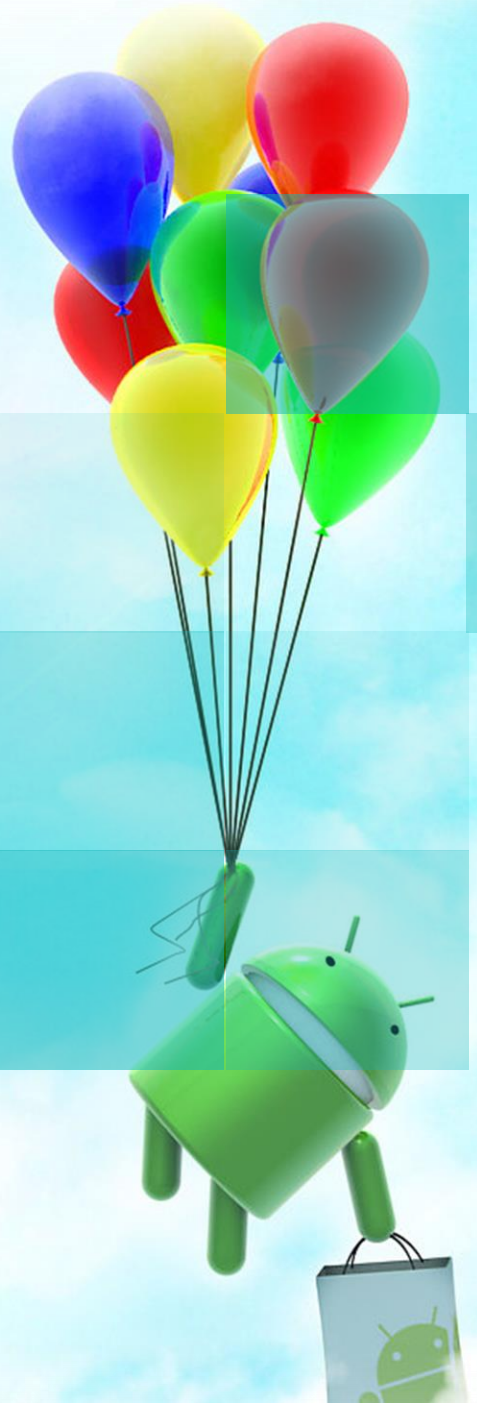
Android
云存储客户端开发



项目6

登录注册模块

任务01 登录功能开发

任务02 注册

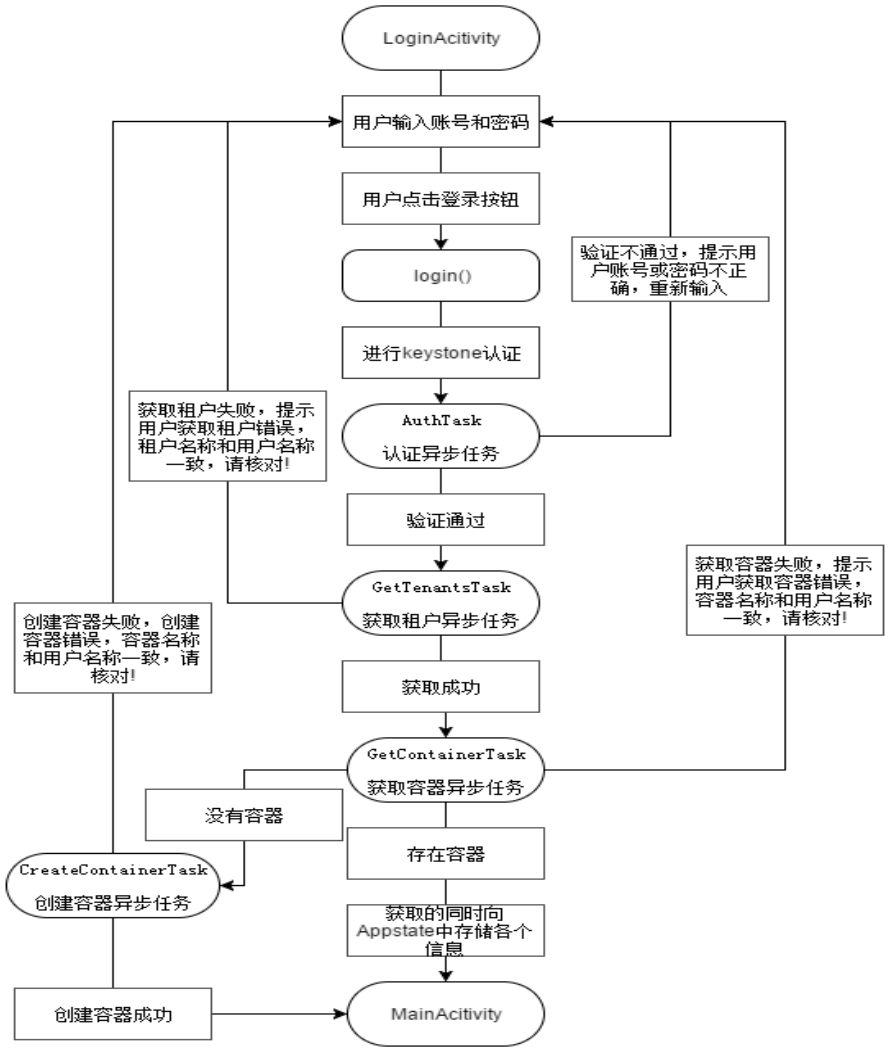
学习任务

完成已注册用户登录到Swift服务和用户注册的功能

1

登录功能开发

登录注册功能-操作流程图



登录功能开发

功能需求

已知已注册可登录的账号用户名为gw001，密码是00000。

登录功能流程如图所示。

1）视图层，界面实现

根据原型图设计实现登录窗口View。

1 登录功能开发

功能需求

2）控制层，用户登录输入处理

3）服务层，用户信息后台登录验证

4）控制层，认证返回结果处理

认证返回值处理：认证可能的情况包含：

（1）没有联网，服务不能访问，提示用户联网；

（2）账户和密码都正确，验证通过，自动登录，展示网盘主窗口。

（3）账户不正确，验证不通过，提示用户重新输入账户和密码。

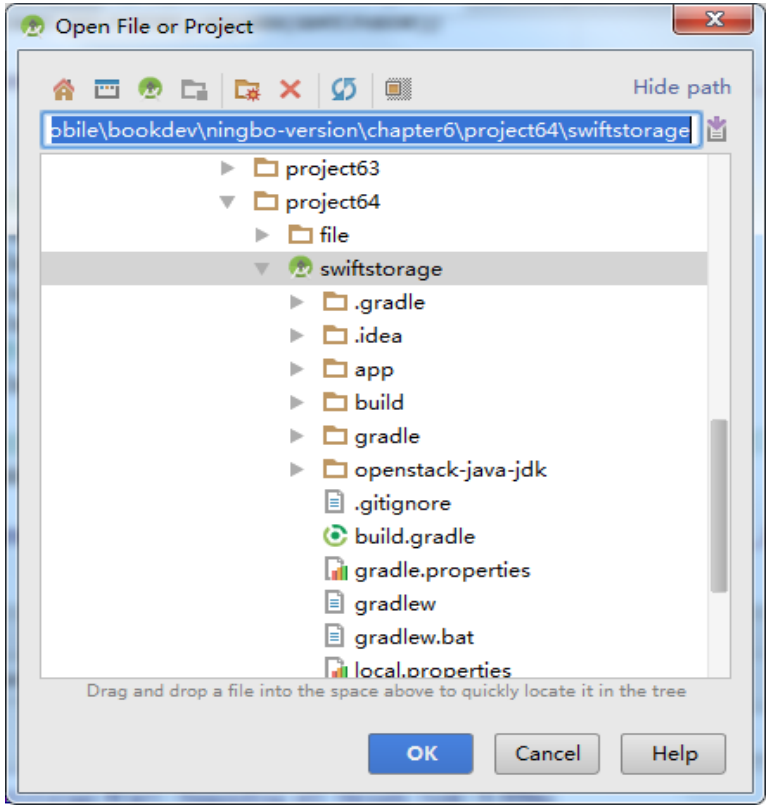
（4）密码不正确，验证不通过，提示用户重新输入账户和密码。

（5）连续输入3次不正确，提示用户找回密码。通过邮件的方式找回。

1 登录注册模块

实现步骤

运行Android Studio，选择File\Open..，点击弹出选择project64目录下面的项目“swiftstorage”，如图所示：



1 登录功能开发

实现步骤

2）界面实现

登录布局包含5个组件，从上到下依次为：图标（ImageView）、用户输入框（EditText）、 密码输入框（EditText）、登录按钮（Button），注册按钮（TextView）,及最下面一个进度条（ProgressBar）。几个组件的说明如表6-1所示。

此布局的路径app\src\main\res\layout\login.xml。

组件	作用	ID
ImageView	图标	imageLogo
EditText	用户名输入框	txtUsername
EditText	密码输入框	txtPassword
Button	登录按钮	btnLogin
TextView	注册按钮	register
ProgressBar	显示登录进度的进度条	servicesProgressBar

1 登录功能开发

实现步骤

3）实现登录视图LoginActivity

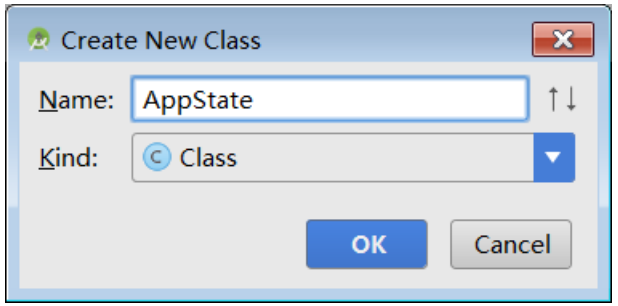
此Activity所在路径为app\src\main\java\com.xiandian.openstack.cloud.swiftstorage\ LoginActivity.java。

此处的登录视图项目4Android基础中已经详细介绍，接下来有关登录的方法都是在此Java类中实现。

4）新建AppState类

这个类为应用状态类，不需要写界面。此类主要用于存储应用当前的状态，单个实例，保存当前账号、租户、容器、选择的路径等信息。新建过程如下：

右击项目根目录，选择new\java class项，在新建类对话框中进行设置，如图。类名设置为AppState。



1 登录功能开发

实现步骤

扩展

单例设计模式

Java中单例模式是一种常见的设计模式，单例模式有以下特点：

（1）单例类只能有一个实例。

（2）单例类必须自己创建自己的唯一实例。

（3）单例类必须给所有其他对象提供这一实例。

单例模式确保某个类只有一个实例，而且自行实例化并向整个系统提供这个实例。在计算机系统中，线程池、缓存、日志对象、对话框、打印机、显卡的驱动程序对象常被设计成单例。

1 登录功能开发

实现步骤

5）调用OpenStackClientService实现登录验证

增加login()方法，该方法通过调用OpenStackClientService，进行验证。实现方法如下：

```
/**
 * 登录。步骤包括，认证，获得租户， 根据租户获得容器，根据容器获得对象。
 */
public void login() {
    //(1) 重新获取用户信息
    userName = txtUsername.getText().toString().trim();
    userPassword = txtPassword.getText().toString().trim();
    openstackSwiftIP = AppState.getInstance().getOpenStackIP().trim();

    //(2)设置用户名、租户名、容器名、回收站容器名称
    tenantName = userName;// 目前环境用户名称和租户名称一致
    containerName = tenantName;// 目前主容器和租户名称一致
    garbageContainerName = "garbage_" + containerName;// 目前回收站容器在主容器名称前garbage_

    //(3) 获取OpenStackClientService服务，并初始化  String keystoneAuthUrl="http://"+openstackSwiftIP+":5000/v2.0";
    OpenStackClientService service = OpenStackClientService.getInstance();
    service.setKeystoneAuthUrl(keystoneAuthUrl);
    service.setKeystoneAdminAuthUrl(keystoneAuthUrl);
    service.setKeystoneEndpoint(keystoneAuthUrl);
    service.setOpenstackIP(openstackSwiftIP);
    service.setTenantName(tenantName);
    service.setKeystoneUsername(userName);
    service.setKeystonePassword(userPassword);
   // (4). 认证异步任务
    AuthTask authTask = new AuthTask();
    authTask.execute();}
```

1 登录功能开发

实现步骤

代码说明，对应参考注释：

（1）读取用户输入的用户名和密码。

（2）租户、用户、容器名称和用户名默认一样，后台swift服务进行对应。

（3）获取OpenStackClientService并对OpenStack的Keystone认证服务keystone 认证Url、服务器地址、容器、用户和密码进行初始化。

（4）完成以上操作，定义一个异步类AuthTask进行访问后台认证和存储服务，访问认证和存储是通过Restful Http调用完成，需要通过启动异步任务，并监控执行结果。

1 登录功能开发

实现步骤

下面再定义AuthTask内部类，AuthTask实现接口AsyncTask，前面基础已经介绍过AsyncTask异步任务。代码如下：

（1）用户认证，首先获得Keystone服务，然后获得访问许可，代码如下：

Keystone keystone = service.getKeystone(); Access access = service.getAccess();

根据反馈情况，进行处理。如果认真成功，获取租户

（2）获取租户，代码如下：

OpenStackClientService osServcie = OpenStackClientService.getService();

Tenant tenant = osServcie.getTanet();

（3）获取租户之后，获取同用户名称一行的容器，代码如下：

OpenStackClientService osServcie = OpenStackClientService.getService();

Container container = osServcie.getContainer(containerName);

（4）如果能正常获得容器，创建Intent，进入主视图MainActivity，代码如下：

Intent intent = new Intent(LoginActivity.this, MainActivity.class); startActivity(intent);

（5）如果容器不存在，系统错误，提示用户容器不存在，联系管理人员。

以上网络Http请求完成，需要放在异步任务中实现，这里全部使用封装各自的内部类实现。

目前swift服务通过注册或后台直接创建，因此不应该不对应容器的情况，一旦出现可能的远程后台操作少了步骤或者系统出现错误。因此这里的客户端默认不进行创建，发现容器不在的情况，由系统管理会做人工处理。

1 登录功能开发

实现步骤

扩展:

AsyncTask是Android默认提供的异常处理机制，OpenStack Sdk的实现方式是HTTPConnect，实质是启动了一个现场Thread完成http请求，代码重复量稍大。读者可以到查阅相关的替代方案，包括OkHttp、Volley、Retrofit等。

至此，读者就已完成登录认证的视图、控制和服务访问的代码，下面读者将进行登录执行来测试认证服务。

6）修改启动Activity设置

首先启动LoginAcitivity，打开Android的配置文件app\manifest\AndroidManifest.xml，以login为启动activity。

代码如下:

```
<activity android:name=". LoginActivity"
        android:label="@string/app_name"
        android:theme="@android:style/Theme.Holo.Light.NoActionBar.Fullscreen"

android:windowSoftInputMode="stateHidden|adjustResize">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
</application>
```

1 登录功能开发

实现步骤

扩展：
AsyncTask是Android默认提供的异常处理机制，OpenStack Sdk的实现方式是HTTPConnect，实质是启动了一个现场Thread完成http请求，代码重复量稍大。读者可以到查阅相关的替代方案，包括OkHttp、Volley、Retrofit等。
至此，读者就已完成登录认证的视图、控制和服务访问的代码，下面读者将进行登录执行来测试认证服务。
6）修改启动Activity设置
首先启动LoginAcitivity，打开Android的配置文件app\manifest\AndroidManifest.xml，以login为启动activity。
代码如下：

```
<activity android:name=". LoginActivity"
      android:label="@string/app_name"
      android:theme="@android:style/Theme.Holo.Light.NoActionBar.Fullscreen"
android:windowSoftInputMode="stateHidden|adjustResize">
      <intent-filter>
         <action android:name="android.intent.action.MAIN" />
         <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
   </activity>
</application>
```

注意：通过设置视图Activity的<intent-filter>中配置让LoginActivity默认启动第一个视图。

```
<action android:name="android.intent.action.MAIN" />

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

1 登录功能开发

功能执行测试

1）执行效果

选择项目Android的“app”，点击工具条中运行Run “app”，执行的效果如图6-5所示，填入已经注册好的用户名和密码，进行登录验证。

1 登录功能开发

扩展练习

通过以上开发，我们完成登录基本功能，我们场景的App客户端，在用户名称和密码设置一般都有要求或限制。请进行以下扩展练习

练习题

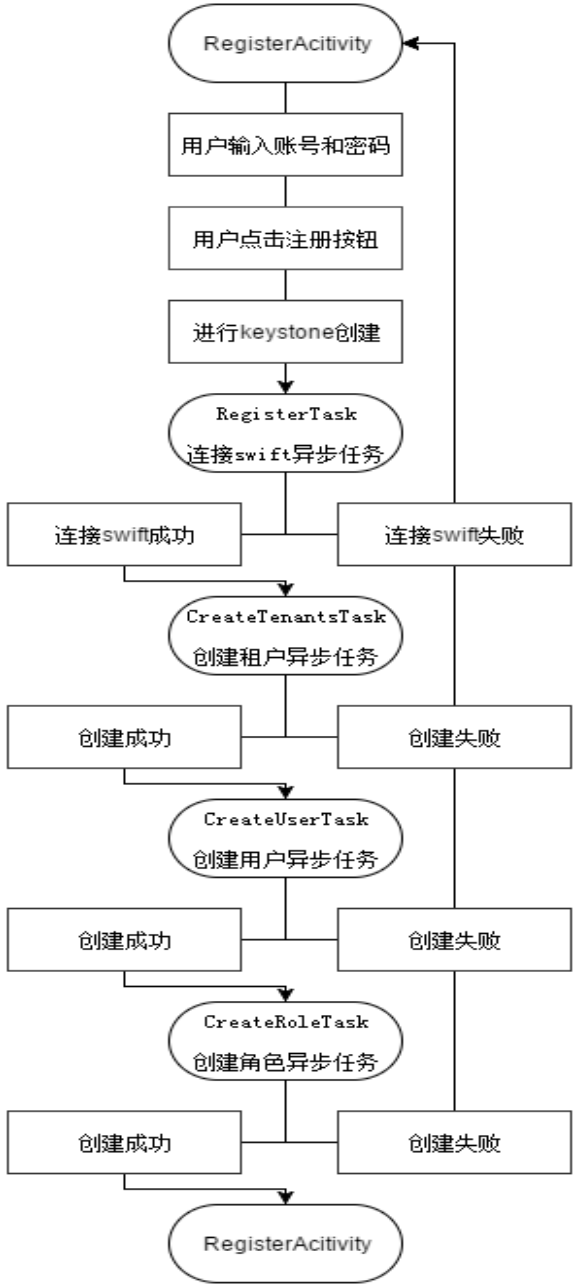
1）输入验证开发练习：

查阅Swift服务的用户名称和密码设置是否有字符限制？如“\”　“/”是否允许输入？如果有字符限制，请在客户端用户输入进行提示，提示用户不能输入那些字符。如果没有限制，请对密码设置强度要求至少6个字符，字符必须是包含字母和数字组合。

2 注册功能开发

功能需求

已知需要注册的账号用户名为gw002，密码是000000。注册功能的流程图如图所示。



2 注册功能开发

功能需求

1）视图层，界面实现

根据原型图设计实现登录窗口View。

2 注册功能开发

功能需求

2）控制层，用户注册输入处理

点击登录窗口的“注册”按钮之后，读取用户在输入框输入的用户名和密码。

3）服务层，用户信息后台注册验证

根据前面SDK的了解，我们知道Keystone服务负责用户的认证和授权，这里调用Keystone的创建租户信息，传递的参数为用户名和密码。

4）控制层，认证返回结果处理

认证返回值处理：认证可能的情况包含：

（1）没有联网，服务不能访问，提示用户联网；

（2）账户和密码都正确，注册通过，自动登录，展示网盘主窗口。

（3）账户已存在，注册不通过，提示用户重新输入账户和密码。

2 注册功能开发

实现步骤

界面实现

注册布局包含5个组件，从上到下依次为：用户输入框（EditText）、 密码输入框（EditText）、注册按钮（Button）。具体说明如表所示。

此布局的路径app\src\main\res\layout\login.xml。

组件	作用	ID
EditText	用户名输入框	txtUsername
EditText	密码输入框	txtPassword
Button	注册按钮	btnLogin

此xlm文件中所涉及的控件的使用，具体请参考前面项目4Android基础。

2 注册功能开发

实现步骤

3）注册视图RegisterActivity

此Activity所在路径为app\src\main\java\com.xiandian.openstack.cloud. swiftstorage\RegisterActivity.java

此处的登录视图项目4Android基础中已经详细介绍，接下来有关登录的方法都是在此Java类中实现。

此类中已存在的变量及他的作用。

4）调用OpenStackClientService实现注册验证

（1）首先连接swift，连接成功后创建租户。此处读者需要重新回顾下Swift存储基础知识。

```
protected TaskResult<Tenant> doInBackground(String... params) {
    OpenStackClientService osServcie = OpenStackClientService.getService();
    try {
        //设置租户的属性
        tenant = new Tenant();tenant.setName(username);
        tenant.setDescription(username); tenant.setEnabled(true);
        //在keystone中创建租户
        keystone = new Keystone(keystoneAuthUrl); keystone.token(access.getToken().getId());
        tenant = keystone.tenants().create(tenant).execute();
        return new TaskResult<Tenant>(tenant);
    } catch (Exception e) {
        osServcie.resetConnection();
        return new TaskResult<Tenant>(e);
    }
}
```

2 注册功能开发

实现步骤

（2）接着创建用户。

```
protected TaskResult<User> doInBackground(String... params) {
    OpenStackClientService osServcie = OpenStackClientService.getService();
    try {
        //设置用户的属性
        user = new User();
        user.setName(username);
        user.setPassword(password);
        user.setEnabled(true);
        user.setEmail(email);
        user.setTenantId(tenant.getId());
        //在keystone中创建用户
        keystone = new Keystone(keystoneAuthUrl);
        keystone.token(access.getToken().getId());
        user = keystone.users().create(user).execute();
        return new TaskResult<User>(user);
    } catch (Exception e) {
        osServcie.resetConnection();
        return new TaskResult<User>(e);
    }
}
```

2 注册功能开发

实现步骤

（3）最后分配角色

```
protected TaskResult<Role> doInBackground(String... params) {
    OpenStackClientService osServcie = OpenStackClientService.getService();
    try {
        //设置用户角色属性
        role = new Role();
        role.setName(username);
        role.setDescription(username);
        //在keystone中创建角色
        keystone = new Keystone(keystoneAuthUrl);
        keystone.token(access.getToken().getId());
        role = keystone.roles().create(role).execute();
        return new TaskResult<Role>(role);
    } catch (Exception e) {
        osServcie.resetConnection();
        return new TaskResult<Role>(e);
    }
}
```

2 注册功能开发

实现步骤

4）启动注册Activity设置

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

2 注册功能开发

功能执行测试

选择项目Android的“app”，点击工具条中运行Run “app”，执行的效果如图所示，填入已经注册好的用户名和密码，进行登录验证。