json，内容如下：
14. {
15. "name": "nw-demo",
16. "main": "example1.html"
17. }

 好了，用你机器上的压缩工具，rar或者7z或者神马神马的，把这两个文件压缩成exampl1.zip，如下图所示：

**注意，用什么工具压缩无所谓，但是一定要压缩成ZIP格式。**

        把example1.zip拷贝到与nw.exe同级的目录下，如下图：



        然后用鼠标把example1.zip拖动到nw.exe上运行（用命令行也可以，不过在windows下还是直接拖比较爽一点），效果如下图：



        这样，第一个例子就完成了，然后你自己可以去玩儿了。

        按照官方的搞法，example1.zip可改名成example1.nw，这样显得更加高大上一些。不过由于小僧的BIG一向比较低，私认为用.zip作为后缀也没啥不可以。

**来点儿JS**

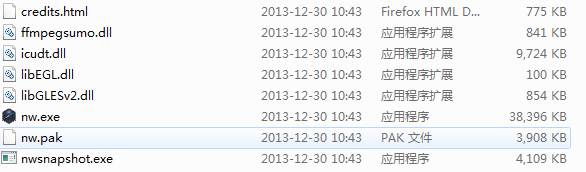
        第一个例子代码非常简单，你可以自己加点儿你最熟悉的JS代码，比如这样：

**[html]** [view plain](http://blog.csdn.net/xiebaochun/article/details/39078041) [copy](http://blog.csdn.net/xiebaochun/article/details/39078041)

1. <!DOCTYPE html**>**
2. **<html>**
3. **<head>**
4. **<title>**Example2**</title>**
5. **<meta** http-equiv="content-type" content="text/html;charset=UTF-8" **/>**
6. **</head>**
7. **<body>**
8. **<h1>**Example2**</h1>**
9. **<script>**
10. alert("This is example2!");
11. **</script>**
12. **</body>**
13. **</html>**

        显然，这里有很多**[HTML5](http://lib.csdn.net/base/html5" \o "HTML5知识库" \t "_blank)**的习惯写法，那当然是必须的对吧？因为webkit在google的调教之下早已成了HTML5的开路先锋了！

**打包和分发**



        如上图，很显然，要想让我们的应用跑起来，以上这些dll和pak文件是少不了的（至于每个文件是用来干嘛的，请参见官方的描述文档）。

        那么，对于我们做的example1这个简单的应用，怎么让我们的用户能运行它呢？当然，让我们的用户自己去下载node-webkit然后解压运行，这也算一种办法，这样的话我们只要把example1.zip文件扔给他就可以了。

        不过这样总觉得有点儿奇怪，要是能把所有相关的内容都打包成一个独立的exe文件就好了，比如打包成example1.exe，这样用户拿到之后直接双击就可以运行，这样显然是极好的。

        OK,let's [**Go**](http://lib.csdn.net/base/go)!

        首先，我们要合并example1.zip和nw.exe，形成一个新文件，名为example1.exe，这一点我们通过windows的命令行下的copy命令可以实现：

**[html]** [view plain](http://blog.csdn.net/xiebaochun/article/details/39078041) [copy](http://blog.csdn.net/xiebaochun/article/details/39078041)

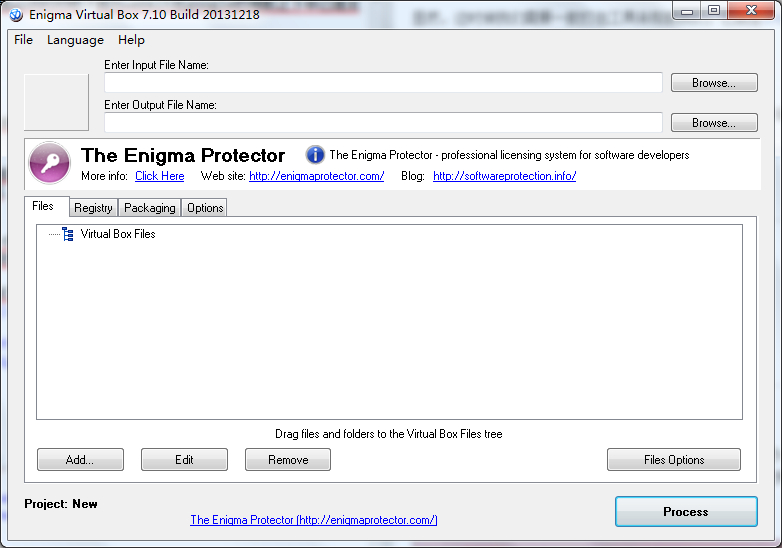
1. copy /b nw.exe+example1.zip example1.exe

**注意，nw.exe必须放在+号前面。**

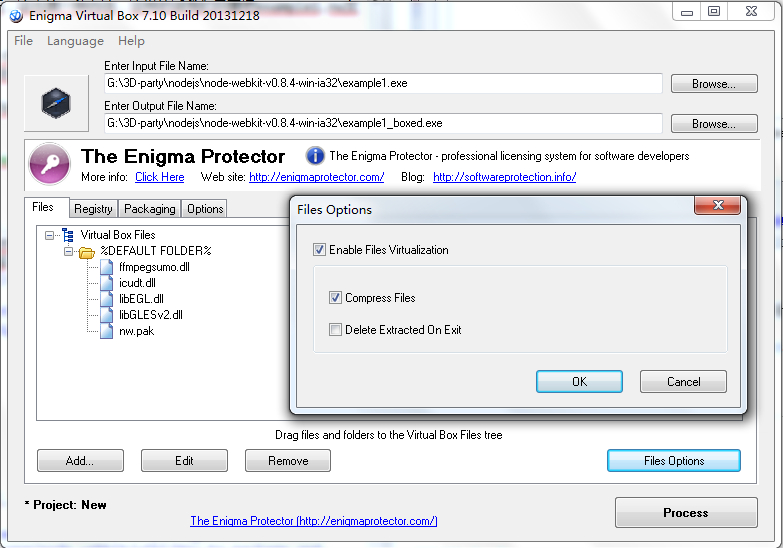
       这样就生成了一个名为example1.exe的文件，双击这个文件，看到的效果和第一个例子相同。很显然，这个example1.exe还是要依 赖那些dll文件才能运行，所以，这时候我们需要一款打包工具来帮助我们把这些example1.exe和dll文件都打包到一起，它就是[Enigma Virtual Box](http://enigmaprotector.com/en/aboutvb.html" \t "_blank)，大家可以点击这里下载安装：

<http://enigmaprotector.com/assets/files/enigmavb.exe>

        安装完成之后启动，看到如下界面：



        打包过程截图（请按照图中的配置操作）：



        点击右下角的Process按钮开始打包，最终获得了一个名为example1\_boxed.exe的文件。把这个文件拷贝到其它任意目录中双击，如果能顺利运行，说明打包成功！你可以把它分发给你的用户了！

        官方的文档上还介绍了其它几种打包分发方式，不过看起来都不是很实用，有兴趣的参见这里：

<https://github.com/rogerwang/node-webkit/wiki/How-to-package-and-distribute-your-apps>

**未竟的主题**

        以上就是关于node-webkit用法的一些简单示例，你可以沿着这个方向自己去探索了。有热心的网友把我们的Ext4.1中文文档打了一个包出来，可以到脚本娃娃的群共享（号83163037）去下载（比较大，不好放上来），入群口令nw

**如何最小化发布？**

        在官方发布的内容里面，icudt.dll这个文件有9724k，nw.exe有38396k，很显然，这个体积略显庞大了一些，如果我们的应用本身很小，只有几百K，而打包出来的东西有几十M显然是不合适的，这样非常不方便拷贝和网络传播。

        很显然，我们还需要一款工具，能够根据需要进行最小化的打包发布。

        不过目前我还没找到这样的好东东，各路大神如有知道的，求指教。

**如何利用nodejs调用本地接口？**

        很显然，既然nodejs都被打包进来了，网上众多的npm模块都是可以拿过来使用的，具体做法官方主页上都有比较详细的介绍，参见这里[https://github.com/rogerwang/node-webkit](https://github.com/rogerwang/node-webkit" \t "_blank)

**其它案例**

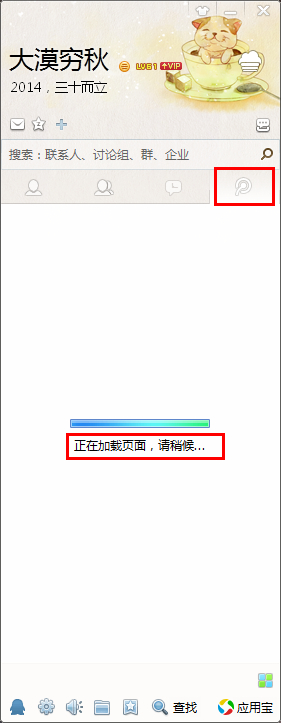
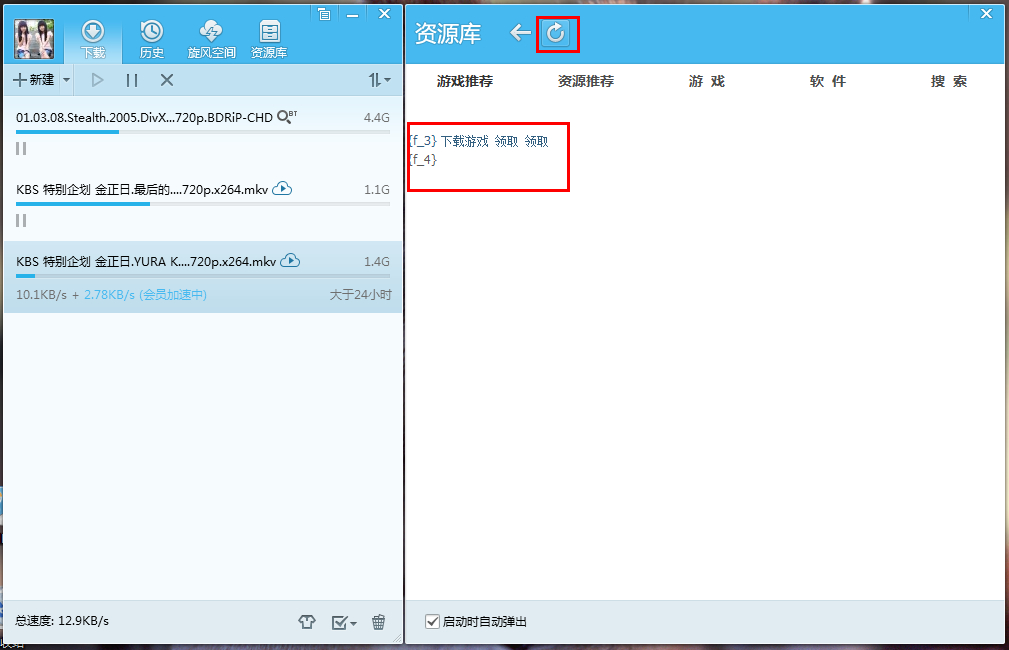
        官方的主页上还展示了众多使用node-webkit制作的应用，各位自行点击查看即可。

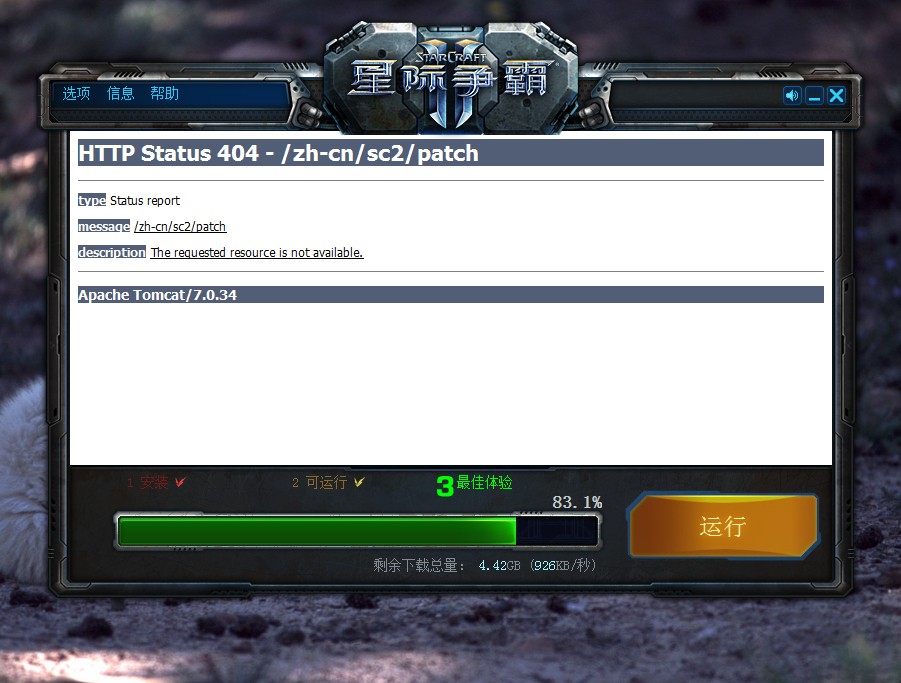
**展望未来**

**其实这条路老早就有人在走**

        网上有很多人在争论，未来究竟是原生的应用会胜出，还是WEB APP会胜出，实际上这两者并不是你死我活的关系，而是会互相融合，走向【混合型应用】这样一种形态。

先看几个界面：



        大家可以把自己的网络断掉测试以上界面，QQ有时候甚至能爆出JS错误，哈哈。

        2011年的时候，我发过一篇文章，《未来五年，桌面应用和WEB走向融合》，当时只是因为看到诸如酷我音乐、腾讯QQ这些桌面应用的开发方式而有 所感悟，对于桌面应用会以什么样的形式和B/S开发模式融合在一起并没有确切的想法。而最近由于打包Ext4.1中文文档的缘故知道了node- webkit，突然有一种恍然大悟的赶脚！

        当然，酷我音乐盒和疼讯目前是用什么样的技术实现的小僧不知道其中的细节（人家也不想让外人知道对吧？）。不过，小僧相信，对于屌丝单兵作战来说，疼讯的实现方式肯定会比node-webkit更加复杂，目测个人无法模仿。

**跨平台+多终端的神器**

        毫无疑问，这货会成为2014年的一款神器。因为它具备了以下这些无可比拟的优势：

1. **人力成本最低**：招一个会C++的码农要花多少钱？招一个牛逼哄哄的既会C++还对QT、MFC等了如指掌的码 农又要花多少钱？对比一下，招一个会JS+CSS的呢？如果你是老板，你怎么想？再说了，市面上会C++的码农更多，还是会JS+CSS的码农更多？无需 小僧多言，其中的奥妙您自个儿体会。
2. **APP的体积更小**：列位一定还记得当年大明湖畔的疼讯QQ只有十几M的体积，俗话说，岁月是把杀猪刀，在集成了疼讯的各种应用之后，如今的QQ身材早已发福走形。可以想象，如果QQ不使用嵌入浏览器内核的方式，恐怕体积早就超过100M了吧！
3. **不需要整体更新APP**：如果是按照传统的方式开发桌面应用，每次升级的时候都需要用户重新下载整个应用，比如疼讯QQ。但是用node-webkit的这种方式，可以很好地避免这个问题，就像网站一样，网站内容可以不断更新，但是浏览器本身的更新不需要那么频繁。
4. **跨平台**：这一点是由NodeJS来保证的。
5. **多终端**：由于nodejs和chromium本身都是可以在桌面平台和移动平台上运行的，因此用这货来支持多终端开发，保持技术栈的一致性是非常合适的。
6. **完善的工具链**：由于nodejs的兴起，它下面的npm模块数量已经有几十万个；而各种基于chrome内核的定制版浏览器就不用再介绍了。加上传统B/S开发模式下已经具备的完整开发和设计工具，用node-webkit来开发桌面应用已经具备了完善的工具链。

**参考资料**

        关于node-webkit作者的介绍参见这里：

[http://www.csdn](http://www.csdn.net/article/2014-01-08/2818066-Node-Webkit)[**.NET**](http://lib.csdn.net/base/dotnet)/article/2014-01-08/2818066-Node-Webkit

        node-webkit在github上的地址：

<https://github.com/rogerwang/node-webkit>

        由于node-webkit的作者提供的文档都是E文的，为了方便大中华局域网内的一众屌丝，我的好基友@老男孩对文档做了一些翻译，参见这里：

<https://github.com/liu78778/node-webkit>

来源： <http://blog.csdn.net/xiebaochun/article/details/39078041>