3 注册模块

这一章我们将带领大家在网络环境下完成在蓦然服务器的注册，主要包含以下内容：

- 界面跳转
- MD5
- 多线程
- 网络编程

开发环境还是在Windows平台下，以后就不再做说明了。

3.1 用户界面的跳转

（1）完成界面布局

注册界面的布局和上一章的登录大同小异，只是多了昵称和重复密码两个输入框，除此之外，我们还可以添加一个进度条或进度对话框。

在activity子包下新建注册界面SignUpActivity，页面布局如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/colorBackground"
    tools:context="com.geekband.demo.moran.ui.activity.SignUpActivity">

    <ProgressBar
        android:id="@+id/sign_up_progress"
        style="?android:attr/progressBarStyleLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center|center_horizontal|center_vertical"
        android:visibility="gone"/>

    <ScrollView
        android:id="@+id/signin"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:id="@+id/signin_form"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">

            <FrameLayout
                android:layout_width="match_parent"
                android:layout_height="@dimen/logo_container_height"
                android:background="@color/colorOrange">

                <ImageView
                    android:layout_width="@dimen/logo_width"
                    android:layout_height="@dimen/logo_height"
                    android:src="@drawable/sign_logo"
                    android:layout_gravity="center|center_horizontal|center_vertical"/>
            </FrameLayout>

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="vertical"
```

```
        android:paddingLeft="@dimen/sign_vertical_padding"
        android:paddingRight="@dimen/sign_vertical_padding">


        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="18dp"
            android:layout_marginBottom="@dimen/sign_vertical_margin"
            android:paddingLeft="@dimen/sign_padding_left"
            android:textSize="@dimen/sign_text_size"
            android:textColor="@color/colorGrey"
            android:text="@string/prompt_email"/>

        <AutoCompleteTextView
            android:id="@+id/email"
            android:inputType="textEmailAddress"
            android:layout_width="match_parent"
            android:layout_height="@dimen/sign_view_height"
            android:paddingLeft="@dimen/sign_padding_left"
            android:maxLines="1"
            android:singleLine="true"
            android:background="@drawable/text_shape"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/sign_vertical_margin"
            android:layout_marginBottom="@dimen/sign_vertical_margin"
            android:paddingLeft="@dimen/sign_padding_left"
            android:textSize="@dimen/sign_text_size"
            android:text="@string/prompt_nickname"
            android:textColor="@color/colorGrey"
            />
        <EditText
            android:id="@+id/nickname"
            android:layout_width="match_parent"
            android:layout_height="@dimen/sign_view_height"
            android:paddingLeft="@dimen/sign_padding_left"
            android:maxLines="1"
            android:singleLine="true"
            android:background="@drawable/text_shape"
            />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/sign_vertical_margin"
            android:layout_marginBottom="@dimen/sign_vertical_margin"
            android:layout_marginLeft="@dimen/sign_padding_left"
            android:textSize="@dimen/sign_text_size"
            android:textColor="@color/colorGrey"
            android:text="@string/prompt_password"
            />

        <EditText
            android:id="@+id/password"
            android:inputType="textPassword"
            android:layout_width="match_parent"
            android:layout_height="@dimen/sign_view_height"
            android:paddingLeft="@dimen/sign_padding_left"
            android:maxLines="1"
            android:singleLine="true"
            android:background="@drawable/text_shape"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/sign_vertical_margin"
            android:layout_marginBottom="@dimen/sign_vertical_margin"
            android:layout_marginLeft="@dimen/sign_padding_left"
            android:textSize="@dimen/sign_text_size"
            android:textColor="@color/colorGrey"
            android:text="@string/prompt_comfirm_password"/>

        <EditText
            android:id="@+id/confirmPassword"
            android:inputType="textPassword"
```

```
                    android:layout_width="match_parent"
                    android:layout_height="@dimen/sign_view_height"
                    android:paddingLeft="@dimen/sign_padding_left"
                    android:maxLines="1"
                    android:singleLine="true"
                    android:background="@drawable/text_shape"/>

                <Button
                    android:id="@+id/sign_up_button"
                    android:layout_width="match_parent"
                    android:layout_height="@dimen/sign_view_height"
                    android:layout_marginTop="16dp"
                    android:text="@string/action_sign_up"
                    android:textStyle="bold"
                    android:textColor="@color/colorWhite"
                    android:background="@drawable/button_shape"/>
                <Button
                    android:id="@+id/sign_in_button"
                    android:layout_width="match_parent"
                    android:layout_marginTop="16dp"
                    android:layout_height="@dimen/sign_view_height"
                    android:layout_gravity="center"
                    android:gravity="center"
                    android:text="@string/sign_in"
                    android:textSize="@dimen/sign_text_size"
                    android:textColor="@color/colorGrey"
                    style="?android:attr/borderlessButtonStyle"/>
            </LinearLayout>
        </LinearLayout>
    </ScrollView>
</LinearLayout>
```

进度条默认不显示，当用户发送网络请求时显示，客户端接收服务端响应后再隐藏。

（2）获取对控件的引用

打开新建的SignUpActivity，完成对界面控件的引用。参考代码如下：

```
public class SignUpActivity extends AppCompatActivity {

    //声明成员变量
    private AutoCompleteTextView mEmail;
    private EditText mNickname;
    private EditText mPassword;
    private EditText mConfirmPassword;
    private Button mSignUp;
    private Button mSignIn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_up);

        //获取对UI组件的引用
        mEmail = (AutoCompleteTextView) findViewById(R.id.email);
        mNickname = (EditText) findViewById(R.id.nickname);
        mPassword = (EditText) findViewById(R.id.password);
        mConfirmPassword = (EditText) findViewById(R.id.confirmPassword);
        mSignUp = (Button) findViewById(R.id.sign_up_button);
        mSignIn = (Button) findViewById(R.id.sign_in_button);

    }
……
}
```

（3）注册点击事件

给注册界面的两个按钮设置点击事件的监听器，这一次使用和上一章不同的写法，在onCreate方法中给两个按钮添加监听器：

```
//设置点击事件监听器
mSignUp.setOnClickListener(listener);
mSignIn.setOnClickListener(listener);
```

给这两个点击事件中传入相同的监听对象，然后我们来定义这个对象如何进行事件处理：

```
private Button mSignIn;

private View.OnClickListener listener = new View.OnClickListener() {
    @Override
    public void onClick(View v) {//传入被点击对象
        switch (v.getId()){//获取该对象的Id
            case R.id.sign_up_button:
                SignUp();//尝试注册
                break;
            case R.id.sign_in_button:
                SignIn();//转到登录
                break;
            default://不要忘记加上,保持编程的严谨性
                break;
        }
    }
};
```

也可以给两个按钮传入不同的监听者，分别编写不同的事件处理程序。

SignUp()和SigIn()就是我们封装的代码逻辑了，SignUp()就是注册任务的入口，我们先来看看SignIn()，也就是跳转到登录。

（4）跳转到登录界面

我们已经有登录和注册两个页面了，现在就来实现两个页面之间的切换，在SignIn()里编写下面的逻辑：

```
private void SignIn() {
        //Intent:意图,要做什么事
        Intent intent=new Intent(SignUpActivity.this,SignInActivity.class);
        startActivity(intent);
    }
```

Intent用来描述你想要做什么，这里是要实现页面跳转，两个参数就是跳转的起点和终点。

然后我们可以把程序运行起来，看一下效果了。打开应用程序清单AndroidManifest.xml，把SignUpActivity设置为启动项：

```
<activity
    android:name=".ui.activity.SignUpActivity"
    android:label="@string/title_activity_sign_up"
    android:theme="@style/AppTheme.NoActionBar" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

为了调试方便，先把注册页面设为启动项，运行后点击转到登录按钮观察效果。

类似地，大家也可以把登录界面的跳转逻辑自行实现了。

3.2 使用MD5保障信息安全

接下来为大家介绍在登录和注册模块会被经常使用的MD5算法。

MD5是一种散列算法，或者称为摘要算法，这是一种不可逆的运算，我们常常利用这种特性来“加密”敏感数据和进行文件校验（这里打引号是因为它不能被解密，没有相应的解密算法，当然也就不应该称其为加密算法了）。

（1）初始化SharedPreferences

为了方便讲解，我们这里使用Android为我们提供的一个用于数据存储的方便的API，也就是SharedPreferences。

SharedPreferences一般用于存储用户的偏好数据，并且可以同步到Google服务器，以允许在不同设备间同步用户设置。

在代码页完成初始化：

```
//创建
private SharedPreferences sharedPref;
……
    onCreate(){
        ……
        //初始化,参数为文件名和保存模式
        sharedPref = getSharedPreferences("moran", MODE_PRIVATE);
    }
```

上面的onCreate()方法是方便提示代码位置，并不是一个新的方法，下同。

初始化的第一个参数是保存的文件名（不用加后缀），第二个参数是保存类型，MODE_PRIVATE表明该文件不允许其他应用查看。

（2）向SharedPreferences保存数据

我们先把网络编程放一边，先来完成下面的逻辑，当用户点击注册按钮时，首先在本地完成数据的校验工作，逻辑代码写在SignIn()方法中，大家可以参照上一章的登录时的校验完成编写，在校验通过后，我们再向SharedPreferences提交数据：

```
if (isValid == false) {
    focusView.requestFocus();
} else {
    //新建Editor对象，往里面添加数据
    SharedPreferences.Editor editor = sharedPref.edit();
    editor.putString("email", email);
    editor.putString("nickname", nickname);
    editor.putString("password", StringUtil.getMD5(password));
    editor.commit();//把数据保存到sharedPref对象中
    Toast.makeText(this, "保存成功", Toast.LENGTH_SHORT).show();
}
```

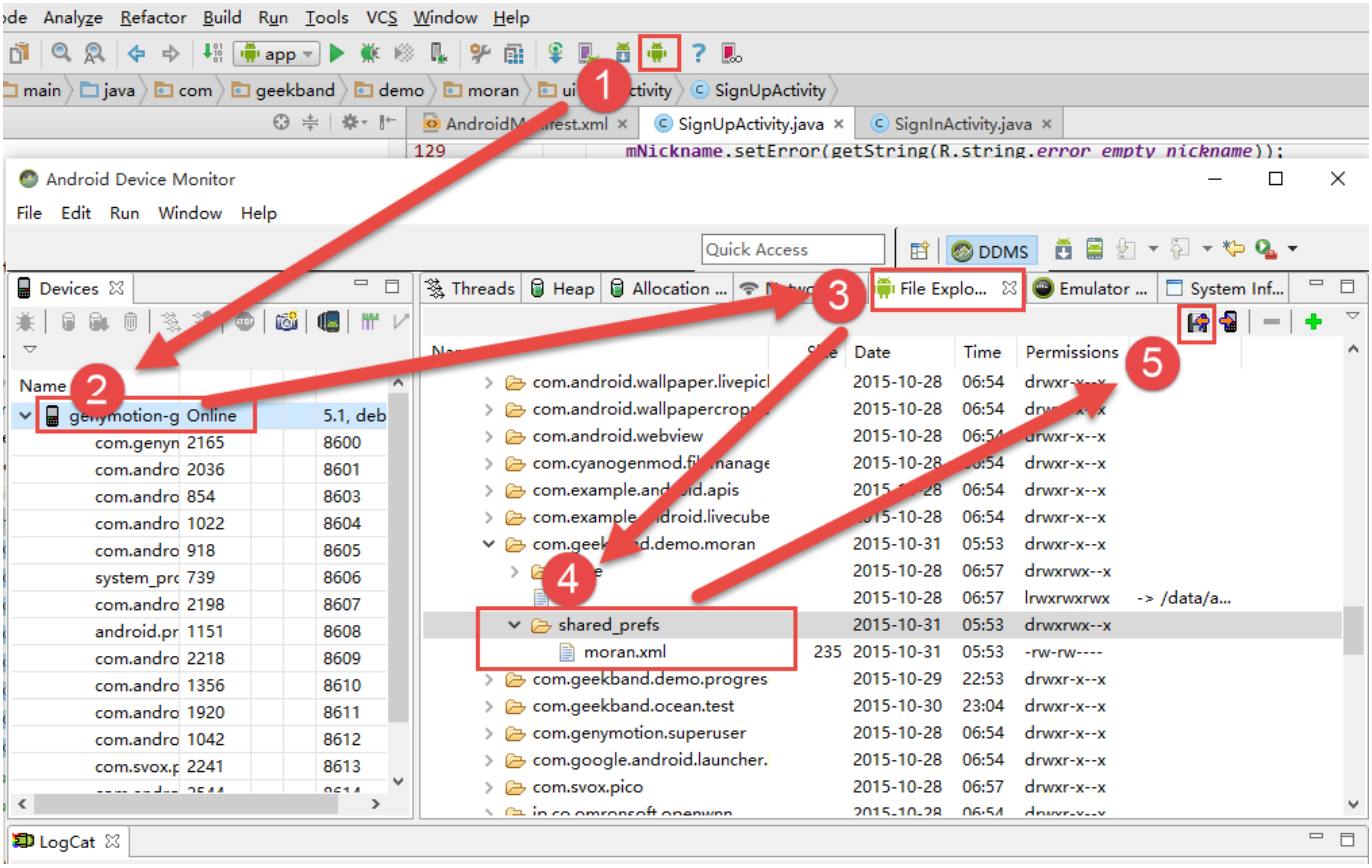
执行完上面的代码后，我们就把数据保存到创建的SharedPreferences里了。注意：保存密码时没有直接存入密码，而是调用StringUtil工具类中的getMD5()，得到它的MD5值。

把下面的代码添加到java/包名/util/StringUtil.java文件里，以后就可以用它来获取一个字符串的MD5值了。

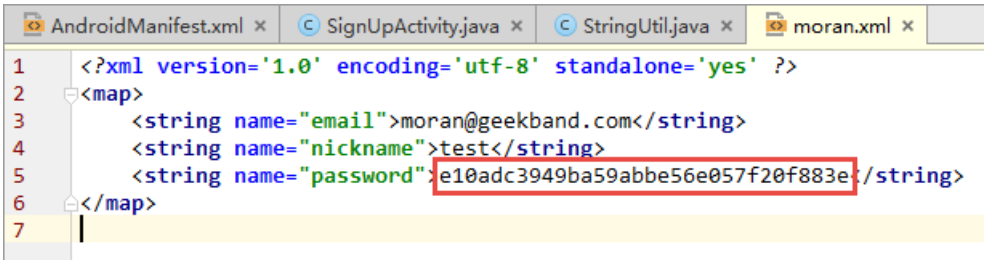
```
public static String getMD5(String password) {
    try {
        MessageDigest digest = MessageDigest.getInstance("md5");
        byte[] results = digest.digest(password.getBytes());
        StringBuilder stringBuilder = new StringBuilder();
        for(byte b : results){
            int number = b&0xff;
            String hex = Integer.toHexString(number);
            if(hex.length()==1){
                stringBuilder.append("0");
            }
            stringBuilder.append(hex);
        }
        return stringBuilder.toString();
    } catch (Exception e) {
        e.printStackTrace();
        return "";
    }
}
```

（3）查看保存数据

在AS菜单栏，选择“Tools|Android|Android Device Monitor”，或如图点击小机器人图标，选择应用所在模拟器，在文件浏览标签页找到data/data/包名，在目录下就可以看到上面创建的用户偏好数据，选择xml文件点击右侧的导出图标到电脑上。



打开这个文件，我们可以看到保存的内容：



可以看到，密码字段的数据存放的是一串很难读懂实际意义的字符串，我们可以把敏感信息用MD5值的形式进行存放和传输，一定程度上保障信息安全。

当然，实际生产应用中一般不会这么简单的使用，我们还会给它“添油加醋”。

3.3 使用多线程进行网络操作

好了，现在我们开始进行网络环境下的开发设计。

（1）完成本地校验

和登录一样，在把请求提交到服务器之前，需要先在本地完成用户输入数据的合法性校验，参考代码如下：

```
public void SignUp() {
    //重置错误提示
    mEmail.setError(null);
    mNickname.setError(null);
    mPassword.setError(null);
    mConfirmPassword.setError(null);

    //获取注册数据
    String email = mEmail.getText().toString().trim();
    String nickname = mNickname.getText().toString().trim();
    String password = mPassword.getText().toString().trim();
    String confirmPassword = mConfirmPassword.getText().toString().trim();

    //初始化获取错误焦点的控件
    boolean isValid = true;
    View focusView = null;

    //密码验证
    if (TextUtils.isEmpty(password)) {
        mPassword.setError(getString(R.string.error_empty_password));
        focusView = mPassword;
        isValid = false;
    } else if (StringUtil.isPasswordValid(password) == false) {
        mPassword.setError(getString(R.string.error_length_password));
        focusView = mPassword;
        isValid = false;
    }

    //确认密码验证
    if (TextUtils.isEmpty(confirmPassword)) {
        mConfirmPassword.setError(getString(R.string.error_empty_password));
        focusView = mPassword;
        isValid = false;
    } else if (confirmPassword.equals(password) == false) {
        mConfirmPassword.setError(getString(R.string.error_match_password));
        focusView = mConfirmPassword;
        isValid = false;
    }

    //昵称验证
    if (TextUtils.isEmpty(nickname)) {
        mNickname.setError(getString(R.string.error_empty_nickname));
        focusView = mNickname;
        isValid = false;
    }

    //邮箱验证
    if (TextUtils.isEmpty(email)) {
        mEmail.setError(getString(R.string.error_empty_email));
        focusView = mEmail;
        isValid = false;
    } else if (StringUtil.isEmail(email) == false) {
        mEmail.setError(getString(R.string.error_pattern_email));
        focusView = mEmail;
        isValid = false;
    }

    if (isValid == false) {
        focusView.requestFocus();
    } else {
        // TODO: 提交服务器
    }
}
```

前面的SharedPreferences可以删掉，不需要了。完成基本的本地校验后，我们再开始网络环境下的编程。

（2）申请网络访问权限

首先确保模拟器能正常联网，模拟器不能联网的请先自行搜索解决办法。

然后，我们需要在应用清单文件里注册网络访问权限，会有两个与网络访问有关的权限，这里我们先申请其中一个：

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

（3）在单独的线程上执行网络操作

网络操作有可能涉及到不可预知的延迟或未响应，所以我们应该总是而且必须是把网络操作放到与主线程（UI线程）隔离的线程上，我们把上面的TODO替换如下：

```
new Thread(){
    @Override
    public void run() {
        // TODO: 提交服务器
    }
}.start();
```

这段代码的含义是在UI线程上创建一个子线程，重写run()方法来完成相应操作，并调用它的start()方法启动这个线程。

（4）尝试封装表单数据

根据蓦然项目的API文档，注册模块需要以POST方式提交，提交的参数有username、password、email、gbid，返回的数据类型为JSON格式。

这里我们把提交的数据也按照JSON格式封装，参考代码如下：

```
new Thread(){
    @Override
    public void run() {

        try {
            //获取密码的md5值，并截取前20个字符
            String pwd=StringUtil.getMD5(password).substring(0,20);
            String gbid="GeekBand-A150010";//换为自己的学号
            JSONObject user = new JSONObject();
            user.put("username", nickname);
            user.put("password", pwd);
            user.put("email", email);
            user.put("gbid", gbid);
            //从全局类获取访问路径
            String url=mAppContext.getUrl(mPath);
            doPostRequest(url,user);//发送POST请求
        } catch (JSONException e) {//捕获异常
            e.printStackTrace();
        }
    }
}.start();
```

原型没有输入学号的地方，这里直接写在代码里，大家记得替换为自己的学号。昵称提交的参数名为username，不要使用自定义的名称。

其中，网络访问的路径，小编放在了自定义的全局变量类中，打开ApplicationContext类文件，添加代码：

```
public class ApplicationContext extends android.app.Application {

    private static String baseUrl="http://moran.chinacloudapp.cn/moran/web";

    public String getUrl(String path){
        return baseUrl+path;
    }
}
```

在SignUpActivity中使用该全局变量：

```
//使用全局变量
private ApplicationContext mAppContext;
private static final String mPath="/user/register";
……
    onCreate(){
        //获取对全局变量的引用
        mAppContext = (ApplicationContext) getApplication();
    }
```

（5）POST提交表单数据

Android包含两种访问HTTP的方式：HttpURLConnection和HttpClient，谷歌官方推荐在Android 2.3及以上版本中使用HttpURLConnection进行网络资源的请求。有关这两种方式的讨论可以查看下面的链接进行了解，这是小编搜到的一些信息：

> “ 为什么在Android 5.1中，org.apache.http包中的类和AndroidHttpClient类均已被废弃。对开发有什么影响？ 关于安卓HTTP请求用HttpUrlConnection还是HttpClient好

这里就使用谷歌推荐的HttpURLConnection方式进行连接：

```
private void doPostRequest(String path, JSONObject data) {
    try {
        //获得传输的实体
        byte[] entity = data.toString().getBytes("UTF-8");
        URL url = new URL(path);
        //实例化一个HTTP连接对象
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setConnectTimeout(5000);//定义超时时间
        connection.setRequestMethod("POST");//使用POST方式发送请求
        connection.setUseCaches(false);//设置不允许使用缓存
        connection.setDoOutput(true);//允许对外输出
        connection.setDoInput(true);//允许对内输入
        connection.setRequestProperty("Content-Type", "application/json");//提交格式为json
        connection.setRequestProperty("Content-Length", String.valueOf(entity.length));
        OutputStream outputStream = connection.getOutputStream();//获得输出流
        outputStream.write(entity);
        outputStream.close();

        if(connection.getResponseCode()==200){
            // TODO: 接收响应数据
        }

    } catch (UnsupportedEncodingException e) {//编码异常
        e.printStackTrace();
    } catch (MalformedURLException e) {//路径异常
        e.printStackTrace();
    } catch (ProtocolException e) {//协议类型异常
        e.printStackTrace();
    } catch (IOException e) {//输入输出异常
        e.printStackTrace();
    }
}
```

getByte()用于将我们提交的字符串转换成字节数组，使用系统默认的字符集进行编码，这里指定使用UTF-8进行编码。

连接时间的单位为毫秒，最好不要超过10秒，以免被系统回收。

由于要向服务器提交提交数据，并接收服务器传回的数据，所以要设置允许输出和输入。

向服务器提交数据，对于客户端而言是输出，所以这里要使用OutputStream对外输出。

当执行connection.getResponseCode()==200时，表单数据就会提交给服务器，服务器处理完毕，返回浏览器端响应，响应码为200表示服务器成功处理了请求。

（6）处理响应数据

当服务器成功处理请求后，返回客户端一个输入流，我们先将输入流转换为一个字节数组：

```
if(connection.getResponseCode()==200){
    InputStream inputStream=connection.getInputStream();
    byte[] is=StreamUtil.readInputStream(inputStream);
    ……
}
```

将输入流转换为字节数组的操作以后会经常使用，所以我们把处理逻辑封装起来，方便以后调用。在java/包名/util下新建一个名为StreamUtil的工具类，参考代码：

```
public class StreamUtil {

    /**
    * 从输入流中获取数据
    * @param inStream 输入流
    * @return
    * @throws Exception
    */
    public static byte[] readInputStream(InputStream inStream) throws Exception{
        ByteArrayOutputStream outStream = new ByteArrayOutputStream();
        byte[] buffer = new byte[1024];
        int len = 0;
        while( (len=inStream.read(buffer)) != -1 ){
            outStream.write(buffer, 0, len);
        }
        inStream.close();
        return outStream.toByteArray();
    }
}
```

（7）抽取需要的信息

现在已经得到服务器返回的数据，还需要把数据进行解析，抽取出我们需要的数据，忽略其他。

查看API文档，注册成功返回的是类似下面的数据：

```
{
"status": 1,
"data": {
"user_id": "6",
"user_name": "testProject3",
"project_id": "3"
},
"message": "Register success"
}
```

现在大家都应该清楚了，这是一串json格式的数据，只是里面还嵌套了一个json，我们简单分析一下：

最外层可以写成{"status": 1,"data": {...},"message": "Register success"}，其中“data”的值又是一个json，即{"user_id": "6","user_name": "testProject3","project_id": "3"}

对于注册模块，暂时我们只需要一个"status"信息和"message"信息就可以了，不过为了查看是否有编码问题，我们先把所有信息显示出来：

```
byte[] is=StreamUtil.readInputStream(inputStream);
String json=new String(is);

Message msg=Message.obtain();
msg.obj=json;
handler.sendMessage(msg);
```

因为现在是在子线程中操作，是无法在界面是显示信息的，所以这里使用一个Handler向主线程传递消息。

在SignUpActivity中，创建一个Handler对象，添加处理逻辑：

```
private Button mSignIn;
……
private Handler handler;
{
    handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        Toast.makeText(getApplicationContext(), msg.obj.toString(), Toast.LENGTH_LONG).show();
        }
    };
}
```

现在运行程序，输入相关信息注册一下吧！如果一切正常，就会在界面上显示所有返回的数据。

接下来，我们就来抽取出需要的信息：

```
String json=new String(is);
SONObject jsonObject=new JSONObject(json);
int status=jsonObject.getInt("status");
String message=jsonObject.getString("message");
Message msg=Message.obtain();
if(status==1){
    msg.what=SUCCESS;
}else {
    msg.what=ERROR;
}

msg.obj=message;
handler.sendMessage(msg);
```

SUCCESS和ERROR为创建的常量：

```
private static final String mPath = "/user/register";
……
private static final int SUCCESS = 1;
private static final int ERROR = 0;
```

返回消息的处理逻辑改为：

```
handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
    switch (msg.what) {
        case SUCCESS:
            Toast.makeText(getApplicationContext(), msg.obj.toString(), Toast.LENGTH_LONG).show();
            break;
        case ERROR:
            Toast.makeText(getApplicationContext(), msg.obj.toString(), Toast.LENGTH_LONG).show();
        default:
            break;
        }
    }
};
```

好了，现在部署运行起来吧！

至此，注册模块也暂时告一段落了，大家现在可以把登录模块的网络连接部分完善起来了，由于提交的密码是取的MD5值，所以登录时应该使用相同的逻辑。

在本章开篇提到了进度条，大家可以先尝试自己完成一下。提示一下，使用多线程操作。