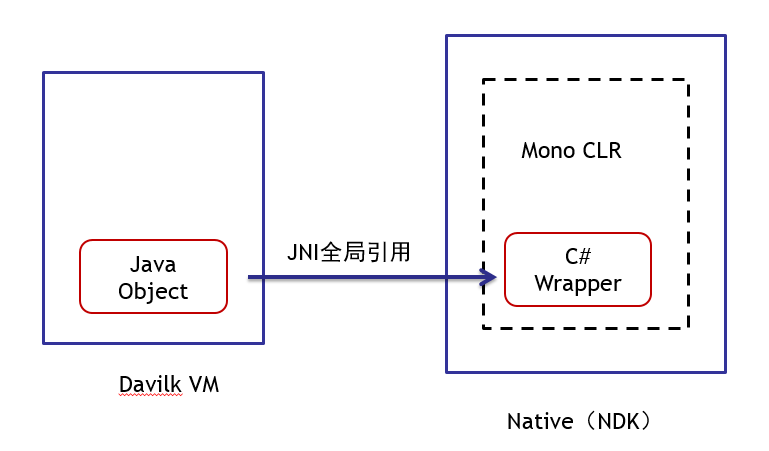
# Xamarin Android第三方库绑定指南

使用Xamarin开发App，最纠结的地方应该就是绑定第三方库了…

## 原理

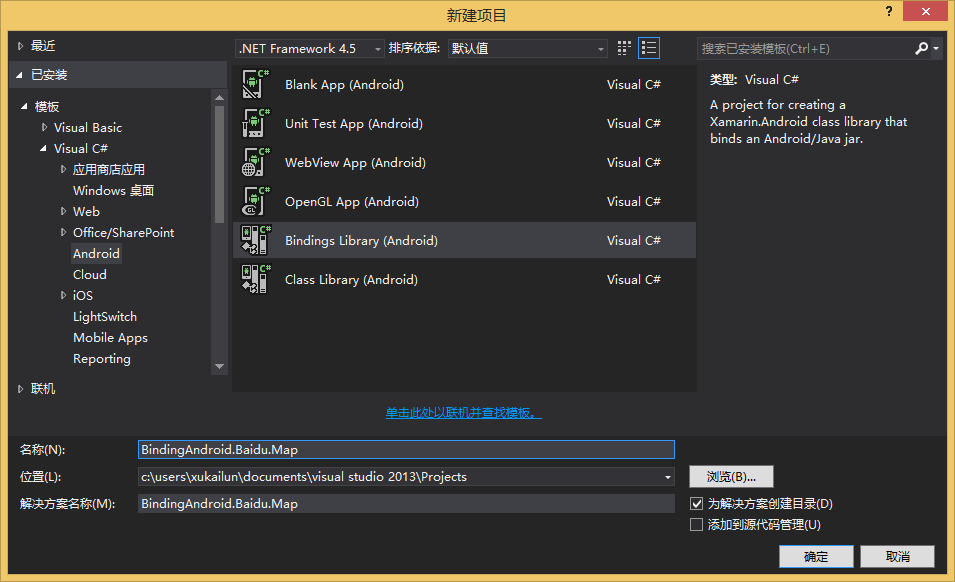
下图能基本解释Xamarin Android的工作原理，详细解释间官方文档[**Android Callable Wrappers**](http://developer.xamarin.com/guides/android/advanced_topics/java_integration_overview/android_callable_wrappers/)。



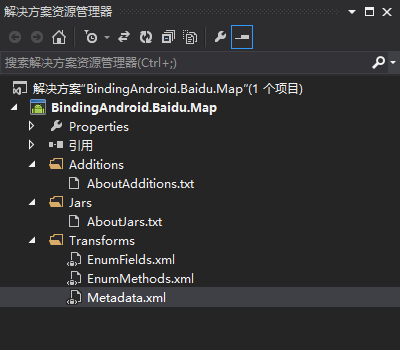
## 项目结构

* 创建Android绑定项目

新建 -> 项目，Visual C# -> Android -> **Binding Library (Android)：**



* 目录结构

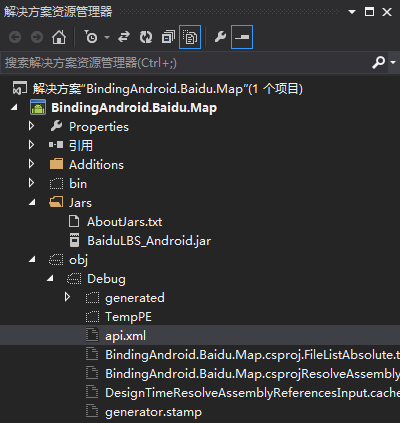


整体的目录结构如上图：

* **Additions**：自动生成的代码无法满足你的要求的时候，可以在这个地方手动添加代码文件，如无必要，留空即可。
* **Jars**：需要绑定的jar库，放在这里。需要生成C# API的jar，生成方式为**EmbeddedJar**；如果只是被引用，而不需要接口给C#，则选择**EmbeddedReferenceJar**。详见[**Binding a Java Library**](http://developer.xamarin.com/guides/android/advanced_topics/java_integration_overview/binding_a_java_library_%28.jar%29/)。
* **Transforms**：生成C# API时需要额外设定的规则。
  + **EnumFields.xml**：这个文件中的规则应用于Java类中的常量字段，使他们能够用C#的枚举替代；
  + **EnumMethod.xml**：这个文件中的规则应用于Java方法中的参数，使他们能够用C#的枚举替代；
  + **Metadata.xml**：这个文件中的规则应用于Java->C# API生成的各类规则，是绑定工作的重点所在。

作为栗子，我们试一下百度地图SDK和定位SDK，新版本的两个SDK有[绑在一起的安装包](http://lbsyun.baidu.com/sdk/download)。把**BaiduLBS\_Android.jar**放到**Jars**目录下，生成类型**EmbeddedJar**。

重新生成，可以看到若干错误，再看一下obj下面的目录结构发生了变化：

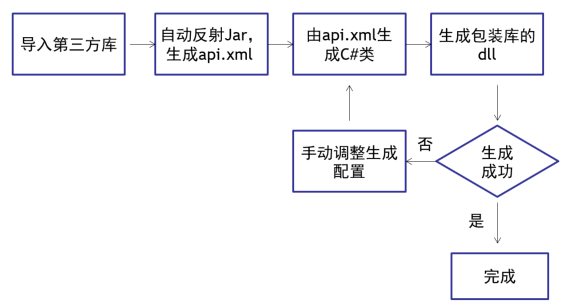


* **obj**：生成过程中间文件所在目录；
  + **Debug**
    - **generated**：自动生成的C#绑定代码所在目录
      * enums
      * **src**
      * metadata
    - **api.xm**l：Java -> C#转换的配置文件，由xamarin自动生成。**非常重要**！

## 绑定步骤

在上一步，我们添加jar，重新生成之后，遇到了错误。下面我们要做的就是通过修改各种配置，修复这些错误，最后使项目生成成功，得到绑定后的dll。

我们的目标就是生成这个dll，并且生成过程中**必须保证0 warning**！



在生成的过程中，xamarin做了这么一系列事情：

1. 反射jar文件，根据文件中暴露给外部的接口，生成**api.xml**。

这个文件描述了jar中需要转换成C#的各种信息。如果这一步出错，错误来源显示为**JARTOXML**。

1. 读取**api.xml**，合并**Transforms**中的规则（主要是**metadata.xml**），生成C#代码。

生成后的C#代码位于**genereated**目录中。如果这一步出错，错误来源显示为**BINDINGSGENERATOR**。

1. 编译**genereated**目录中的C#代码，合并**Additions**中的代码，最终打包生成dll。

如果这一步出错，错误来源一般多是各个生成的cs文件。

在上面的步骤中，我们发现我们可以做有这两件事儿：

1. 在**Transforms**中添加我们想要自定义规则；
2. 在**Additions**中添加我们想要的自定义代码；

在**多数情况**下，我们只需要做1）就能实现目标，某些情况下需要2），极端情况下需要完全手动实现转换。

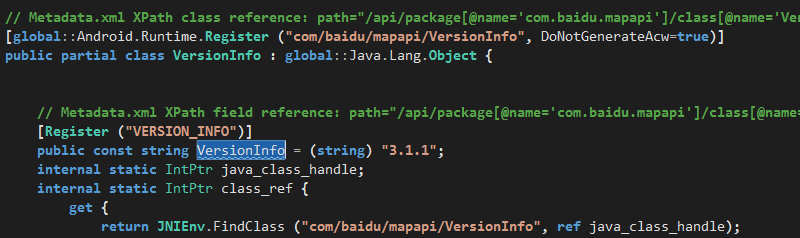
## 绑定细节

* 成员名称不能与它们的封闭类型相同：

C:\Users\xukailun\AppData\Local\Temp\Image.png

**问题**：这是地图SDK生成时唯一个error，产生的原因很简单，常量**VERSION\_INFO**再规范化成**VersionInfo**后，与类名相同。

这类问题会比较多，除了此例中规范化之后恰好重名的情况，还有不少是Java中原本就重名的（注：一般出现在混淆后的代码中，Java允许内部类、方法和成员变量使用同样的名字，而C#却不允许，理由很显然，就当思考题了）。



**解决方法**：

需要对这个常量改名，但是不能在自动生成的这个cs里改，因为它每次都会被覆盖。

应当在**metadata.xml**中添加配置：

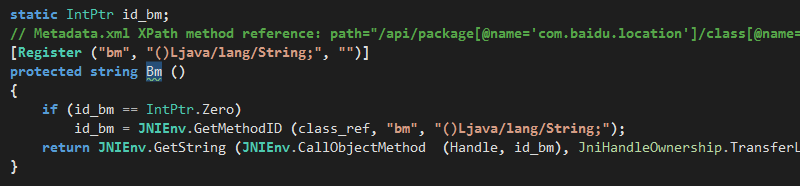
|  |
| --- |
| <attr path="/api/package[@name='com.baidu.mapapi']/class[@name='VersionInfo']/field[@name='VERSION\_INFO']" name="managedName">SDKVersionInfo</attr> |

**attr**表示对api.xml中的结点做修改操作，path后接一个**XPath**表达式，用于定位结点name的值表示对这个结点做哪一种操作（参考：[API Metadata Reference](http://developer.xamarin.com/guides/android/advanced_topics/java_integration_overview/binding_a_java_library_%28.jar%29/api_metadata_reference/)），**managedName**表示修改名称，最后修改为SDKVersionInfo，躲避重名。

* 在密封类中声明了新的保护成员

C:\Users\xukailun\AppData\Local\Temp\Image.png

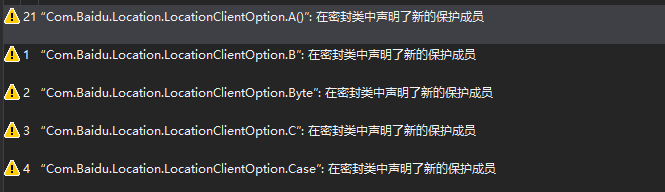
**问题**：定位SDK的**BDLocation**，有一个叫bm的保护类型字段，然而这个类本身又是final的，因此到了C#中编译器叫会warning。这个warning其实无伤大雅，但是为了0 warning，还是解决一下吧。



**解决方法**：在**metadata.xml**中添加配置，屏蔽掉bm这个字段，使之不出现在C#中：

|  |
| --- |
| <remove-node path="/api/package[@name='com.baidu.location']/class[@name='BDLocation']/method[@name='bm']"/> |

* 需要同时屏蔽多个成员变量



**问题**：我们使用remove-node这种方式可以屏蔽字段、方法或者类（注：实际上所有**api.xml**中出现的结点都可以用这种方式屏蔽），这里需要屏蔽多个。

**解决方法**：上文中提到，path的值是一个**XPath**，所以我们实际上可以用这个XPath指定多个结点。比如我们需要remove掉LocationClientOption下面所有的字段：

|  |
| --- |
| <remove-node path="/api/package[@name='com.baidu.location']/class[@name='LocationClientOption']/field"/> |

类似的，我们需要屏蔽com.baidu.location下面所有只有一个字母长度的类（这些类都是混淆出来的），可以这么写：

|  |
| --- |
| <remove-node path="/api/package[@name='com.baidu.location']/class[string-length(@name)=1]"/> |

* 某个类找不到它的基类



**问题**：这个类b的有个基类a，他俩显然是被混淆了的。对于xamarin来说，被混淆的类它一般是不会做转换的。这里的b被转换了，原因一般是b在其他一些地方被用到。

**解决方法**：

遇到这种情况，处理方式比较灵活：

* 把a恢复出来，还是用att，name=” obfuscated”，值写成false：

|  |
| --- |
| <attr path="/api/package[@name='com.baidu.mapapi.map']/class[@name='a']" name="obfuscated">false</attr> |

* 把b也干掉：

|  |
| --- |
| <remove-attr path="/api/package[@name='com.baidu.mapapi.map']/class[@name='b']"/> |

* 类中包含一个未知类型的字段

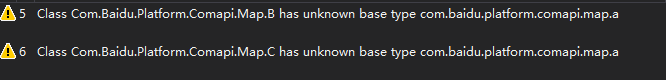


**问题**：就是有一个被混淆的类，而以它为类型的一个公开成员变量出现在一个公开的类里。这种问题显然是作者没有好好写代码，这种成员根本不应该公开的。

**解决方法**：把这个成员变量屏蔽了。不过这里成员变量的名字，得用**jd-gui**之类的工具去看一下。

|  |
| --- |
| <remove-node path="/api/package[@name='com.baidu.location']/class[@name='BDNotifyListener']/field[@name='mNotifyCache']"/> |

* 非公开API的问题



**问题**：仔细看，你会发现包括MapRender在内，虽然com.baidu.platform.comapi.map下面有一些公开的类，但是转换时问题多多。查阅API文档，发现这里整个包都不在其列，应当是非公开的API。

**解决方法**：屏蔽整个包。

|  |
| --- |
| <remove-node path="/api/package[@name='com.baidu.platform.comapi.map']"/> |