2018

Android 移动开发实战

程源

广东机电职业技术学院

2018/9/1

目录

3 实现用户注册界面

3.1 本章目标

本章我们将升级 NEC Vocab 应用，为其添加第二个 activity—用户注册界面。LoginActivity 控制着当前的"登录"界面，用户可以通过本讲将要添加的 RegisterActivity 设置用户名、手机号码和密码，并将其存储在 SharedPreferences 中。本章结束时，我们应该可以看到如图 3-1 的运行效果。

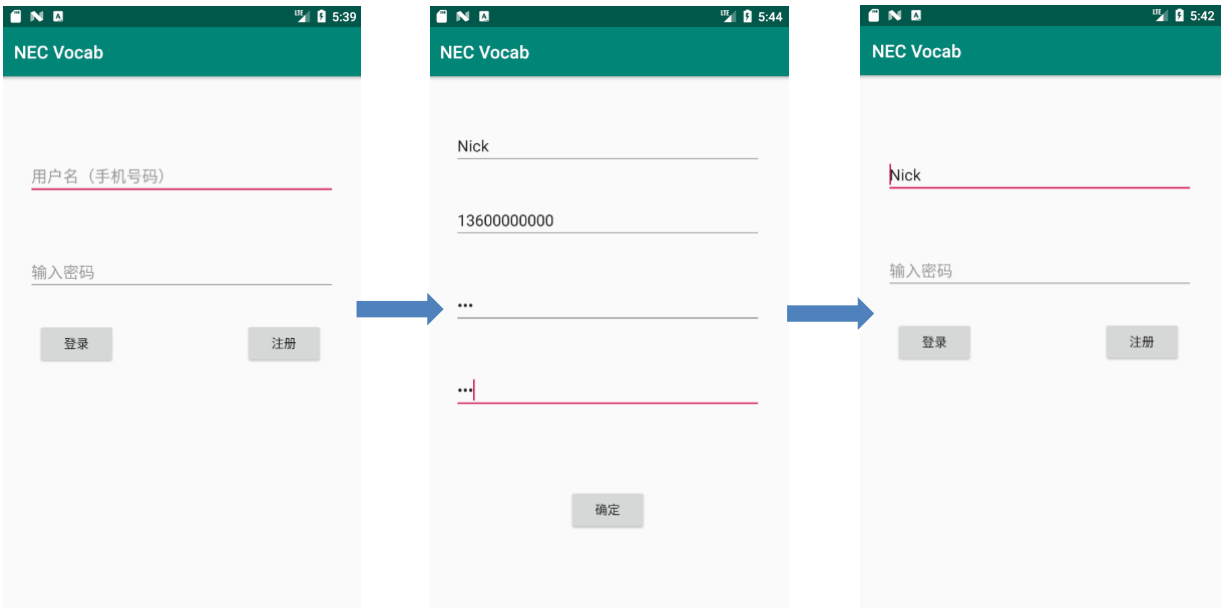

图 3-1 本章结束时的运行效果

通过本讲对 NEC Vocab 应用的升级，我们可以从中学到如下知识点：

- 为应用创建一个新的 activity 和配套布局；
- 从一个 activity 中启动另一个 activity；
- 在父 activity（启动方）与子 activity（被启动方）之间进行数据传递；
- 利用 SharedPreferences 存储轻量级数据；
- 创建消息提示。

3.2 创建 RegisterActivity

为实现应用能够注册的目标，需要为 NEC Vocab 项目新增一个 RegisterActivity。

3.2.1 创建新 Activity

在 Peoject 视图中展开 **NECVocab→app→src→main→java**，选中包 com.studio.aime.necvocab，点击右键，在弹出窗口选择 **New→Activity→Empty Activity**，再在随后的弹出窗口的 **Activity Name:**后面填写 RegisterActivity，其余保持默认，点击 **Finish** 按钮。Android Studio 同时也为我们创建了一个名为 activity_register 的布局文件。

3.2.2 在 manifest 配置文件中声明 activity

事实上，New Activity 向导也为我们完成了 manifest 的配置工作。为更好地理解 Android 应用，我们在这里介绍一下 manifest。

manifest 配置文件是一个包含元数据的 XML 文件，用来向 Android 操作系统描述应用。该文件总是以 AndroidMainfest.xml 命名，可以在 res/目录中找到。AndroidManifest.xml 是每个 Android 程序中必须的文件，位于整个项目的根目录下，描述了 package 中组件，如 Activities、Services 和 Content Providers 等，以及它们各自的实现类，各种能被处理的数据和启动位置。

NEC Vocab 应用目前的 AndroidMainfest 源码如图 3-2 所示。

第 3 行的 package="com.studio.aime.necvocab"表示整个 java 应用程序的主要包名，其中 necvocab 的是我们新建的工程名。

第 6 行的 android:icon="@drawable/ic_launcher 表示应用程序的 LOGO 图片。

第 7 行的 android:label="@string/app_name"，这里显示的是 NEC Vocab，表示的是应用程序的名称。

第 11 行的 android:name=" .LoginActivity "表示整个应用程序的主程序的名称。

第 12 行的<intent-filter>是意图过滤器，用来过滤掉用户的一些动作和操作。

第 13 行的<action android:name="android.intent.action.MAIN" />指明了整个工程的入口程序。

第 15 行的<category android:name="android.intent.category.LAUNCHER" />用来决定应用程

序是否在程序列表中进行显示。

第 18 行的 android:name=".RegisterActivity" 则是 New Activity 向导添加的声明 RegisterActivity 部分。这里 **android:name** 属性是必须的。属性前面的“.”告诉操作系统：在 manifest 配置文件的头部包属性值指定的包路径下（com.studio.aime.necvocab）可以找到 activity 的类文件。

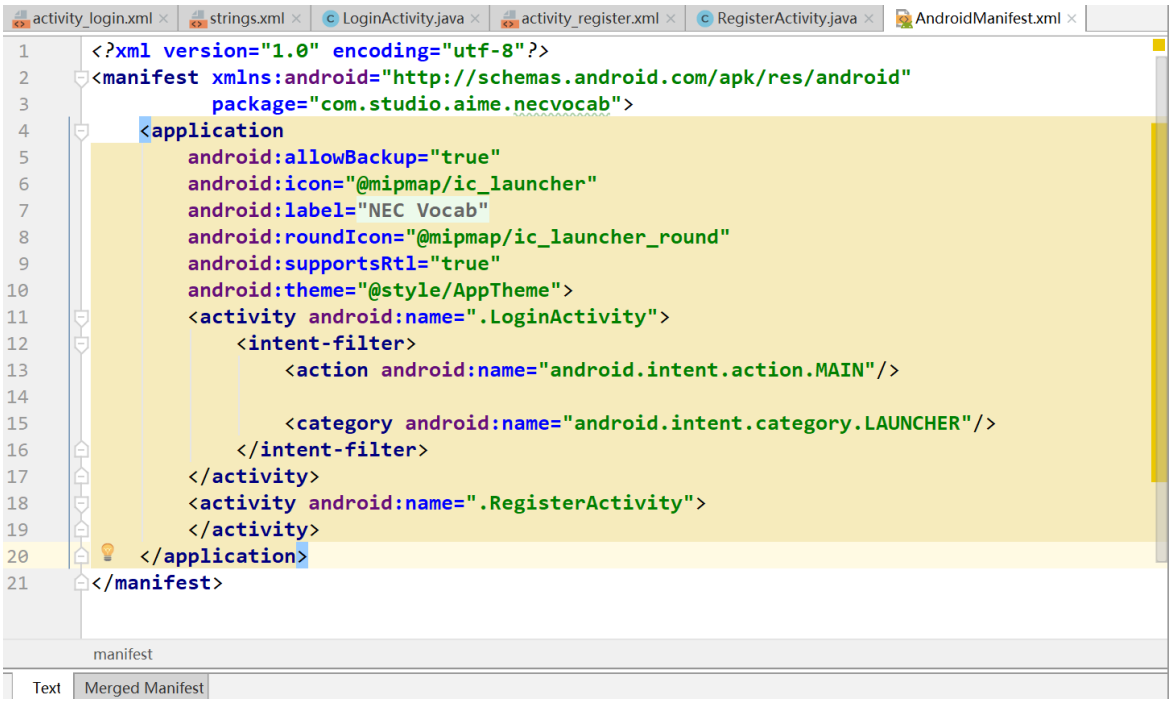

图 3-2 NEC Vocab 目前的 AndroidMainfest 文件

应用的所有 activity 都必须在 manifest 中声明，这样操作系统才能够使用它。

创建 LoginActivity 时，因使用了 **New Application**（新建应用）向导，向导已自动完成了声明工作。同样，**New Activity** 向导也会自动声明 RegisterActivity。

在 manifest 配置文件里还有很多重要内容，但我们现在先集中精力配置好 RegisterActivity，并使其运行起来。在后续章节中，我们将学习到更多有关 manifest 配置文件的知识。

3.2.3 为 RegisterActivity 添加组件

按照开发设想，用户在 LoginActivity 用户界面上点击“注册”按钮，应该立即产生进入 RegisterActivity 实例，并显示其用户界面。用户注册界面至少需要能录入“输入用户名”、“输入手机号码”、“输入密码”和“再次输入密码”的四个 EditText 组件和一个完成“注册”的 Button 组件。

为此我们首先更新字符串资源。如代码清单 3-1 所示，我们在 strings.xml 文件添加应用布

局需要的三个新的字符串。

代码清单 3-1 添加字符串资源（strings.xml）

```
<resources>
    <string name="app_name">NEC Vocab</string>
    <string name="username">用户名（手机号码）</string>
    <string name="mobile">输入手机号码</string>
    <string name="password">输入密码</string>
    <string name="repassword">再次输入密码</string>
    <string name="login">登录</string>
    <string name="register">注册</string>
    <string name="OK">确定</string>
</resources>
```

接下来，如图 3-3 所示，修改默认的布局，采用与第 2.3.4 节完全相同的方法添加 EditText 和 Button 组件，并为其设置好约束。

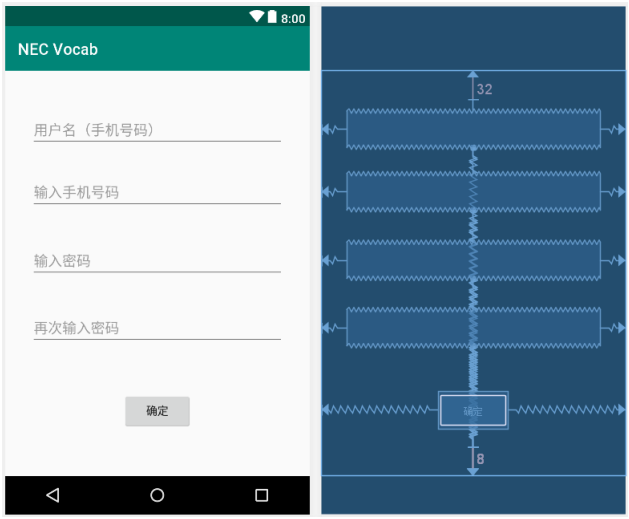

图 3-3 “注册”界面的布局文件

修改它们的 ID、layout_width、hint 和 text 等，并用和 2.3.5 节完全相同的方法为这些组件添加字符串资源。

设置完成后，选择编辑区的“**Text**”标签页，查看 activity_register.xml 文件，所含的**+id** 和 **@string** 应该与代码清单 3-2 相似。

代码清单 3-2 “注册”界面布局参考代码（activity_register.xml）

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

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools=http://schemas.android.com/tools
    ......
    tools:context=".RegisterActivity">

    <EditText
        android:id="@+id/username"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        ......
        android:hint="@string/username"
        android:inputType="textPersonName"
        ...../>

    <EditText
        android:id="@+id/mobile"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        ......
        android:hint="@string/mobile"
        android:inputType="phone"
        ...../>

    <EditText
        android:id="@+id/password"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        ......
        android:hint="@string/password"
        android:inputType="textPassword"
        ...../>

    <EditText
        android:id="@+id/repassword "
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        ......
        android:hint="@string/repassword"
        android:inputType="textPassword"
        ...../>

    <Button
        android:id="@+id/ok_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        ......
        android:text="@string/OK"
        ...../>
</android.support.constraint.ConstraintLayout>
```

可以看到，@string/username、@string/mobile、@string/password、@string/repassword 和

@string/OK 都是已经在 strings.xml 文件中定义好的，这里直接引用即可。

下面来看看如何启动 RegisterActivity。

3.3 启动 RegisterActivity

从一个 activity 启动另一个 activity 最简单的方法是调用如下 Activity 方法：

```
public void startActivity(Intent intent)
```

我们可能以为 startActivity(Intent intent)是一个静态（static）方法，启动 activity 就是针对 Activity 子类调用该方法。实际上并非如此，**activity** 调用 **startActivity(…)**方法时，调用请求实际上发给了**操作系统**。

准确地说，**该方法发送的请求是发送给操作系统的 ActivityManager 的**。ActivityManager 负责创建 Activity 实例并调用其 onCreate(…)方法。activity 的启动示意如图 3-4 所示。

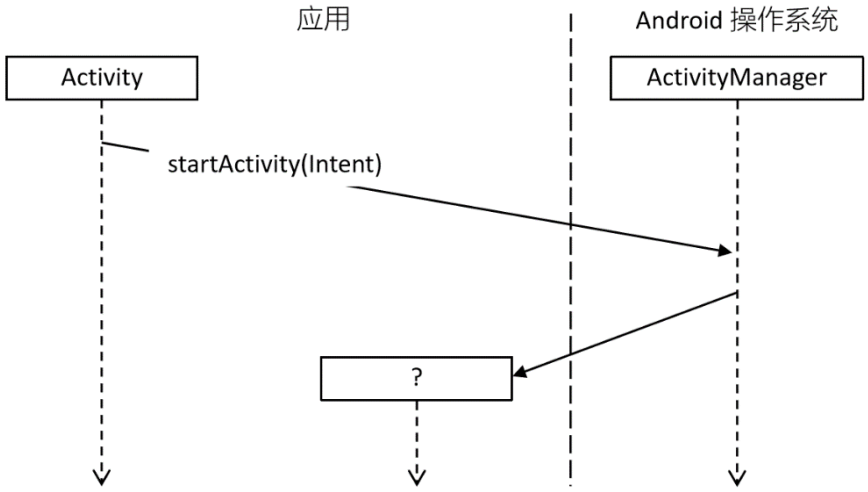

图 3-4 启动一个 activity

那么 ActivityManager 是如何知道应该启动哪一个 activity 呢？

答案就在于传入 startActivity(…)中的 **Intent** 参数。

3.3.1 基于 Intent 的通信

Intent 对象是 component 用来与操作系统通信的一种媒介工具。到目前为止，我们见过的唯一的 component 就是 activity。实际上还有其他一些 component，如 service、broadcast receiver

以及 content provider 等。

Intent 是一种多功能的通信工具。Intent 类提供了多个构造方法以满足不同的使用需求。

在 NEC Vocab 应用中，我们使用 **intent** 告知 **ActivityManager** 应该启动哪一个 **activity**。因此可以使用以下构造方法：

```
public Intent(Context packageContext, Class<?> class)
```

传入该方法的 Class 对象指定 ActivityManager 应该启动的 activity；Context 对象告知 ActivityManager 在哪一个包里可以找到该 Class 对象，如图 3-5 所示。

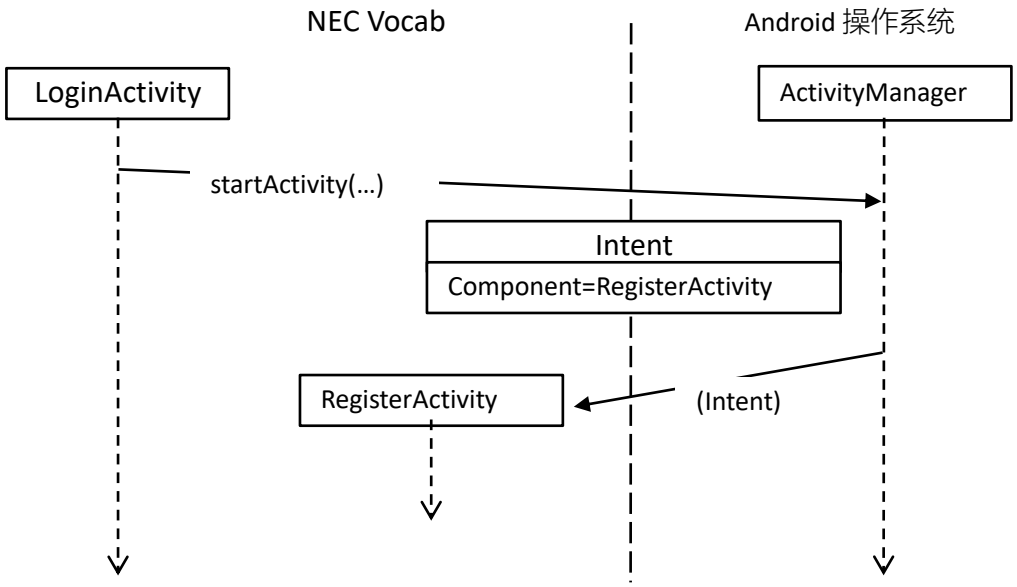

图 3-5 intent：ActivityManager 的信使

下面，我们将在 LoginActivity 中设置监听器，以便检测我们手指的点击事件，并在监听器中创建包含 RegisterActivity 类的 Intent 实例，然后将其传入 startActivity(Intent)方法中。如代码清单 3-3 所示。

代码清单 3-3 启动 RegisterActivity（LoginActivity.java）

```
public class LoginActivity extends AppCompatActivity {
    private EditText mUsername;
    private EditText mPassword;
    private Button mLogin;
    private Button mRegister;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        mUsername = findViewById(R.id.username);
```

```
            mPassword = findViewById(R.id.password);
            mLogin= findViewById(R.id.login);
            mRegister = findViewById(R.id.register);
            mLogin.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    //目前什么都不做
                }
            });
            mRegister.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    //启动 Register Activity
                    Intent intent = new Intent(LoginActivity.this, RegisterActivity.class);
                    startActivity(intent);
                }
            });
        }
    }
```

在启动 activity 以前，ActivityManager 会检查确认指定的 Class 是否已经在配置文件 manifest 中声明。如果已经完成声明，则启动 activity，应用将正常运行。否则，就会抛出 ActivityNotFoundException 异常。这就是我们必须在 manifest 配置文件中声明应用的所有 activity 的原因。

下面运行 NEC Vocab 应用，单击“注册”按钮，屏幕上将会显示新 activity 实例的用户界面（图 3-6）。单击“**后退**”按钮，RegisterActivity 实例会被销毁，继而返回到 LoginActivity 实例的用户界面。

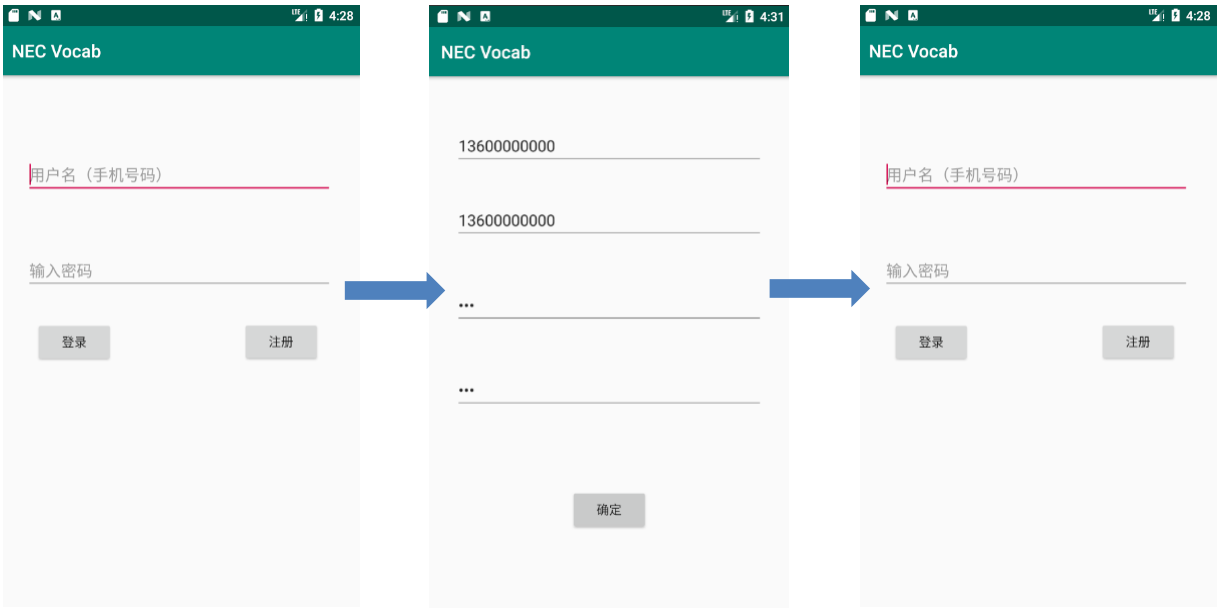

图 3-6 启动第二个 Activity

当然，我们希望用户在“注册”界面填入相关信息之后，点击“**确定**”按钮，能够返回“登录”界面，同时“登录”界面能够直接显示刚才注册的“用户名”，用户只需输入“密码”即可进行登录。

为了这一改进，我们还需要引入一个新的概念——Activity 间的数据传递。

3.4 在 Activity 间传递数据

我们还需要为 RegisterActivity 添加四个 EditText 组件和一个用于注册的 Button 组件，并设置监听器，并通过 Intent 启动 LoginActivity，如代码清单 3-4 所示。

代码清单 **3-4** 为 **RegisterActivity** 设置组件和监听器（**RegisterActivity.java**）

```
public class RegisterActivity extends AppCompatActivity {
    private EditText mUserName;
    private EditText mMobile;
    private EditText mPassword;
    private EditText mRePassword;
    private Button mRegister;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        mUserName = findViewById(R.id.username_register);
        mMobile = findViewById(R.id.mobile);
        mPassword = findViewById(R.id.password);
        mRePassword = findViewById(R.id.repassword);
        mRegister = findViewById(R.id.ok_button);

        mRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //启动 Login Activity
                Intent intent = new Intent(RegisterActivity.this, LoginActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

由于所添加的代码和 LoginActivity.java 中的代码类似，这里不再赘述。现在运行 NEC Vocab 应用，形式上和图 3-6 完全相同，只是现在点击“**确定**”按钮，即可返回“登录”界面。

既然 LoginActivity 和 RegisterActivity 都已经准备就绪，接下来就可以考虑它们之间的数据传递了。

图 3-7 展示了 LoginActivity 和 RegisterActivity 两个 activity 间传递的数据信息。LoginActivity 启动后，RegisterActivity 会将当前注册的用户名通知给它。

用户注册后，单击"**确定**"键回到 LoginActivity，RegisterActivity 随即被销毁。在 RegisterActivity 被销毁前的瞬间，它需要将用户注册的"用户名"数据传递给 LoginActivity。

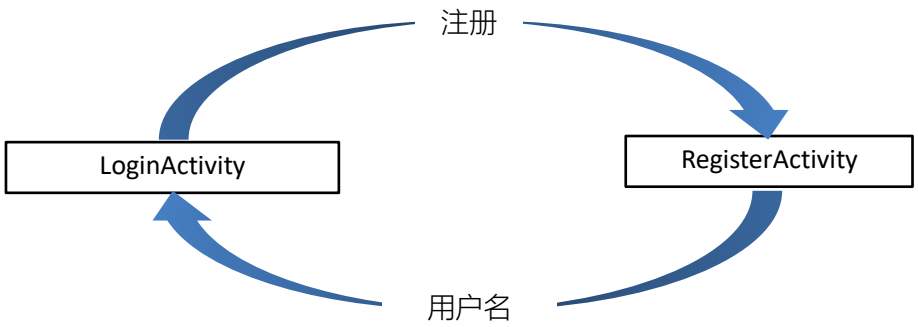

图 3-7 LoginActivity 与 RegisterActivity 之间的对话

下面我们学习如何将数据从 RegisterActivity 传递到 LoginActivity。

3.4.1 使用 intent extra

前面 Intent 通过指定 Context 与 Class 对象启动了 LoginActivity。如何将当前"用户名"通知给 LoginActivity？我们可以将"用户名"值作为 extra（附加）信息附加在 startActivity(Intent)方法的 Intent 上发送出去。extra 与 activity 间的通信与数据传递关系图如图 3-8 所示。

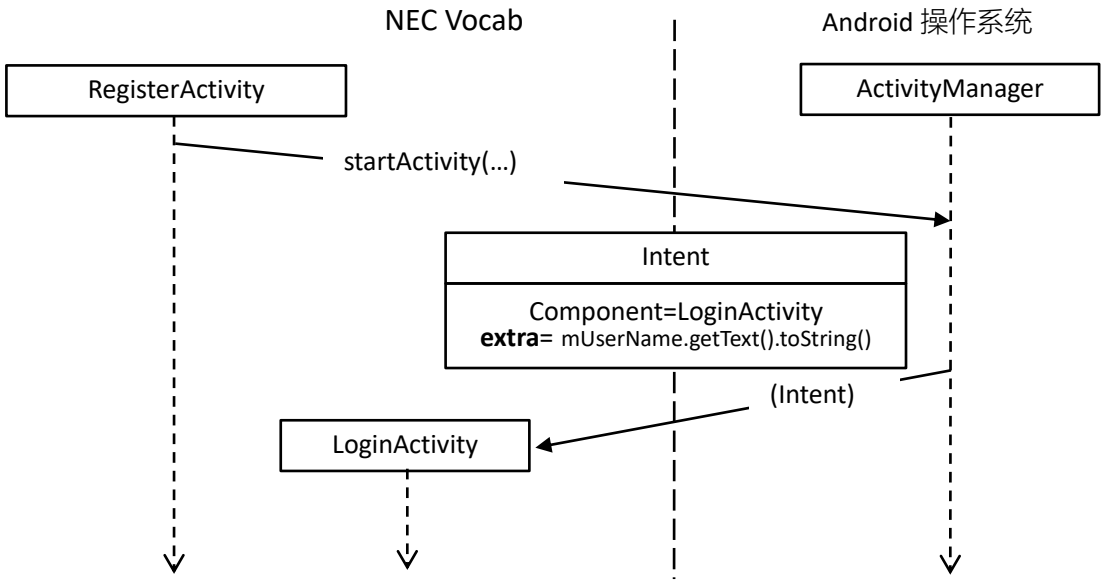

图 3-8 Intent extra：activity 间的通信与数据传递

extra 信息可以是任意数据，它包含在 Intent 中，由启动方发送出去。接收方 activity 接收到操作系统转发的 Intent 后，访问并获取包含在其中的 extra 数据信息。为了将 extra 数据信息添加给 intent，Intent 类为我们提供了 Intent.putExtra(...)方法：

```
public Intent putExtra(String name, String value)
```

尽管 Intent.putExtra(...)方法有多种形式，但它只有两个参数。一个参数是固定为 String 类型的 key，另一个参数值可以是多种数据类型。

首先在 LoginActivity.java 中，为 extra 数据信息新增 key-value 对中的 key，如代码清单 3-5 所示。

代码清单 3-5 添加 extra 常数（LoginActivity.java）

```
public class LoginActivity extends AppCompatActivity {
    public static final String USERNAME =
            "com.studio.aime.necvocab.username";
.....
```

Activity 可能在不同地方启动，我们应该为 activity 获取和使用的 extra 定义 key。在代码清单 3-5 中，我们使用包名来修饰 extra 数据信息，这样可以避免来自不同应用的 extra 间发生命名冲突。

接下来要将 extra 附加到 intent 上。考虑到 RegisterActivity 没有理由知道 LoginActivity 需要通过其 Intent 中的 extra 传送什么信息。因此我们将其封装在一个称为 newIntent(...) 方法中。

下面在 LoginActivity 中创建该方法。

代码清单 3-6 在 LoginActivity 中创建 newIntent(...)方法（LoginActivity.java）

```
public class LoginActivity extends AppCompatActivity {
    public static final String USERNAME =
            "com.studio.aime.necvocab.username";
    public static Intent newIntent(Context packageContext, String username){
        Intent intent = new Intent(packageContext, LoginActivity.class);
        intent.putExtra(USERNAME, username);
        return intent;
    }
.....
```

这个静态方法使我们能正确地创建一个与 LoginActivity 需要相配的 Intent。这里 String 类型参数 username 作为键值对的值，USERNAME 作为键值对的键被放入 Intent。使用 newIntent(...)方法，activity 子类将很容易地配置其发送的 intent。

现在更新 RegisterActivity 中的 onClick（.....）方法，使用 newIntent 方法传递“用户名”，如

代码清单 3-7 所示。

代码清单 **3-7** 传递一个"用户名"（**RegisterActivity.java**）

```
public class RegisterActivity extends AppCompatActivity {
    ……
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        ……
        mRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(RegisterActivity.this, LoginActivity.class);
                Intent intent = LoginActivity.newIntent(RegisterActivity.this,
                        mUserName.getText().toString().trim());
                startActivity(intent);
            }
        });
    }
}
```

这里 trim()方法返回调用字符串对象的一个副本，但是所有起始和结尾的空格都被删除了，例如：

```
String s="    Hello World    ".trim();
```

就是把"Hello World"放入 s 中。

代码清单 3-7 中只有一个 extra。但如果有需要，也可以附加多个 extra 到同一个 Intent 上，形式上与前面相同，只是添加更多的参数到 newIntent(…)方法中。

3.4.2 获取 extra 信息

我们已经利用 Intent 从 RegisterActivity 启动了 LoginActivity，并通过 extra 携带了"用户名"信息。那么 LoginActivity 是如何获取 extra 的信息的呢？

要从 extra 获取数据，需要用到 getIntent().getStringExtra(String name)方法。（实际上 Intent 有多种获取 extra 数据的方法，我们可以用 getXXXXExtra(…) 表示，这里 XXXX 代表数据类型，可以是 Boolean、Int、String 等。）

在 LoginActivity 代码的 onCreate(Bundle)方法中插入代码实现从 extra 中获取信息，然后将信息存入成员变量中，如代码清单 3-8 所示。

代码清单 **3-8** 获取 **extra** 信息（**LoginActivity.java**）

```
……
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    ……
    mUsername.setText(getIntent().getStringExtra(USERNAME));
……
```

在代码清单 3-8 中，getIntent()方法返回来自启动 LoginActivity 的 Intent。然后调用 Intent 的 getStringExtra(String)方法获取"用户名"并存入 mUsername.setText 中。

现在运行 NEC Vocab 应用，应该可以看到如图 3-9 的运行效果。

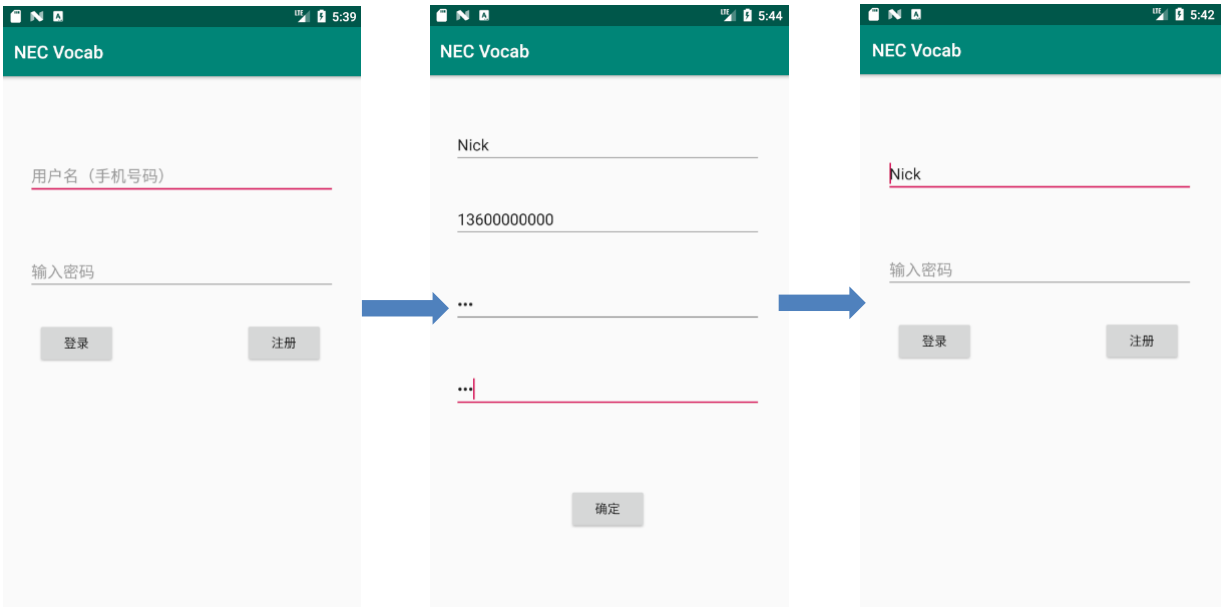

图 3-9 "登录"界面显示"用户名"信息

3.5 使用 SharedPreferences 存放数据

目前我们虽然可以从 LoginActivity 启动 RegisterActivity，并且能够在点击"确定"键后返回 LoginActivity，通过将注册的用户名传递给 LoginActivity。但我们的"注册"信息并没有保存，当我们离开"注册"界面后，这些信息将会丢失。

下面我们来完善 RegisterActivity，我们将使用 SharedPreferences 来保存注册信息。

SharedPreferences 是 Android 平台上一个轻量级的存储类，用来保存应用的一些常用配置。其原理是通过 Android 系统生成一个 xml 文件保到：/data/data/包名/shared_prefs 目录下，用

类似**键-值对**的方式来存储数据。SharedPreferences 提供了常规的 Long、Int、String 等类型数据的保存接口。

3.5.1 SharedPreferences 介绍

存储数据

保存数据一般分为四个步骤：

- 使用 Activity 类的 getSharedPreferences 方法获得 SharedPreferences 对象；
- 使用 SharedPreferences 接口的 edit 获得 SharedPreferences.Editor 对象；
- 通过 SharedPreferences.Editor 接口的 putXXX 方法保存 key-value 对；
- 通过过 SharedPreferences.Editor 接口的 commit 方法保存 key-value 对。

读取数据

读取数据一般分为两个步骤：

- 使用 Activity 类的 getSharedPreferences 方法获得 SharedPreferences 对象；
- 通过 SharedPreferences 对象的 getXXX 方法获取数据；

用到的方法

获取 SharedPreferences 对象

（根据 name 查找 SharedPreferences，若已经存在则获取，若不存在则创建一个新的）

```
public abstract SharedPreferences getSharedPreferences (String name, int mode)
```

参数：

name：命名

mode：模式，包括

- MODE_PRIVATE（只能被自己的应用程序访问）
- MODE_WORLD_READABLE（除了自己访问外还可以被其它应该程序读取）
- MODE_WORLD_WRITEABLE（除了自己访问外还可以被其它应该程序读取和写入）

若该 Activity 只需要创建一个 SharedPreferences 对象的时候，可以使用 getPreferences 方法，不需要为 SharedPreferences 对象命名，只要传入参数 mode 即可

```
public SharedPreferences getPreferences (int mode)
```

获取 Editor 对象（由 SharedPreferences 对象调用）

abstract SharedPreferences.Editor edit()

写入数据（由 Editor 对象调用）

```
//写入 boolean 类型的数据
putBoolean(String key, boolean value)
//写入 float 类型的数据
putFloat(String key, float value)
//写入 int 类型的数据
putInt(String key, int value)
//写入 long 类型的数据
putLong(String key, long value)
//写入 String 类型的数据
putString(String key, String value)
//写入 Set<String>类型的数据
putStringSet(String key, Set<String> values)
```

参数

key：指定数据对应的 key

value：指定的值

移除指定 key 的数据（由 Editor 对象调用）

remove(String key)

参数

key：指定数据的 key

清空数据（由 Editor 对象调用）

clear()

提交数据（由 Editor 对象调用）

commit()

读取数据（由 SharedPreferences 对象调用）

```
//读取所有数据
Map<String, ?> getAll()
//读取的数据为 boolean 类型
boolean getBoolean(String key, boolean defValue)
//读取的数据为 float 类型
float getFloat(String key, float defValue)
//读取的数据为 int 类型
int getInt(String key, int defValue)
```

```
//读取的数据为 long 类型
long getLong(String key, long defValue)
//读取的数据为 String 类型
String getString(String key, String defValue)
//读取的数据为 Set<String>类型
Set<String> getStringSet(String key, Set<String> defValues)
```

参数

key：指定数据的 key

defValue：当读取不到指定的数据时，使用的默认值 defValue

说了这么多，下面我们来看看如何在 RegisterActivity 和 LoginActivity 里使用 SharedPreferences。

3.5.2 利用 SharedPreferences 保存数据

首先考虑 RegisterActivity。在 RegisterActivity 里，我们希望用户能够在四个 EditText 组件的输入栏中输入用户名（通常是手机号码）、手机号码、密码以及再次输入密码，当用户点击“确定”按钮时，系统能够将用户的输入信息保存在 SharedPreferences。

修改 RegisterActivity.java 文件，声明一个 SharedPreferences 对象，用于存放用户的“注册”信息，如代码清单 3-9 所示。

代码清单 3-9 声明 SharedPreferences 对象（RegisterActivity.java）

```
public class RegisterActivity extends AppCompatActivity {
    ......
    private SharedPreferences mPreferences;                                    ①
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ......
        mPreferences = getSharedPreferences("register_data", Context.MODE_PRIVATE);   ②
        mRegister = findViewById(R.id.register_bt);
        mRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                switch (v.getId()) {                                           ③
                    case R.id.ok_button:                                       ④
                        SharedPreferences.Editor edit = mPreferences.edit();   ⑤
                        edit.putString("username",mUserName.getText().toString().trim());   ⑥
                        edit.putString("mobile", mMobile.getText().toString().trim());
                        edit.putString("password",mPassword.getText().toString().trim());
                        edit.commit();                                         ⑦
```

```
                    Intent intent = LoginActivity.newIntent(RegisterActivity.this,
                            mUserName.getText().toString().trim());
                    startActivity(intent);
                }
            }
        });
    }
}
```

这里添加的代码比较多，需要进行一些说明。

标记①是声明一个名为 mPreferences 的 SharedPreferences 对象。

标记②创建一个新的 Shared preferences 文件。可以通过如下两个方法中的某个方法来创建一个新的 Shared preferences 文件或者访问一个已经存在的文件

1、getSharedPreferences()，如果应用中有多个 Shared Preferences 文件需要保存，这个方法很适合。第一个参数为你给这个文件指定的 ID，这里使用的是"register_data"，可以通过应用的上下文来调用它。getSharedPreferences 是 Context 类中的方法，可以指定 filename 以及 mode。

2、getPreferences()，如果应用只需要保存一个 Shared Preferences 文件，这个方法很适合。对于只有一个 Shared Preferences 文件的应用，不需要为其指定名称，系统在创建时会默认一个名称。getPreferences 是 Activity 类中的方法，只需指定 mode。

标记③利用 switch 语句通过 v.getId()获得组件 ID。

标记④，当该 ID 为 R.id.ok_btutton，即“确定”按钮时，首先获取一个 SharedPreferences.Editor 对象 edit，见标记⑤。

标记⑥及接下来的两行代码：利用 edit 对象的 putString 方法将用户名、手机号码和密码分别存放在键名为 username、mobile 和 password 的键值对的值中保存。我们这里没有保存“再次输入密码”的信息，因为该信息只用于比较两次密码输入是否相等（将在后面介绍）。

最后不要忘记标记⑦所示的 commit()方法。

当然，在一切完成之后，系统通过 startActivity()返回到“登录”界面。

运行 NEC Vocab 应用，在登录界面点击“注册”，随即进入“注册”界面，输入用户名和密码，点击“确定”，便返回“登录”界面。如图 3-10 所示，一切都还不错。

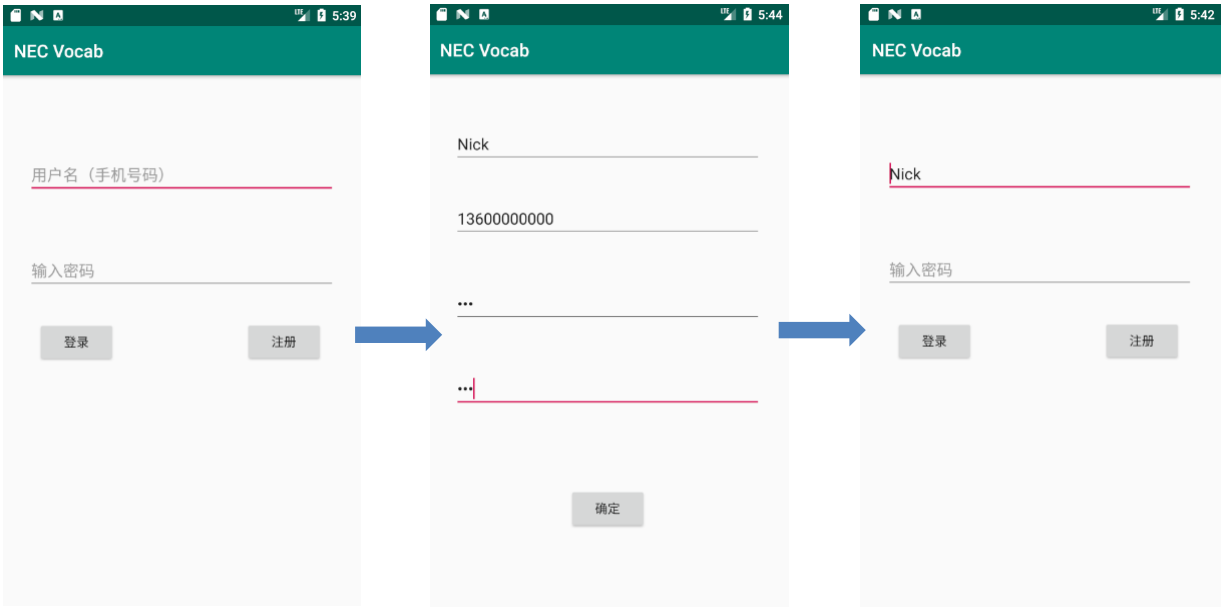

图 3-10 “登录”→“注册”→“登录”

3.5.3 读取 SharedPreferences 数据

下面我们修改 LoginActivity 代码，使其在点击**登录**键时，能将输入的“用户名”和“密码”与已经注册的用户名和密码进行比对。为此，LoginActivity 也需要声明一个 SharedPreferences 对象，并获取 SharedPreferences 对象中存放的“用户名”和“密码”信息，其完整代码如代码清单 3-10 所示。

代码清单 **3-10** 升级 **LoginActivity**（**LoginActivity.java**）

```
public class LoginActivity extends AppCompatActivity {
    public static final String USERNAME =
            "com.studio.aime.necvocab.username";
    public static Intent newIntent(Context packageContext, String username){
        Intent intent = new Intent(packageContext, LoginActivity.class);
        intent.putExtra(USERNAME, username);
        return intent;
    }
    private EditText mUsername;
    private EditText mPassword;
    private Button mLogin;
    private Button mRegister;
    private SharedPreferences mPreferences;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        mUsername = findViewById(R.id.username);
        mPassword = findViewById(R.id.password);
        mLogin= findViewById(R.id.login);
```

```
        mRegister = findViewById(R.id.register);
        mPreferences = getSharedPreferences("register_data", Context.MODE_PRIVATE);①
        mUsername.setText(getIntent().getStringExtra(USERNAME));
        mLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if ((mUsername.getText().toString().trim().equals(
                        mPreferences.getString("username", "")))          ②
                        && (mPassword.getText().toString().trim().equals(
                                mPreferences.getString("password", ""))))
                {
                    //什么都不做
                }else {
                    //什么都不做
                }
            }
        });

        mRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(LoginActivity.this,
                        RegisterActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

这里需要指出的是，标记①的代码与 RegisterActivity 中的相应的代码完全相同，因为 LoginActivity 需要从"register_data"指定的文件中获取数据。标记②中 getString（.....）方法的第一个参数是获取键为"username"的对应的值，第二个参数为省缺值，这里使用空值：""。接下来的"password"与此完全相同。

现在运行 NEC Vocab 应用，看看运行效果。

现在我们虽然能将输入的"用户名"和"密码"与已经注册的用户名和密码进行比对，但还需要给用户一个提示消息：如果完全符合，则提示"登陆成功"，否则提示"用户名或密码不正确"。

3.6 创建提示消息

3.6.1 消息对话框

接下来要实现的就是，分别单击"登录"按钮，弹出我们称为 toast 的提示消息。Android 的

toast 是用来通知用户的简短弹出消息，用户无需输入或进行任何操作。这里，我们要用 toast 来告知用户其"登录"成功与否，如图 3-11 所示。

图 3-11 toast 反馈消息提示

调用来自 Toast 类的以下方法，可创建一个 toast：

```
public static Toast makeText(Context context, int resId, int duration)
```

该方法的 Context 参数通常是 Activity 的一个实例（Activity 本身就是 Context 的子类）。第二个参数是 toast 要显示字符串消息的资源 ID。Toast 类必须借助 Context 才能找到并使用字符串的资源 ID。第三个参数通常是两个 Toast 常量中的一个，用来指定 toast 消息显示的持续时间。 创建 toast 后，可调用 Toast.show()方法在屏幕上显示 toast 消息。在 LoginActivity 代码里，对"登录"按钮的监听器调用 makeText(...)方法，如代码清单 3-11 所示。在添加 makeText(...)时，可利用 Android Studio 的代码自动补全功能，让代码输入更加轻松。

代码清单 3-11 升级 LoginActivity（LoginActivity.java）

```
public class LoginActivity extends AppCompatActivity {
    ......
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ......
        mLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if ((mUsername.getText().toString().trim().equals(
                        mPreferences.getString("username", "")))
                        && (mPassword.getText().toString().trim().equals(
                                mPreferences.getString("password", ""))))
```

```
                {
                    Toast.makeText(LoginActivity.this,
                            "登录成功", Toast.LENGTH_SHORT).show();
                }else {
                    Toast.makeText(LoginActivity.this,
                            "用户名或密码不正确", Toast.LENGTH_SHORT).show();
                }
            }
        });
    ......
    }
```

使用代码自动补全 代码自动补全功能可以节约大量开发时间。参照代码清单 3-11，依次输入代码。当输入到 Toast 类后的点号时，Android Studio 会弹出一个窗口，窗口内显示了建议使用的 Toast 类的常量与方法。 要选择需要的建议方法，使用上下方向键。（如果不想使用代码自动补全功能，请不要按 Tab 键、Return/Enter 键，或使用鼠标点击弹出窗口，只管继续输入代码直至完成。） 在列表建议清单里，选择 makeText(Context context, int resID, int duration) 方法，代码自动补全功能会自动添加完整的方法调用。

3.6.2 TextView 的 setError()方法

注册界面还有个问题：即使什么都没有输入，仍然能完成注册。这当然不是我们所希望的。我们在注册时，比如输入用户名或密码为空时会有提示，对了！就是这些提示：

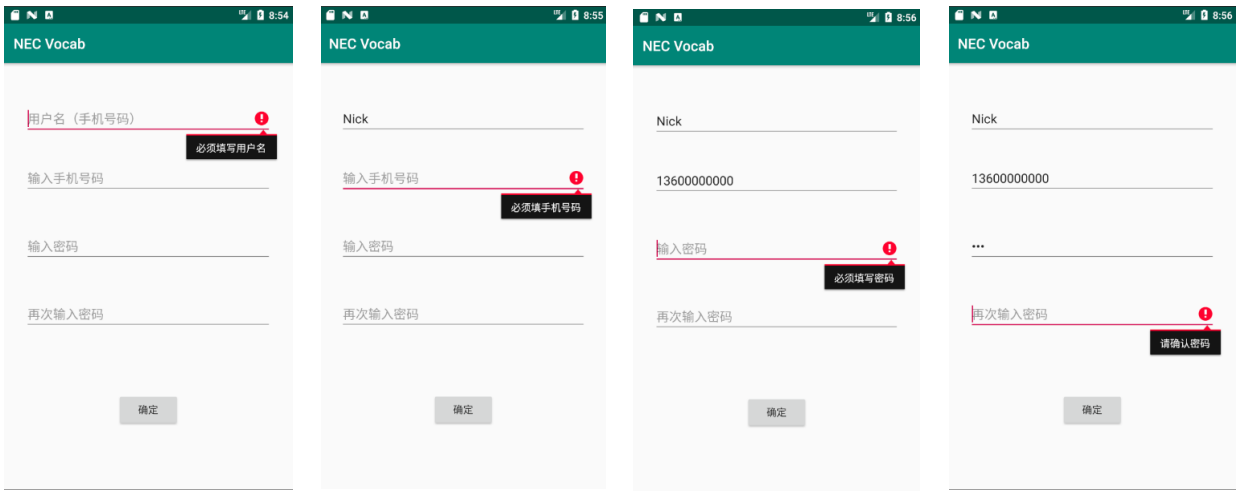

图 3-12 注册时用户输入为空时会有提示

Android EditText 在输入的时候，有很多中类型，如手机号，邮箱，或者数字；那在用户输入的时候，提示错误信息是必须的；为了不占用屏幕空间，而且方便开发者，可以使用 EditText

自带的功能。首先你要对 EditText 的内容检查，在不符合业务的时候提示错误，如提示如下：

```
editText.setError("必须填写用户名");
```

下面我们来看看具体实现。

首先我们在字符串资源里添加错误提示的内容。打开 strings.xml 文件，添加如下代码：

代码清单 3-12 添加字符串资源（strings.xml）

```
<resources>
    ......
    <string name="error_username_required">必须填写用户名</string>
    <string name="error_mobile_required">必须填手机号码</string>
    <string name="error_password_required">必须填写密码</string>
    <string name="error_repassword_required">请确认密码</string>
</resources>
```

接下来，我们来修改 RegisterActivity。如代码清单 3-13 所示。代码比较简单，加上注释，比较好理解。为了便于对比，这里列出了完整的代码。

代码清单 3-13 设置用户输入为空时的提示（RegisterActivity.java）

```
public class RegisterActivity extends AppCompatActivity {
    private static final String TAG = "RegisterActivity";
    private EditText mUserName;
    private EditText mMobile;
    private EditText mPassword;
    private EditText mPasswordCheck;
    private Button mRegister;
    private SharedPreferences mPreferences;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        mUserName = findViewById(R.id.username);
        mMobile = findViewById(R.id.mobile);
        mPassword = findViewById(R.id.password);
        mPasswordCheck = findViewById(R.id.password_check);
        mPreferences = getSharedPreferences("register_data", Context.MODE_PRIVATE);
        mRegister = findViewById(R.id.ok_button);
        mRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                switch (v.getId()) {
                    case R.id.ok_button:
                        if (checkInputAndPermission()) {
                            SharedPreferences.Editor edit = mPreferences.edit();
                            edit.putString("username", mUserName.getText().toString().trim());
                            edit.putString("mobile", mMobile.getText().toString().trim());
                            edit.putString("password", mPassword.getText().toString().trim());
```

```
                                edit.commit();
                                Intent intent = LoginActivity.newIntent(RegisterActivity.this,
                                        mUserName.getText().toString().trim());
                                startActivity(intent);
                            }
                        }
                    }
                });
            }

            private boolean checkInputAndPermission() {
                //检查，重置错误.
                mUserName.setError(null);
                mMobile.setError(null);
                mPassword.setError(null);
                mPasswordCheck.setError(null);
                // 在注册完成后点击确定时尝试存储值.
                String username = mUserName.getText().toString();
                String mobile = mMobile.getText().toString();
                String password = mPassword.getText().toString();
                String passwordCheck = mPasswordCheck.getText().toString();

                boolean cancel = false;
                View focusView = null;

                // 如果用户输入了一个用户名，检查其有效的。.
                if (TextUtils.isEmpty(username)){// && !isPhoneValid(password)) {
                    mUserName.setError(getString(R.string.error_username_required));
                    focusView = mUserName;
                    cancel = true;
                }

                if (TextUtils.isEmpty(mobile)){// && !isPhoneValid(password)) {
                    mMobile.setError(getString(R.string.error_mobile_required));
                    focusView = mMobile;
                    cancel = true;
                }
                // 如果用户输入了一个密码，检查其有效的。.
                if (TextUtils.isEmpty(password)){// && !isPhoneValid(password)) {
                    mPassword.setError(getString(R.string.error_password_required));
                    focusView = mPassword;
                    cancel = true;
                }

                // Check for a valid password, if the user entered one.
                if (TextUtils.isEmpty(passwordCheck)){// && !isPhoneValid(password)) {
                    mPasswordCheck.setError(getString(R.string.error_password_check_required));
                    focusView = mPasswordCheck;
                    cancel = true;
                }

                //判断两次密码输入是否相同，如果不同，提示密码不一样
                if(password.equals(passwordCheck)==false){
                    Toast.makeText(this, getString(R.string.pwd_not_the_same),Toast.LENGTH_SHORT).show();
                    return false;
```

```
            }


            if (cancel) {
                // 如果出现错误；不尝试注册，并将焦点放到第一个错误字段上，返回 FALSE。
                focusView.requestFocus();
                return false;
            } else {
                // 可以在这里添加显示进度条，并启动一个后台任务执行用户这次的代码.
                return true;
            }
        }

}
```

现在运行 NEC Vocab 应用，应该能看到如图 3-12 的输入检查错误提示。