App远程更新说明

**一、为何要实现远程更新**

当我们要部署新版本App的时候，总是面临着要重新提供给用户一个最新的apk文件使他们更新到最新版本的App。但是当用户基数较大的时候，这种直接提供用户apk文件让他们手动更新的形式就变得不现实了，此时就应该考虑到有没有一种能够检测当前版本号并与服务器上的版本号进行比对，如果版本号低于服务器上的版本号则进行强制更新的方法。

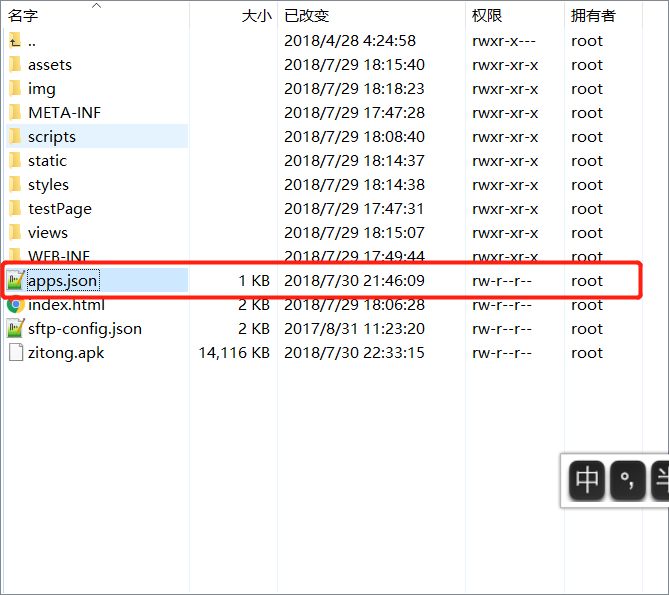
**二、如何实现远程更新**

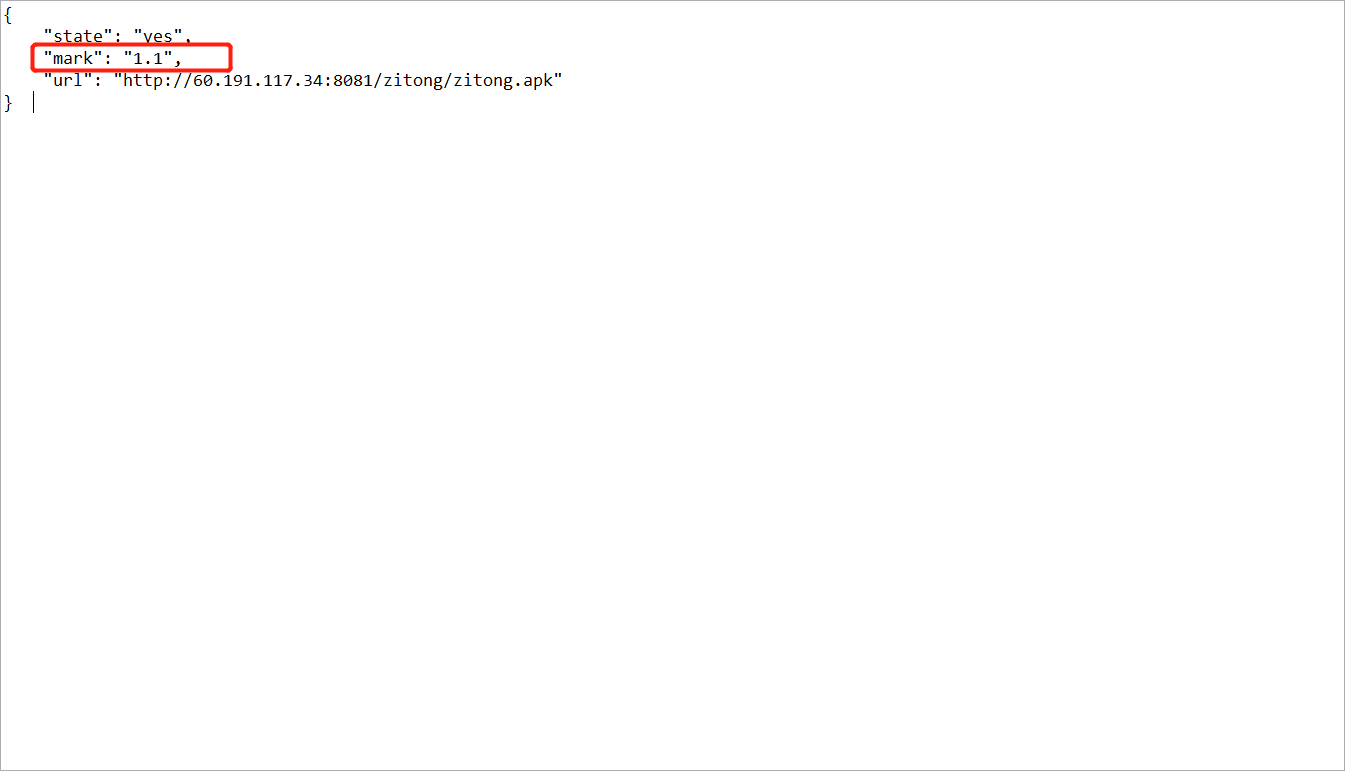
目前我们的App源码中已经实现了远程自动更新的方法，我在这里简单阐述一下原理。

在/js/update.js中，是实现自动更新的具体源码，大家可以去查阅一下。当然，如果不理解也可以，因为我们只要会用就行了，轮子别人已经帮我们造好了，我们要用上这个轮子。

首先，假设用户版本是1.1，在index.html文件中的第87行，有一句svn(1.1);

意思是每一次用户打开App都会跟服务器上Tomcat同个项目目录下的apps.json文件里的mark值进行比对。如果都是一致的，则跳过强制更新，否则就更新。





这个时候，我们要更新1.2版本了，那么我们需要修改三个地方。

1. 修改index.html的第87行，将svn(1.1)修改为svn(1.2)
2. 修改manifest.json中版本号为1.2

这个时候再将你修改过的app进行打包并发布到服务器上同样的Tomcat项目目录下，注意文件名必须是zitong.apk，最后

3. 修改服务器上app.json文件中的mark值为1.2

用户端的版本是将1.1与服务器中的app.json文件的mark值进行对比，因为此时我们将mark值修改为1.2了，所以用户只要一打开App就会提醒更新。当用户更新成功之后，就相当于将你发布在服务器上最新的apk文件安装了一遍，而你最新的apk文件里比较的版本号已经修改是1.2了，并且.json文件也是1.2，所以检测到是最新版本就不进行更新了，而用户也用到了最新版本的文件。

**三、总结**

技术从来都是为了服务而产生了，有了需求才有实现的方案。先技术，后思想的意思是你一开始不需要懂为什么能实现，你要知道你为什么要实现，最后实现了再去问这是怎么实现的。

最后感谢一下这个App自动更新是刘盟盟同学引入并实现了update.js方法。

Ps：现在的App版本和服务器.json版本，我都初始化成1.1版本了，下一次更新就是从1.2开始命名。