软件更新是软件版本更新迭代要处理的问题，如何把最新版本软件及时推送到用户端是一件很重要的事。一般开发者是通过自行在平台添加更新版本的apk，这样做，如果是在一两个应用平台发布应用，那还说得过去，工作量还不是很大。但大家都知道，软件发布往往不止在一个平台，这样的工作量往往比较大，版本推送不够及时。面对这个问题，软件自动更新是一项很实用的技术。

**软件自动更新实现实现思路**

1客户端（主程序）调用升级程序，升级程序连接到最新的服务器上。

2、升级程序获取服务器上的xml配置文件中最新程序的更新日期或版本号或文件大小等。

3、升级程序获取客户端（主程序）的xml配置文件的更新日期或版本号大小等，然后两者进行比较；

* 如果如果新版本日期>原有程序的最新日期，则提示用户是否升级；
* 或如果新版本版本号>原有程序的版本号，则提示用户是否升级；
* 再或如果新版本文件大小>原有程序的文件大小，则提示用户是否升级。

本文主要采用一般的做法，就是通过版本号来进行对比。

4、如果用户选择升级，则下载文件列表。

5.在本地建立与远程IIS或者FTP相应的临时目录，然后下载到这个临时目录文件下；

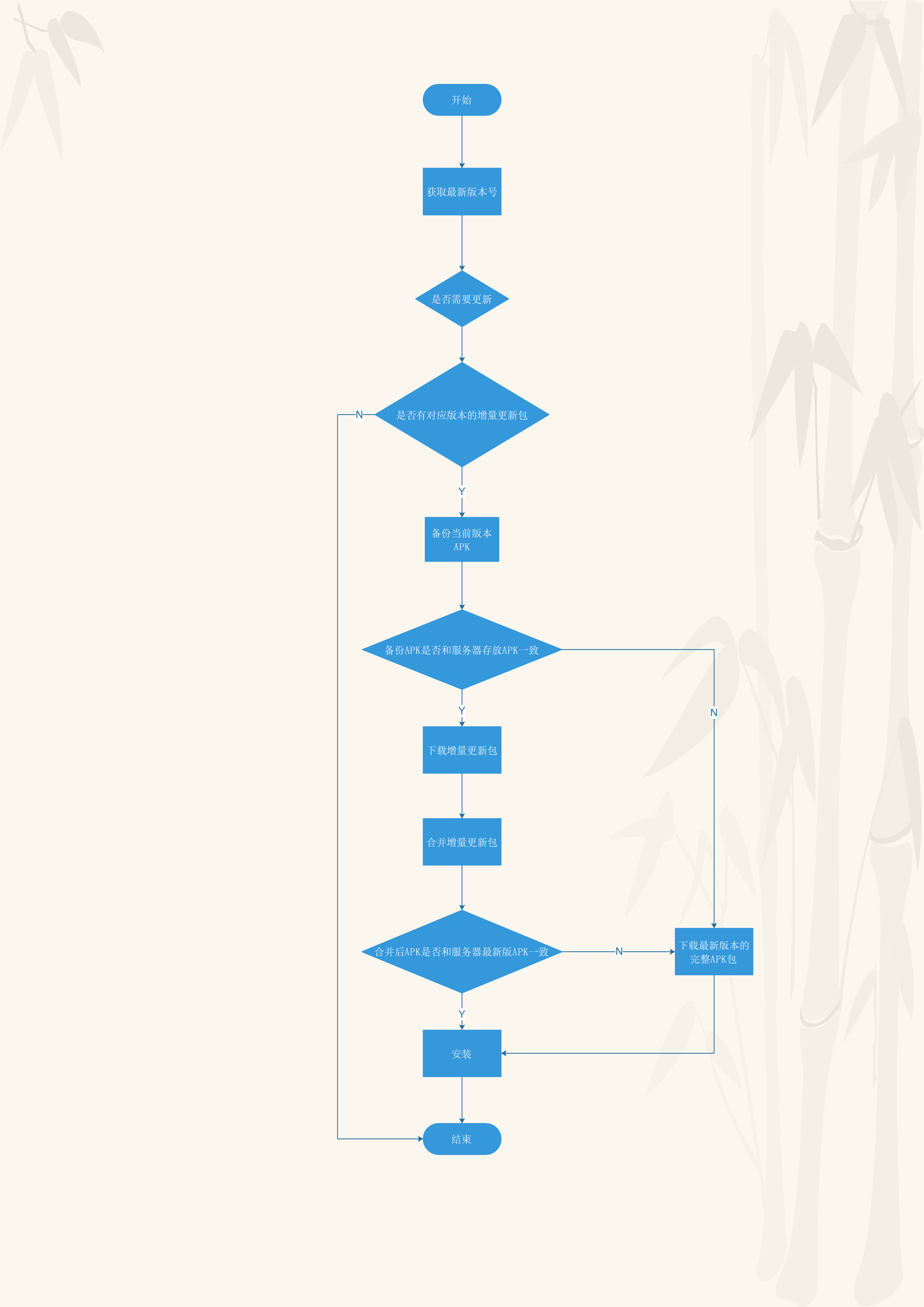
6.删除旧的主程序，拷贝临时文件夹中的文件到相应的位置；

7.结束升级流程并重新启动主程序。

**软件更新有两种实现思路，整体更新和部分更新。**

整体更新即是指重新下载整个软件替换掉原有的软件，实现起来比较简单，不容易出错，但是对于比较大的安装包是比较麻烦的。

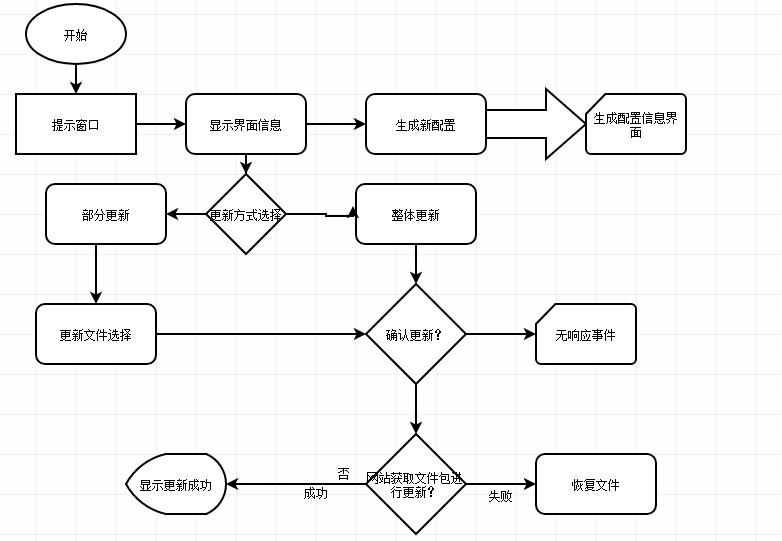
而增量更新是指只替换更新的部分，服务端通过新版本APK和旧版本APK生成patch补丁（也称为差分包），目前使用比较多的实现方案多是BSDiff/Patch(可下载BsDiff工具)，客户端更新的时候只需要下载差分包到本地，然后从system/app取出旧版本APK，通过差分包来合成新版本的APK，这个过程实际上就是打补丁。过程比较麻烦，但是可以节省一点更新时间和网络带宽，对于大型程序比较适用，比如一些游戏APP甚至有几十G到几百G的内存空间，对于这种程序如果用整体更新就不太合适。



增量更新实现的方式多种多样，现代客户端的升级程序已经相当复杂了，包括了支持灰度放量的新版本检测，http断点续传下载，MD5完整性校验，bsdiff/courgette 二进制补丁更新，双目录迭代升级等技术。这里对实现方法不做过多深究。

**更新模拟**

具体实现因各平台程序差别而略有差别，但是主流的思路都如上。因为笔者对WINFORM程序开发不是很懂，所以简单的以JAVA写了一个模拟软件更新的程序，仅仅为了表示软件更新的思路。



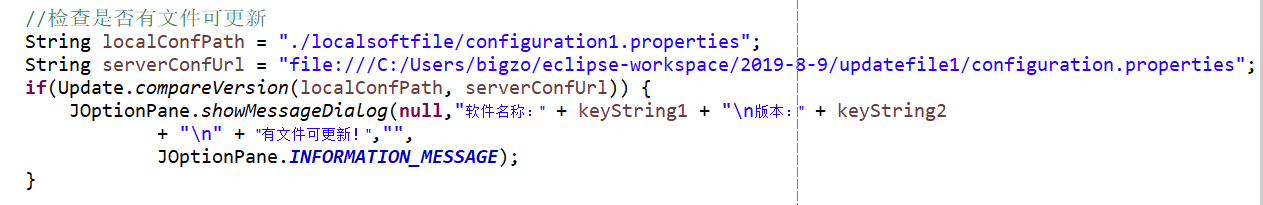
流程图

检查版本号



这里用properties文件来模拟xml文件，检查版本号是通过将服务器的版本号下载到本地进行对比，如果本地版本号小于服务端版本号，则返回true。

调用cmopareVersion函数对比版本号，如果可更新就提示用户软件可更新

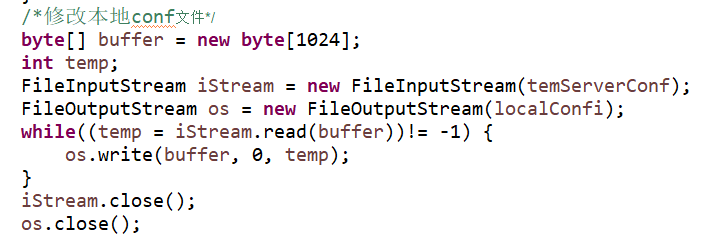


**更新方式**

这里提供了两种更新方式，整体更新和部分更新，整体更新比较简单，因为知识简单模拟，没有实际的APP软件包可供操作，部分更新使用的方法是通过配置清单进行选择（早期实现增量更新的一种简单方式），通过一个配置文件，来说明最新的客户端中都包含了那些文件，这些文件的MD5值是什么，然后网络路径是什么。这样客户端拿到这个配置清单的时候，就可以轻松的判断本地的那些文件是可以继续用的。哪些文件是过时了的，这样客户端通过配置心中的网络路径位置获取最新的对应文件就好了，这个配置文件的生成是需要程序自动生成的（现在多使用bisdiff工具生成差分包）。

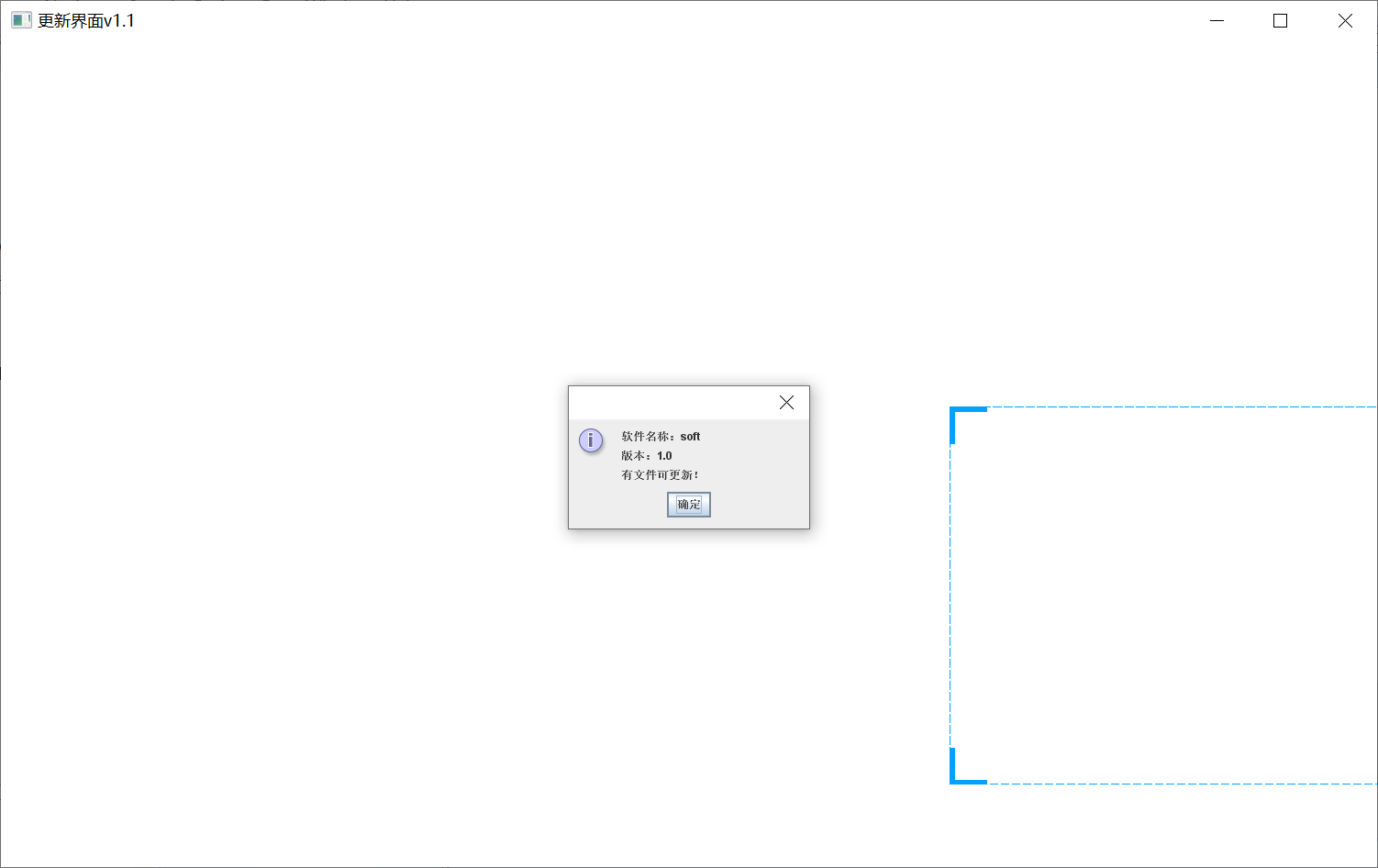


更新完要修改本地配置文件

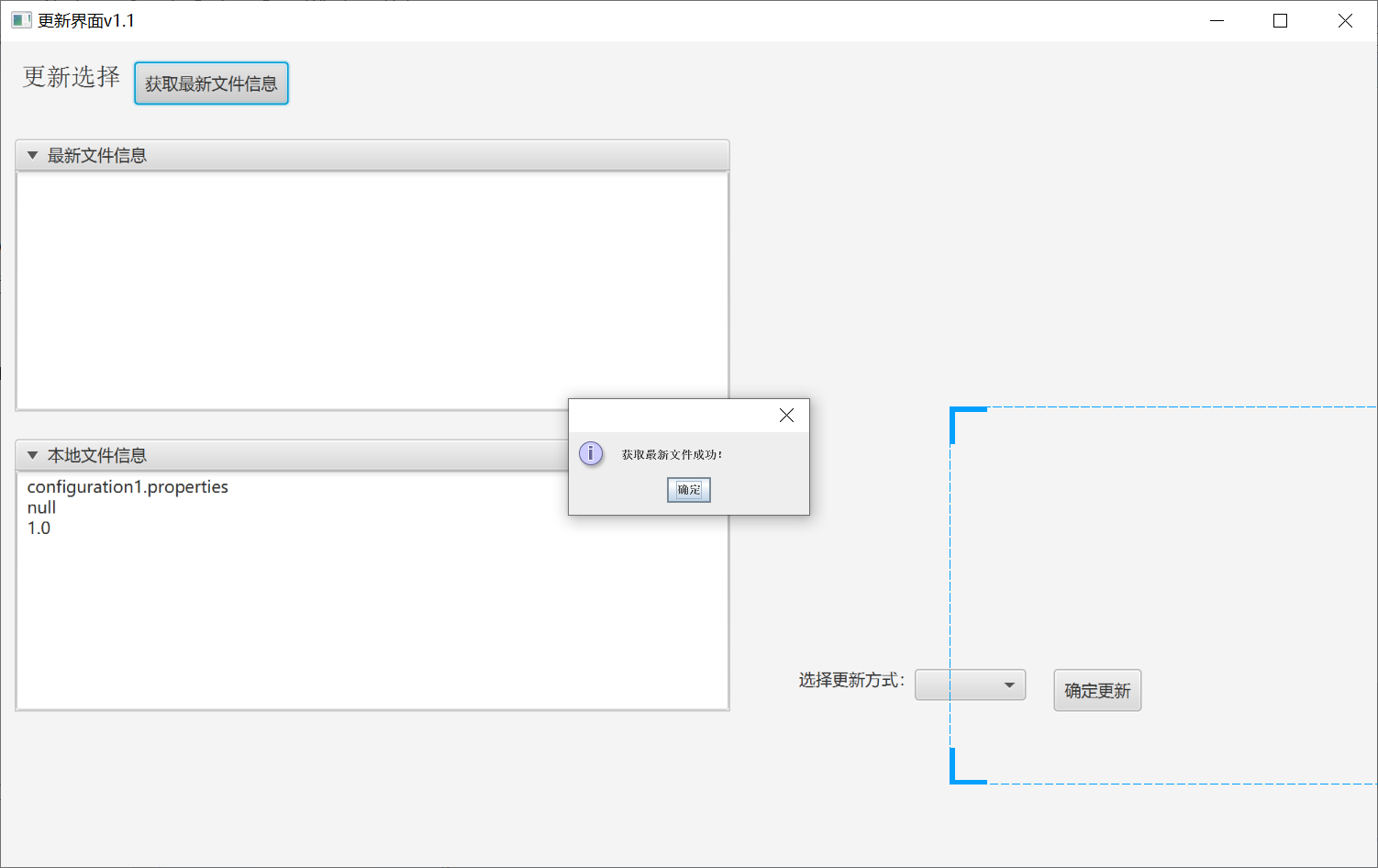


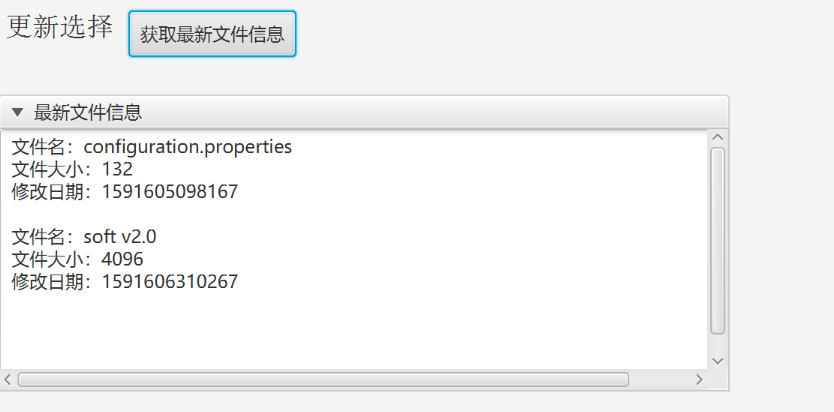
**模拟界面**

1 进入提示需要更新



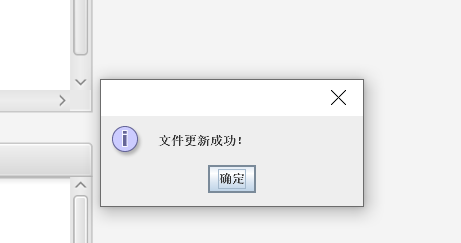
2 点击获取最新文件按钮，模拟软件检查更新按钮





3 更新





再次获取更新信息，提示已经是最新版本了，本地文件夹也已经更新完毕。

