C#移动跨平台开发（2）Xamarin移动跨平台解决方案是如何工作的？ - 文章 - 伯乐在线

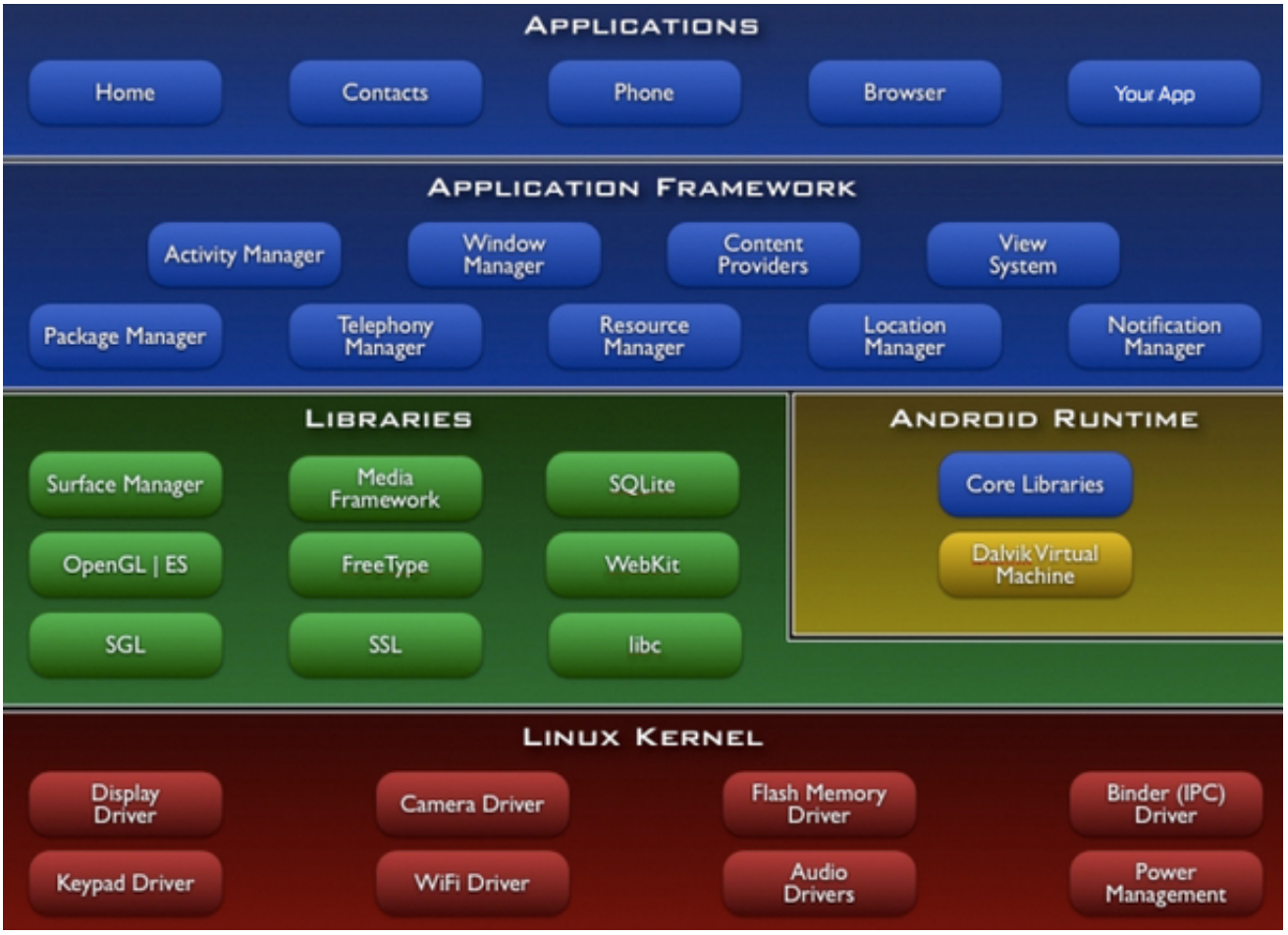
概述

上一篇 C#移动跨平台开发（1）环境准备发布之后不久，无独有偶，微软宣布了开放.NET框架源代码并且会为Windows、Mac和Linux开发一个核心运行时（Core CLR），这也是开源的！IT媒体网站纷纷转载，博客园的C#开发者们热泪盈眶（泥煤都等这一天好久了！）

与此同时VS2015预览版更是直接集成了Android模拟器，但是其实里面并没有说集成IOS模拟器，我不知道大家是怎么得出可以直接用VS来开发Android和IOS应用的。不管怎么说，这都是个好消息。那么问题来了，C#如何来开发Android和IOS应用？微软会怎么做我们不确定，但是我们倒是可以来看看Xamarin是如何做的。

Android系统架构

我想下面这张图做Android开发的同学应该很熟悉，下面我们就通过来了解Android系统的架构入门来看看Xamarin会怎么样去做？



- Linux Kernel 操作系统层
- Libraries And Android Runtime 各种库和Android 运行环境
- Application Framework 应用框架层 （由Java编写）
- Applications 应用程序层（由Java编写并且在Dalvk虚拟机来运行）

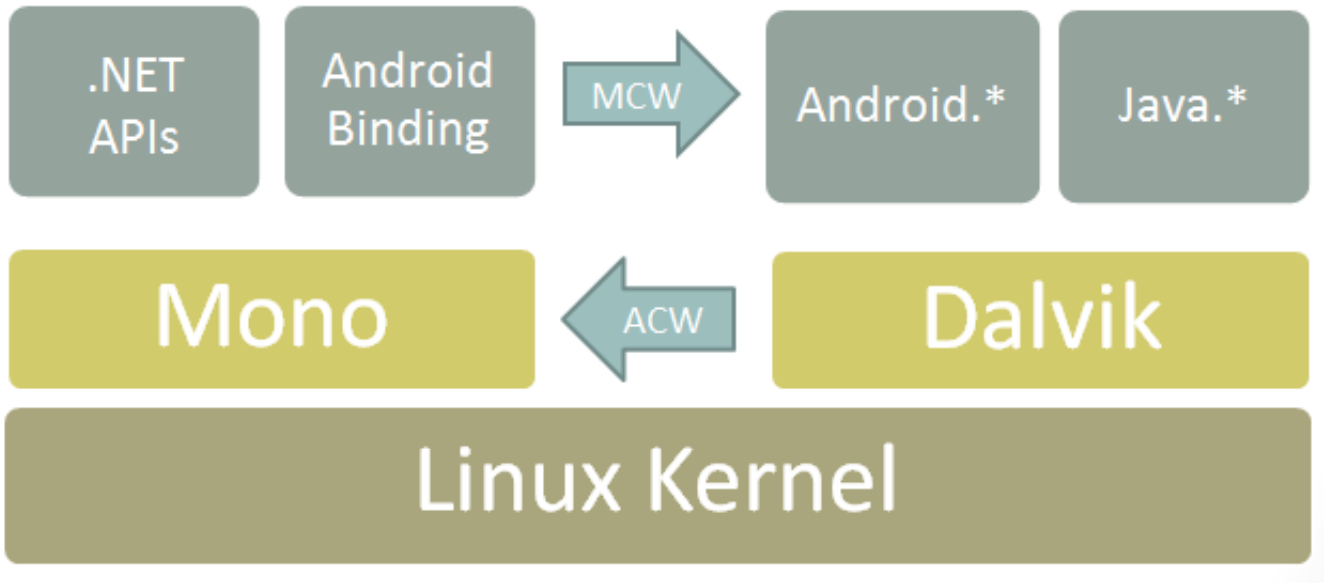
现在做Android开发的同学只要熟悉这些应用框架层的一些接口和类库就可以给方便的来实现自己的Android应用程序。

关于Dalvk虚拟机与Java运行环境的区别

1. Dalvik主要是完成对象生命周期管理，堆栈管理，线程管理，安全和异常管理，以及垃圾回收等等重要功能。
2. Dalvik负责进程隔离和线程管理，每一个Android应用在底层都会对应一个独立的Dalvik虚拟机实例，其代码在虚拟机的解释下得以执行。
3. 不同于Java虚拟机运行java字节码，Dalvik虚拟机运行的是其专有的文件格式
4. Dex文件格式可以减少整体文件尺寸，提高I/o操作的类查找速度。
5. 是为了在运行过程中进一步提高性能，对dex文件的进一步优化。
6. 所有的Android应用的线程都对应一个Linux线程，虚拟机因而可以更多的依赖操作系统的线程调度和管理机制
7. 有一个特殊的虚拟机进程Zygote，他是虚拟机实例的孵化器。它在系统启动的时候就会产生，它会完成虚拟机的初始化，库的加载，预制类库和初始化的操作。如果系统需要一个新的虚拟机实例，它会迅速复制自身，以最快的数据提供给系统。对于一些只读的系统库，所有虚拟机实例都和Zygote共享一块内存区域。

大家注意第2点和第7点有助于我们理解Xamarin.Android的工作机制。

Xamarin.Android 架构



Java编写的Android应用程序通过调用 Android.* 和 Java.* 这些命名空间下的类来实现一些系统的功能包括：声音、显示、OpenGl等一些通过Java API不能实现的功能或者说是与硬件、系统平台相关的功能。那这里的问题是当我们用C#来编写的时候，这些功能怎么去调用？C#写的Android 应用程序又是如何初始化的？

Android Callable Wrappers (ACW)

当一个C#开发的Android程序运行的时候，除了一个Dalvik的虚拟机实例，还有一个Mono的虚拟机实例在运行。那个Dalvik虚拟机实体就像一个宿主，我们的APP在宿主上运行，而我们所有用C#写的方法都会以ACW的形式被调用。在Java代码中以native的式式invoke，就像invoke其它C或者C++的代码一样。

Momodroid.exe 在编译阶段会为我们的C#类生成对应的ACW。

1
2
3
4
5
6
7
8
9
10
11
12
13
14

```
15
    using System;
    using Android.App;
    using Android.OS;
    namespace Mono.Samples.HelloWorld
    {
                public class HelloAndroid : Activity
                {
                            protected override void OnCreate (Bundle
savedInstanceState)
                            {
                                        base.OnCreate (savedInstanceState);
                                        SetContentView (R.layout.main);
                            }
                }
}
```