

大话企业级 Android 开发 · 第十二部分

本教程说明及版权声明

- 《大话企业级 Android 开发》是国士工作室为了方便中国 Android 开发者，推动 Android 企业级应用开发，特投入大量心血撰写的书籍，并在网络上免费发布，希望为移动互联网和智能手机时代贡献绵薄之力！所有相关文档版权均属国士工作室所有。
- 本教程是由国士工作室参考官方文档，综合市面相关书籍，经过充分的吸收消化，结合开发实践的一部原创作品，为了本教程及早与广大读者同仁见面、分享，特采用定稿一部分就发布一部分的连载方式发布。读者可以在本博客获取最新内容。
- 未经国士工作室授权，禁止将此文档及其衍生作品以标准（纸质）书籍形式发行。
- 本文档受有关法律的版权保护，对本文档内容的任何未经同意的复制和抄袭行为，将导致相应的法律责任。未经国士工作室同意，任何团体及个人不能用此教程牟利，违者必究。但是：在不收取其他人费用的前提下，您可以自由传播此文档，但必须保证版权信息、文档及其自带标示的完整性。
- 如果对该文档有任何疑问或者建议，请进入官方博客 <http://www.cnblogs.com/guoshiandroid/> 留言或者直接与国士工作室联系（后附联系方式），我们会慎重参考您的建议并根据需要对本文档进行修改，以造福更多开发者！
- 《大话企业级 Android 开发》的最新及完整内容会在国士工作室官方博客定期更新，请访问国士工作室博客 <http://www.cnblogs.com/guoshiandroid/> 获取更多更新内容。

关于国土工作室

我们(国土工作室)是一支专注于 Android 平台企业级应用开发的技术团队，对娱乐多媒体应用有着深刻的理解及研发能力，致力服务于企业用户。为音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络应用向移动互联网发展提供解决方案和技术支持，为企业提供 Android 培训服务等多种业务。

我们尤其擅长于提供从 Android 客户端到服务端的一站式解决方案和技术支持，服务端可以采用 Java EE，也可以采用轻量级流行的 LAMP 技术体系。目前，研发出了比 KU6、优酷更加强大和完善的 Android 视频网站娱乐多媒体客户端软件，并在持续升级中。

目前，我们正在务实而卓有成效的与音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络服务商合作，发展迅速，渴望有志之士的加入，和我们一起为成为世界最好的 Android 软件开发和咨询、培训公司而奋斗，为移动互联网和智能手机时代贡献力量！

联系我们

电话:15711060468

Email:guoshiandroid@gmail.com

博客: <http://www.cnblogs.com/guoshiandroid/>

1 文件：普通文件 I/O

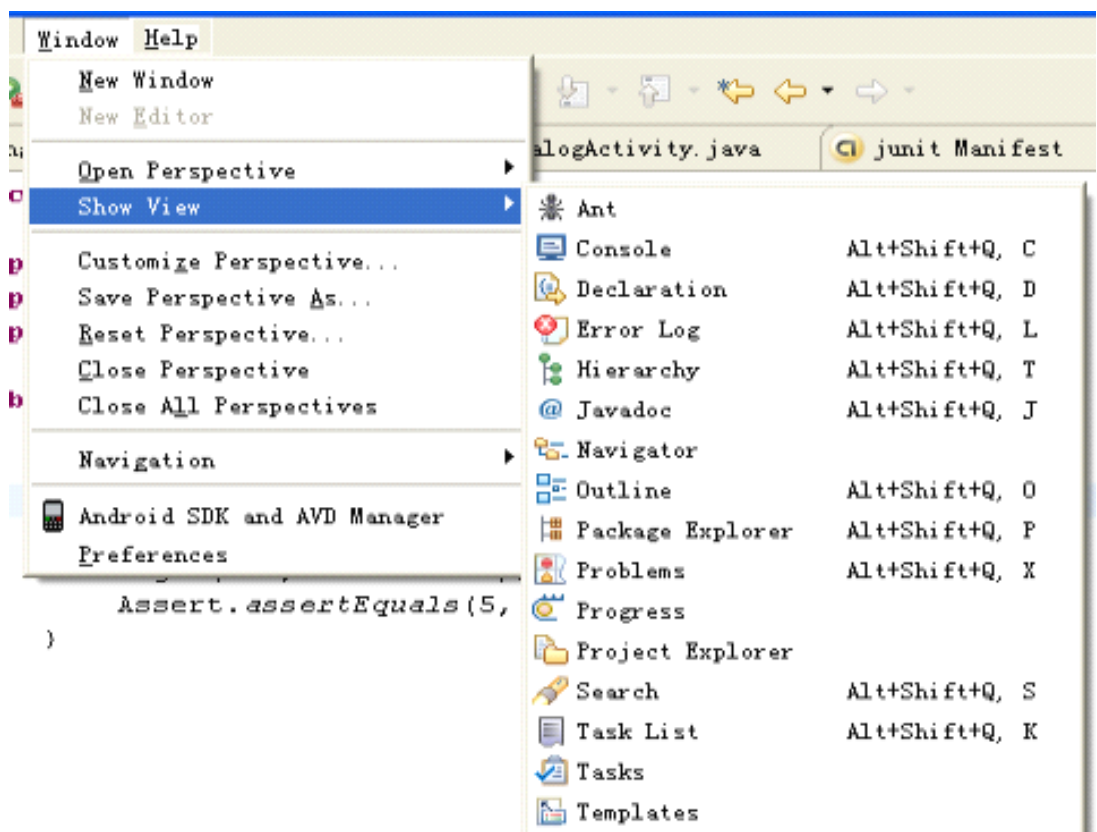
小安：我交了一个女朋友，但是却把女朋友的生日忘记了，在生日那天没有第一个给女朋友祝福，女朋友很是生气，我后悔极了。我在想：如果在 Android 上开发一款记事本软件一定会很有用，但是对于文件的操作 Android 是如何实现的？有什么特别的要求呢？

大致：你的想法很好，在 Android 中文件存储是很重要的一个知识点，不单单可以用来存储数据，还可以用来保存用户想要保存的信息。如果你对 java 的 I/O 很熟悉的话，学习 Android 的文件操作就很简单了。

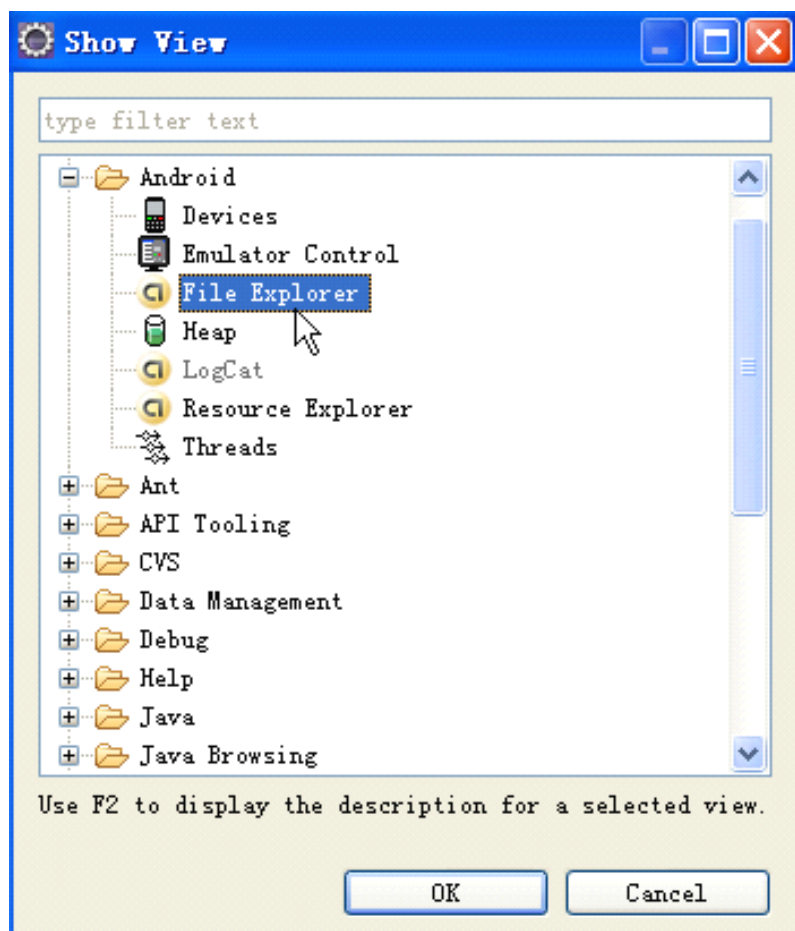
1.1 文件存储数据

小安：博士，Java 的 I/O 我熟悉，但是 Android 上创建文件难道和它一样？

大致：当然有所不同了，在介绍文件存取数据之前，我们先观察一下 Android 平台的文件系统目录。观察文件系统目录，需要再打开一个新的视图——File Explorer。打开方式如下图：

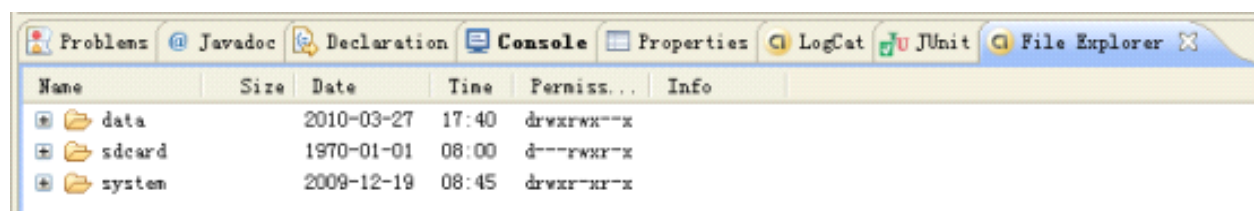


然后单击【other】，选择【File Explorer】如下图：



单击【OK】。

查看 File Explorer 视图。如下图，列出三个文件夹。



由于 Android 系统是基于 Linux 系统的，而 Linux 系统的根目录是“/”，因此访问 Android 根目录下的 data 文件夹的路径为“/data”。

/data 专门用来存放 Android 中各个应用的数据，以后我们保存文件，数据库文件都是存在该目录下。

/sdcard 访问该目录下的文件相当于访问 sdcard 中的文件。

/system 用来存放 Android 的操作系统文件及自带的软件。

创建项目

Layout 对应 View 层。Modle 层则需要我们去编写。本项目中我们的业务方法就是数据存放。
新建 Android 项目 FileRW 如下图所示：

New Android Project
Creates a new Android Project resource.

Project name:

Contents

- ☒ Create new project in workspace
- ☐ Create project from existing source
- ☒ Use default location

Location:

- ☐ Create project from existing sample

Samples:

Build Target

Target Name	Vendor	Platform	AP...
<input type="checkbox"/> Android 2.0	Android Open Source Project	2.0	5
<input checked="" type="checkbox"/> Android 2.1	Android Open Source Project	2.1	7

Standard Android platform 2.1

Properties

Application name:

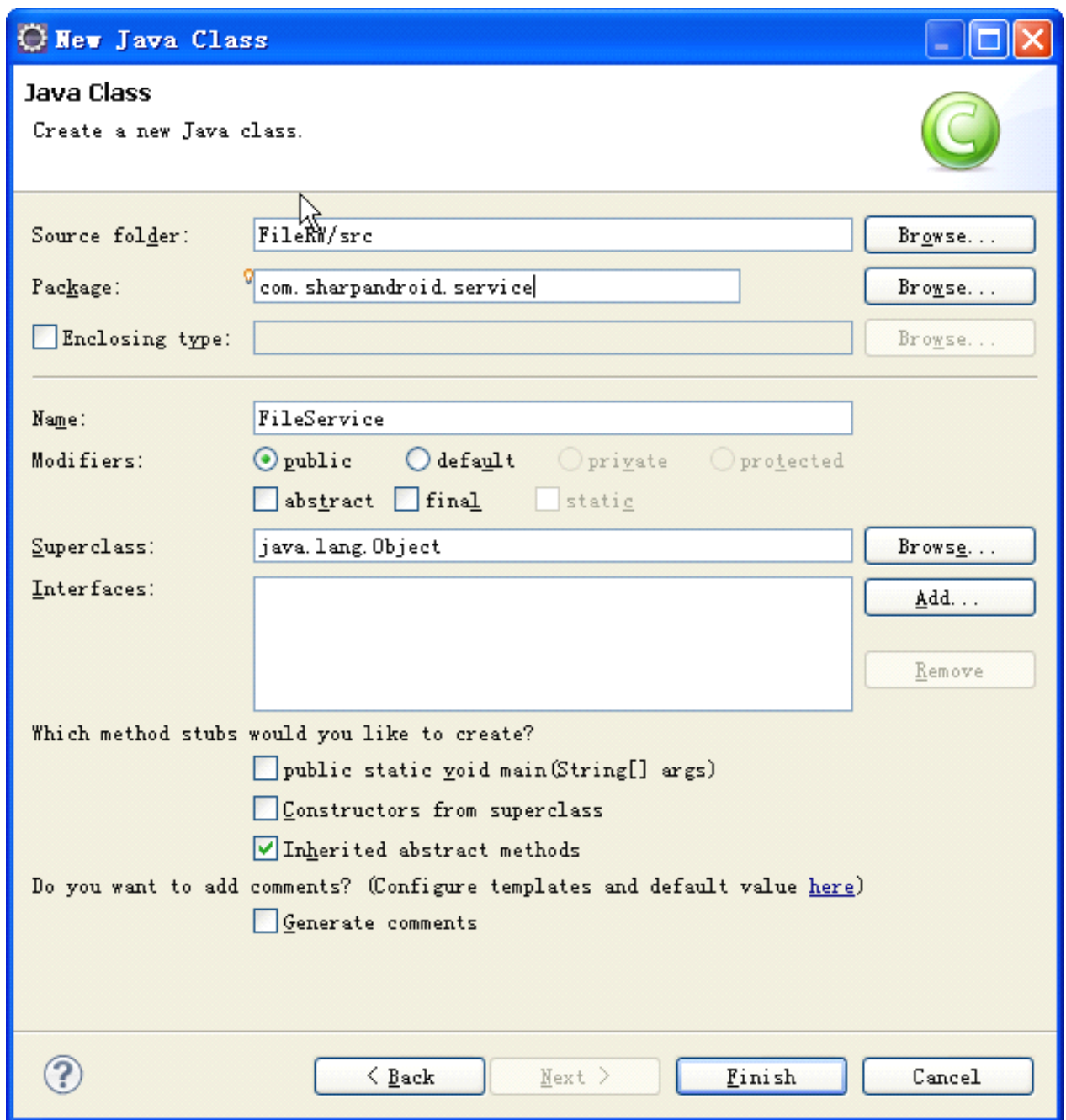
Package name:

☒ Create Activity:

Min SDK Version:

实现文件保存

先需要创建一个业务类 FileService。如下图：



点击【Finish】。

在 J2EE 开发中，业务层一般会面向接口编程，其好处在于解耦。而我们在开发 Android 应用时只要能做到清晰、可读性强就可以了。面向接口编程会对性能有所影响。Android 的文档建议使用内部类，可以少创建一些类，这样做都是为了提升性能。

完成 save 方法之后，FileService.java 代码如下。

```
import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
public class FileService {
    private Context context;
    public FileService(Context context) {
        this.context = context;
    }
}
```



```

    public void save(String content) throws Exception{
        FileOutputStream outputStream =
context.openFileOutput("sharpandroid.txt", Context.MODE_PRIVATE);
        outputStream.write(content.getBytes()); //写数据
        outputStream.close(); //关闭输出流
    }
}

```

说明：

- private Context context;
public FileService(Context context) {
 this.context = context;
 }

将当前应用的上下文Context作为其属性，当创建该类时传入。这样在该类的其他方法中就可以利用该Context获取对应用的信息。修改构造方法的目的是确保该属性在创建时会被传入。

- FileOutputStream outputStream =
context.openFileOutput("sharpandroid.txt",
Context.MODE_PRIVATE);

通过上下文打开一个文件输出流。

第一个参数：文件的名称，不能包含路径分隔符“/”，如果文件不存在，Android会自动创建它。创建的文件保存在/data/data/<package name>/files目录，如：
/data/data/com.sharpandroid.file/files/sharp.txt，在File Explorer视图中展开/data/data/<package name>/files目录就可以看到该文件。

第二个参数：用于指定操作模式，有四种模式，分别为：

Context.MODE_PRIVATE 表示文件只能被创建该文件的应用所访问。该模式也是默认的模式。如果创建时文件不存在，会创建文件，如果存在会覆盖该文件。

Context.MODE_APPEND 表示模式会自动检查文件是否存在，存在就往文件追加内容，否则就创建新文件。

Context.MODE_WORLD_READABLE 表示当前文件可以被其他应用读取。

Context.MODE_WORLD_WRITEABLE 表示当前文件可以被其他应用写入。

如果希望文件被其他应用读和写，可以这样操作：

```

openFileOutput("sharpandroid.txt", Context.MODE_WORLD_READABLE +
Context.MODE_WORLD_WRITEABLE);

```

配置测试环境

编写 AndroidManifest.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"

```

```

        android:versionCode="1"
        android:versionName="1.0">
        <application android:icon="@drawable/icon"
        android:label="@string/app_name">
            <uses-library android:name="android.test.runner" />
            <activity android:name=".FileActivity"
                android:label="@string/app_name">
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />
                    <category
        android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
            </activity>
        </application>
        <uses-sdk android:minSdkVersion="7" />
        <instrumentation
        android:name="android.test.InstrumentationTestRunner"
            android:targetPackage="com.sharpandroid.file"
        android:label="Tests for My App" />
    </manifest>

```

除了<uses-library android:name="android.test.runner" />存放的位置之外，记得要将targetPackage属性修改的跟应用包一致。

创建测试

创建 FileServiceTest.java 测试类。

New Java Class

Java Class
Create a new Java class.

Source folder: FileRW/src Browse...

Package: com.sharppandroid.file Browse...

☐ Enclosing type: com.sharppandroid.service.FileService Browse...

Name: FileServiceTest

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: android.test.AndroidTestCase Browse...

Interfaces: Add...
Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

? < Back Next > Finish Cancel

添加 testSave 测试方法，之后 FileServiceTest 测试类代码如下：

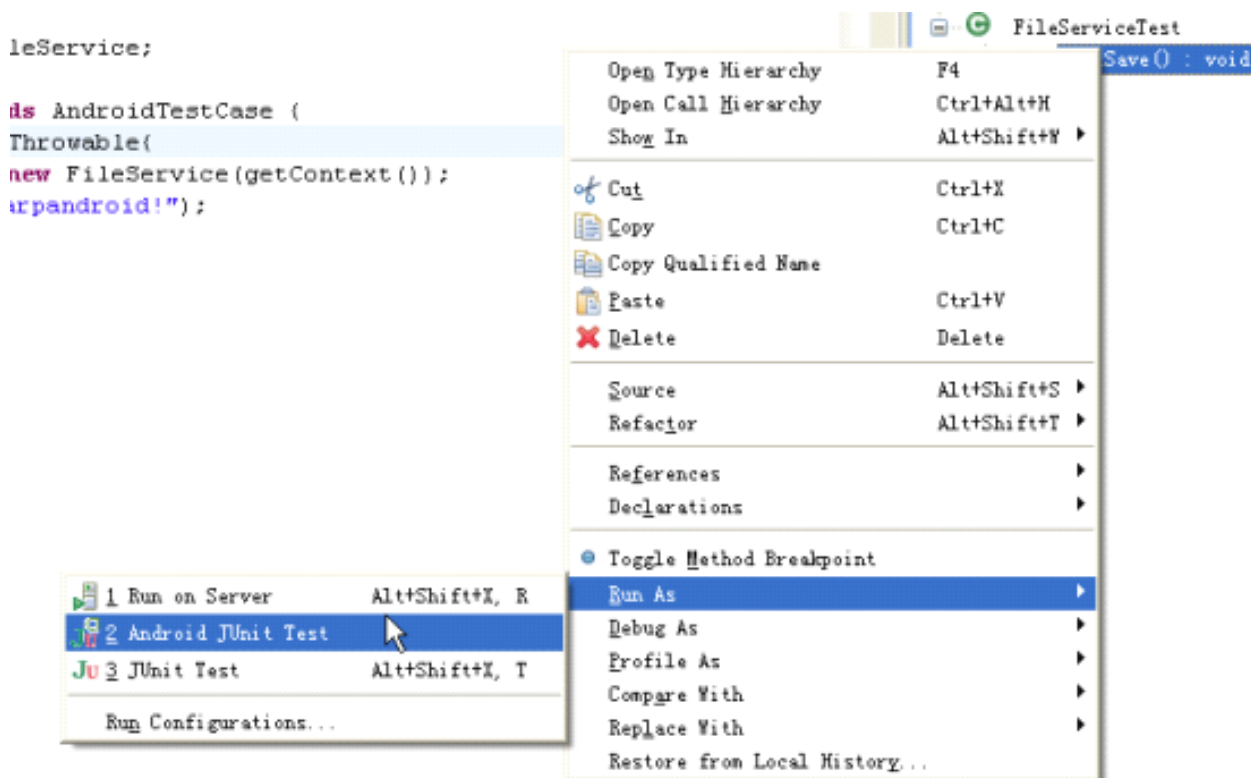
```
package
import java.io.InputStream;
import java.io.OutputStream;
import android.content.Context;
import android.test.AndroidTestCase;
import android.util.Log;
public class FileServiceTest extends AndroidTestCase {
    public void testSave() throws Throwable{
        FileService fileService = new FileService(getContext());
        fileService.save("我们是sharppandroid!");
    }
}
```

说明:

- `FileService fileService = new FileService(getContext());`
由于该测试用例继承自`AndroidTestCase`,因此该类可以直接采用`getContext`方法得到当前应用的`Context`。

执行测试:

测试方法如下图:



由于该测试框架还有些 bug,有时点击运行之后并不能正常运行,我们可以再次运行,直到出现运行结果为止。执行之后 JUnit 控制台为绿条通过。



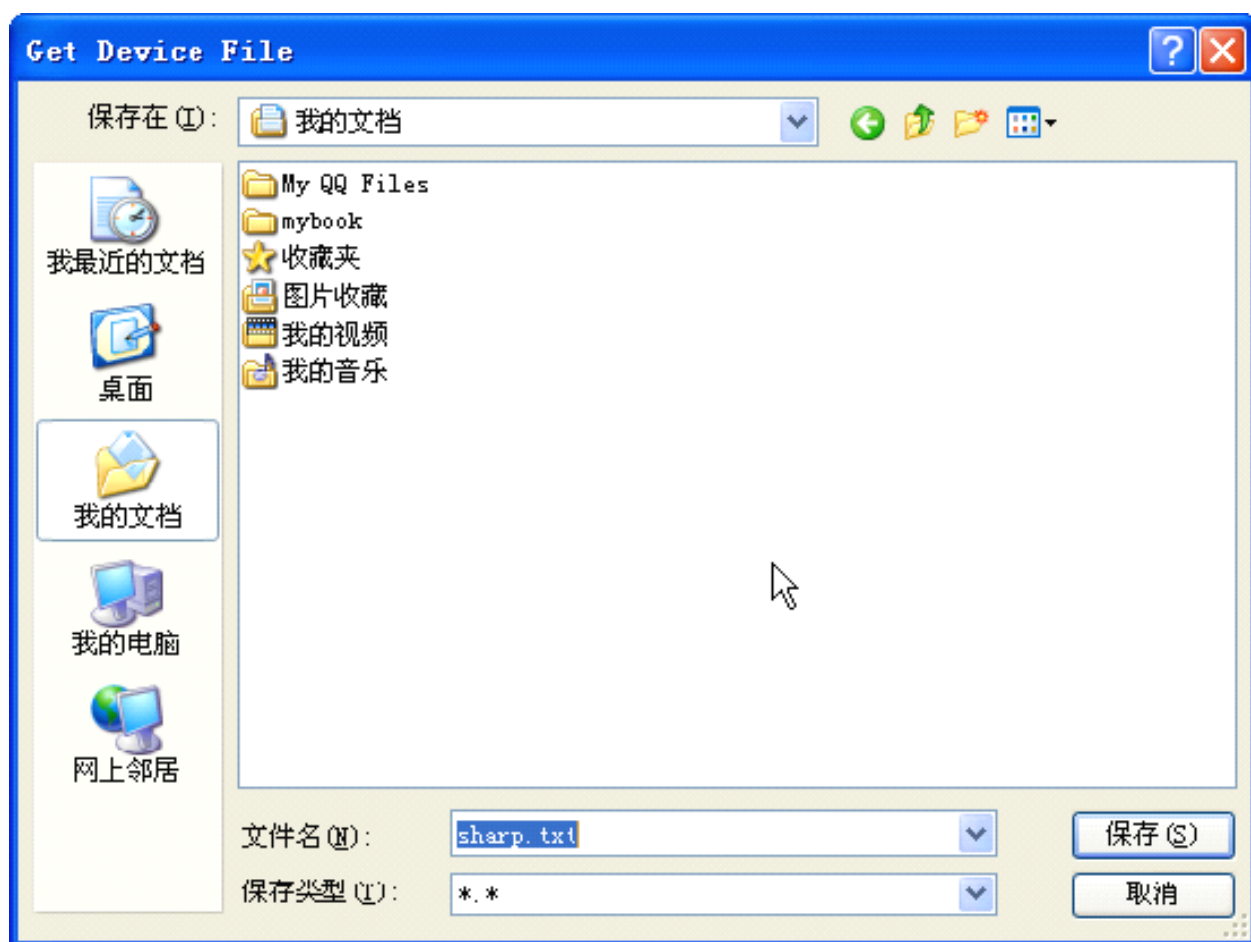
但我们需要检查下“`/data/data/<package name>/files`”路径下是否存在该文件。在文件浏览器中查看结果如下。

本例的`<package name>`是 `com.sharpandroid.file`

+	com.google.android.providers.enhanced		2010-03-28	21:15	drwxr-xr-
+	com.sharpandroid.activity		2010-03-28	11:01	drwxr-xr-
-	com.sharpandroid.file		2010-03-30	11:01	drwxr-xr-
-	files		2010-03-30	11:01	drwxrwx--
	sharp.txt	22	2010-03-30	11:01	-rw-rw--
+	lib		2010-03-30	11:01	drwxr-xr-

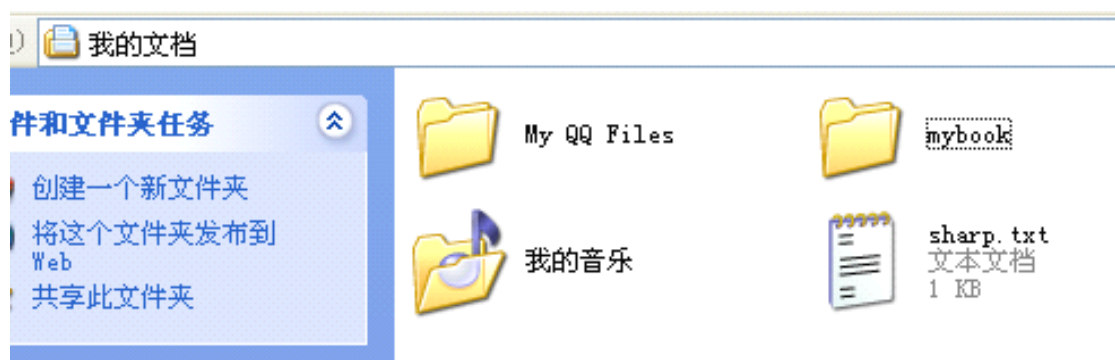
文件导出到 windows 系统中，进行查看。方式如下。

首先点击文件浏览器标题栏右侧的按钮，弹出如下界面。



选择保存的位置，我们保存在“我的文档”中，点击【保存】。

打开【我的文档】，会发现里面多了一个名称为“sharp.txt”文件。



打开文件，其内容如下。



与我们在 testSave 方法中传入的数据一致，因此保存方法测试通过。

读取文件

在 FileService.java 文件中增加 read 方法和 readFile 方法。其代码如下。

```
/**
 * 读取内容
 */
public String read() throws Throwable{
    FileInputStream inStream =
context.openFileInput("sharpandroid.txt");
    byte[] data = readFile(inStream);
    return new String(data);
}
/**
 * 读取文件数据
 */
public byte[] readFile(InputStream inStream) throws Throwable{
    int len = 0;
    byte[] buffer = new byte[1024];
    ByteArrayOutputStream outStream = new
ByteArrayOutputStream();
    while((len = inStream.read(buffer)) != -1){
        outStream.write(buffer, 0, len);
    }
    outStream.close();
    return outStream.toByteArray();
}
```

说明：

- FileInputStream inStream =
context.openFileInput("sharpandroid.txt");

通过上下文打开一个文件输入流。

第一个参数：文件的名称，不能包含路径分隔符“/”。读取的文件路径在
/data/data/<package name>/files目录，如：
/data/data/com.sharpandroid.file/files/sharp.txt。

- public byte[] readFile(InputStream inStream)
该方法将传入其中的文件输入流进行读取，并将其写入一个
ByteArrayOutputStream中。之后再将 ByteArrayOutputStream 中的内容以字
节数组形式返回。

测试读操作

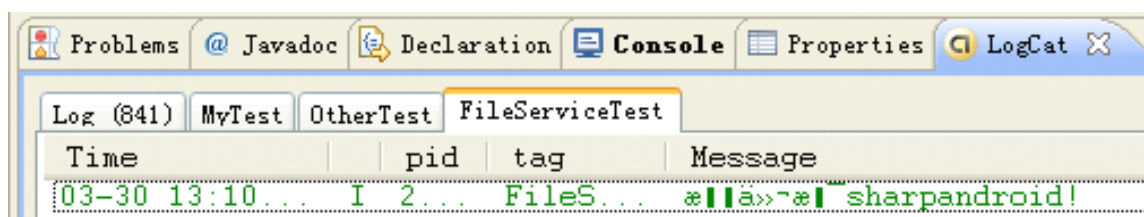
在 FileServiceTest.java 类中增加 testRead()方法。代码如下。

```

FileService fileService = new FileService(getContext());
Log.i(TAG, fileService.read());
}

```

执行该测试方法，测试通过，观察JUnit控制台，绿条通过。
在LogCat中添加过滤器“FileServiceTest”，其结果如下。

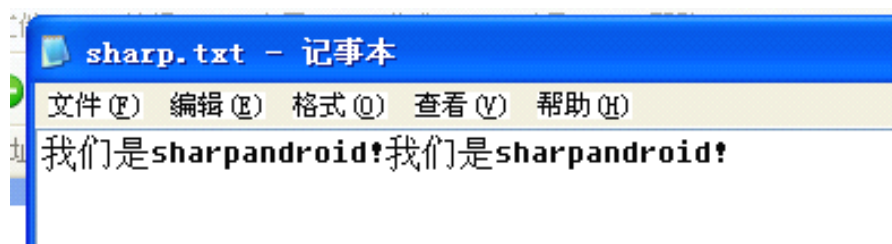


我们向文件中写入的内容是“我们是 sharpandroid”，但由于目前的 LogCat 不支持中文，所以“我们是”三个中文在 LogCat 中显示为乱码。

测试操作模式

测试 Context.MODE_APPEND 模式

刚才我们设置的操作模式为私有模式 Context.MODE_PRIVATE。在这里我们将模式改为 Context.MODE_APPEND，其他的地方不做改变，再次执行测试用例。如上步一样导出并打开 sharp.txt 文件，结果如下。



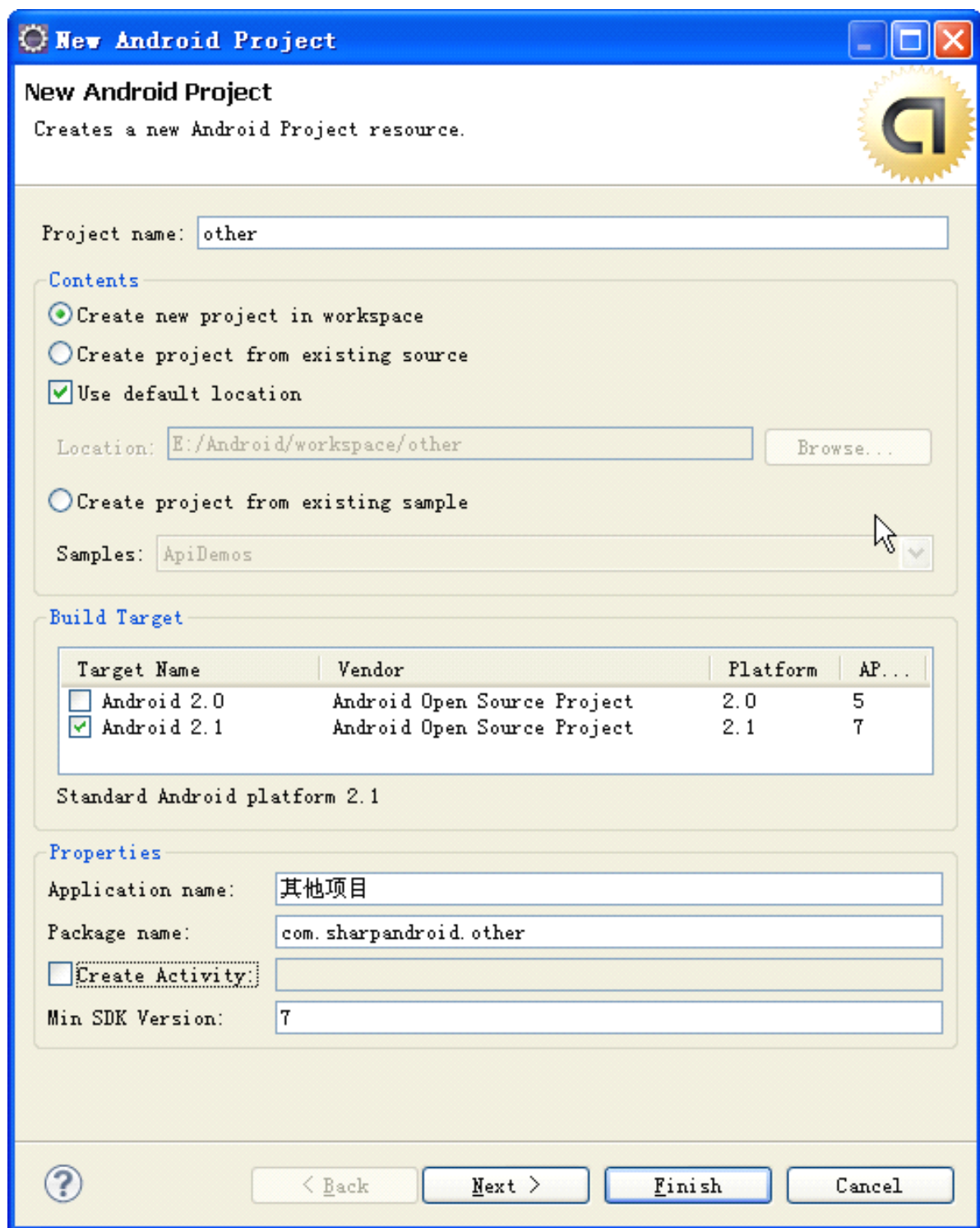
大家可以看出这里在文件后面追加内容的效果。

如果将模式再改回 Context.MODE_PRIVATE，执行测试用例，导出并打开 sharp.txt 文件，其内容如上图。覆盖掉了原有信息。

测试 Context.MODE_WORLD_READABLE 模式

将 save 方法中模式修改为 Context.MODE_WORLD_READABLE。
执行 testSave 方法。

为了测试该文件能否被其他应用访问，我们需要再创建一个应用 other。在 other 中我们采用单元测试来进行测试，因此可以不用创建 Activity。
创建选项如图：



接着在 other 应用中配置单元测试环境。其功能清单文件内容如下。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.sharpandroid.other"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"
    android:label="@string/app_name">
```

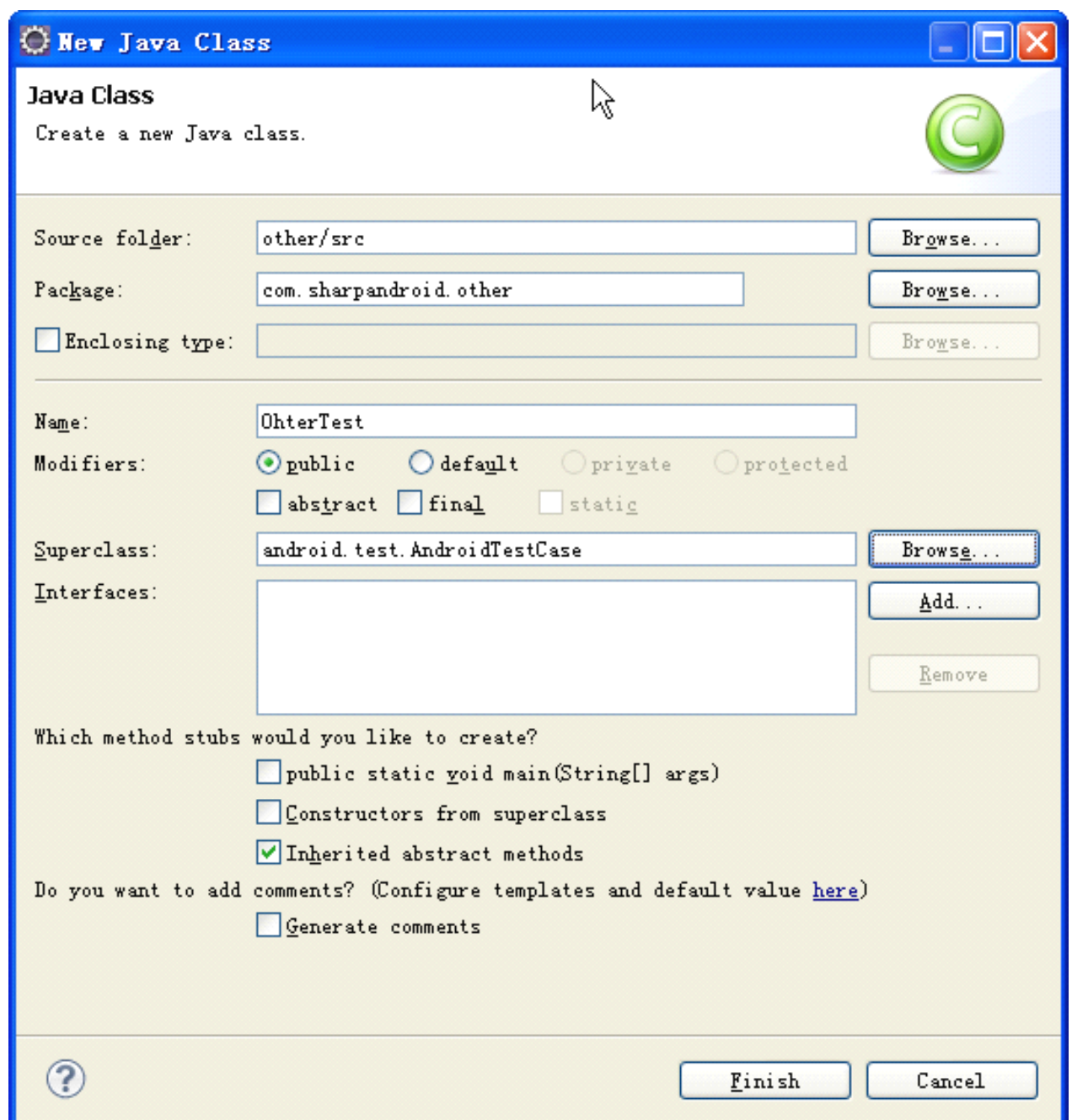


```

</application>
<uses-sdk android:minSdkVersion="7" />
<instrumentation
    android:name="android.test.InstrumentationTestRunner"
    android:targetPackage="com.sharpandroid.other"
    android:label="Tests for My App" />
</manifest>

```

创建测试类 OhterTest.java ， 创建如图：



完成之后，创建 testAccessOtherAppFile 方法。OtherTest.java 内容如下。

```

import java.io.OutputStream;
import android.content.Context;
import android.test.AndroidTestCase;
import android.util.Log;
public class OtherTest extends AndroidTestCase {
    private static final String TAG = "OtherTest";
    /**
     * 读取其他应用中的文件
     */
    public void testAccessOtherAppFile() throws Throwable{
        File file = new File("/data/data/
com.sharppandroid.file/files/sharp.txt");
        FileInputStream inStream = new FileInputStream(file);
        byte[] data = readFile(inStream);
        String content = new String(data);
        Log.i(TAG, content);
    }
    /**
     * 读取文件数据
     */
    public byte[] readFile(InputStream inStream) throws Throwable{
        int len
= 0;
        byte[] buffer = new byte[1024];
        ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
        while((len = inStream.read(buffer)) != -1){
            outputStream.write(buffer, 0, len);
        }
        outputStream.close();
        return outputStream.toByteArray();
    }
}

```

说明:

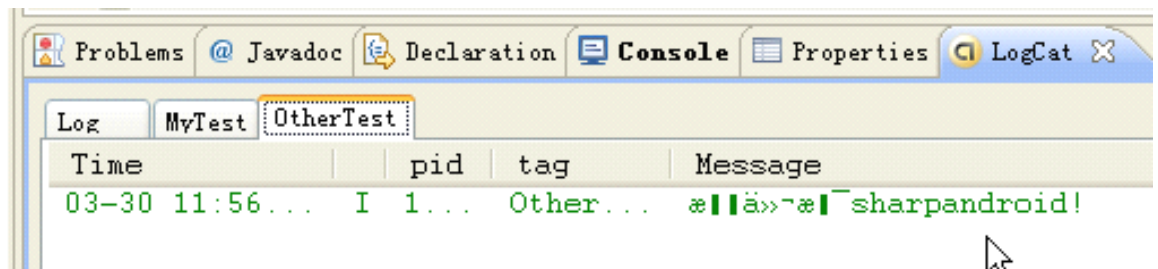
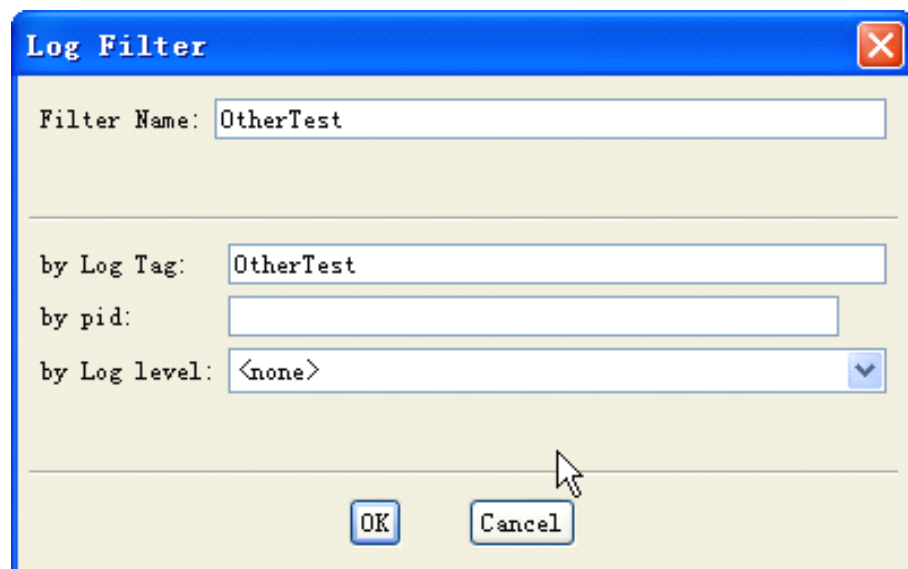
- File file = new File("/data/data/ com.sharppandroid.file/files/sharp.txt");
创建文件，该文件就是在file项目中存储的sharp.txt文件。其中“/data/data/
com.sharppandroid.file/files/sharp.txt”是sharp.txt的绝对路径。注意不要将
com.sharppandroid.file进行拆分或者以com/sharppandroid/file的形式书写，它们是一个字符串。
- FileInputStream inStream = new FileInputStream(file);
获取输入流。
- byte[] data = readFile(inStream);
调用readFile方法，将输入流以字节数组的形式返回。
- String content = new String(data);
用字节数组构建一个字符串。

将字符串打印到LogCat控制台。

执行 `testAccessOtherAppFile()` 方法。

观察 JUnit 控制台，绿条通过。

观察 LogCat 控制台，并添加 OtherTest 过滤器，结果如下。



这时如果我们在 other 应用中向 file 应用的 `sharp.txt` 文件写数据是否能够成功呢？

我们在 `otherTest.java` 类中添加 `testWriteOtherAppFile()` 方法，测试项其他应用的文件中写数据。

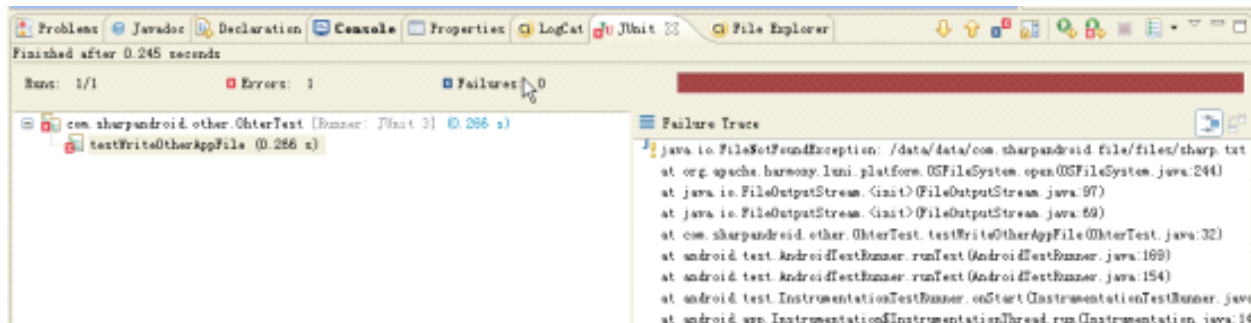
`testWriteOtherAppFile()` 代码如下。

```
/**
 * 编写其他应用中的文件
 */
public void testWriteOtherAppFile() throws Throwable{
    File file = new File("/data/data/com.sharpandroid.file/files/sharp.txt");
    FileOutputStream outStream = new FileOutputStream(file);
    outStream.write("we are sharpandroid".getBytes());
    outStream.close();
}
```

- `FileOutputStream outStream = new FileOutputStream(file);`
创建一个文件输出流。

向相应的文件中写 "we are sharpandroid".

执行 testWriteOtherAppFile 方法。没有通过。



报的异常 “**java.io.FileNotFoundException: /data/data/com.sharpandroid.file/files/sharp.txt**”，大家肯定有疑问，明明该文件存在，就算不能访问，也不应报FileNotFoundException这个异常，报没有权限的错误才是符合我们逻辑的。其实之所以报java.io.FileNotFoundException异常是出于安全的考虑。因为该应用没有访问该文件的权限，既然没有权限，文件对其而言就不可见。

测试 Context.MODE_WORLD_WRITEABLE 模式

如上，将 save 方法中的权限模式改为 **Context.MODE_WORLD_WRITEABLE**，再次执行 testSave () 方法。

之后，依次执行 testAccessOtherAppFile ()、testWriteOtherAppFile () 方法。

情况刚好与上面测试 **Context.MODE_WORLD_READABLE** 模式相反。

testAccessOtherAppFile () 没有测试通过。

执行 testWriteOtherAppFile () 通过。此时再导出并打开 “/data/data/com.sharpandroid.file/files/sharp.txt” 文件，内容如下。



该字符串是我们在 testWriteOtherAppFile () 中重新输入的。

测试外部读写

如上，将 save () 方法中的权限模式改为 **Context.MODE_WORLD_READABLE+Context.MODE_WORLD_WRITEABLE**,

其实可以写成两个常量所代表的数字1+2的结果3也可以。

再次执行 testSave () 方法。

之后，依次执行 testAccessOtherAppFile ()、testWriteOtherAppFile () 方法。都通过了测试，导出查看文件内容如上图所示。

存原有数据，实现追加效果，只需要在权限模式中加上 `Context.MODE_APPEND`，即为 **`Context.MODE_APPEND+Context.MODE_WORLD_READABLE+Context.MODE_WORLD_WRITEABLE`**

本节中，sharp.txt 文件存放于/data 下，也就是存放于手机自身的存储空间上面。

1. 2SDCard 数据存取

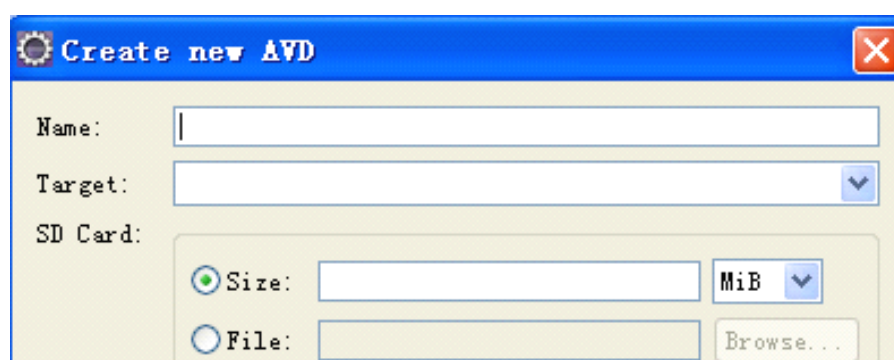
小安：文件都被保存到了手机的存储空间上，要是空间使用完了，怎么办？

大致：现在的手机一般都有 SD 卡，前面介绍的使用 Activity 的 `openFileOutput()` 方法保存文件，文件是存放在手机空间上，一般手机的存储空间不是很大，存放些小文件还行，如果要存放像视频这样的大文件，是不可行的。对于像视频这样的大文件，我们可以把它存放在 SDCard，你可以把它看作是移动硬盘或 U 盘。

下图为 sdcard 的外观。sdcard 可以插入手机的插槽中使用。



在使用 AVD 的 SDCard 之前首先要确定在创建 AVD 时是否为其创建了 SDCard。



如果没有在 Size 中填上大小，那么该 AVD 就没有 SDCard。
该实例的功能与上节中的文件读写类似，唯一的区别在于文件存放的位置不同。

创建项目

创建 Android 项目 SDCardRW，如下图所示：

New Android Project
Creates a new Android Project resource.

Project name: SDCardRW

Contents

- ☒ Create new project in workspace
- ☐ Create project from existing source
- ☒ Use default location

Location: E:/Android/workspace/SDCardRW Browse...

☐ Create project from existing sample

Samples: ApiDemos

Build Target

Target Name	Vendor	Platform	AP...
<input type="checkbox"/> Android 2.0	Android Open Source Project	2.0	5
<input checked="" type="checkbox"/> Android 2.1	Android Open Source Project	2.1	7

Standard Android platform 2.0

Properties

Application name: SDCard数据读写器

Package name: com.sharpandroid.sdcard

☐ Create Activity:

Min SDK Version: 7

The minimum SDK version number that the application requires. Must be an integer

? < Back Next > Finish Cancel

文件保存业务

先需要创建一个业务类 FileService。创建时界面如下图。

New Java Class

Java Class
Create a new Java class.

Source folder: SDCardRW/src Browse...

Package: com.sharppandroid.service Browse...

☐ Enclosing type: Browse...

Name: FileService

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add...
Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

? Finish Cancel

点击【Finish】。

完成 save 方法之后，FileService.java 代码如下。

```
import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
public class FileService {
    private Context context;
    public FileService(Context context) {
        this.context = context;
    }
    /**
     * 保存内容
```

```

        public void saveToSDCard(String content) throws Exception{
            File file = new
File(Environment.getExternalStorageDirectory(), "SDCard.txt");
            FileOutputStream outputStream = new FileOutputStream(file);
            outputStream.write(content.getBytes());
            outputStream.close();
        }
    }
}

```

说明：

- private Context context;
public FileService(Context context) {
 this.context = context;
 }

将当前应用的上下文Context作为其属性，当创建该类时传入。这样在该类的其他方法中就可以利用该Context获取对应用的信息。修改构造方法的目的是确保该属性在创建时会被传入。

```

File file = new File(Environment.getExternalStorageDirectory(),
"SDCard.txt");

```

第一个参数：文件存放的文件夹路径

Environment.getExternalStorageDirectory()方法用于获取SDCard的目录，当然要获取SDCard的目录，你也可以这样写：

```

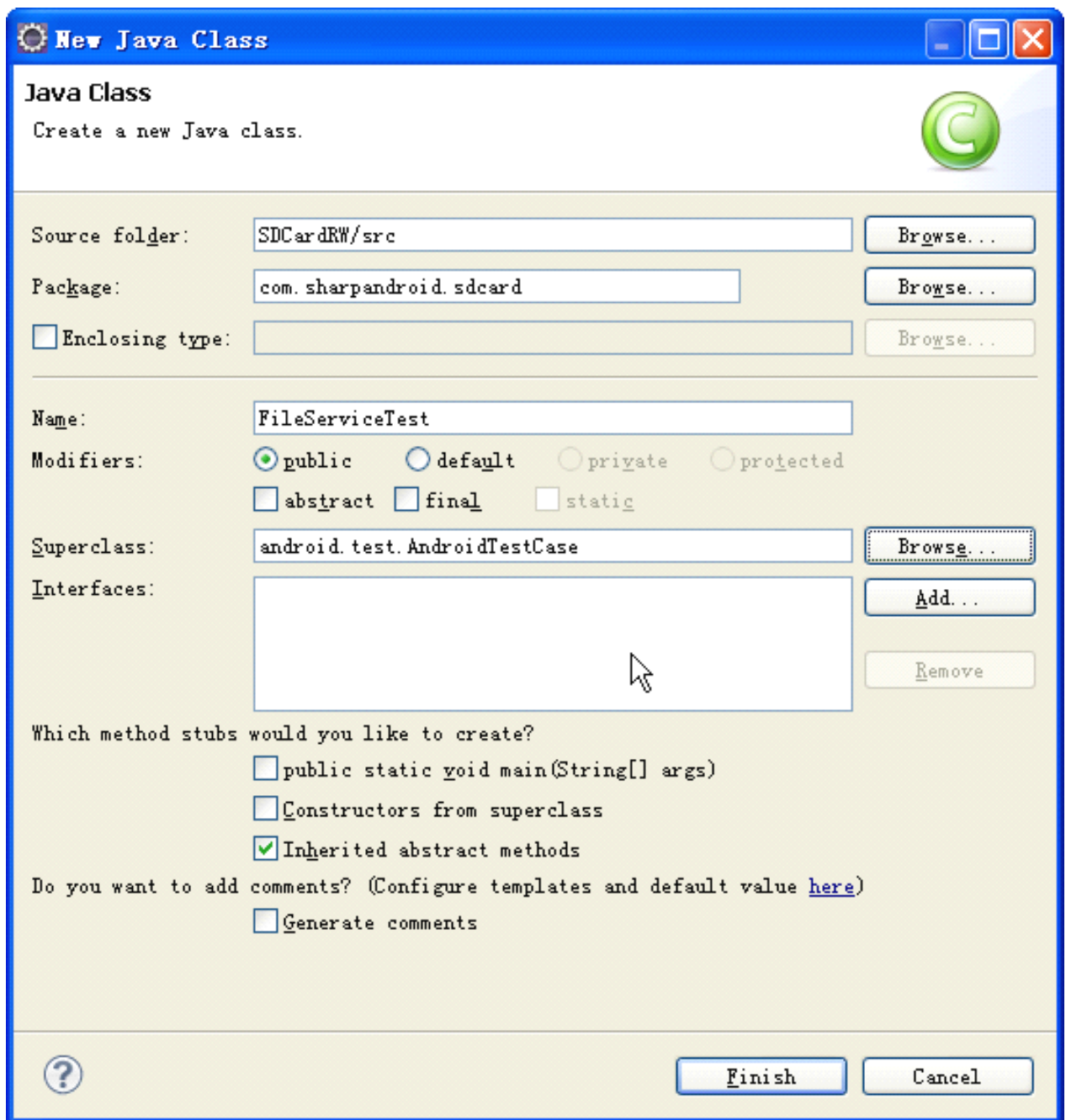
File sdCardDir = new File("/sdcard"); //获取SDCard目录
File saveFile = new File(sdCardDir, "SDCard.txt");
//上面两句代码可以合成一句： File saveFile = new File("/sdcard/
SDCard.txt");

```

第二个参数：文件的名称

测试保存方法

创建 FileServiceTest.java 测试类。



添加 testSave 测试方法，之后 FileServiceTest 测试类代码如下。

```
import java.io.InputStream;
import java.io.OutputStream;
import android.content.Context;
import android.test.AndroidTestCase;
import android.util.Log;
public class FileServiceTest extends AndroidTestCase {
    private static final String TAG = "FileServiceTest";
    public void testSaveToSDCard() throws Throwable{
        if(Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
            //该状态代表SDCard已经安装在手机上，并且可以进行读写访问
```

```

        fileService.saveToSDCard("明天会更好!");
    }else{
        Log.i(TAG, "sdcard不存在或者写保护了");
    }
}
}

```

说明:

- `if(Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED))`
判断SDCard是否可用。`Environment.getExternalStorageState()` 获取外存储设备的状态, 如果等于`Environment.MEDIA_MOUNTED`, 则说明可用。
- 在实际开发中, 由于手机可能并没有安装SDCard, 所以只要用到SDCard之前都要加上该判断, 以增强程序容错性。

执行测试

配置单元测试环境, 配置后代码如下:

AndroidManifest.xml

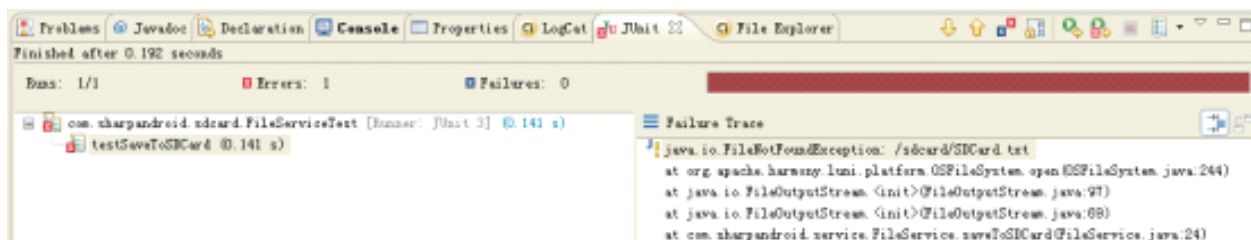
```

<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.sharpandroid.sdcard"
        android:versionCode="1"
        android:versionName="1.0">
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name">
        <uses-library android:name="android.test.runner" />
    </application>
    <uses-sdk android:minSdkVersion="7" />
    <instrumentation
        android:name="android.test.InstrumentationTestRunner"
        android:targetPackage="com.sharpandroid.sdcard"
        android:label="Tests for My App" />
</manifest>

```

执行testSave方法。

执行之后, 观察JUnit控制台, 结果如下:



报的异常是：“**java.io.FileNotFoundException: /sdcard/SDCard.txt**”，这是由于使用 SDCard 是需要添加访问权限的，而我们并没有添加权限，因此根据 Android 的安全机制，对该程序而言 SDCard 为不存在。

访问 SDCard

在 AndroidManifest.xml 添加访问 SDCard 权限的代码如下。

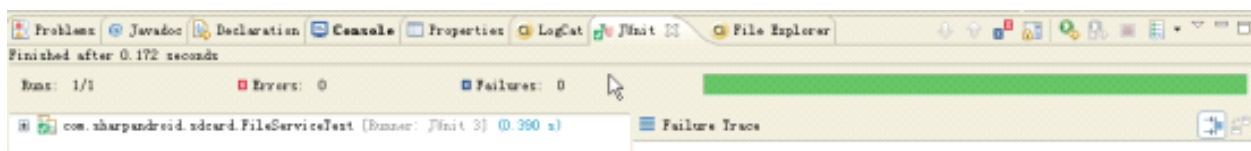
<!-- 在 SDCard 中创建与删除文件权限 -->

<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>

<!-- 往 SDCard 写入数据权限 -->

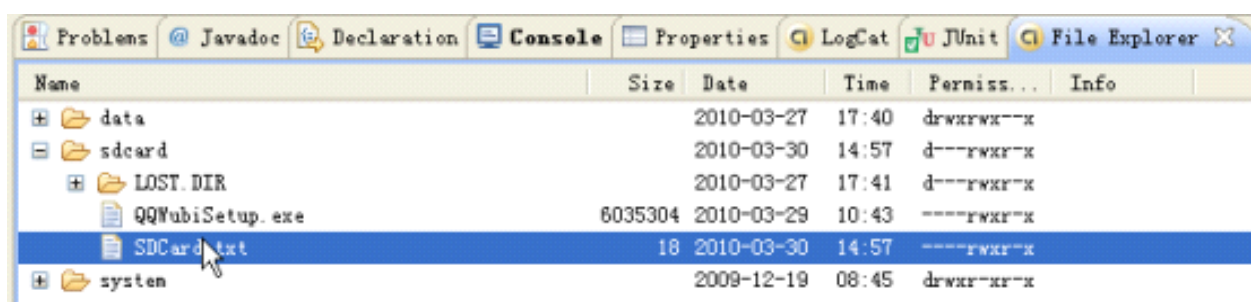
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

再次执行 testSave 方法，结果如下图。



测试通过。

但我们需要检查下“/sdcard”路径下是否存在该文件。在文件浏览器中查看结果如下。

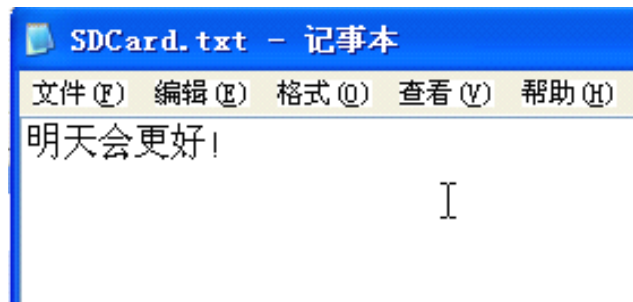


说明文件已经存在。由于在文件浏览器中无法打开文件，查看到它的内容，我们可以将文件导出到 windows 系统中，进行查看。方式如下。

首先点击文件浏览器标题栏右侧的  按钮，弹出如下界面。

选择保存的位置，我们将其保存在“我的文档”中，点击【保存】。
打开【我的文档】，会发现里面多了一个名称为“SDCard.txt”文件。

打开文件，其内容如下。



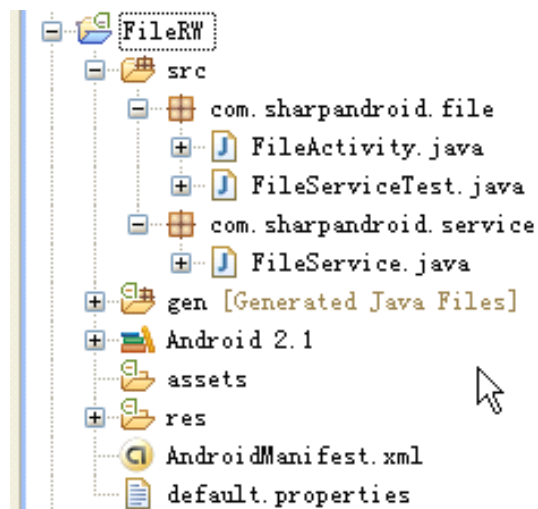
与我们在 testSave 方法中传入的数据一致，因此保存方法测试通过。

获取文件路径

Activity 还提供了 getCacheDir()和 getFilesDir()方法：

getCacheDir()方法用于获取/data/data/<package name>/cache 目录

getFilesDir()方法用于获取/data/data/<package name>/files 目录



1.3SharedPreferences（共享参数）

小安：我发现在应用软件中有很多的配置文件，这些文件的操作和上面讲解的文件操作有什么不同吗？

大致：当然有区别了，配置文件是还要被解析的。在配置文件中会用到 SharedPreferences，我们可以做一实例进行测试。

1.3.1 SharedPreferences 简介

为了保存软件的设置参数，Android 平台为我们提供了一个 SharedPreferences 类，它是一个轻量级的存储类，特别适合于保存软件配置参数。使用 SharedPreferences 保存数据，其背后是用 xml 文件存放数据，文件存放在/data/data/<package name>/shared_prefs 目录下

```
SharedPreferences pre = Context.getSharedPreferences("soft", Context.MODE_WORLD_READABLE);
```

在这里我们可以调用 activity 为我们提供的方法，这个方法有两个参数：

1. 文件名。在这里要特别注意。因为在 Android 中已经确定了 SharedPreferences 是以 xml 形式保存，所以，在填写文件名参数时，不要给定“.xml”后缀，android 会自动添加。只要直接写上文件名，即可。它会直接被保存在/data/data/<package name>/shared_prefs 路径下。它是采用键值对的形式保存参数。当你需要获得某个参数值时，按照参数的键索引即可。
2. 第二个可以理解为创建模式和之前的文件存储的模式是一样的。

Context.MODE_PRIVATE

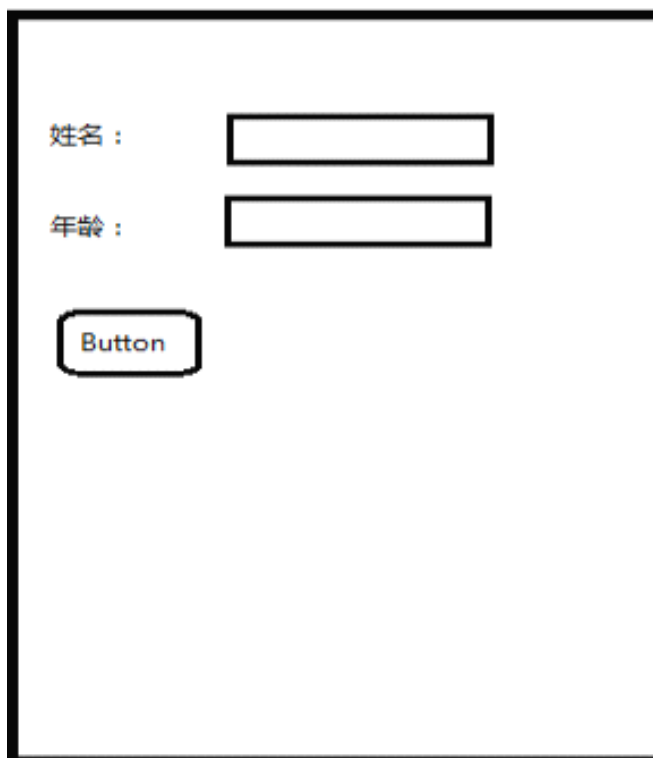
Context.MODE_APPEND

Context.MODE_WORLD_READABLE

Context.MODE_WORLD_WRITEABLE

1.3.2 设计思路

下面我们通过实现一个简单的示例，来掌握 SharedPreferences 的基本应用。当用户在 EditText 中输入相应的值，而后通过 SharedPreferences 来保存相应参数。草图：



1.3.3 界面设计

编写 Main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        >
        <TextView
            android:id="@+id/nameView"
            android:layout_width="60px"
            android:layout_height="wrap_content"
            android:text="name:"
            />
        <EditText
            android:id="@+id/name"
            android:layout_width="60px"
            android:layout_height="wrap_content"
            android:layout_alignTop="@id/nameView"
            android:layout_alignLeft="@id/nameView"
            />
    </RelativeLayout>
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"
        />
</LinearLayout>
```

```

</RelativeLayout>
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    >
    <TextView
        android:id="@+id/ageView"
        android:layout_width="60px"
        android:layout_height="wrap_content"
        android:text="age:"
    />
    <EditText
        android:id="@+id/age"
        android:layout_width="60px"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/ageView"
        android:layout_toRightOf="@id/ageView"
    />
</RelativeLayout>
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    >
    <Button
        android:id="@+id/save"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="保存"
    />
    <Button
        android:id="@+id/load"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@id/save"
        android:layout_toRightOf="@id/save"
        android:text="读取"
    />
</RelativeLayout>
<TextView
    android:id="@+id/text01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    >

```

</LinearLayout>

说明:

要注意控件之间的布局。在 `EditText` 里输入用户的值，当点击 `Button`，保存用户的值。设置文本框的长度。

这里有涉及到了 Android 的显示单位的知识，在这里我们复习一下：

- `px (pixels)` 像素
一般 HVGA 代表 320x480 像素，这个用的比较多。
- `dip` 或 `dp (device independent pixels)` 设备独立像素
这个和设备硬件有关，一般为了支持 WVGA、HVGA 和 QVGA 推荐使用这个，不依赖像素。
- `sp (scaled pixels — best for text size)` 比例像素
主要处理字体的大小，可以根据系统的字体自适应。

下面几个不太常用：

- `in (inches)` 英寸
 - `mm (millimeters)` 毫米
 - `pt (points)` 点，1/72 英寸
- 建议在设置非文字控件，可以使用 `dip` 设置布局。

在绝对布局中，如何让年龄的 `EditText` 在年龄的文本框的右边显示呢？可以在布局文件的 `EditText` 部分中添加该属性：

`android:layout_toRightOf`

并且让年龄的 `EditText` 与年龄的文本框顶部对齐，要加入以下属性：

`android:layout_alignTop`

在年龄的输入框内，逻辑只需要接收数字值就好，那么，我们可以添加一下代码：

`android:numeric="integer"`

这样可以限制输入的值为整数。类似与网页中注册时的拼写检查。

当布局页面完成后，可以先运行观看软件布局是否理想。因为插件自带的预览功能，使用起来并不理想。但是我们可以使用软件：**DroidDraw**。它就是专门为 Android 开发程序设计界面的工具。它能提供基本的预览功能。虽然一些高级的控件它并不支持。但是对于初学者，学习基本布局还是非常有用的。在学习期间，推荐使用。下载地址：
<http://www.droiddraw.org>。

效果图：



1.3.4 代码处理

当点击 Button，触发点击事件，利用 SharedPreferences 将值保存。

```
Button button = (Button)findViewById(R.id.setbutton);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        //获得输入框控件
        EditText nameText = (EditText)findViewById(R.id.name);
        EditText ageText = (EditText)findViewById(R.id.age);
    }
});
```

以上代码，是再简单不过的获得了相应的控件的代码。那么我们思考一下这样的编写是否合理呢？当每点击 button 一次，系统就会进行一次 nameText、ageText 控件的搜索过程，这里按照 Android 的编程习惯，可以将使用频繁的控件作为成员变量，减少不必要的资源浪费。在 Android 编程中，性能优化是很重要的。下面就是进入 SharedPreferences 的具体操作代码。

```
SharedPreferences sharedPreferences = getSharedPreferences("sharppandroid ",
Context.MODE_PRIVATE);
Editor editor = sharedPreferences.edit();
editor.putString("name", "NEwii");
editor.putInt("age", 21);
editor.commit();
```

获得 SharedPreferences 的编辑器。在这里，一般程序员可能想要使用类似 put 等方法，将值放入。所以，需要注意一下，SharedPreferences 需要使用一个编辑器。

放入数据，编辑器已经为用户提供了常用数据类型。

这一步，经常被开发者遗忘。一定要注意，虽然它前面的操作类似于 map，但它又有别于 map，不是放进去就 OK，还需要 commit() 一下。对于没有类似 JavaEE 开发经验的读者可能不是很理解，为什么还需要调用 commit()。在这里涉及到了一个事务提交的概念，它是

name 字段已经修改，这样就出现了数据不同步的现象。commit()就是为了解决这个问题。部分代码如下：

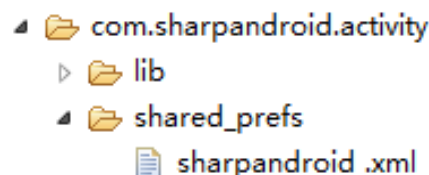
```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    nameText = (EditText)findViewById(R.id.name);
    ageText = (EditText)findViewById(R.id.age);
    Button setbutton = (Button)findViewById(R.id.setbutton);
    setbutton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            //获得输入框的内容
            SharedPreferences sharedPreferences = getSharedPreferences("sharpandroid ",
            Context.MODE_PRIVATE);

            String name = nameText.getText().toString();
            String age = ageText.getText().toString();

            Editor editor = sharedPreferences.edit();
            editor.putString("name", name);
            editor.putInt("age", Integer.parseInt(age));
            editor.commit();
            //添加一个提示，当添加成功可以看到该提示。
            Toast.makeText(PreferencesActivity.this, "保存成功.",
            Toast.LENGTH_LONG).show();

        }
    });
}
```

运行程序。当出现成功的提示，说明程序运行成功。可以在 ADT 的 File Explorer 中查看到生成的文件。在 data/data 下：



```
com.sharpandroid.activity
├── lib
│   └── shared_prefs
│       └── sharpandroid.xml
```

135

将文件导出，内容就是：

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<int name="age" value="21" />
<string name="name">NEwii</string>
</map>
```

学会了怎样写入 SharedPreferences，下面就是读取 SharedPreferences 的值的方法。

数据。

当再添加 **Button** 时，我们可能想又要再重写一次 `View.OnClickListener()` 了？这样一来，两个 **Button** 需要两个点击事件的监听器，那么三个、四个 **Button** 呢？那么我们有办法省去每次都使用匿名内部类的麻烦呢？下面我们将使用另外一种形式，更好的解决这个问题，代码如下：

```
package cn.sharpandroid.sharedPTest;

import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class SharedPreferenceTest extends Activity {
    EditText nameEdit ;
    EditText ageEdit;
    TextView text;
    Button button01;
    Button button02;
    String info = "提交成功";
    public void onCreate(Bundle savedInstanceState) {
        super. onCreate(savedInstanceState);
        setContentView(R.layout. main);

        nameEdit = (EditText)findViewById(R.id. name);
        ageEdit = (EditText)findViewById(R.id. age);
        text = (TextView)findViewById(R.id. text01);
        button01 = (Button)findViewById(R.id. save);
        button02 = (Button)findViewById(R.id. load);

        button01.setOnClickListener(listener);
        button02.setOnClickListener(listener);
    }

    private View.OnClickListener listener = new View.OnClickListener() {
        public void onClick(View v) {
            SharedPreferences sp = getSharedPreferences("preferences",
```

```

        Button button = (Button)v;
        switch (button.getId()) {
            case R.id. save:
                Editor editor = sp.edit();
                editor.putString("name", nameEdit.getText().toString());
                try {editor.putInt("age",
Integer.parseInt(ageEdit.getText().toString()));
                } catch (Exception e) {info = "提交失败";}
                editor.commit();
                DisplayToast(info);
                break;
            case R.id. load:
                sp.getString("name", null);
                sp.getInt("age", 0);
                text.setText(sp.getString("name", null)+sp.getInt("age", 0));
                break;
        }
    }
};

public void DisplayToast(String context){
    Toast.makeText(SharedPreferencesTest.this, context, 200).show();
}

```

}说明:

- 新建一个类，专门用于处理 Button 的点击事件，并实现 OnClickListener 接口，注意你要实现的是 android.view.View.OnClickListener。并建立一个构造器，使 Activity 得上下文可以传进来，供其他代码使用。
- 在这里我们还要清楚 context 与 activity 的关系。因为传入的 context 没有 findViewById() 方法，以至于得不到我们需要使用的控件，所以需要将其强制转换一下，转换成 activity 使用。
- 在这里，所有的Button的事件都会触发ClickEvent类的onClick()方法。那么我们怎样区分那个Button被点击呢？这是时候我们可以用到onClick()方法的参数。View代表着控件，那么可以通过View的getId()的方法，区分相应的控件。

总结：通过整体代码的改造，可以先使用以前的思路，完成该功能，然后再使用该方法，完成该功能。对比两者Activity的代码量。注意，这里不仅仅是代码被转移，在这里又引出了一个非常重要的概念——MVC设计模式。这个概念非常之重要，对代码进行了一个面向对象的封装，可以避免在一个Activity中写过多的代码，影响Activity的可读性与可维护性。

- SharedPreferences通过get，然后输入键，获得与其相对应的值。但是值得注意的是，其第二个参数为缺省的返回值。

获得后用 `TextView` 用以显示，获得具体代码不在此赘述。

使 `SharedPreferences` 可以让其他程序访问的方法

- 首要创建的 `SharedPreferences`，并且使用创建模式为：

`Context.MODE_WORLD_READABLE + Context.MODE_WORLD_WRITEABLE`

- 首先取得创建应用的上下文。

```
Context                                otherContext                                =  
this.getContext().createPackageContext("cn.sharpandroid.soft",  
Context.CONTEXT_IGNORE_SECURITY);  
SharedPreferences preferences = otherContext.getSharedPreferences("sharpandroid",  
Context.MODE_PRIVATE)
```

小安：有了这些知识，对于保存数据和配置软件就足够了。

大致：错，还有一个最重要的保存数据的知识没有接触到，等你遇到这类问题之后，自然会来找我的。