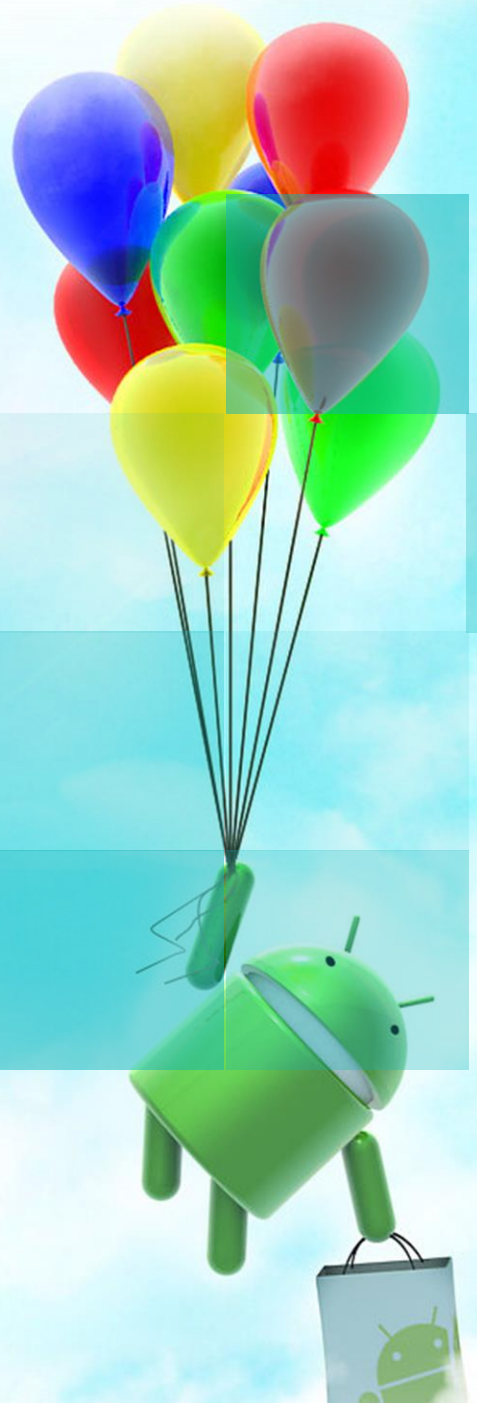
Android
云存储客户端开发



项目4

Android 基础

任务01 项目程序解读

任务02 新建带抽屉导航的App

任务03 定义App名称和图标

任务04 实现导航Navigation Drawer

任务05 实现Toolbar工具条

任务06 实现文件列表

任务07 弹出框、进度条

任务08 异步任务模拟文档下载

学习任务

- 新建带抽屉导航的APP
- 定义APP名称和图标
- 实现导航NagivitorDrewr导航
- 实现Toolbar工具条
- 实现文件列表：ListView内容列表
- 实现GridView视图
- 弹出框、(Progressbar)进度条
- 异步任务网络访问

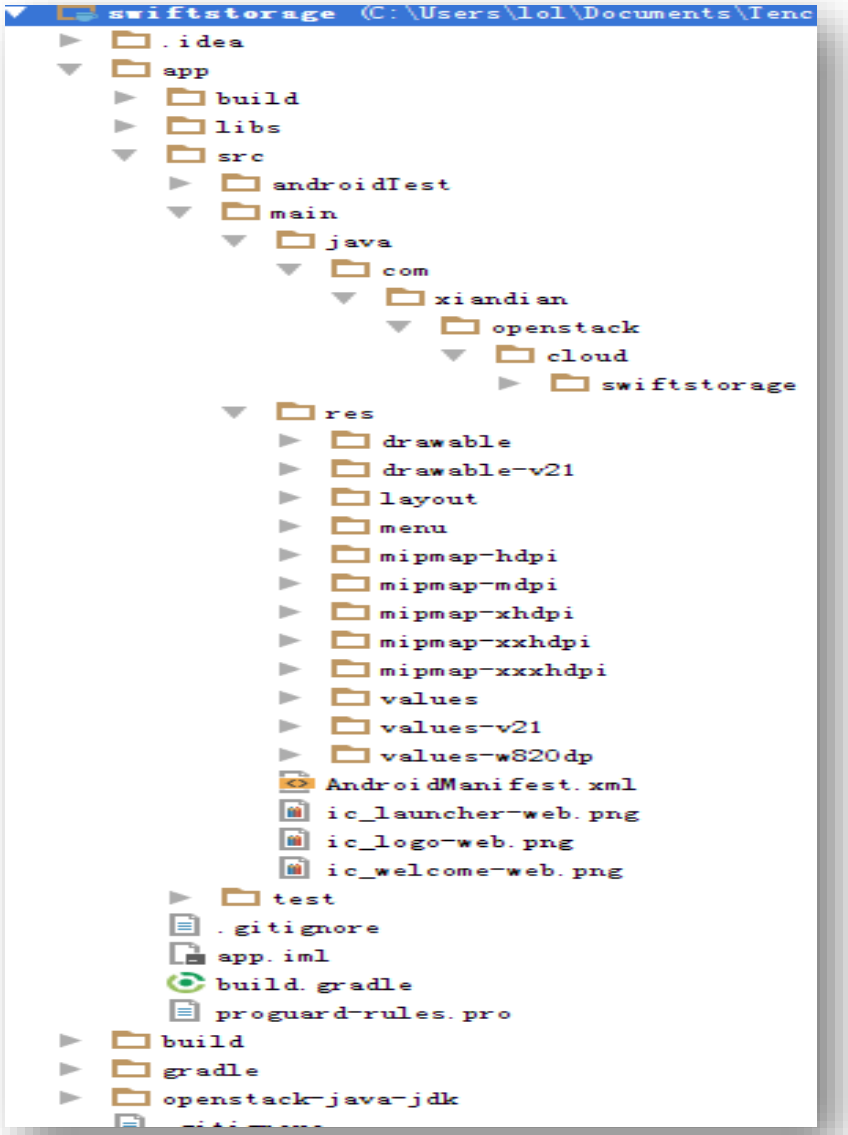
本章单元介绍

本单元主要讲解Android基础知识，通过几个阶段性项目把网盘APP的界面实现。在实现过程中学习和掌握Android开发的基础知识和常见控件的应用，同时为后面的功能实现作准备。

1

项目程序解读

项目结构图



1 项目程序解读

App文件夹下的内容及其作用

1、build
存放项目的输出

2、libs
存放我们这个项目需哦需要的第三方jar文件

3、src
存放我们编辑的源代码

4、build.gradle
模块中的gradle文件， Gradle是一种依赖管理工具，基于Groovy语言，面向Java应用为主，它抛弃了基于XML的各种繁琐配置，取而代之的是一种基于Groovy的内部领域特定（DSL）语言。
一个 Android Studio 项目中，会存在多个.gradle文件。其中project目录下存在一个build.gradle文件和每一个module会存在一个build.gradle文件。
工程中的build.gradle内容主要包含了两个方面：一个是声明仓库的源，这里可以看到是指明的jcenter(), 之前版本则是mavenCentral(), jcenter可以理解成是一个新的中央远程仓库，兼容maven中心仓库，而且性能更优。另一个是声明了android gradle plugin的版本.android studio 1.0正式版必须要求支持gradle plugin 1.0的版本。

5、openstack-java-jdk
该文件夹下存放我们云存储开发过程中的sdk，我们开发的APP几乎所有的功能实现都来自于这里

1 项目程序解读

App文件夹下的内容及其作用—Src文件夹下的内容及其作用

1、java文件夹

此文件夹用于存放我们编写的程序代码，为了便于代码的管理，我们将不同的功能放在不同的包里。

Java文件夹下的内容及其作用

（1）application包
定义全局信息

（2）base包中的TaskResult
对异步任务执行结果进行封装

（3）fragment包
存放各个fragment，方便屏幕适配

（4）fs包
云存储的各类功能进行封装

（5）utils包
对各种常用功能进行封装，便于调用

1 项目程序解读

App文件夹下的内容及其作用—Src文件夹下的内容及其作用

2、 res文件夹

该文件夹存放项目中用到的各种资源如布局、菜单、图片等等。

res文件夹下的内容及其作用

（1） layout文件夹
存放布局资源

（2） menu文件夹
存放菜单资源

（3） drawable文件夹
存放图片资源

（4） values文件夹
存放值资源

1 项目程序解读

App文件夹下的内容及其作用—Src文件夹下的内容及其作用

3、 清单文件：AndroidManifest.xml

AndroidManifest.xml 是每个android程序中必须的文件。它位于整个项目的根目录，描述了package中暴露的组件（activities, services, 等等），他们各自的实现类，各种能被处理的数据和启动位置。 除了能声明程序中的Activities, ContentProviders, Services, 和 Intent Receivers, 还能指定permissions和instrumentation（安全控制和测试）

2

新建带抽屉导航的App

相关知识

NavigationDrawer是Google在Material Design中推出的一种侧滑导航栏设计风格。Google 为了支持这样的导航效果，推出一个新控：DrawerLayout 。

一般情况下，在DrawerLayout 布局下只会存在两个子布局，一个内容布局和一个侧滑菜单布局，这两个布局关键在于android:layout_gravity属性的设置。如果你想把其中一个子布局设置成为左侧滑菜单，只需要设置 android:layout_gravity="start" 即可（也可以是 left，右侧滑则为end 或 right ），而没有设置的布局则自然成为内容布局。

2

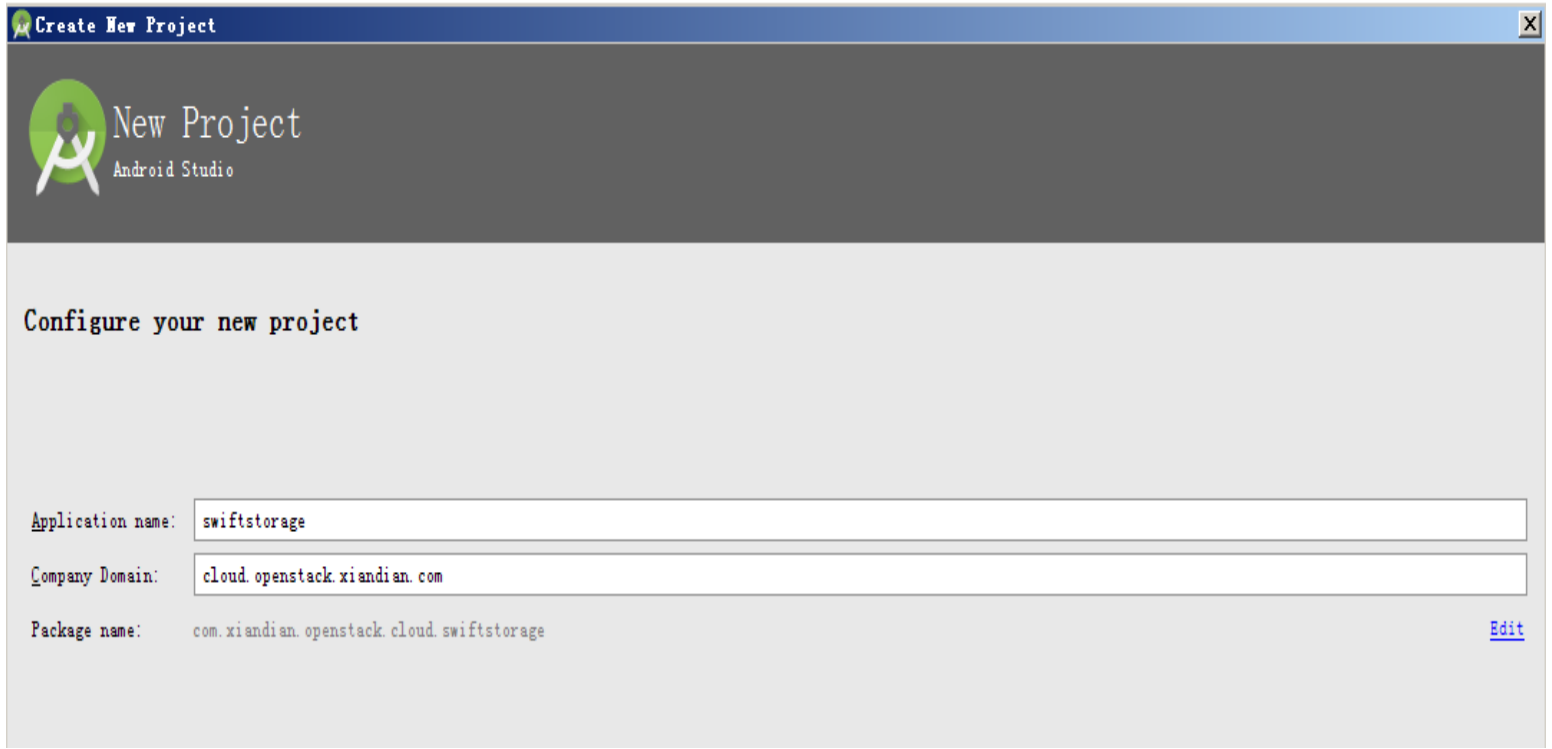
新建带抽屉导航的App

实现步骤

（1）道路项目

在Android Studio中选择菜单File→New Project，进入“New Project”向导

（2）修改项目名称

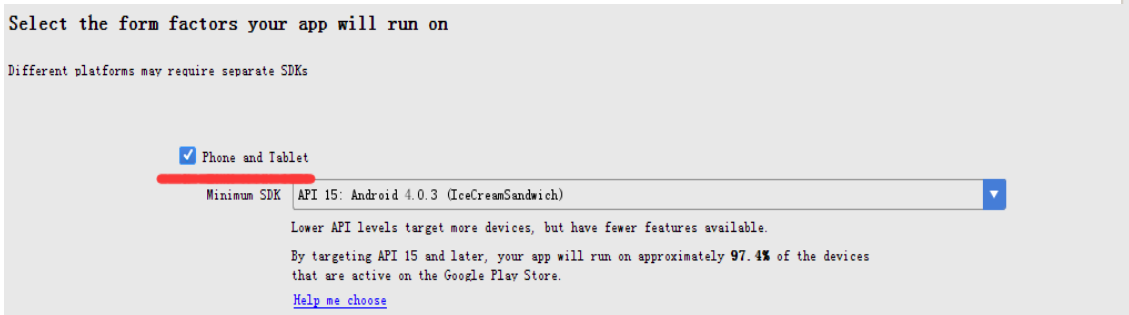


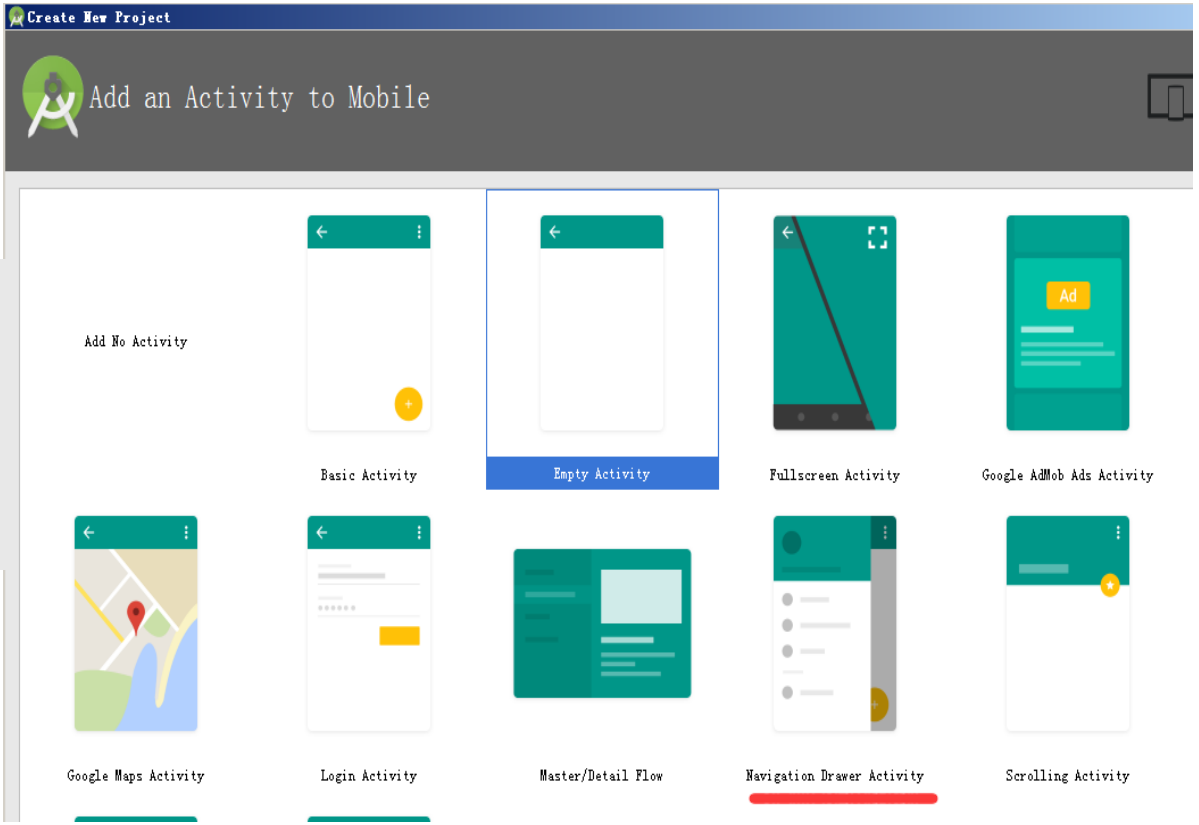
2 新建带抽屉导航的App

实现步骤

(3) 选择SDK

(4) 选择“Navigation Drawer Activity”类型



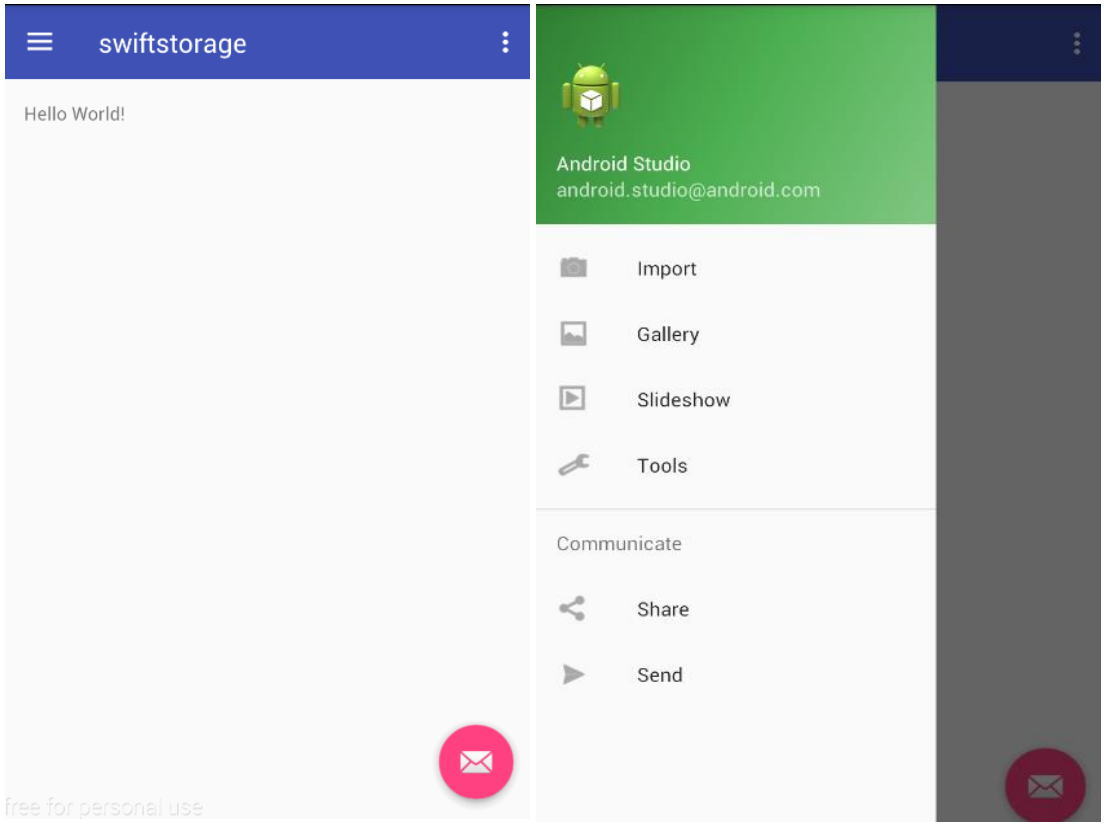


2

新建带抽屉导航的App

实现步骤

(5) 设置“Title”为“swiftstorage”
(6) 运行效果



3

定义App名称和图标

相关知识

目前网络上常见的图片格式有3种：
分别为JPEG、GIF和PNG这三种格式各有特点。

- JPEG 是一般的照片标准格式

不支持透明

- GIF 被限制在256色

因此对于大块纯色和简单图像非常好GIF
支持透明，适合制作呈现简单动画但是会产生锯齿边缘

- PNG 是GIF和JPEG的漂亮结合

具有JPEG格式图片的质量和GIF格式图片的透明度，而且没有锯齿

Android占时还不支持GIF，所以常用的图片格式只有：

.JPG和.PNG两种

3

定义App名称和图标

实现步骤

（1）修改App名称，在资源目录里找到“strings.xml”，配置“app_name”，修改为“Swift云存储”，代码如下：

```
<resources>
    <string name="app_name">Swift云存储</string>
    <string name="app_swift_app_info">OpenStack Swift Android Client</string>
    <string name="navigation_drawer_open">Open navigation drawer</string>
    <string name="navigation_drawer_close">Close navigation drawer</string>
</resources>
```

3

定义App名称和图标

实现步骤

（2）更改App图标，将我们准备好的图片“ic_launcher.png”拷贝到资源中的“mipmap”目录中，修改清单文件“AndroidManifest.xml”，确认“android:icon”属性值为刚才添加的图片，代码如下：

```
android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
```

3 定义App名称和图标

实现步骤

（3）修改完成后的执行效果，同时与未修改前的图标与名称

4 实现导航Navigation Drawer

相关知识

Drawer 作为新一代 Android Design 的代表之一，具有非常高的泛用性。很多旧的导航方式都可以无违和的被 Drawer 替代。

Navigation Drawer是从屏幕的左侧滑出，显示应用导航的视图。Navigation Drawer是Android团对在2013 google IO大会期间更新的Support库（V13）中新加入的重要的功能。

4

实现导航Navigation Drawer

实现步骤

（1）找到导航菜单文件定义的位置

导航菜单信息存放在资源文件menu下的activity_main_drawer.xml文件中。因此我们需要编辑这个文件生成我们需要的数据。下面代码即是其中某个菜单项的值。

```
<item
    android:id="@+id/nav_camera"
    android:icon="@drawable/ic_menu_camera"
    android:title="Import" />
```

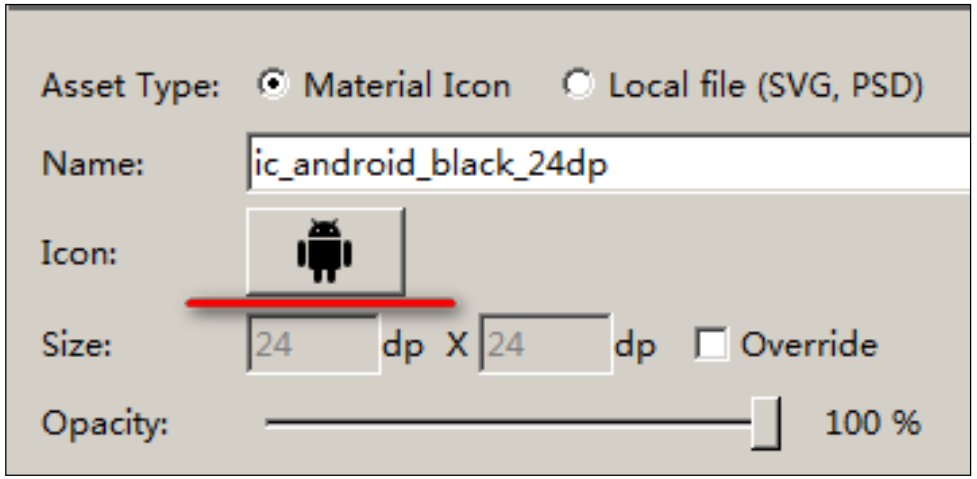
4 实现导航Navigation Drawer

实现步骤

（2）设定菜单项的图标

我们可以用任务4.2中增加logo的方法来设定菜单项的图标,但我们需要对不同的分辨率做适配的话, 每个图标都需要至少做5个分辨率, 非常麻烦。
Android Studio提供了一个向导来生成矢量图标。

在drawable目录右键点击后,点击New,点击Vector Asset。

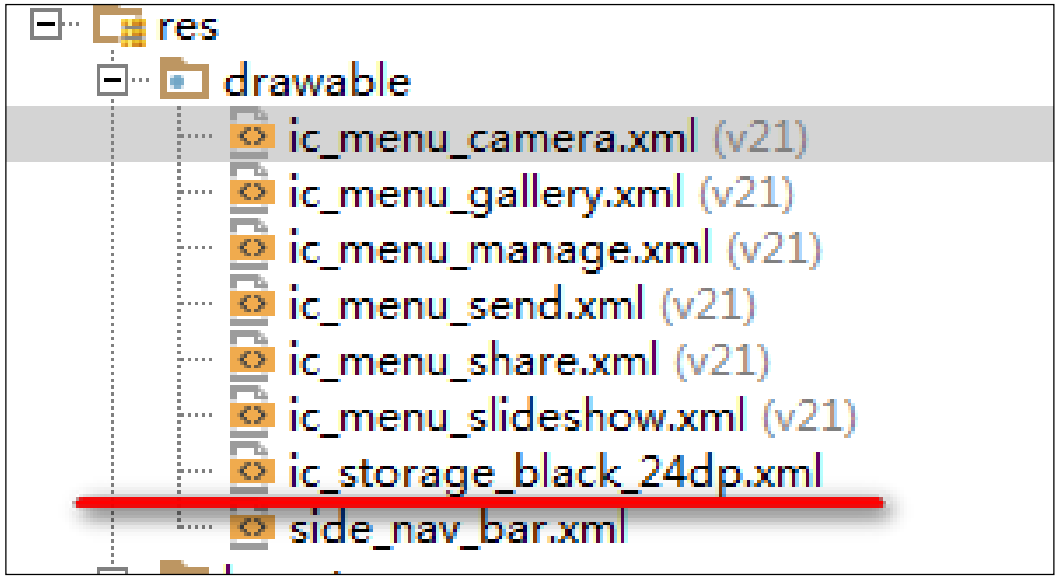


选择上图红线标注的按钮，出现“Select Icon”向导，在左边列表找到“Device”，右边找到“”图标，随后点击“OK”，其他选项保持默认，完成图标的添加。添加成功后，可以看到资源中多了个“ic_storage_black_24dp.xml”的文件

4

实现导航Navigation Drawer

实现步骤



在该文件上右键，选择“Refactor”→“Rename”，将其重新命名为“ic_menu_swiftstorage.xml”。

4

实现导航Navigation Drawer

实现步骤

打开menu下的activity_main_drawer.xml 文件，只保留一个菜单项，编辑文件，代码如下：

```
<group android:checkableBehavior="single">
    <item
        android:id="@+id/nav_swiftdisk"
        android:icon="@drawable/ic_menu_swiftstorage"
        android:title="@string/menu_swiftdisk" />
</group>
```

4

实现导航Navigation Drawer

实现步骤

打开MainActivity.java文件，编辑菜单响应事件，将不存在的菜单都去掉，代码如下：

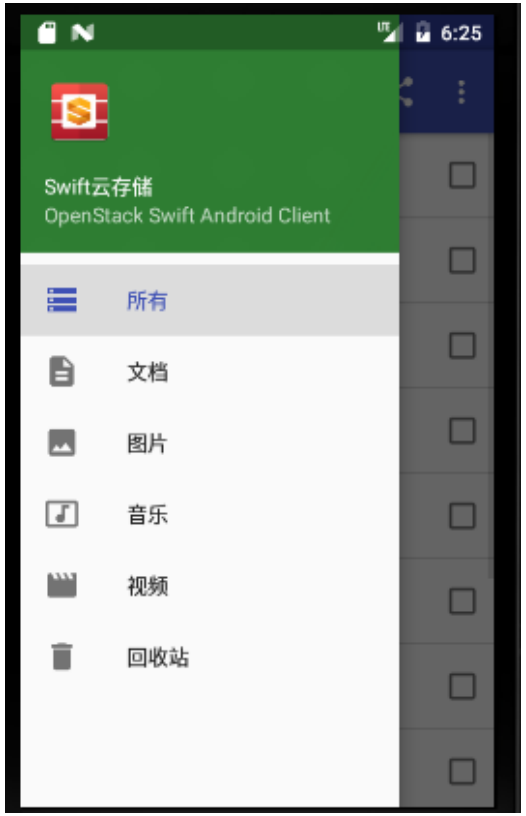
```
@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {//导航选择操作
    // Handle navigation view item clicks here.
    int id = item.getItemId();
    if (id == R.id.nav_swiftdisk) {}
    else {}
    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    drawer.closeDrawer(GravityCompat.START);
    return true;
}
```

4

实现导航Navigation Drawer

实现步骤

重复操作，完成所有后效果如下：



4 实现导航Navigation Drawer

实现步骤

（3）文字的添加

将文字内容直接写进菜单，不利于以后做国际化，更好的做法是将文字写进资源文件，然后在需要的时候引用。

打开资源文件“String.xml”，添加<string name="menu_swiftdisk">所有</string>，然后更改activity_main_drawer.xml 文件，将“所有”菜单项的“title”属性改为android:title="@string/menu_swiftdisk" />，代表这个文本内容引用自String资源中名称为menu_swiftdisk的资源。

其他文字内容相应的添加到String资源中。

4

实现导航Navigation Drawer

实现步骤

（4）导航头的制作

导航条由一张图片和两段文字构成，修改“nav_header_main.xml”文件的内容，完成制作。

4 实现导航Navigation Drawer

实现步骤

（5）Fragment的制作

可以把Fragment当成Activity一个界面的一部分，甚至Activity的界面由完全不同的Fragment组成，更帅气的是Fragment有自己的声明周期和接收、处理用户的事件，这样就不必要在一个Activity里面写一堆事件、控件的代码了。更为重要的是， 可以动态的添加、替换、移除某个Fragment。

使用Fragment最简单的一种方式，把Fragment当成普通的控件，直接写在Activity的布局文件中，用布局文件调用Fragment。

下面以创建图片的Fragment来举例说明Fragment的用法。

4 实现导航Navigation Drawer

实现步骤

创建一个ImgFragment的类

```
Public class ImgFragment extends Fragment implements SwipeRefreshLayout.OnRefreshListener,
        SFileListViewAdapter.ItemClickCallable, SFileEditable, View.OnClickListener {
}
```

继承Fragment，重写onCreateView决定Fragment布局

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    View view = inflater.inflate(R.layout.img_video, null);
    resources = getResources();
    InitTextView(view);
    InitViewPager(view);
    tvTabHot.setTextColor(resources.getColor(R.color.noselected));
    return view;
}
```

4

实现导航Navigation Drawer

实现步骤

在Activity中声明此Fragment，就当和普通的View一样

```
else if (id == R.id.nav_pic) {
    //调用图片的CategoryFragment并传入参数MIME_IMAGE
    if (picCategoryFragment == null) {
        picCategoryFragment = new ImgFragment();
        Bundle bundle = new Bundle();
        bundle.putStringArray(Constants.CATEGORY_TYPE, Constants.MIME_IMAGE);
        picCategoryFragment.setArguments(bundle);
    }
    fragmentManager.beginTransaction().replace(R.id.container, picCategoryFragment).commit();
    currentFragment = picCategoryFragment;
    //改变工具栏
    setToolbarTitles(getString(R.string.menu_pic), "");
}
```

5

实现Toolbar工具条

相关知识

Toolbar是在 Android 5.0 开始推出的一个 Material Design 风格的导航控件，Google 非常推荐大家使用 Toolbar 来作为Android客户端的导航栏，以此来取代之前的 Actionbar 。与 Actionbar 相比， Toolbar 明显要灵活的多。它不像 Actionbar 一样，一定要固定在Activity的顶部，而是可以放到界面的任意位置。

Toolbar 是在 Android 5.0 才开始加上的，Google 为了将这一设计向下兼容，自然也少不了要推出兼容版的 Toolbar 。为此，我们需要在工程中引入 appcompat-v7 的兼容包，使用 android.support.v7.widget.Toolbar 进行开发。

5 实现Toolbar工具条

实现步骤

（1）修改主菜单main.xml
编辑menu中main.xml的文件，输入代码如下

```
<item android:id="@+id/action_search"
    android:title="@string/action_search"
    android:icon="@drawable/ic_action_search"
    android:visible="true"
    app:showAsAction="collapseActionView|always"
    app:actionViewClass="android.support.v7.widget.SearchView"
    android:layout_width="wrap_content" />
<item android:id="@+id/action_sort"
    android:title="@string/action_sort"
    android:visible="true"
    app:showAsAction="never"
    android:icon="@drawable/ic_action_sort"
    android:layout_width="wrap_content" />
```

5 实现Toolbar工具条

```
<item android:id="@+id/action_share"
    android:title="@string/action_share"
    android:icon="@drawable/ic_action_share"
    android:visible="true"
    app:showAsAction="collapseActionView|always"
    android:layout_width="wrap_content" />
<item
    android:id="@+id/action_select_all"
    android:icon="@drawable/ic_action_select_all"
    android:title="@string/action_select_all"
    android:visible="true"
    app:showAsAction="never" />
```

添加相应的String资源代码如下

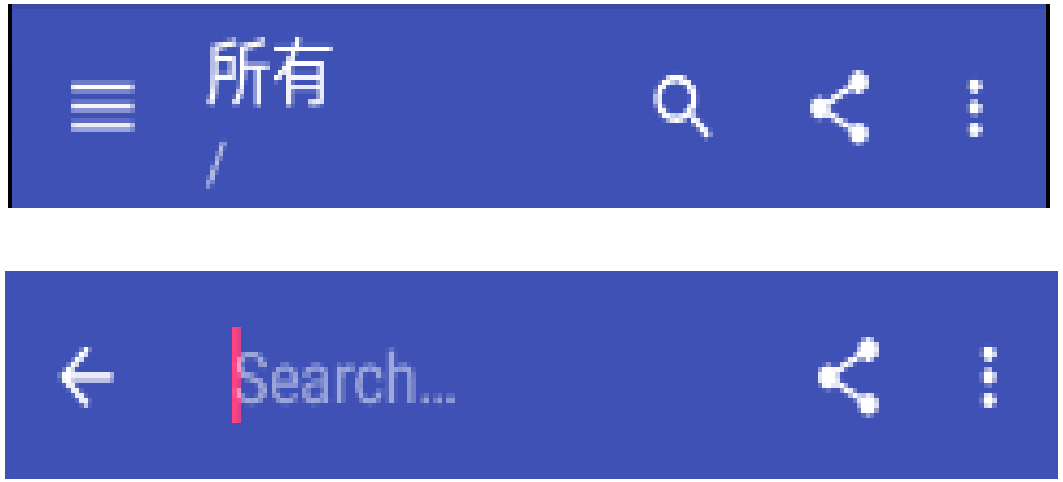
```
<string name="action_share">分享</string>
<string name="action_search">搜索</string>
<string name="action_select_all">全选</string>
<string name="action_sort">排序</string>
```

5

实现Toolbar工具条

实现步骤

（2）执行测试
效果如下



6 实现文件列表

相关知识

ListView组件在应用程序中可以说是不可或缺的一部分，ListView主要是显示列表数据，同时可以滚动查看，一个ListView通常有两个职责:（1）将数据填充到布局；（2）处理用户的选择点击等操作。

一个ListView的创建需要3个元素：

（1）ListView中的每一列的View。

（2）填入View的数据或者图片等。

（3）连接数据与ListView的适配器。

也就是说，要使用ListView，首先要了解什么是适配器。适配器是一个连接数据和AdapterView的桥梁，通过它能有效地实现数据与AdapterView的分离设置，使AdapterView与数据的绑定更加简便，修改更加方便。

6 实现文件列表

实现步骤

（1）创建SFileData
该类用于封装我们的文件数据，主要有文件名、大小等属性，为了便于管理，我们将这个文件存放在包model下。
（2）定义数据适配器类SFileListViewAdapter
该类用于将我们云存储上的文件添加到数据适配器中。该文件中，最重要的是完成以下几个重载方法。

public int getCount()
获取要填充到ListView中的数据总数
public Object getItem(int index)
获取指定索引的条目
public long getItemId(int position)
返回当前条目的索引
public View getView(final int position, View convertView, ViewGroup parent)
将数据填充到布局文件中，该方法获取布局文件，并将数据绑定到对应的控件上。

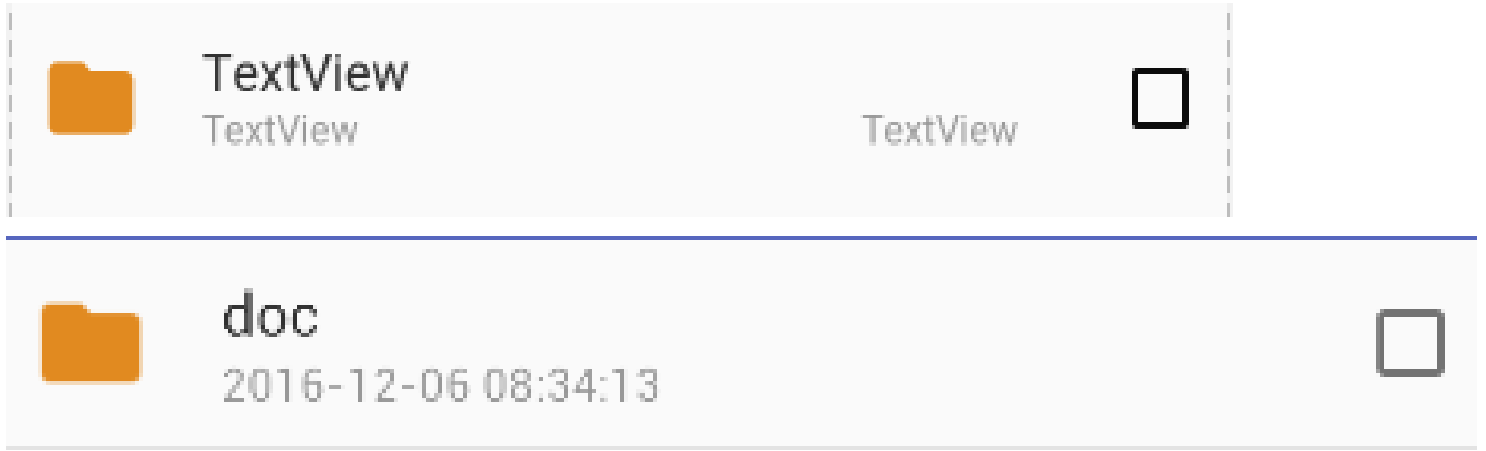
6

实现文件列表

实现步骤

（3）定义ListView布局文件main_list_item.xml
按照要求，设定好相应的布局文件，该布局文件将会被上面的getView方法加载，因此，我们的数据会被填充到相应的布局上。

（4）执行效果



7

弹出框、进度条

相关知识

1）弹出框的使用

AlertDialog的构造方法全部是Protected的，所以不能直接通过new一个AlertDialog来创建出一个AlertDialog。

使用AlertDialog.Builder创建对话框需要了解以下几个方法：

setTitle：为对话框设置标题

setIcon：为对话框设置图标

setMessage：为对话框设置内容

setView：给对话框设置自定义样式

setItems：设置对话框要显示的一个list，一般用于显示几个命令时

setMultiChoiceItems：用来设置对话框显示一系列的复选框

setNeutralButton ：普通按钮

setPositiveButton ：给对话框添加"Yes"按钮

setNegativeButton：对话框添加"No"按钮

create：创建对话框

show：显示对话框

弹出框、进度条

相关知识

2） 进度条的使用

在登录的过程中，可能会进行一些比较耗时的操作，最好为用户呈现操作的进度，避免用户因不知道执行的情况而焦虑，这就用到了进度条。

XML文件中有两个重要属性，其中android:progressBarStyle属性表示默认进度条样式，android:progressBarStyleHorizontal属性表示水平样式。

另外进度条还有以下几个比较重要的方法。

（1）getMax()方法返回这个进度条的范围的上限；

（2）getProgress()方法返回进度；

（3）getSecondaryProgress()方法返回次要进度；

（4）incrementProgressBy(int diff)方法指定增加的进度；

（5）isIndeterminate()方法判断指示进度条是否在不确定模式下；

（6）setIndeterminate(boolean indeterminate)方法用来设置是否在不确定模式下；

（7）setVisibility(int v)用来设置该进度条是否可视。

另外进度条还有一个重要事件，onSizeChanged(int w, int h, int oldw, int oldh)。当进度值改变时引发此事件。

7

弹出框、进度条

实现步骤

（1）下面代码实现了重命名功能的弹出对话框的功能

```
private AlertDialog renameDialog;
@Override
public void rename(String oldFilePath, String newFilePath) {
    //启动交互Dialog
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
    LayoutInflater inflater = getActivity().getLayoutInflater();
    View view = inflater.inflate(R.layout.input_text_edit_dialog, null);
    view.findViewById(R.id.btnEnter).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            EditText editText = (EditText) renameDialog.findViewById(R.id.edit_text);
            Toast.makeText(context,"您刚刚输入的是： "+editText.getText().toString(),Toast.LENGTH_SHORT).show();
renameDialog.dismiss();
        }
    });
```

7

弹出框、进度条

实现步骤

```
view.findViewById(R.id.btnCancel).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            renameDialog.cancel();
        }
    });
    builder.setTitle(R.string.title_dialog_rename);
    builder.setView(view);
    renameDialog = builder.create();
    renameDialog.show();
}
```

7

弹出框、进度条

实现步骤

（2）运行测试效果

8

异步任务模拟文档下载

相关知识

在使用AsyncTask时处理类需要继承AsyncTask，提供三个泛型参数，并且重载AsyncTask的四个方法（至少重载一个）。

三个泛型参数如下：

1）Param 任务执行器需要的数据类型

2）Progress 后台计算中使用的进度单位数据类型

3）Result 后台计算返回结果的数据类型

在设置参数时通常是这样的：String... params，这表示方法可以有0个或多个此类型参数；有时参数可以设置为不使用，用Void...即可。

8

异步任务模拟文档下载

相关知识

四个方法如下：

1）onPreExecute() 执行预处理，它运行于UI线程，可以为后台任务做一些准备工作，比如绘制一个进度条控件。

2）doInBackground(Params...) 后台进程执行的具体计算在这里实现， doInBackground(Params...)是AsyncTask的关键，此方法必须重载。在这个方法内可以使用publishProgress(Progress...)改变当前的进度值。

3）onProgressUpdate(Progress...) 运行于UI线程。如果在doInBackground(Params...) 中使用了publishProgress(Progress...)，就会触发这个方法。在这里可以对进度条控件根据进度值做出具体的响应。

4）onPostExecute(Result) 运行于UI线程，可以对后台任务的结果做出处理，结果就是doInBackground(Params...)的返回值。此方法也要经常重载，如果Result为null表明后台任务没有完成(被取消或者出现异常)。

8

异步任务模拟文档下载

相关知识

创建一个异步任务类DownloadTask，将进度条对象作为构造方法的参数传入，覆写doInBackground(Params...)方法。当调用publishProgress(i)方法时触发onProgressUpdate(Progress...) 方法，更新进度条的值，简要代码如下。

```
//该方法不运行在UI线程中,主要用于异步操作,通过调用publishProgress()方法
//触发onProgressUpdate对UI进行操作
@Override
protected String doInBackground(Integer... params) {
    DelayOperator dop = new DelayOperator();
    for (i = 10;i <= 100;i+=10)
    {
        dop.delay();
        publishProgress(i);
    }
    return null;
}
```