实验七：数据存储（一）

1 实验目的

1. 学习 SharedPreferences 的基本使用。
2. 学习 Android 中常见的文件操作方法。
3. 复习 Android 界面编程。

2 实验内容

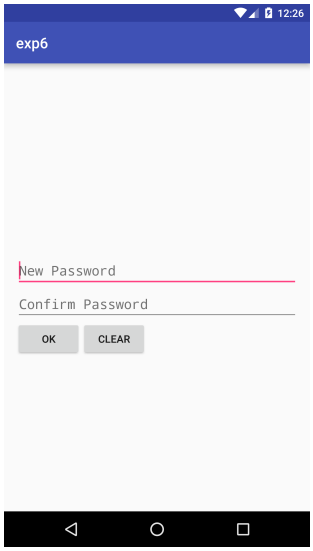


Figure 1: 首次进入，呈现创建密码界面

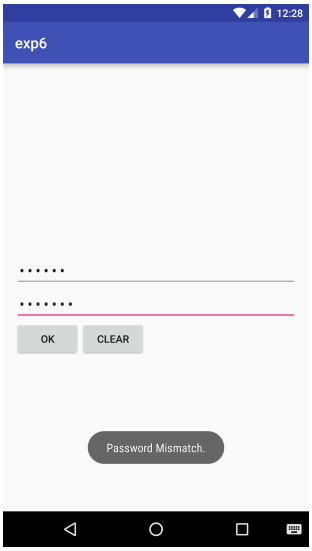


Figure 2: 若密码不匹配，弹出 Toast 提示

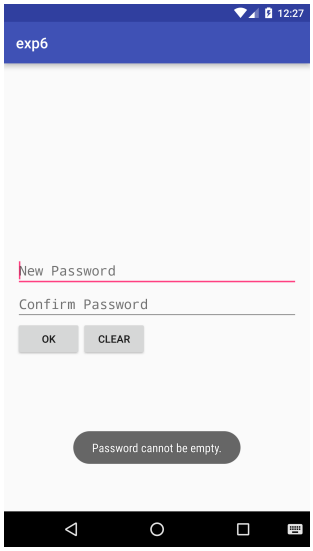


Figure 3: 若密码为空，弹出 Toast 提示

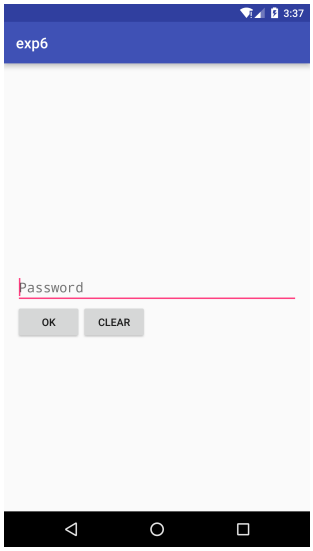


Figure 4: 退出后第二次进入呈现输入密码界面

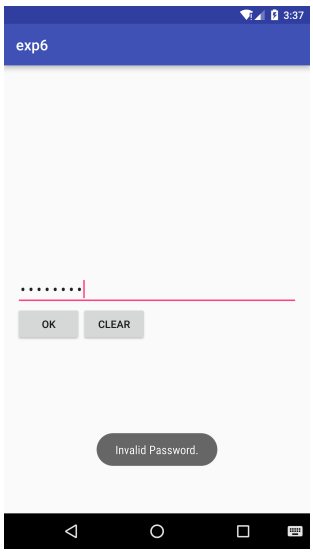


Figure 5: 若密码不正确，弹出 Toast 提示

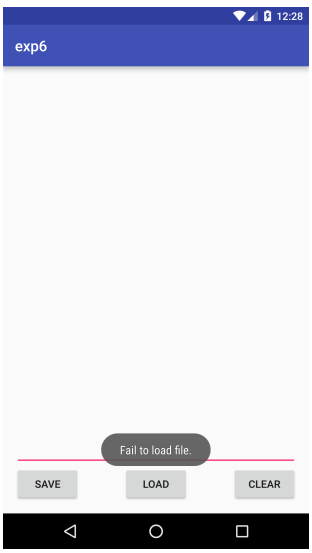


Figure 6: 文件加载失败，弹出 Toast 提示

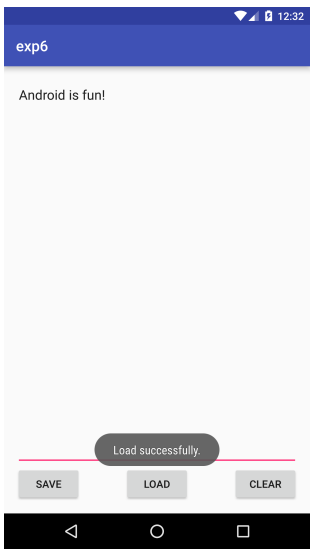


Figure 7: 成功导入文件，弹出 Toast 提示

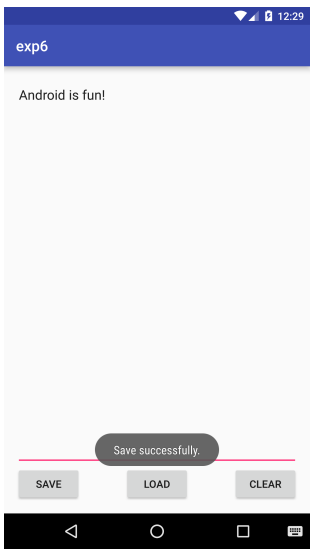


Figure 8: 成功保存文件，弹出 Toast 提示

1. 如 Figure 1 至 Figure 8 所示，本次实验演示应用包含两个 Activity.
2. 首先是密码输入 Activity:

- 若应用首次启动，则界面呈现出两个输入框，分别为新密码输入框和确认密码输入框。
- 输入框下方有两个按钮：
  - OK 按钮点击后：
    * 若 New Password 为空，则发出 Toast 提示。见 Figure 3.
    * 若 New Password 与 Confirm Password 不匹配，则发出 Toast 提示。见 Figure 2。
    * 若两密码匹配，则保存此密码，并进入文件编辑 Activity.
  - CLEAR 按钮点击后：**清除两输入框的内容**。
- 完成创建密码后，退出应用再进入应用，则只呈现一个密码输入框。见 Figure 4.
  - 点击 OK 按钮后，若输入的密码与之前的密码不匹配，则弹出 Toast 提示。见 Figure 5
  - 点击 CLEAR 按钮后，清除密码输入框的内容。
- **出于演示和学习的目的，本次实验我们使用 SharedPreferences 来保存密码。但实际应用中不会使用这种方式来存储敏感信息，而是采用更安全的机制。见这里和这里。**

3. 文件编辑 Activity:

   - 界面底部有三个按钮，高度一致，顶对齐，按钮水平均匀分布。三个按钮上方除 ActionBar 和 StatusBar 之外的全部空间由一个 EditText 占据（保留 margin）。EditText 内的文字竖直方向置顶，左对齐。
   - 在编辑区域输入任意内容，点击 SAVE 按钮后能保存到**指定文件（文件名随意）**. 成功保存后，弹出 Toast 提示。见 Figure 8.
   - 点击 CLEAR 按钮，能清空编辑区域的内容。
   - 点击 LOAD 按钮，能够从同一文件导入内容，并显示到编辑框中。若成功导入，则弹出 Toast 提示。见 Figure 7. 若读取文件过程中出现异常（如文件不存在），则弹出 Toast 提示。见 Figure 6.

4. **特殊要求：进入文件编辑 Activity 后，若点击返回按钮，则直接返回 Home 界面，不再返回密码输入 Activity.**

3 基础知识

3.1 SharedPreferences 的使用

3.1.1 SharedPreferences 的读取

在 Android 中，用于获取 SharedPreferences 的接口是 getSharedPreferences(String, int) 函数。两个参数的意义：

- String: Desired preferences file. If a preferences file by this name does not exist, it will be created when you retrieve an editor.
- int: Operating mode. Use 0 or MODE_PRIVATE for the default operation.

我们对 SharedPreferences 的**读取**操作是通过 getSharedPreferences(String, int) 函数返回的 SharedPreferences 对象的方法来完成的。查阅文档可以看到，SharedPreferences 支持如下几种方法读取之前存储的数据：

- abstract Map<String, ?> getAll()
- abstract boolean getBoolean(String key, boolean defValue)
- abstract float getFloat(String key, float defValue)
- abstract int getInt(String key, int defValue)
- abstract long getLong(String key, long defValue)
- abstract String getString(String key, String defValue)
- abstract Set<String> getStringSet(String key, Set<String> defValues)

所有方法都需要传入一个 defValue 参数，在给定的 key 不存在时，SharedPreferences 会直接返回这个默认值。

3.1.2 SharedPreferences 的写入

所有对 SharedPreferences 的写入操作，都必须通过 SharedPreferences.edit() 函数返回的 Editor 对象来完成。举例：

```
Context context = getActivity();
SharedPreferences sharedPref = context.getSharedPreferences("MY_PREFERENCE", Context.MODE_PRIVATE);
// Alternatively, if you need just one shared preference file for your activity, you can use the
    getPreferences() method:
//  SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putInt("KEY_SCORE", newHighScore);
editor.commit();
```

3.2 Android 中的文件操作

Android 中的存储空间分为两种：Internal Storage 和 External Storage.

3.2.1 Internal Storage

- 默认情况下，保存在 Internal Storage 的文件只有应用程序可见，其他应用，以及用户本身是无法访问这些文件的。向 Internal Storage 写入文件的示例代码如下：

```
try (FileOutputStream fileOutputStream = openFileOutput(FILE_NAME, MODE_PRIVATE)) {
    String str = "Hello, World!";
    fileOutputStream.write(str.getBytes());
    Log.i("TAG", "Successfully saved file.");
} catch (IOException ex) {
    Log.e("TAG", "Fail to save file.");
}
```

  若对应的文件不存在，openFileOutput(String, int) 函数会直接新建文件。注意传入的文件名参数不能含有 path separators（即 '/'）. 该函数返回一个 FileOutputStream 对象，可以调用 write() 方法写入内容。

- 相应地，文件的读取可以使用 openFileInput(String) 来读取文件。该函数返回一个 FileInputStream，调用 read() 方法读取内容。

```
try (FileInputStream fileInputStream = openFileInput(FILE_NAME)) {
    byte[] contents = new byte[fileInputStream.available()];
    fileInputStream.read(contents);
} catch (IOException ex) {
    Log.e("TAG", "Fail to read file.");
}
```

3.2.2 External Storage

Android 支持使用 Java 的文件 API 来读写文件，但是关键的点在于要有一个合适的路径。如果你要存储一些公开的，体积较大的文件（如媒体文件），External Storage 就是一个比较合适的地方。如文档中所说：

> All Android devices have two file storage areas: “internal” and “external” storage. These names come from the early days of Android, when most devices offered built-in non-volatile memory (internal storage), plus a removable storage medium such as a micro SD card (external storage). Some devices divide the permanent storage space into “internal” and “external” partitions, **so even without a removable storage medium, there are always two storage spaces and the API behavior is the same whether the external storage is removable or not.**

无论是否支持外置 SD 卡，所有的 Android 设备都会将存储空间分为 internal 和 external 两部分。

- 要往 External Storage 写入文件，需要在 AndroidManifest.xml 文件中声明权限：

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

- 随后调用 getExternalFilesDir(String type) 或 Environment.getExternalStoragePublicDirectory() 来获取 SD 卡路径。两者的区别在于：前者指向的目录会在应用卸载时被删除，而后者不会。
- 上面的两个函数均返回一个 File 对象，代表一个目录路径，使用这个 File 对象，再结合文件名，即可创建 FileInputStream 或 FileOutputStream 来进行文件读写。
- 举例：

```
void createExternalStoragePrivateFile() {
    // Create a path where we will place our private file on external
    // storage.
    File file = new File(getExternalFilesDir(null), "DemoFile.jpg");

    try {
        // Very simple code to copy a picture from the application's
        // resource into the external file.  Note that this code does
        // no error checking, and assumes the picture is small (does not
        // try to copy it in chunks).  Note that if external storage is
        // not currently mounted this will silently fail.
        InputStream is = getResources().openRawResource(R.drawable.balloons);
        OutputStream os = new FileOutputStream(file);
        byte[] data = new byte[is.available()];
        is.read(data);
        os.write(data);
        is.close();
        os.close();
    } catch (IOException e) {
        // Unable to create file, likely because external storage is
        // not currently mounted.
        Log.w("ExternalStorage", "Error writing " + file, e);
    }
}
```

4 参考实现

4.1 如何使文件编辑 Activity 的 EditText 占据上方全部空间?

使用 LinearLayout 和 layout_weight 属性。

> if there are three text fields and two of them declare a weight of 1, while the other is given no weight, the third text field without weight will not grow and will only occupy the area required by its content. **The other two will expand equally to fill the space remaining after all three fields are measured.**

更多请查阅文档。

4.2 当 Activity 不可见时，如何将其从 activity stack 中除去（按返回键直接返回 Home)?

AndroidManifest.xml 中设置 noHistory 属性。

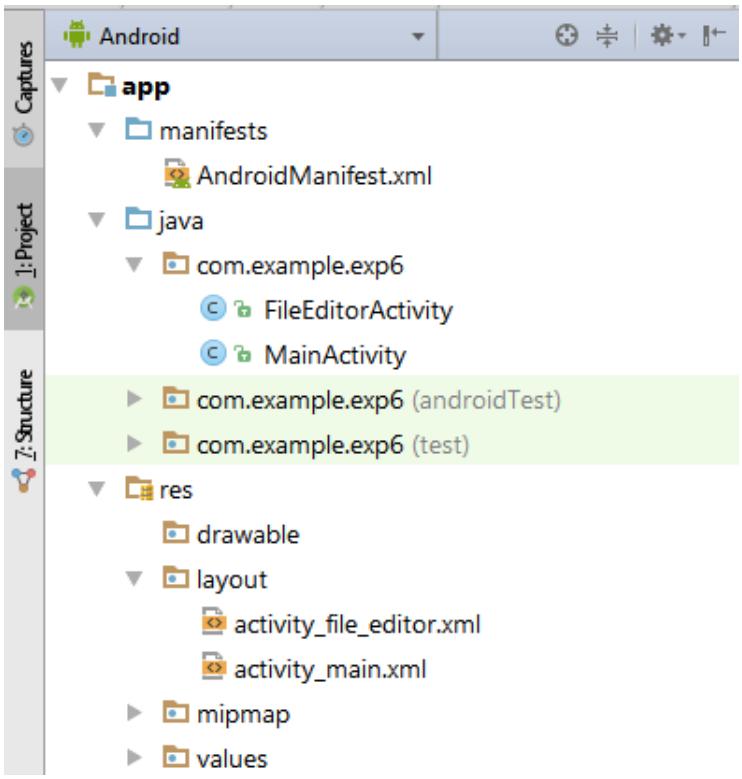
> Whether or not the activity should be removed from the activity stack and finished (its finish() method called) when the user navigates away from it and it’s no longer visible on screen —“true” if it should be finished, and “false” if not. The default value is “false”.

4.3 如何根据需要隐藏/显示特定的控件?

见文档:

> Set visibility: You can hide or show views using setVisibility(int).

4.4 参考工程目录结构



5 检查内容

1. 布局显示与 Figure 1 至 Figure 8 一致。
2. 应用逻辑与上文描述一致：
   - 异常情况弹出 Toast 提示。
   - 创建密码后重新启动应用，直接显示单个输入框输入密码。
   - 文件编辑界面：能够正常保存和读取文件。
3. 在实验报告中简要描述 Internal Storage 和 External Storage 的区别，以及它们的适用场景。

6 提交说明

1. Deadline： 下一次实验课前一天晚上 12 点。
2. 提交位置： ftp://222.200.185.18:1890/ 对应文件夹下
3. 命名要求：附件命名及格式要求：学号_姓名_labX.zip（姓名中文拼音均可）
   重复提交命名格式要求：学号_姓名_labX_Vn.zip
4. 目录结构：如下图所示，其中项目代码文件为项目文件夹，提交之前先clean

```
14331111_huashen_lab1 --
                       |
                       --  lab1实验报告.pdf
                       |
                       --  lab1_code（包含项目代码文件）
```