手机卫士第一天

代码的组织结构
推荐使用根据功能模块划分
- Activity com.itheima.mobilesafe.activty
- 后台服务 com.itheima.mobilesafe.service
- 广播接受者 com.itheima.mobilesafe.receiver
- 数据库 com.itheima.mobilesafe.db.dao
- 对象(java bean) com.itheima.mobilesafe.domain/bean
- 自定义控件 com.itheima.mobilesafe.view
- 工具类 com.itheima.mobilesafe.utils
- 业务逻辑 com.itheima.mobilesafe.engine

在创建项目时的选项解释
- minimum SDK 要求最低的安装版本, 安装apk前,系统会判断当前版本是否高于(包含)此版本, 是的话才允许安装

- maxSdkVersion 要求最高的安装版本(一般不用)

- Target SDK 目标SDK, 一般设置为开发时使用的手机版本, 这样的话,系统在运行我的apk时,就认为我已经在该做了充分的测试, 系统就不会做过多的兼容性判断, 从
而提高运行效率

- Compile With 编译程序时使用的版本

代码获取软件的versioncode和versionname

```
PackageManager packageManager = getPackageManager();//获取包管理器

try {

    PackageInfo packageInfo = packageManager.getPackageInfo(getPackageName(),
PackageManager.GET_CONFIGURATIONS);//获取包的信息

    int versionCode = packageInfo.versionCode; //获取版本号

    String versionName = packageInfo.versionName; //获取版本的名称号

    splash_tv_version.setText("版本:" + versionName);//将获取到的版本名称号显示到splash界面上

} catch (NameNotFoundException e) {

    e.printStackTrace();

}
```

设置文本的阴影效果

```
<TextView

    android:id="@+id/splash_tv_version"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:layout_centerHorizontal="true"

    android:layout_centerVertical="true"

    android:textSize="20sp"

    android:shadowColor="#ff0000"

    android:shadowDx="1"

    android:shadowDy="1"
```

```
android:shadowRadius="10"

android:text="@string/test_text"

android:textColor="#ffffff" />
```

在模拟器或者真机的调试中, 如果要访问上位机的资源, 那么上位机的ip地址为: 10.0.0.2

从网络上获取数据的一般步骤

```
URL url; //创建url对象
try {
   url = new URL("http://10.0.2.2:8080/update.json");
   HttpURLConnection connection = (HttpURLConnection)url.openConnection(); //获取连接对象
   connection.setConnectTimeout(5000); //设置连接超时
   connection.setReadTimeout(5000); //设置请求超时
   connection.setRequestMethod("GET"); //设置请求方法
   connection.connect(); //开始连接
   if (connection.getResponseCode() == HttpURLConnection.HTTP_OK)
   {
      InputStream in = connection.getInputStream();
      //处理获取到的数据
    }
} catch (MalformedURLException e) {//url异常, 即网络异常
   e.printStackTrace();
} catch (IOException e) {//io异常
   e.printStackTrace();
}
```

涉及网络的耗时操作必须放到子线程中去执行, 否则会导致主线程阻塞, 如果主线程阻塞时间超过5秒, 所在的activity就会崩溃

组件的visibility属性详解

组件的visibility属性有三个值:visible, invisible, gone

这三个值分别代表的意思是:

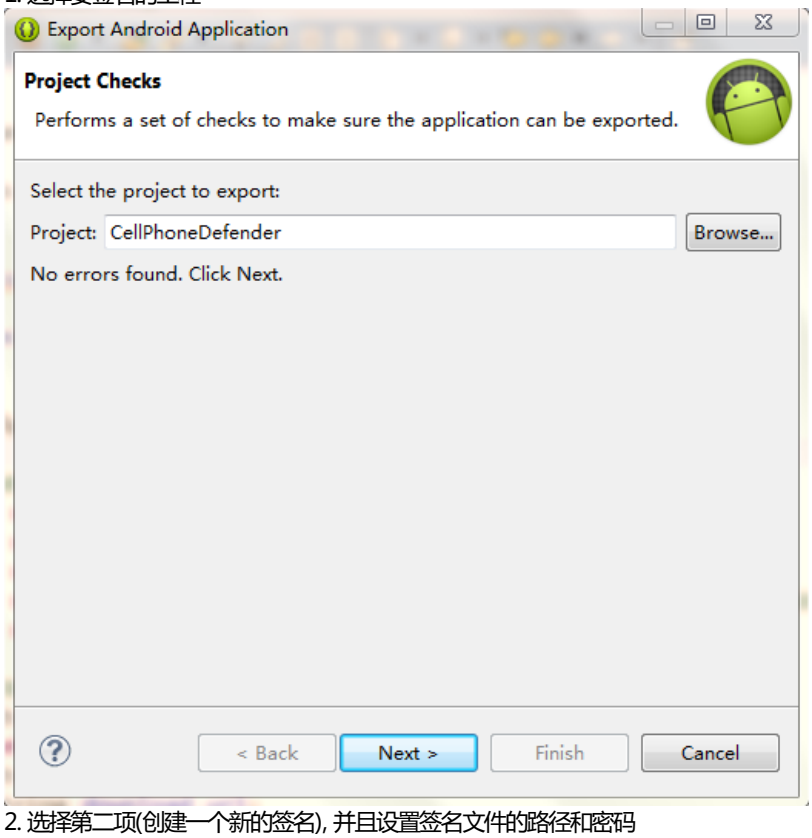
visibile: 组件可见

invisible: 组件不可见, 但是还占着地方

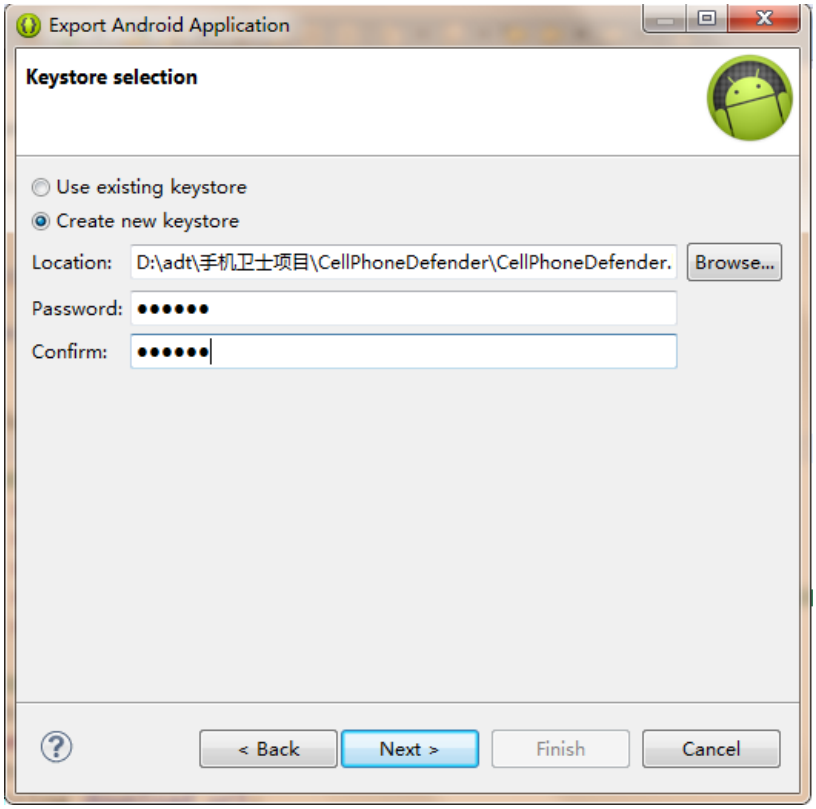
gone: 组件滚蛋了, 即连地方都没有占, 直接滚蛋了

关于软件正式签名的步骤:

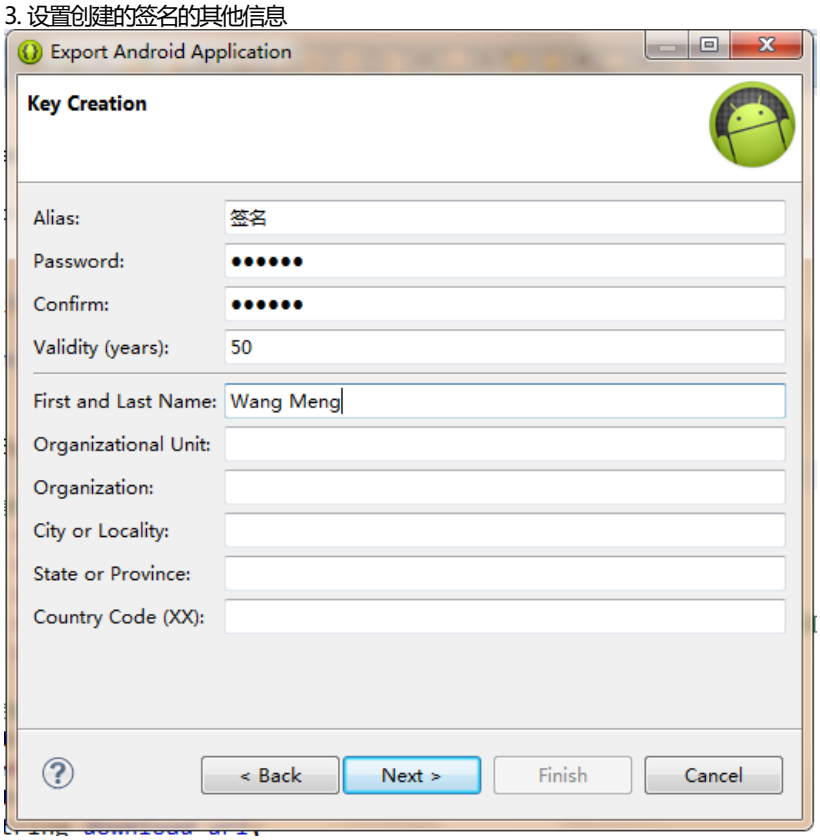
1. 选择要签名的工程



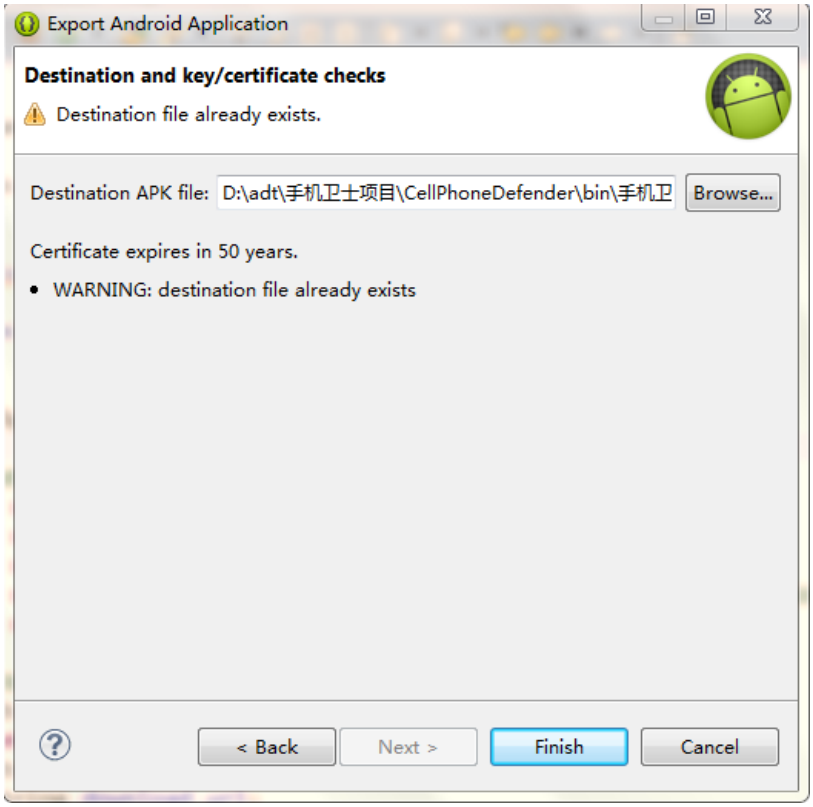
2. 选择第二项(创建一个新的签名), 并且设置签名文件的路径和密码



3. 设置创建的签名的其他信息



4. 选择要签名的文件(.apk文件)



软件签名成功

注:

1. 在正式发布应用时, 必须用正式签名来打包, 这样可以防止别人恶意的覆盖你的应用(比如你的竞争对手通过起相同的包名和应用名来覆盖你的应用, 如果有签名的话他们就覆盖不了你的应用了.)
2. 如果在eclipse直接运行项目时使用的是测试签名, 测试签名也有密码, 使用年限一般是1年.
3. 请保护好你的正式签名, 正式签名很重要.

设置某activity为没有标题栏的样式可以在其style定义行中添加:<item name="android:windowNoTitle">true</item>

例:

```
<style name="AppTheme" parent="AppBaseTheme">
    <!-- 没有标题栏的设置 -->
    <item name="android:windowNoTitle">true</item>
</style>
```

常用的快捷键:

常用快捷键

- ctrl + shift + o 导包

- ctrl + shift + t 快速查找某个类

- 先按ctrl + 2 ,再点L, 创建变量并命名

- ctrl + o , 在当前类中,快速查找某个方法

- ctrl + k, 向下查找某个字符串

- ctrl + shift + k, 向上查找某个字符串

- alt + 左方向键 跳转上一个页面

- ctrl + shift + f 代码格式化

关于this, getApplicationContext()和getContext()的详解

在android中常常会遇到与context有关的内容
浅论一下context ： 在语句 AlertDialog.Builder builder = new AlertDialog.Builder(this); 中，要求传递的 参数就是一个context，在这里我们传入的是this，那么这个this究竟指的是什么东东呢？ 这里的this指的是Activity.this，是这个语句所在的Activity的this，是这个Activity 的上下文。网上有很多朋友在这里传入this.getApplicationContext（ ），这是不对的。AlertDialog对象是依赖于一个View的，而View是和一个Activity对应的。 于是，这里涉及到一个生命周期的问题，this.getApplicationContext（ ）取的是这个应 用程序的Context，Activity.this取的是这个Activity的Context，这两者的生命周期是不同 的，前者的生命周期是整个应用，后者的生命周期只是它所在的Activity。而AlertDialog应 该是属于一个Activity的，在Activity销毁的时候它也就销毁了，不会再存在；但是，如果传 入this.getApplicationContext（ ），就表示它的生命周期是整

个应用程序，这显然超过了它 的生命周期了。 所以，在这里我们只能使用Activity的this。
new AlertDialog.Builder(getApplicationContext())时发生错误：
E/AndroidRuntime(5844): android.view.WindowManager$BadTokenException: Unable to add window -- token null is not for an application
于是查了查：
getApplicationContext() 生命周期是整个应用，应用摧毁它才摧毁 Activity.this的context属于activity ，activity 摧毁他就摧毁
activity.this要返回一个activity，而getApplicationContext()就不一定返回一个activity
getApplicationContext() 返回应用的上下文，生命周期是整个应用，应用摧毁它才摧毁
Activity.this的context 返回当前activity的上下文，属于activity ，activity 摧毁他就摧毁

getBaseContext() 返回由构造函数指定或setBaseContext()设置的上下文

设置TextView中文字内容的跑马灯效果

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:text="萌哥手机卫士,你值得拥有,你用不了吃亏,你用不了上当,你值得拥有!"
    android:textSize="20sp"
    android:singleLine="true"
    android:ellipsize="marquee" //设置展示方式为跑马灯的效果
    android:focusable="true" //设置该组件获得焦点, 只有获得焦点, 跑马灯效果才能实现
    android:focusableInTouchMode="true"
    />
```