##### Windows下环境配置

IDE的使用，Windows下，推荐使用Visual studio 2012或更高版本；在Mac下，推荐使用Xcode5.0或更高版本。

**Visual studio 2013**

**下载地址：**<http://www.visualstudio.com/downloads/download-visual-studio-vs>

**CrossApp\_alpha\_v0.0.1**

**下载地址：**

**CrossApp工作Git ：**<http://git.oschina.net/9miao/CrossApp>

**CrossApp GitHub ：**<https://github.com/9miao/CrossApp>

**adt-bundle-windows-x86\_64-20130917**

下载地址：<http://developer.android.com/sdk/index.html>

**android-ndk-r9c**

下载地址：<http://developer.android.com/tools/sdk/ndk/index.html>

**jdk-7u45-windows-x64**

下载地址：<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

**python-2.7.3.amd64**

下载地址：<https://www.python.org/downloads/windows>

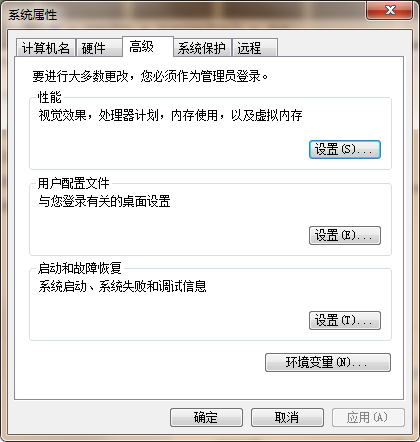
以上是windows下配置跨平台开发环境所需的工具，这里列出的工具比较常用，当然以上列出的工具也可以使用其他的版本，但配置的方法基本一样。在windows下配置环境相对复杂些，因为windows本身是不集成python和java环境的，需要自己手动配置这两个环境。请事先下载好配置环境所需的所有工具包

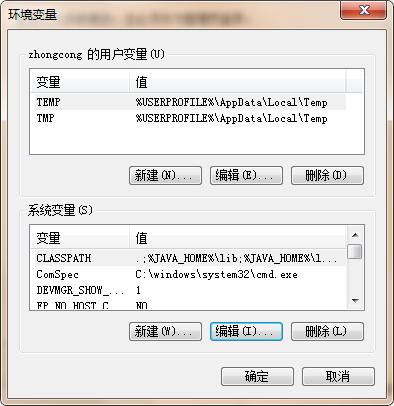
安装java环境

双击jdk，直至安装完成，安装的过程没有什么特别要求。接下来就是配置jdk的环境

变量，右键计算机->系统高级设置->环境变量。







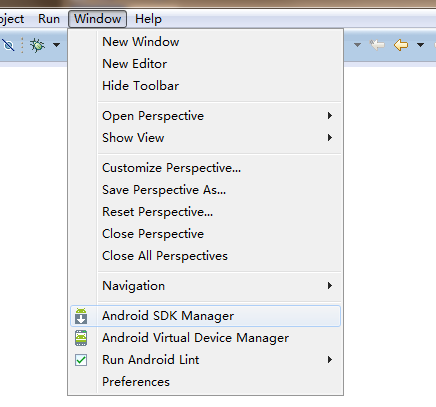
在系统变量中点击新建，新建一个变量名为JAVA\_HOME的变量，变量值为jdk的安装目录，例如：C:\Program Files\Java\jdk1.7.0\_45。再新建一个名为CLASSPATH的变量，变量的值为：.;%JAVA\_HOME%\lib;%JAVA\_HOME%\lib\tools.jar(最前面的.;一定要加上)，在系统变量中找到Path变量，点击编辑，在变量值最前面添加一个分号，然后在分号的前面添加%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin。这样java的环境变脸已经配置好了，打开dos，输入java、javac、java –version等命令就能看见相关的信息。

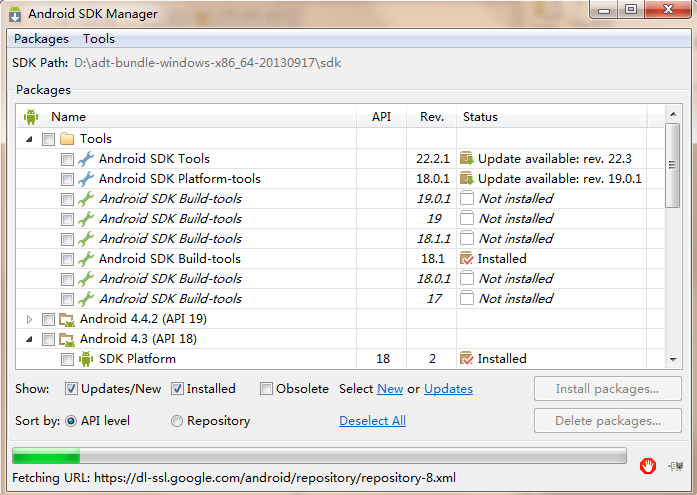
配置pyhton环境

双击python安装包直至安装完成，安装完成后和java一样，需要在系统变量中配置一下python的系统路径，打开系统变量Path，在变量值的最后面先添加一个分号，然后在分号的后面添加python的安装路径，例如：C:\Python27，打开dos，输入python，即可进入python的命令行。

更新adt以及配置ndk

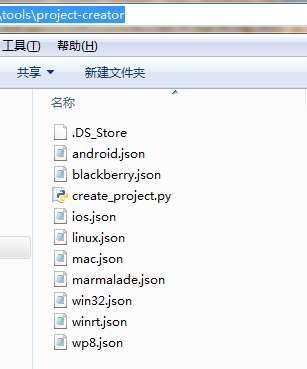
其次，将下载好的adt和ndk解压在指定目录就行，运行adt里面的sdk目录下的eclipse，在菜单栏中点击windows，点击Android SDK Manager在线更新sdk，有更新的话尽量全选，这个更新的过程会话费较多的时间，没办法，那就等吧！直至sdk更新完成(后面的在xcode下配置环境同样需要经过这一步)



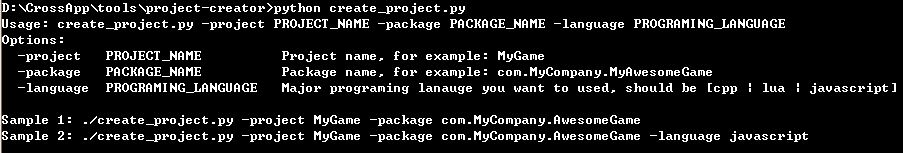


再者，这里需要注意的，不需要安装cygwin，而是直接使用ndk，因为从ndk-r7开始已经集成了linux环境，单就跨平台移植无需再安装cygwin。

**现在用CrossApp来创建第一个工程**

解压CrossApp，进入CrossApp\tools\project-creator目录，找到create\_project.py,

创建工程使用的就是这个.py文件，打开dos，通过cd命令，将当前目录定位到CrossApp\tools\project-creator，运行python create\_project.py就可以看见新建工程的示例说明，这里我们新建一个名为MyFirstApp的工程

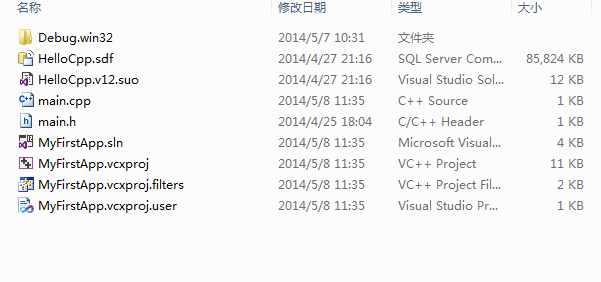


当看见Have Fun则表示工程新建成功了

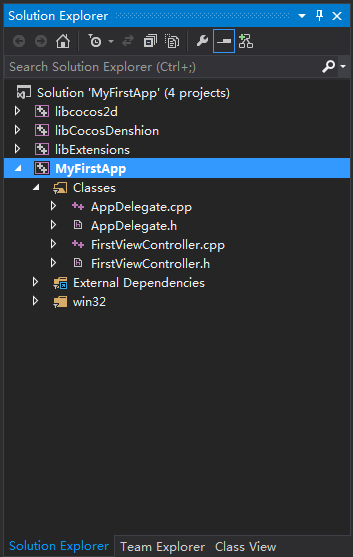


最后，工程创建好了以后，在CrossApp的projects目录下就能看见我刚才新建好的工程，进入新建的工程，可以直接运行proj.win32下的.sln，



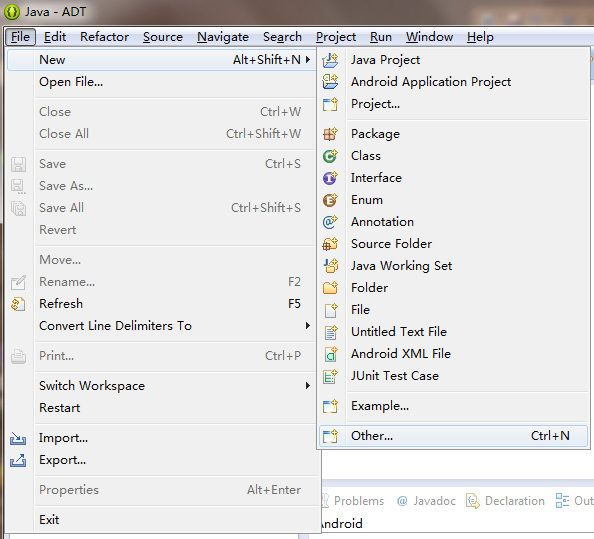


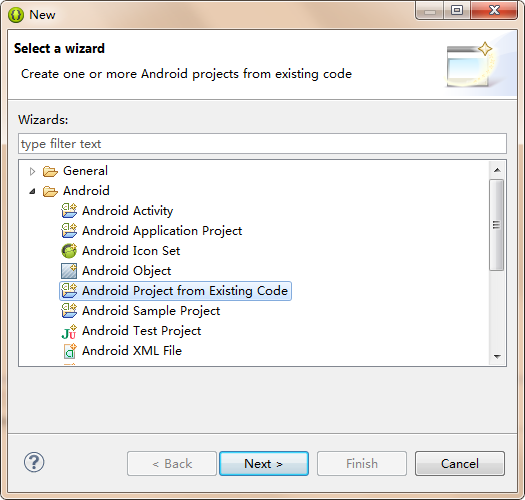
双击运行MyFirstApp.sln,进入vs工程界面，vs界面右面的解决方案资源管理器就是我们新建的工程的代码结构，展开MyFirstApp下的Classes目录，可以看见工程已经默认为我们创建好了两个类，其中AppDelegate就是我们工程的入口类，我们只需要在Classes下面添加代码就可以开始编写我们自己的App了。

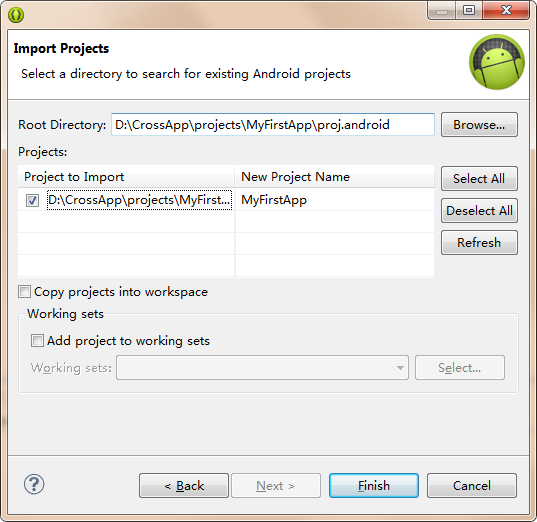


按Ctrl+F5或者点击菜单栏DEBUG->Start Without Debugging可以直接运行工程，就能看见我们的HelloWorld界面。

如果需要移植到android，先将工程导入eclipse，File->New->Other->Android project from existing code,点击browse找到我们刚创建的工程目录下的proj.android,完成。

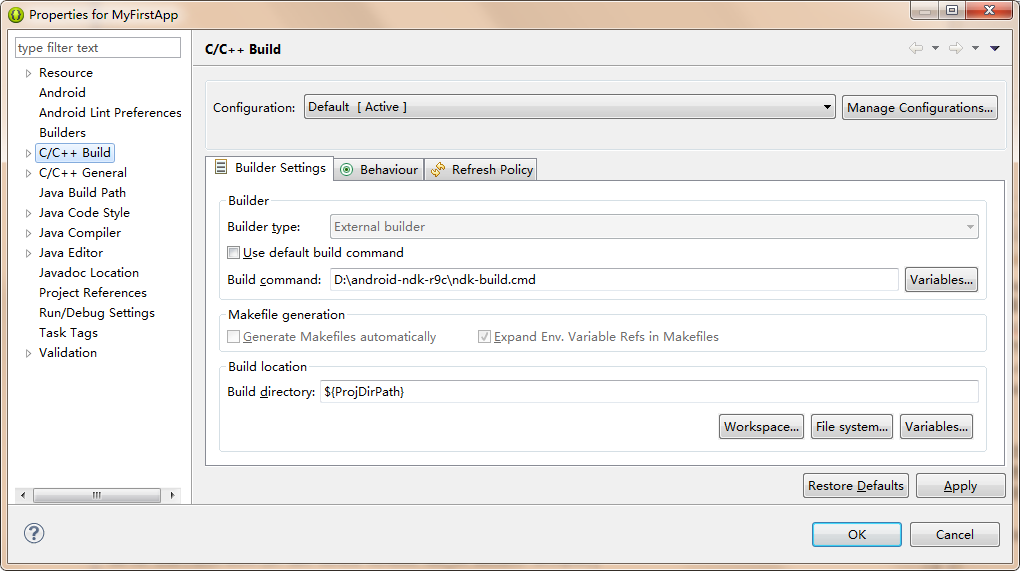






在eclipse中展开工程，打开jni目录下的Android.mk文件，我们看见AppDelegate.cpp和HelloWorldSence.cpp这两个文件已经默认为我们添加好了。在工程中会看见很多错误，这是缺少CrossApp库导致的，将CrossApp\CrossApp\platform\android\java\src路径下的文件夹复制到工程的\proj.android\src目录下，刷新一下工程就可以了。如果assets目录为空，则需要我们将工程的Resources目录下的所有文件都手动复制到assets目录下。

然后需要为工程配置ndk编译环境，在菜单栏选择Project，点击properties，在打开面板中点击C/C++ Build，在Build command一项中，把默认的command替换为自己nkd目录下的nkd-build.cmd，例如：D:\android-ndk-r9c\ndk-build.cmd。



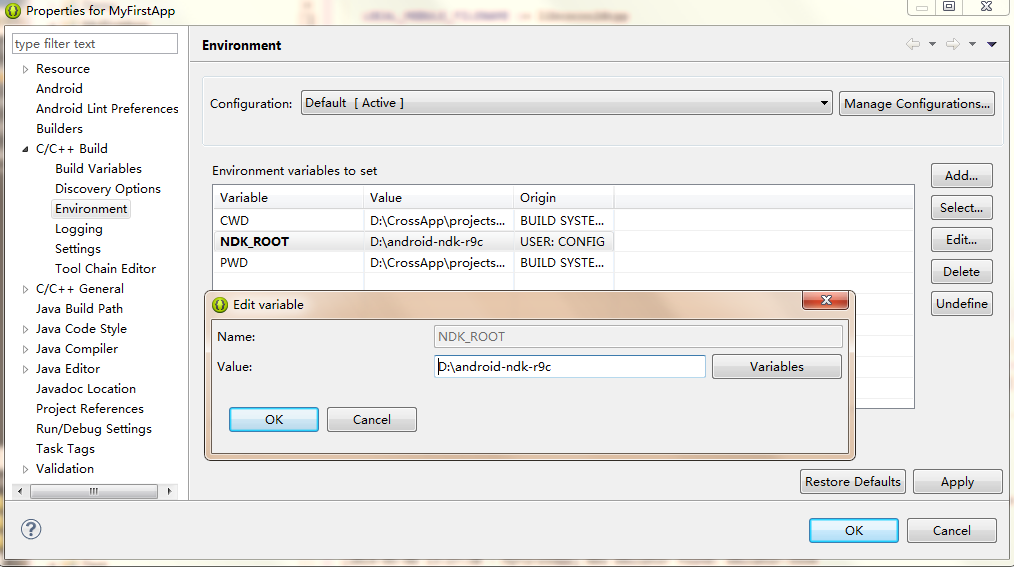
现在可以右键工程->run as android application，但编译时在console中又出现了错误，提示找不到cocos2d-x的相关路径，不用担心，我们只需要将对应的路径添加进去就可以了，在android.mk中找到$(call import-module,cocos2dx)，在这句前面添加如下两段代码：

$(call import-add-path, D:/CrossApp)

$(call import-add-path, D:/ CrossApp /CrossApp/platform/third\_party/android/prebuilt)

路径修改为自己的安装路径就行。

  路径修改为自己的安装路径就行，如果出现Please define NDK\_ROOT错误,则要求我们添加一下ndk的路径，在properties页面展开C/C++ Build，在Environment新建一个名为NDK\_ROOT的变量，值为NDK的解压目录。



​ 以上步骤完成后就可以开始编译我们的CrossApp工程，最好接上真机，直接在真机上运行查看效果，使用模拟器通常会报错。

  至此windows下android开发环境搭建完成。