移动应用开发实验报告

lab7：数据存储（一）

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。

实验过程

整体设计思路： 首先登录界面的密码使用SharedPreferences存储，登录后的文件界面使用文件存储。所以要写两个界面的xml文件和两个Activity文件，首先进入登录界面，正确输入密码后进入文件界面。登录界面只要利用SharedPreferences存储，然后以后登录时和之前保存的密码判断是否一致即可；文件界面主要有保存，读取和删除三个功能，保存调用openFileOutput和write方法，将输入框中的文件名和内容写入；读取调用openFileInput和read方法，将输入框中的文件名对应的内容显示出来；删除直接调用deleteFile方法将对应文件名的文件删除即可。总体来说比较简单。

登录界面：
xml布局没什么好说的，对齐即可，把两个button用链条连接起来也行，因为用的约束布局。然后注意因为我们的输入框要输入密码，所以要把输入框的inputype设置为password，这里我用的是numberpassword，因为输入的时候只有数字键盘，比较简单，而且demo也是数字密码。

登录界面的要求是，第一次启动程序注册密码，这时需要输入两次密码，第二次确定之前的密码，需要和之前的输入一致。然后该密码会利用SharedPreferences存储，之后再进入程序时，则只需要输入密码，不需确认密码，即只会显示一个输入框。

```
//如果注册过，则将一个输入框置为不可见并改变hint
if (!TextUtils.isEmpty(this.sp.getString("Password", null)))
{
    this.is_registed = true;
    this.newpassword.setVisibility(View.INVISIBLE);
    this.confirmpassword.setHint("Password");
}
```

这里sp为SharedPreferences类型变量，存储之前保存的密码。若不为空，则将一个密码输入框利用setVisibility置为不可见，并将另一个输入框的hint设为Password。is_registed是boolean类型的变量，用来表示是否已经注册，用于后面判断。

为了存储密码，需要定义SharedPreferences变量和SharedPreferences.Editor变量进行存储。

```
private SharedPreferences.Editor edit;
private SharedPreferences sp;
...
this.sp = getSharedPreferences("Password", 0);

this.edit = this.sp.edit();
```

getSharedPreferences方法获取对象，并将名称和访问模式作为参数传入。这里的0为私有模式。

然后对ok按钮设置监听器设置其点击的触发事件处理。
首先获取输入框的输入内容，然后判断是否已经注册过，若注册过，则之间和存储的密码进行判断是否相等，若相等，则直接跳转到文件界面；若尚未注册过，即第一次安装程序，则判断输入是否合法，即是否有空输入，两次输入是否相等，若都正确，则利用edit保存当前密码，并跳转到文件界面。

```
public void ok(View v){
    String str1 = MainActivity.this.newpassword.getText().toString();
    String str2 = MainActivity.this.confirmpassword.getText().toString();
    //如果已经注册过,直接登录跳转
    if (MainActivity.this.is_registed){
        if (str2.equalsIgnoreCase(MainActivity.this.sp.getString("Password", null)))
        {
            Intent localIntent = new Intent();
            localIntent.setClass(MainActivity.this, FileEdit.class);
            MainActivity.this.startActivity(localIntent);
        }
    else{
        Toast.makeText(MainActivity.this.getApplicationContext(), "Password Mismatch",
Toast.LENGTH_SHORT).show();
    }
    }
    //否则进行注册保存密码并跳转
    else
    {
        if ((TextUtils.isEmpty(str1)) || (TextUtils.isEmpty(str2)))
        {
            Toast.makeText(MainActivity.this.getApplicationContext(), "Password cannot be
empty", Toast.LENGTH_SHORT).show();
        }
        else if (!str1.equalsIgnoreCase(str2))
        {
            Toast.makeText(MainActivity.this.getApplicationContext(), "Password Mismatch",
Toast.LENGTH_SHORT).show();
        }
        else
        {
            MainActivity.this.edit.putString("Password", str1);
            MainActivity.this.edit.commit();
            Intent localIntent1 = new Intent();
            localIntent1.setClass(MainActivity.this, FileEdit.class);
            MainActivity.this.startActivity(localIntent1);
        }
    }
}
```

这里我直接定义的onClick属性进行监听。

然后还有一个清楚clear功能，即把输入内容清空，这里直接调用输入框的setText方法将内容设置为空""即可。通用定义onClick属性实现监听。

```
public void clear(View v){
    MainActivity.this.newpassword.setText("");
    MainActivity.this.confirmpassword.setText("");
}
```

文件界面：
文件的存储我们利用文件流实现，文件的保存使用FileOutputStream文件输出流变量调用openFileOutput方法进行文件的保存；文件的读取使用FileInputStream文件输入流变量调用openFileInput方法实现文件的读取。删除文件直接调用deleteFile方法即可。

为了方便使用，我们为每个button定义一个函数来实现对应功能。

保存： 首先定义FileOutputStream变量，调用openFileOutput方法，将输入的文件名作为文件名称，这里也使用私有模式。然后将内容框中的内容通过write函数写入，这里要注意的是调用write方法写入是要将获取的string变量利用getBytes函数转化成当前默认编码的字节数组，然后再存入。所以在后面读取的时候也要先利用byte数组将内容读出，再转化成string返回。最后要记得调用close函数关闭文件流。

```
//保存
public void saveContent(String paramString1, String paramString2)
{
    try
    {
        FileOutputStream localFileOutputStream =
getApplicationContext().openFileOutput(paramString1, 0);
        localFileOutputStream.write(paramString2.getBytes());
        localFileOutputStream.close();
        Toast.makeText(getApplicationContext(), "Sava succesfully", Toast.LENGTH_SHORT).show();
        return;
    }
    catch (IOException localIOException)
    {
        while (true)
        {
            Toast.makeText(getApplicationContext(), "Fail to save file",
Toast.LENGTH_SHORT).show();
            localIOException.printStackTrace();
        }
    }
}
```

读取： 读取文件时利用文件输入流变量FileInputStream调用openFileInput方法进行读取，然后利用read函数读取到byte数组中。因为之前保存时只能将string转化成byte数组存储，所以读取的时候也要先读出byte数组，然后再转化成string。

```
//读取
public String getContent(String paramString)
{
    try
    {
        FileInputStream localFileInputStream =
getApplicationContext().openFileInput(paramString);
        byte[] arrayOfByte = new byte[localFileInputStream.available()];
        localFileInputStream.read(arrayOfByte);
        localFileInputStream.close();
        Toast.makeText(getApplicationContext(), "Load succesfully", Toast.LENGTH_SHORT).show();
        String str = new String(arrayOfByte);
        return str;
    }
    catch (IOException localIOException)
    {

            Toast.makeText(getApplicationContext(), "Fail to load file",
Toast.LENGTH_SHORT).show();
            localIOException.printStackTrace();
            String str = new String("");
            return str;
    }
}
```

删除： 删除文件直接调用deleteFile方法，将要删除的文件名作为参数传进去，即可删除。

```
//删除
public void DeleteFile(String paramString)
{
    getApplicationContext().deleteFile(paramString);
    Toast.makeText(getApplicationContext(), "Delete succesfully", Toast.LENGTH_SHORT).show();
}
```

然后在监听器中调用对应方法时，直接利用当前Activity调用对应方法即可。

```
//save按钮监听器
public void save(View v){
    FileEdit.this.saveContent(FileEdit.this.fileName.getText().toString(),
FileEdit.this.fileText.getText().toString());
}
//load按钮监听器
public void load(View v){
    String str = FileEdit.this.getContent(FileEdit.this.fileName.getText().toString());
    FileEdit.this.fileText.setText(str);
}
//delete按钮监听器
public void delete(View v){
    FileEdit.this.DeleteFile(FileEdit.this.fileName.getText().toString());
}
//clear按钮监听器
public void cle(View v){
    FileEdit.this.fileText.setText("");
}
```

最后清除按钮直接将内容框setText为空即可。

其他问题：
点击返回键返回HOME:
项目要求登录到文件界面后点击返回键直接退出程序，即不会再返回输入密码的登录界面。这个功能通过在AndroidManifest文件中设置登录界面Activity的noHistory属性实现。将其noHistory属性设为true之后，对应Activity就会被从Activity栈中移除，即此时再点返回键就会返回到再上一层，即退出程序。

EditText占据指定大小空间：
这里通过设置layout_weight属性实现，但是要注意的是该属性只能在线性布局使用，所以文件界面的布局要用线性布局写。将其他控件设置好后，对内容输入框设置layout_weight值为1，并将高度设为0dp即可。

```
<EditText
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:hint="File Content Here"
    android:id="@+id/content"
    android:layout_weight="1"
    android:gravity="top|left"/>
```

内容输入框置顶左对齐：
这里只要在内容框添加gravity属性，设置为top|left即可。

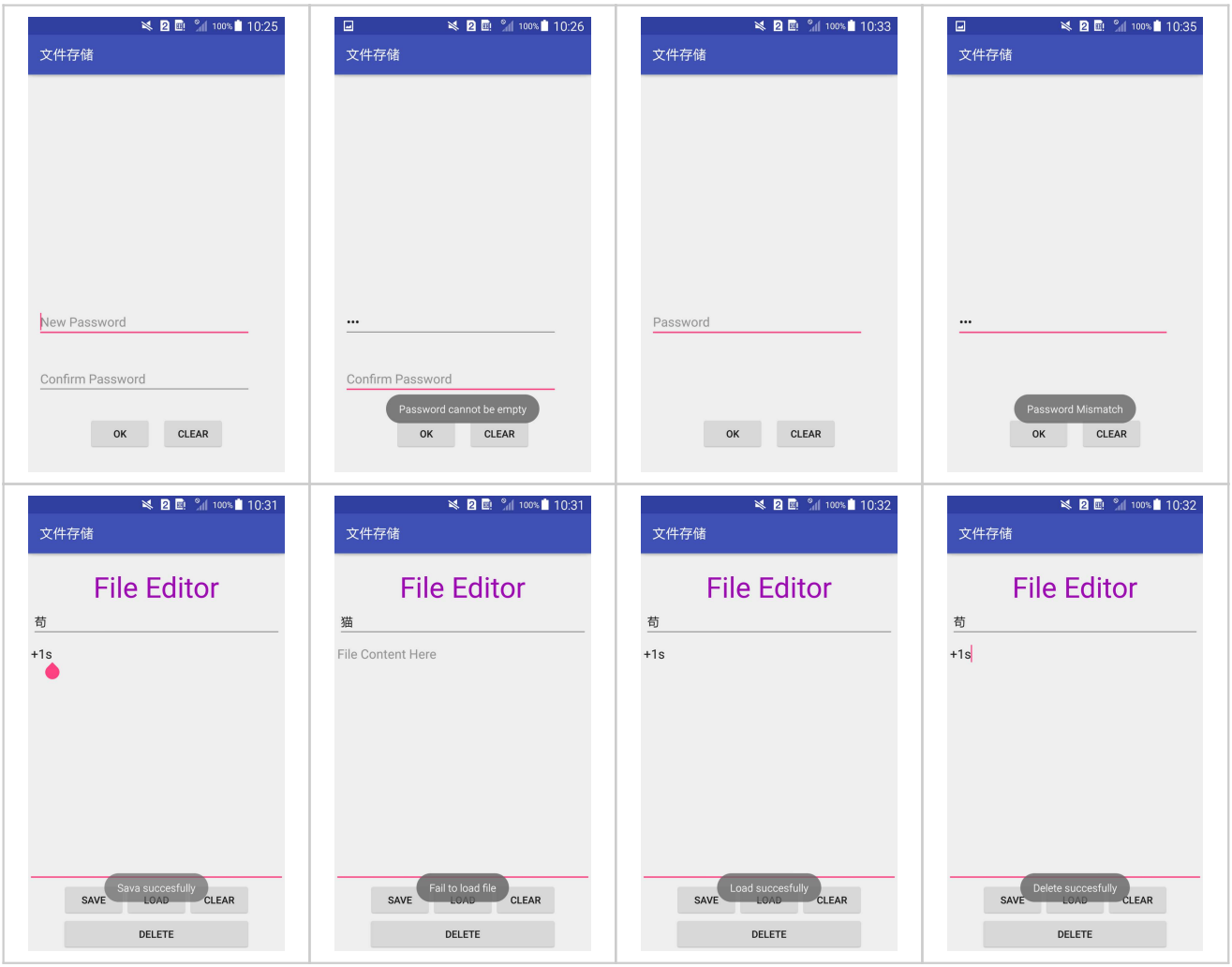
关于internal storage 和 external storage：
本次实验用到的是关于internal storage存储。

> Internal Storage 把数据存储在设备内部存储器上，存储在/data/data//files目录下。 默认情况下在这里存储的数据为应用程序的私有数据，其它应用程序不能访问。 卸载应用程序后，内部存储器的/data/data/目录及其下子目录和文件一同被删除。

而Android中还有一种存储空间为external storage，其与internal storage的区别为：

> Internal storage 是属于应用程序的，文件管理器看不见。
> External storage 在文件浏览器里是可以看见的 /mnt 。
> 这两个概念都是相对于应用来说的，应该理解为逻辑上的概念,不应理解为物理上的外部SD卡和手机或移动设备内存。
> 一个应用把数据存在external storage上时,那么数据成为共有的,所有人都可见的和可用的。 存在internal storage上时,只有这个应用本身可以看到和使用。 很多没有插ＳＤ卡的设备,系统会虚拟出一部分存储空间用来做公共存储(主要是音乐,文档之类的media)。

运行结果：



实验总结

实验过程中遇到的问题：

1. 界面布局方面。这次实验要求在文件界面的内容输入框占据除其他控件外的剩余控件，一开始知道是要用weight写，但是不知道要怎么分配，因为weight是所占比重，将内容框设为1的话其他控件怎么设。后来试了一下，只要在内容输入框设为1后其就会自动占据所有剩余空间，不需要写其他控件的weight属性。
2. SharedPreferences存储方式太轻量级。虽然说期中project用到的SQLite是轻量级数据库，但这里的SharedPreferences更轻量级，只能存储一些简单的属性设置，或者这里的自动登录保存账号密码，不能存储更多的东西。感觉有点鸡肋，可能是效率上快，所以一般就只用在这些地方。使用方法也比较简单，存一些简单的东西还是挺方便的。
3. 文件流的存写。一开始把文件的输入输出流搞混了，认为输出流是读的，因为是要读出来，输入流是写的，因为是写入。然后才发现正好是反过来的。还有一个问题是保存写入时要先把string转化成字节数组，读的时

候也要先读字节数组再转化成string。这应该是内部存储结构等方面的原因，如我们在期中project上传图片保存到数据库时也要先转化成字节数组再保存。

4. 异常的处理。文件存储时我们通过异常处理来显示错误信息，即当抛出异常时即文件操作出了错误，如读取了不存在的文件，此时在catch异常时弹出错误信息，并将内容框setText为空。

本次实验实现简单的一些数据存储，利用SharedPreferences和文件流存储。感觉还是比较简单的，毕竟之前project用数据库存储数据比这两种方法麻烦一点。不过数据库存储除了在第一次安装时要等一段时间，而且以后进行操作时也不如这里顺畅，毕竟更轻量级。所以对于一些简单的数据存储，如自动登录的账号密码存储，用SharedPreferences就很方便。这些存储方法也是各有利弊的，所以要结合起来用才能使其功能最大化。