DailyLife Android应用

开发工具：Android studio  
开发语言：Java

开发框架：MVC

后端数据提供：Bmob云数据库

Bmob云数据库：

Bmob 提供了实时数据与文件存储功能，轻松实现应用“云与端”的数据连通。数据存储除了常规应用文本信息的存储，还可以存储图片、视频、音频、地理位置等信息。

使用 Bmob 几乎能免去所有服务器端编码工作量。

集成教程文档：<http://doc.bmobapp.com/data/android/index.html>

BaseActivity/BaseFragment：每一个Activity和Fragment都是很多模板代码的，为了减少模板代码，封装进Base类其实是一种比较方便且可行的选择.

当创建新的Activity或者Fragment的时候需要想想有没有逻辑可以复用，就去找Base类，或许写Base类的人不同，发现一个项目中可能会存在多个Base类，甚至Base类仍然有多个Base子类实现不同逻辑。这个时候就需要去查看分析每个Base类分别实现了什么功能，决定继承哪个。

如果一个项目中只有一个Base类的话，仍需要看看Base类实现了什么逻辑，没有实现什么逻辑，防止重复写样板代码。

当出现Base类实现了的，而自己本身并不想需要，例如不想监听广播或者不想用ViewModel，对于不想监听广播的情况就要特殊做适配，例如往Base类加标志位。对于不想用ViewModel但是由于泛型限制，还是只能传进去，不然没法继承。

当发现自己集成Base类出BUG了，就要考虑改子类还是改Base类，由于大量的类都集成了Base类，显然改Base类比较麻烦，于是改自己比较方便。

如果一个Activity中展示了多个Fragment，可能会有业务逻辑的重复，其实只需要一个就好了。

**OKHttp框架：**

HTTP是现代应用网络请求的方式，是一个简单的请求-响应协议。在数据和媒体中进行交换。有效地请求使您的东西加载更快，并节省带宽。OKHttp是安卓开发中十分常用的网络请求框架。

**为什么使用OKHttp**

      1：缓存响应数据来减少重复的网络请求

       2：可以从很多常用的连接问题中自动恢复；（Okhttp处理了很多的网络疑难杂症：会从很多常用的连接问题中自动恢复。如果您的服        务器配置了多个IP地址，当第一个IP连接失败时，OkHttp会尝试连接下一个IP。）

       3：支持gzip压缩响应体

       4：使用简单，可扩展性非常的强；（责任链模式使得很容易添加一个自定义拦截器对请求和返回结果进行处理）

**三：用来干什么**

      1：支持一般的get请求，post请求

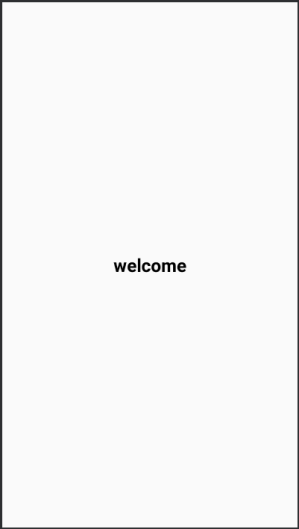
       2：基于Http的文件上传，文件下载，上传下载的进度回调

       3：图片加载

       4：支持请求回调，对象返回

       5:   表单请求

欢迎页面——SplashActivity



作为App的启动页面，并设置Activity为全屏

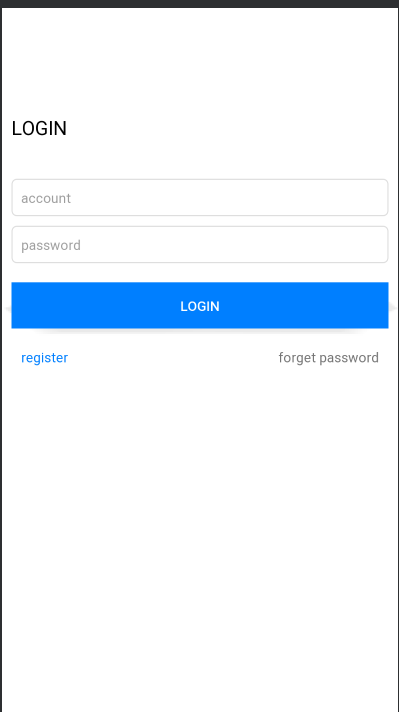
setContentView(R.layout.xxx)之前调用：

requestWindowFeature(Window.FEATURE\_NO\_TITLE);  
getWindow().setFlags(WindowManager.LayoutParams.FLAG\_FULLSCREEN, WindowManager.LayoutParams.FLAG\_FULLSCREEN);  
if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.KITKAT) {  
 getWindow().addFlags(WindowManager.LayoutParams.FLAG\_TRANSLUCENT\_STATUS);  
 getWindow().addFlags(WindowManager.LayoutParams.FLAG\_TRANSLUCENT\_NAVIGATION);  
}

并使用handler进行延时1秒跳转至登录页面LoginActivity

new Handler().postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 startActivity(new Intent(SplashActivity.this,LoginActivity.class));  
 }  
},1000);

**LoginActiv登录界面：**



用户可通过输入手机号和密码登录，手机号采用11位校验，并提供注册和忘记密码重设功能。

监听用户点击登录按钮事件：

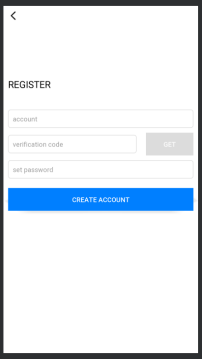
String phone = accountEdit.getText().toString();//获取用户输入的号  
String pwd = pwdEdit.getText().toString();//获取用户输入的密码  
  
if (TextUtils.isEmpty(phone)) {//判断输入的手机号是否为空  
 showToast("Please input phone");//提示用户手机号不能为空  
 return;  
}  
  
if (TextUtils.isEmpty(pwd)) {//判断输入的密码是否为空  
 showToast("Please input password");  
 return;  
}

//查询方法，先查询该手机号是否注册  
BmobQuery<User> query = new BmobQuery<>();  
query.addWhereEqualTo("phone", phone);//设置用户输入的手机  
query.findObjects(new FindListener<User>() {  
 @Override  
 public void done(List<User> list, BmobException e) {

//查询方法回调事件  
 if (e == null) {  
 if (list == null || list.size() == 0){//账号不存在即手机号未注册  
 Toast.makeText(mActivity, "The account does not exist", Toast.LENGTH\_SHORT).show();  
 }else {  
 User user = list.get(0);//账号存在 则判断密码是否输入正确  
 if (user != null){  
 if (pwd.equals(user.getPwd())){  
 Toast.makeText(mActivity, "Login success", Toast.LENGTH\_SHORT).show();  
 MyApplication.getInstance().setLoginUser(user);  
 SharedPreferenceUtil.putString(mContext,Constants.PHONE,phone);  
 startActivity(new Intent(mActivity,MainActivity.class));//登录成功跳转至主页  
 finish();  
 }else {

//提示密码输入错误  
 Toast.makeText(mActivity, "Password entered incorrectly", Toast.LENGTH\_SHORT).show();  
 }  
 }  
 }  
 } else {  
 Toast.makeText(mActivity, mActivity.getResources().getString(R.string.unknown\_error), Toast.LENGTH\_SHORT).show();  
 }  
 }  
});

**RegisterActivity注册页面:**



注册页面和重设密码页面公用一个页面，通过变量type判断，type等于0是注册页面，type等于1为重设密码。

通过用户输入的手机号获取短信验证码校验。

点击获取验证码按钮：

BmobSMS.requestSMSCode(phone, "DailyLife", new QueryListener<Integer>() {  
 @Override  
 public void done(Integer smsId, BmobException e) {  
 if (e == null) {

//获取成功，并开启倒计时功能  
 countDownTimer.start();  
 getCodeBtn.setBackgroundColor(mActivity.getResources().getColor(R.color.dcdcdc));  
 getCodeBtn.setEnabled(false);//倒计时中 获取验证码按钮不可用  
 if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.JELLY\_BEAN) {  
 getCodeBtn.setBackground(null);  
 }  
 Toast.makeText(mActivity, "The verification code is successfully obtained", Toast.LENGTH\_SHORT).show();  
 Log.e(TAG, "done: " + "ID：" + smsId + "\n");  
 } else {  
 Toast.makeText(mActivity, "Failed to obtain the verification code.", Toast.LENGTH\_SHORT).show();  
 Log.e(TAG, "done: " + "Failed：" + e.getErrorCode() + "-" + e.getMessage() + "\n");  
 }  
 }  
});

监听事件：

if (TextUtils.isEmpty(phone)) {  
 showToast("Please input phone");  
 return;  
}  
if (TextUtils.isEmpty(code)) {  
 showToast("Please input code");  
 return;  
}  
if (TextUtils.isEmpty(password)) {  
 showToast("Please input password");  
 return;  
}

//先查询该手机号是否已经注册  
BmobQuery<User> query = new BmobQuery<>();  
query.addWhereEqualTo("phone", phone);  
query.findObjects(new FindListener<User>() {  
 @Override  
 public void done(List<User> list, BmobException e) {  
 if (e == null) {  
 if (type == 1) {//如果为重设密码状态  
 if (list == null || list.size() == 0) { //重设密码时 账号不存在  
 Toast.makeText(mActivity, "The account does not exist", Toast.LENGTH\_SHORT).show();  
 } else {  
 User user = list.get(0);//重设密码 账号存在  
 if (user != null) {  
 user.setPwd(password);//重新设置密码

//调用更新方法 进行更新密码操作  
 user.update(new UpdateListener() {  
 @Override  
 public void done(BmobException e) {  
 if (e == null) {  
 Toast.makeText(mActivity, "Succeeded in resetting the password.", Toast.LENGTH\_SHORT).show();  
 SharedPreferenceUtil.putString(mActivity, Constants.PHONE, phone);  
 finish();  
 } else {  
 Toast.makeText(mActivity, mActivity.getResources().getString(R.string.unknown\_error), Toast.LENGTH\_SHORT).show();  
 }  
 }  
 });  
 }  
 }  
 } else {  
 if (list == null || list.size() == 0) {  
 CheckVerificationCode(phone, password, code);  
 } else {  
 Toast.makeText(mActivity, "The phone number has been registered", Toast.LENGTH\_SHORT).show();  
 }  
 }  
 } else {  
 Toast.makeText(mActivity, mActivity.getResources().getString(R.string.unknown\_error), Toast.LENGTH\_SHORT).show();  
 }  
 }  
});

**验证短信验证码是否正确**

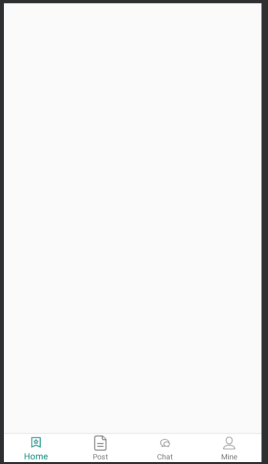
private void CheckVerificationCode(String phone, String password, String code,User user) {

//  
 BmobSMS.verifySmsCode(phone, code, new UpdateListener() {  
 @Override  
 public void done(BmobException e) {  
 if (e == null) {  
 if (type == 1){   
 if (user != null) {  
 user.setPwd(password);//重新设置密码  
 //调用更新方法  
 user.update(new UpdateListener() {  
 @Override  
 public void done(BmobException e) {  
 if (e == null) {  
 Toast.makeText(mActivity, "Succeeded in resetting the password.", Toast.LENGTH\_SHORT).show();  
 SharedPreferenceUtil.putString(mActivity, Constants.PHONE, phone);  
 finish();  
 } else {  
 Toast.makeText(mActivity, mActivity.getResources().getString(R.string.unknown\_error), Toast.LENGTH\_SHORT).show();  
 }  
 }  
 });  
 }  
 }else {

//创建新用户  
 User user = new User();  
 user.setPhone(phone);  
 user.setPwd(password);

//调用保存方法  
 user.save(new SaveListener<String>() {  
 @Override  
 public void done(String s, BmobException e) {  
 if (e == null) {  
 Toast.makeText(mActivity, "Successful registration", Toast.LENGTH\_SHORT).show();  
 SharedPreferenceUtil.putString(mActivity, Constants.PHONE, phone);  
 finish();  
 } else {  
 Toast.makeText(mActivity, "Registration failure", Toast.LENGTH\_SHORT).show();  
 }  
 }  
 });  
 }  
  
   
 } else {  
 Toast.makeText(mActivity, "Verification code error", Toast.LENGTH\_SHORT).show();  
 //mTvInfo.append("Verification code error" + e.getErrorCode() + "-" + e.getMessage() + "\n");  
 }  
 }  
 });  
}

**MainActivity 主页面：**

  
主页面采用viewpager+bottomnavigation实现底部导航栏和左右页面联动。

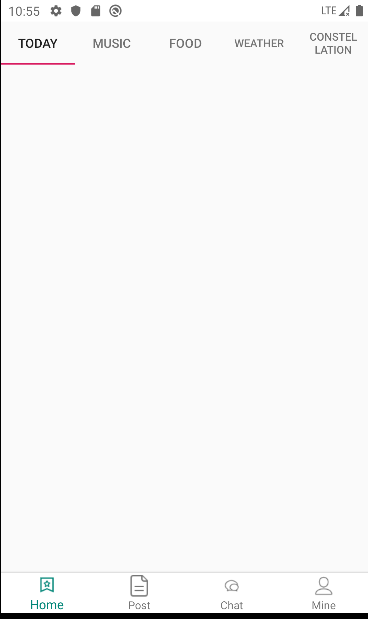
MainActivity.java代码：

public class MainActivity extends BaseActivity {  
  
 @BindView(R.id.view\_pager)  
 ViewPager viewPager;  
 @BindView(R.id.bottom\_navigation)  
 BottomNavigationView bottomNavigation;  
  
 private HomeFragment homeFragment;  
 private PostFragment postFragment;  
 private ChatFragment chatFragment;  
 private MineFragment mineFragment;  
  
 private List<Fragment> mFragments;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 //setContentView(R.layout.activity\_main);  
 }  
  
 @Override  
 protected int setLayout() {  
 return R.layout.activity\_main;  
 }  
  
 @Override  
 protected void initView() {  
 if (isDark) {  
 ImmersionBar.with(mActivity).statusBarColor(R.color.white).statusBarDarkFont(false).init();  
 } else {  
 ImmersionBar.with(mActivity).statusBarColor(R.color.white).statusBarDarkFont(true).init();  
 }

//实例化四个fragment  
 homeFragment = new HomeFragment();  
 postFragment = new PostFragment();  
 chatFragment = new ChatFragment();  
 mineFragment = new MineFragment();  
 mFragments = new ArrayList<>();//初始化fragment容器

//将四个fragment添加到容器中  
 mFragments.add(homeFragment);  
 mFragments.add(postFragment);  
 mFragments.add(chatFragment);  
 mFragments.add(mineFragment);  
 //viewpager绑定适配器  
 viewPager.setAdapter(new FragmentsPagerAdapter(getSupportFragmentManager()));  
 // navigationView点击事件  
 bottomNavigation.setOnNavigationItemSelectedListener(menuItem -> {  
 // menu文件夹中bottom\_nav\_menu.xml里加的android:orderInCategory属性就是下面item.getOrder()取的值  
 viewPager.setCurrentItem(menuItem.getOrder());//与viewpager绑定  
 return true;  
  
 });  
  
 /\*\*  
 \* 监听ViewPager页面变化事件  
 \* \*/  
 viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {  
 @Override  
 public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {  
  
 }  
  
 @Override  
 public void onPageSelected(int position) {  
 bottomNavigation.getMenu().getItem(position).setChecked(true);//与底部导航绑定  
 }  
  
 @Override  
 public void onPageScrollStateChanged(int state) {  
  
 }  
 });  
 }  
  
 public class FragmentsPagerAdapter extends FragmentPagerAdapter{  
 private FragmentManager mFragmentManager;  
 public FragmentsPagerAdapter(FragmentManager fm) {  
 super(fm);  
 mFragmentManager = fm;  
 }  
 @Override  
 public Fragment getItem(int position) {  
 return mFragments.get(position);  
 }  
 @Override  
 public int getCount() {  
 return mFragments.size();  
 }  
 }  
  
  
}

**homeFragment：**



Home界面使用Tablayout+viewpager实现多页面左右滑动，TabLayout作为导航组件来说，使用场景非常的多，也意味着要满足各种各样的需求。

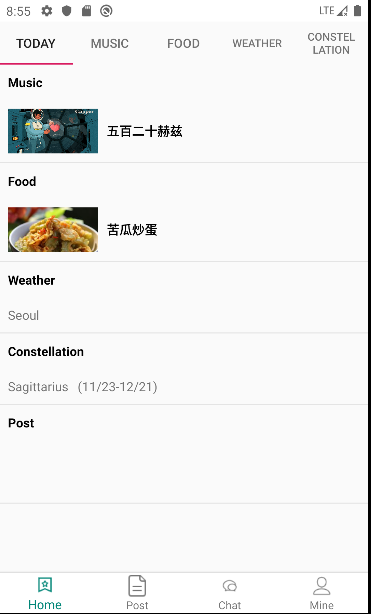
完整代码：

public class HomeFragment extends BaseFragment {  
 @BindView(R.id.tab\_layout)  
 TabLayout tabLayout;  
 @BindView(R.id.view\_pager)  
 ViewPager viewPager;  
  
 private List<Fragment> mFragments;//fragment容器  
 private ArrayList<String> mTitle;//Tablayout标题容器  
  
   
  
 @Override  
 protected int getLayoutId() {  
 return R.layout.fragment\_home;  
 }  
  
 @Override  
 protected void initEventAndData() {  
 mFragments = new ArrayList<>();//初始化fragment容器

//将子fragment添加至容器  
 mFragments.add(TodayFragment.newInstance());  
 mFragments.add(MusicFragment.newInstance());  
 mFragments.add(FoodFragment.newInstance());  
 mFragments.add(WeatherFragment.newInstance());  
 mFragments.add(ConstellationFragment.newInstance());  
  
 mTitle = new ArrayList<>();

//将标题添加至容器  
 mTitle.add("Today");  
 mTitle.add("Music");  
 mTitle.add("Food");  
 mTitle.add("Weather");  
 mTitle.add("Constellation");  
  
  
 /\*\*  
 \* 预加载  
 \*/  
 viewPager.setOffscreenPageLimit(mFragments.size());  
 /\*\*  
 \* 设置适配器  
 \*/  
 viewPager.setAdapter(new FragmentPagerAdapter(getChildFragmentManager()) {  
 @Override  
 public Fragment getItem(int position) {  
 return mFragments.get(position);  
 }  
 @Override  
 public int getCount() {  
 return mFragments.size();  
 }  
 @Override  
 public CharSequence getPageTitle(int position) {  
 return mTitle.get(position);  
 }  
 });  
 tabLayout.setupWithViewPager(viewPager);//将Tablayout与viewpager进行关联  
  
 }  
  
  
 public class FragmentsPagerAdapter extends FragmentPagerAdapter {  
 private FragmentManager mFragmentManager;  
 public FragmentsPagerAdapter(FragmentManager fm) {  
 super(fm);  
 mFragmentManager = fm;  
 }  
 @Override  
 public Fragment getItem(int position) {  
 return mFragments.get(position);  
 }  
 @Override  
 public int getCount() {  
 return mFragments.size();  
 }  
 }  
  
}

**TodayFragment（今日咨询推荐页面）：**



页面用于各项推荐数据，

使用OkHttp网络框架请求数据，并对返回的json数据用GSON解析。

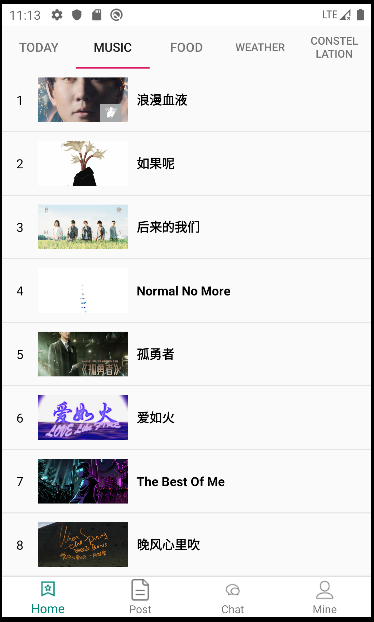
public class TodayFragment extends BaseFragment {  
  
 @BindView(R.id.recycler\_view)  
 RecyclerView recyclerView;  
  
 NewsListAdapter newsListAdapter;  
  
 public static TodayFragment newInstance() {  
 TodayFragment fragment = new TodayFragment();  
 Bundle args = new Bundle();  
  
 fragment.setArguments(args);  
 return fragment;  
 }  
  
  
 @Override  
 protected int getLayoutId() {  
 return R.layout.fragment\_today;  
 }  
  
 @Override  
 protected void initEventAndData() {

//设置列表的加载方向：纵向  
 recyclerView.setLayoutManager(new LinearLayoutManager(mContext));  
 newsListAdapter = new NewsListAdapter();  
 newsListAdapter.bindToRecyclerView(recyclerView);  
 newsListAdapter.setOnItemClickListener((adapter, view, position) -> {  
 NewsData bean = newsListAdapter.getItem(position);  
 if (bean == null)return;  
 Intent intent = new Intent(mContext, NewsDetailsActivity.class);  
 intent.putExtra(Constants.ID,bean.getDocid());  
 intent.putExtra(Constants.TITLE,bean.getTitle());  
 startActivity(intent);  
  
 });  
 getNewsDatat();  
  
 }  
  
 private void getNewsDatat() {  
 //1.拿到okHttpClient对象,可以设置连接超时等  
 OkHttpClient okHttpClient = new OkHttpClient();  
  
 //2.构造Request请求对象，可以增加头addHeader等  
 Request.Builder builder = new Request.Builder();  
 //url()中可以放入网址  
 Request request = builder.  
 get().  
 url("https://v2.alapi.cn/api/new/toutiao?start=1&token="+ MyApplication.getInstance().getToken())  
 .build();  
 //3.将Request封装为Call  
 Call call = okHttpClient.newCall(request);  
 //4.执行call  
 //方法一Response response=call.execute();//汇抛出IO异常，同步方法  
 //方法二,异步方法，放到队列中,处于子线程中，无法更新UI  
 call.enqueue(new Callback() {  
 //请求时失败时调用  
 @Override  
 public void onFailure(Call call, IOException e) {  
  
 }  
  
 //请求成功时调用  
 @Override  
 public void onResponse(Call call, Response response) throws IOException {  
 //处于子线程中，能够进行大文件下载，但是无法更新UI  
 final String res = response.body().string();//请求成功时返回的东西  
 Log.e(TAG, "onResponse: " + res);  
  
 //InputStream is=response.body().byteStream();  
 // 执行IO操作时，能够下载很大的文件，并且不会占用很大内存  
 /\*\*  
 \* runOnUiThread方法切换到主线程中，或者用handler机制也可以  
 \*/  
  
 NewsRoot newsRoot = new Gson().fromJson(res, NewsRoot.class);  
 mActivity.runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 // 更新ui  
 if (newsRoot != null && newsRoot.getData() != null && newsListAdapter != null) {  
 newsListAdapter.setNewData(newsRoot.getData());  
 }  
 }  
 });  
  
 }  
 });  
 }  
  
}

**NewsDetailsActivity（详情页面）**

public class NewsDetailsActivity extends BaseActivity {  
  
 @BindView(R.id.back\_img)  
 ImageView backImg;  
 @BindView(R.id.title\_tv)  
 TextView titleTv;  
 @BindView(R.id.right\_tv)  
 TextView rightTv;  
 @BindView(R.id.content\_tv)  
 TextView contentTv;  
 @BindView(R.id.news\_title\_tv)  
 TextView newsTitleTv;  
 @BindView(R.id.info\_tv)  
 TextView infoTv;  
  
 private String docId;//新闻ID  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 //setContentView(R.layout.activity\_news\_details);  
 }  
  
 @Override  
 protected void initView() {  
 if (isDark) {  
 ImmersionBar.with(mActivity).statusBarColor(R.color.white).statusBarDarkFont(false).init();  
 } else {  
 ImmersionBar.with(mActivity).statusBarColor(R.color.white).statusBarDarkFont(true).init();  
 }  
  
 docId = getIntent().getStringExtra(Constants.ID);//接收上个页面传递过来的数据  
 newsTitleTv.setText(getIntent().getStringExtra(Constants.TITLE));  
  
 //1.拿到okHttpClient对象,可以设置连接超时等  
 OkHttpClient okHttpClient = new OkHttpClient();  
  
 //2.构造Request请求对象，可以增加头addHeader等  
 Request.Builder builder = new Request.Builder();  
 //url()中可以放入网址  
 Request request = builder.  
 get().  
 url("https://v2.alapi.cn/api/new/detail?docid=" + docId + "&token="+ MyApplication.getInstance().getToken())  
 .build();  
 //3.将Request封装为Call  
 Call call = okHttpClient.newCall(request);  
 //4.执行call  
 //方法一Response response=call.execute();//汇抛出IO异常，同步方法  
 //方法二,异步方法，放到队列中,处于子线程中，无法更新UI  
 call.enqueue(new Callback() {  
 //请求时失败时调用  
 @Override  
 public void onFailure(Call call, IOException e) {  
  
 }  
  
 //请求成功时调用  
 @Override  
 public void onResponse(Call call, Response response) throws IOException {  
 //处于子线程中，能够进行大文件下载，但是无法更新UI  
 final String res = response.body().string();//请求成功时返回的东西  
 //InputStream is=response.body().byteStream();  
 // 执行IO操作时，能够下载很大的文件，并且不会占用很大内存  
 /\*\*  
 \* runOnUiThread方法切换到主线程中，或者用handler机制也可以  
 \*/  
 Log.e(TAG, "onResponse: "+res);  
 NewDetailsRoot newDetailsRoot = new Gson().fromJson(res, NewDetailsRoot.class);  
 if (newDetailsRoot != null && newDetailsRoot.getData() != null) {  
 String str = newDetailsRoot.getData().getBody();  
 String info = "来自："+newDetailsRoot.getData().getSource()+" "+newDetailsRoot.getData().getPtime()+"\n"+newDetailsRoot.getData().getCategory();  
 if (newDetailsRoot.getData().getImg() != null && newDetailsRoot.getData().getImg().size() > 0) {  
 for (NewDetailsData.Img img : newDetailsRoot.getData().getImg()) {  
 if (str.contains(img.getRef())) {  
 String[] ss = img.getPixel().split("\\\*");  
 str = str.replaceAll(img.getRef(), "<img height=\\\"" + ss[0] + "px\\\" width=\\\"100%\\\" src=\\\"" + img.getSrc() + "\\\"/>");  
 }  
 }  
 Log.e(TAG, "run: " + str);  
 }  
 final Spanned sp = Html.fromHtml(str, source -> {  
 InputStream is;  
 try {  
 is = (InputStream) new URL(source).getContent();  
 Drawable d = Drawable.createFromStream(is, "src");  
 d.setBounds(0, 0, d.getIntrinsicWidth(),  
 d.getIntrinsicHeight());  
 is.close();  
 return d;  
 } catch (Exception e) {  
 return null;  
 }  
 }, null);  
 mActivity.runOnUiThread(() -> {  
 // 更新ui  
 contentTv.setText(sp);  
 infoTv.setText(info);  
  
 });  
 }  
 }  
 });  
  
  
 }  
  
 @Override  
 protected int setLayout() {  
 return R.layout.activity\_news\_details;  
 }  
  
 @OnClick({R.id.back\_img, R.id.title\_tv})  
 public void onViewClicked(View view) {  
 switch (view.getId()) {  
 case R.id.back\_img:  
 finish(); //返回键 直接退出页面  
 break;  
 case R.id.title\_tv:  
 break;  
 }  
 }  
  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 finish();  
 }  
}

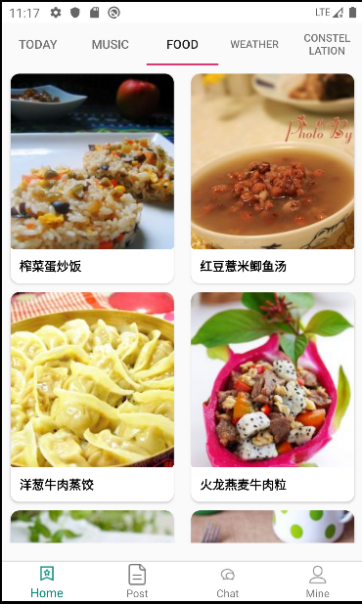
**MusicFragment（音乐页面）：**



使用公用免费API查询 网易云音乐热歌榜，url地址："https://api.uomg.com/api/rand.music?sort=热歌榜&format=json"

public class MusicFragment extends BaseFragment {  
  
 @BindView(R.id.recycler\_view)  
 RecyclerView recyclerView;  
  
 private MusicListAdapter musicListAdapter;  
 private int pageSize = 1;  
 private Set<String> stringSet = new HashSet<>();  
  
 public static MusicFragment newInstance() {  
 MusicFragment fragment = new MusicFragment();  
 Bundle args = new Bundle();  
  
 fragment.setArguments(args);  
 return fragment;  
 }  
  
  
 @Override  
 protected int getLayoutId() {  
 return R.layout.fragment\_music;  
 }  
  
 @Override  
 protected void initEventAndData() {  
  
  
 recyclerView.setLayoutManager(new LinearLayoutManager(mContext));  
 musicListAdapter = new MusicListAdapter();  
 musicListAdapter.bindToRecyclerView(recyclerView);  
 musicListAdapter.setOnItemClickListener((adapter, view, position) -> {  
 MusicData bean = musicListAdapter.getItem(position);  
 if (bean == null)return;  
 Uri uri = Uri.parse(bean.getUrl());  
 Intent intent = new Intent(Intent.ACTION\_VIEW, uri);  
 startActivity(intent);  
 });  
  
 getMusicData();  
  
  
 }  
  
 private void getMusicData() {  
 //1.拿到okHttpClient对象,可以设置连接超时等  
 OkHttpClient okHttpClient = new OkHttpClient();  
 //2.构造Request请求对象，可以增加头addHeader等  
 Request.Builder builder = new Request.Builder();  
 //url()中可以放入网址  
 //使用公用免费API查询 网易云音乐热歌榜  
 Request request = builder.  
 get().  
 url("https://api.uomg.com/api/rand.music?sort=热歌榜&format=json")  
 .build();  
 //3.将Request封装为Call  
 Call call = okHttpClient.newCall(request);  
 //4.执行call  
 //方法一Response response=call.execute();//汇抛出IO异常，同步方法  
 //方法二,异步方法，放到队列中,处于子线程中，无法更新UI  
 call.enqueue(new Callback() {  
 //请求时失败时调用  
 @Override  
 public void onFailure(Call call, IOException e) {  
  
 }  
  
 //请求成功时调用  
 @Override  
 public void onResponse(Call call, Response response) throws IOException {  
 //处于子线程中，能够进行大文件下载，但是无法更新UI  
 final String res = response.body().string();//请求成功时返回的东西  
 //InputStream is=response.body().byteStream();  
 // 执行IO操作时，能够下载很大的文件，并且不会占用很大内存  
 /\*\*  
 \* runOnUiThread方法切换到主线程中，或者用handler机制也可以  
 \*/  
 Log.e(TAG, "onResponse: "+res);  
 MusicRoot musicRoot = new Gson().fromJson(res, MusicRoot.class);  
  
 if (musicRoot != null && musicRoot.getData() != null) {  
 mActivity.runOnUiThread(() -> {  
 if (stringSet.add(musicRoot.getData().getName())){  
 musicListAdapter.addData(musicRoot.getData());  
 pageSize++;  
 }  
 if (pageSize <= 20){ //只查询前20首  
 getMusicData();  
 }  
 });  
 }  
 }  
 });  
 }  
  
  
}

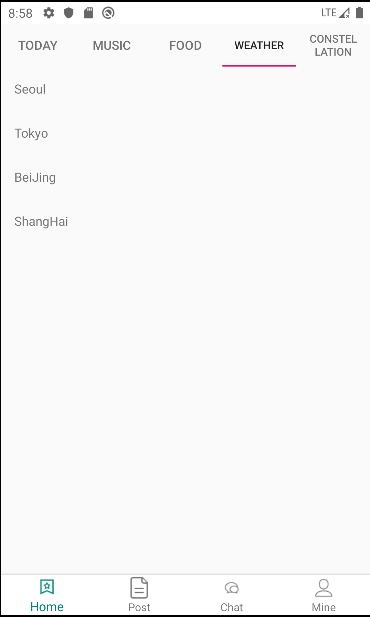
**FoodFragment（美食页面）**



美食页面采用recyclerview展示数据，并使用GridlayouMananger实现网格列表布局。

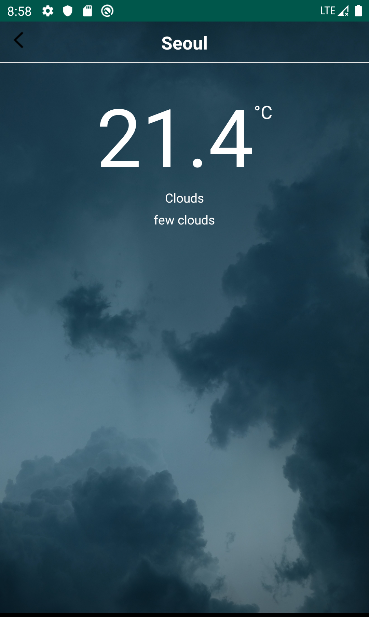
public class FoodFragment extends BaseFragment {  
  
 @BindView(R.id.recycler\_view)  
 RecyclerView recyclerView;  
 private String[] keywords = {"蛋","鱼","豆腐","牛肉"};  
 private int num = 10;  
 private FoodListAdapter foodListAdapter;  
  
 public static FoodFragment newInstance() {  
 FoodFragment fragment = new FoodFragment();  
 Bundle args = new Bundle();  
  
 fragment.setArguments(args);  
 return fragment;  
 }  
  
  
 @Override  
 protected int getLayoutId() {  
 return R.layout.fragment\_food;  
 }  
  
 @Override  
 protected void initEventAndData() {  
 //使用网格布局，一行2列  
 recyclerView.setLayoutManager(new GridLayoutManager(mContext, 2));  
 foodListAdapter = new FoodListAdapter();  
 foodListAdapter.bindToRecyclerView(recyclerView);  
 foodListAdapter.setOnItemClickListener((adapter, view, position) -> {  
 Food food = foodListAdapter.getItem(position);  
 if (food == null)return;  
 Intent intent = new Intent(mContext, FoodDetilsActivity.class);  
 intent.putExtra(Constants.DATA,food);  
 startActivity(intent);  
  
 });  
 getBmobData();  
  
  
 //getData();  
  
 }  
  
 private void getBmobData() {  
  
 BmobQuery<Food> query = new BmobQuery<>();  
 query.order("-createdAt")//查找所有数据，并根据事件降序排列  
 .findObjects(new FindListener<Food>() {  
 @Override  
 public void done(List<Food> object, BmobException e) {  
 if (e == null) {  
 if (object != null){  
 Collections.shuffle(object);//随机打乱顺序  
 foodListAdapter.setNewData(object);  
 }  
 } else {  
  
 }  
 }  
 });  
 }  
  
 private void getData() {  
 //1.拿到okHttpClient对象,可以设置连接超时等  
 OkHttpClient okHttpClient = new OkHttpClient();  
 //2.构造Request请求对象，可以增加头addHeader等  
 Request.Builder builder = new Request.Builder();  
 String url = "https://way.jd.com/jisuapi/search?keyword="+keywords[3]  
 /\*+ keywords[new Random().nextInt(keywords.length)]\*/ +  
 "&num=" + num + "&appkey=" + MyApplication.getInstance().getApiKey();  
 Log.e(TAG, "url: "+url);  
 //url()中可以放入网址  
 Request request = builder.  
 get().url(url)  
 .build();  
 //3.将Request封装为Call  
 Call call = okHttpClient.newCall(request);  
 //4.执行call  
 //方法一Response response=call.execute();//汇抛出IO异常，同步方法  
 //方法二,异步方法，放到队列中,处于子线程中，无法更新UI  
 call.enqueue(new Callback() {  
 //请求时失败时调用  
 @Override  
 public void onFailure(Call call, IOException e) {  
  
 }  
  
 //请求成功时调用  
 @Override  
 public void onResponse(Call call, Response response) throws IOException {  
 //处于子线程中，能够进行大文件下载，但是无法更新UI  
 final String res = response.body().string();//请求成功时返回的东西  
 //InputStream is=response.body().byteStream();  
 // 执行IO操作时，能够下载很大的文件，并且不会占用很大内存  
 /\*\*  
 \* runOnUiThread方法切换到主线程中，或者用handler机制也可以  
 \*/  
 Log.e(TAG, "onResponse: " + res);  
 setData(res);  
 }  
 });  
 }  
  
 private void setData(String res) {  
 FoodRoot foodRoot = new Gson().fromJson(res, FoodRoot.class);  
 if (foodRoot != null && foodRoot.getResult() != null && foodRoot.getResult().getResult() != null  
 && foodRoot.getResult().getResult().getList() != null) {  
 mActivity.runOnUiThread(() -> {  
  
 foodListAdapter.addData(foodRoot.getResult().getResult().getList());  
 });  
 }  
 }  
  
  
}

WeatherFragment：（天气页面）



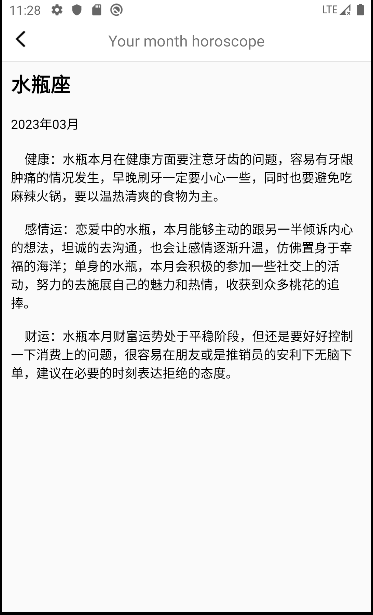
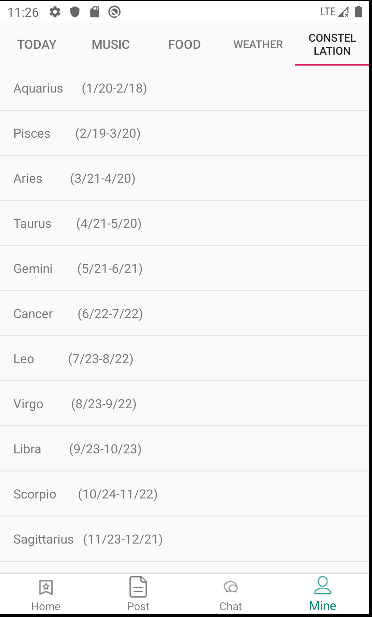
用列表展示城市列表数据，点击某个城市跳转进入具体城市的天气详情页面：

WeatherActivity：使用Handler机制更新UI



public class WeaterActivity extends BaseActivity {  
  
  
 @BindView(R.id.back\_img)  
 ImageView backImg;  
 @BindView(R.id.title\_tv)  
 TextView titleTv;  
 @BindView(R.id.right\_tv)  
 TextView rightTv;  
 @BindView(R.id.city\_tv)  
 TextView cityTv;  
 @BindView(R.id.center\_tv)  
 TextView centerTv;  
 @BindView(R.id.wind\_tv)  
 TextView windTv;  
 @BindView(R.id.humidity\_tv)  
 TextView humidityTv;  
 @BindView(R.id.list\_view)  
 ListView listView;  
 @BindView(R.id.fragment\_layout)  
 FrameLayout frameLayout;  
  
  
 //通过Handler 机制循环取出数据  
 @SuppressLint("HandlerLeak")  
 private Handler handler = new Handler() {  
  
 @Override  
 public void handleMessage(@NonNull Message msg) {  
 super.handleMessage(msg);  
 if (msg.what == 0) {  
 if (jsonRootBean != null) {  
 if (jsonRootBean.getResult() != null) {  
 if (jsonRootBean.getResult().getRealtime() != null) {  
 centerTv.setText(jsonRootBean.getResult().getRealtime().getTemperature());  
 windTv.setText(jsonRootBean.getResult().getRealtime().getDirect() + " " +  
 jsonRootBean.getResult().getRealtime().getPower());  
 humidityTv.setText("湿度 " + jsonRootBean.getResult().getRealtime().getWid());  
 if ("晴".equals(jsonRootBean.getResult().getRealtime().getInfo())) {  
 frameLayout.setBackgroundResource(R.drawable.ic\_sunny);  
 }  
 if ("阴".equals(jsonRootBean.getResult().getRealtime().getInfo())) {  
 frameLayout.setBackgroundResource(R.drawable.ic\_cloudy\_day);  
 }  
 if (!TextUtils.isEmpty(jsonRootBean.getResult().getRealtime().getInfo()) &&  
 jsonRootBean.getResult().getRealtime().getInfo().contains("雨")) {  
 frameLayout.setBackgroundResource(R.drawable.ic\_rain);  
 }  
 }  
 if (jsonRootBean.getResult().getFuture() != null) {  
 //将数据和context通过自定义Adapter的构造方法传给Adapter  
 MyAdapter adapter = new MyAdapter(jsonRootBean.getResult().getFuture(), mActivity);  
 listView.setAdapter(adapter);  
 }  
  
 }  
  
 }  
 }  
 }  
  
  
 };  
 private String city;  
 private String key = "639e250e02d5dd2b4f6647f817a3680b";  
 private JsonRootBean jsonRootBean;  
  
  
 @Override  
 protected void initView() {  
 city = getIntent().getStringExtra(Constants.DATA);  
 cityTv.setText(city);  
 getData(city);  
 }  
  
  
 private void getData(String city) {  
  
 //1.拿到okHttpClient对象,可以设置连接超时等  
 OkHttpClient okHttpClient = new OkHttpClient();  
 //2.构造Request请求对象，可以增加头addHeader等  
 Request.Builder builder = new Request.Builder();  
 String url = "http://apis.juhe.cn/simpleWeather/query?city=" + city + "&key=" + key;  
 Log.e(TAG, "url: " + url);  
 //url()中可以放入网址  
 Request request = builder.  
 get().url(url)  
 .build();  
 //3.将Request封装为Call  
 Call call = okHttpClient.newCall(request);  
 //4.执行call  
 //方法一Response response=call.execute();//汇抛出IO异常，同步方法  
 //方法二,异步方法，放到队列中,处于子线程中，无法更新UI  
 call.enqueue(new Callback() {  
 //请求时失败时调用  
 @Override  
 public void onFailure(Call call, IOException e) {  
 Log.e("onFailure", e.getMessage());  
  
 }  
  
 //请求成功时调用  
 @Override  
 public void onResponse(Call call, Response response) throws IOException {  
 //处于子线程中，能够进行大文件下载，但是无法更新UI  
 /\*\*  
 \* runOnUiThread方法切换到主线程中，或者用handler机制也可以  
 \*/  
 final String result = response.body().string();//请求成功时返回的东西  
 Log.e("onResponse", result);  
 if (!TextUtils.isEmpty(result)) {  
 jsonRootBean = new Gson().fromJson(result, JsonRootBean.class);  
 //定义消息对象  
 Message message = new Message();  
 message.what = 0;  
 handler.sendMessage(message);  
 }  
  
 }  
 });  
 }  
  
 @Override  
 protected int setLayout() {  
 return R.layout.activity\_weater;  
 }  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 }  
  
 @OnClick({R.id.back\_img, R.id.title\_tv})  
 public void onViewClicked(View view) {  
 switch (view.getId()) {  
 case R.id.back\_img:  
 finish();  
 break;  
 case R.id.title\_tv:  
 break;  
 }  
 }  
  
 @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
 finish();  
 }  
  
  
 public class MyAdapter extends BaseAdapter {  
 private List<Future> productions = null;  
 private Context context = null;  
  
 public MyAdapter(List<Future> productions, Context context) {  
 super();  
 this.productions = productions;  
 this.context = context;  
 }  
  
 @Override  
 public int getCount() {  
 return productions.size();  
 }  
  
 @Override  
 public Object getItem(int position) {  
 return productions.get(position);  
 }  
  
 @Override  
 public long getItemId(int position) {  
 return position;  
 }  
  
 @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
 //获取到ListView的子项  
 convertView = LayoutInflater.from(context).inflate(R.layout.list\_view\_item, null);  
  
 TextView tv\_left = convertView.findViewById(R.id.left\_tv);  
 TextView tv\_right = convertView.findViewById(R.id.right\_tv);  
 //待组装的每一个item的bean，从bean中取出数据  
 Future future = productions.get(position);  
 if (position == 0) {  
 tv\_left.setText("今天·" + future.getWeather());  
 tv\_right.setText(future.getTemperature());  
 } else if (position == 1) {  
 tv\_left.setText("明天·" + future.getWeather());  
 tv\_right.setText(future.getTemperature());  
 } else if (position == 2) {  
 tv\_left.setText("后天·" + future.getWeather());  
 tv\_right.setText(future.getTemperature());  
 } else {  
 tv\_left.setText(future.getDate() + "·" + future.getWeather());  
 tv\_right.setText(future.getTemperature());  
 }  
 /\* purchase.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Toast.makeText(context, "点的是第" + (position + 1) + "行", Toast.LENGTH\_LONG).show();  
 }  
 });\*/  
 return convertView;  
 }  
 }  
}

**ConstellationFragment：星座页面**



public class ConstellationFragment extends BaseFragment {  
  
 @BindView(R.id.recycler\_view)  
 RecyclerView recyclerView;  
 List<String> stringList = new ArrayList<>();  
 ConstellationAdapter constellationAdapter;  
  
  
 public static ConstellationFragment newInstance() {  
 ConstellationFragment fragment = new ConstellationFragment();  
 Bundle args = new Bundle();  
  
 fragment.setArguments(args);  
 return fragment;  
 }  
  
  
 @Override  
 protected int getLayoutId() {  
 return R.layout.fragment\_constellation;  
 }  
  
 @Override  
 protected void initEventAndData() {  
  
 recyclerView.setLayoutManager(new LinearLayoutManager(mContext));  
 //创建12星座数据  
 stringList.add("Aquarius (1/20-2/18)");  
 stringList.add("Pisces (2/19-3/20)");  
 stringList.add("Aries (3/21-4/20)");  
 stringList.add("Taurus (4/21-5/20)");  
 stringList.add("Gemini (5/21-6/21)");  
 stringList.add("Cancer (6/22-7/22)");  
 stringList.add("Leo (7/23-8/22)");  
 stringList.add("Virgo (8/23-9/22)");  
 stringList.add("Libra (9/23-10/23)");  
 stringList.add("Scorpio (10/24-11/22)");  
 stringList.add("Sagittarius (11/23-12/21)");  
 stringList.add("Capricorn (12/22-1/20)");  
  
 constellationAdapter = new ConstellationAdapter();  
 constellationAdapter.bindToRecyclerView(recyclerView);  
 constellationAdapter.setNewData(stringList);  
 constellationAdapter.setOnItemClickListener((adapter, view, position) -> {  
 Intent intent = new Intent(mContext, ConstellationActivity.class);  
 intent.putExtra(Constants.POSITION,position);  
 startActivity(intent);  
 });  
 }  
}