# Day01 所有框架及在首页上的使用

## 学习目的

### 1.继续提高设计程序能力

### 2.提高实战能力(按照真实的开发流程推进)

### 3.百度外卖框架

UI组件：fragment+ recycleview，

设计模式：MVP

技术组件：Retrofit + gson文件 + Gson + recycleview + Picasso

能力：常见功能 + 业务能力

### 4.项目需求

1. 项目经理需要考虑的内容
   1. MVP框架，分层的方式，Presenter
   2. 层与层之间Dagger2方式分隔
   3. 常见需求分析：整合ButterKnife、Retorfit、Ormlite工具。
   4. 常见抽取类BasePresenter
   5. 业务流程梳理
2. 程序员
   1. 首页处理
   2. 进入到商铺，选餐
   3. 购物车管理
   4. 结算
   5. 支付
   6. 订单

## 每日学习内容

1.了解MVP、RecycleView、Dagger2、Retrofit

2.短信登录、消息推送、观察者模式

3.多列表联动、View绝对位置

4.购物车、地址管理

5.支付宝、地图展示

6.地图配送

## 从MVC到MVP

1. MVC缺点：控制器里有大量业务逻辑
   1. 计算参数
   2. 请求服务器
   3. 异常处理
   4. 模拟睡眠

@Override  
protected HeimaAsycTask.Result doInbackground() {  
 SystemClock.sleep(Constant.SLEEP\_TIME);  
 GameProtocol gameProtocol = new GameProtocol();  
 Map<String, String> params = new HashMap<>();  
 params.put("index", "0");  
 gameProtocol.setParams(params);  
 try {  
 mDatas = gameProtocol.loadData();  
 if (mDatas == null || mDatas.size() == 0) {  
 return HeimaAsycTask.Result.EMPTY;  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 return HeimaAsycTask.Result.ERROR;  
 }  
 return HeimaAsycTask.Result.SUCCESS;  
}

2.MVC缺点：非常规模块功能不适合向父类抽取

* 1. 下载文件
  2. 购物车
  3. 支付
  4. 地图
  5. 地址

3.C层到P层的改变

MVC特点：

1. **Controller: Activity(对应整个页面) + fragment(对应部分页面)**
2. **View: views（自定义view） + xml**

**3.Model: beans + dao(本地缓存）**

控制器是view和model的桥梁，最重要。

Activity 控制 整个页面的layout

Fragment 控制 部分页面的layout

Holder 控制 一小分部view或者单个自定义view

MVP特点：

**1.Presenter: MainActivityPresenter**

**2.View: MainActivity的addView(动态添加view) + xml**

**3.Model: beans + dao (本地缓存）**

**Presenter控制所有业务逻辑，Activity只展示界面。**

**谷歌市场 --- 》 把业务有关的代码往父类抽取，减少控制器代码**

**百度外卖 ---》 把与业务有关的代码往Presenter类抽取，每个模块独立，类似HmDownloadManager。**

3.具体案例

//1.对javabean进行赋值  
 final User user=new User();  
 user.username=username;  
 user.password=password;  
 //2.开启子线程  
 new Thread(){  
 @Override  
 public void run() {  
 //3.执行业务逻辑  
 UserLoginNet net=new UserLoginNet();  
 if(net.sendUserLoginInfo(user)){  
 // 登陆成功  
 mMainActivity.success();  
 }else{  
 //登陆失败  
 mMainActivity.failed();  
 }  
 }  
 }.start();  
}

**Dagger2的使用**

1.解耦原因

• 组合关系：Activity中还有业务的引用

@Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
 mMainActivityPresenter = new MainActivityPresenter(this);

...

2.常见的解耦方式

• 利用配置文件，使用反射获取到需要加载的对象。

• 设计模式：单例、工厂、观察者……

3.注解回顾

1. 注解本身是一个类，用于标注其他类或者方法的特性。
2. 注解类有点像属性，可以多个使用，标注对象的功能。
3. 生活中的例子

@Game,@Code

Student student1;

@Code,@Love(girlfriend = “xiaohong”)

Student student2;

Student{

String name;

Int age;

}

3.dagger2的使用

**apt插件的安装：**

1.在project的gradle中添加

buildscript {

dependencies {

classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'

}

}

2.apt插件的使用

在modle的gradle中添加

apply plugin: 'com.neenbedankt.android-apt'

3.添加dagger2的依赖

dependencies {

compile 'com.google.dagger:dagger:2.6'

apt 'com.google.dagger:dagger-compiler:2.6'

}

四个注解使用

@Module、@Provides、@Component、@Inject

@Module：标注对象创建的容器

@Module  
public class MainActivityModule {  
  
 MainActivity mMainActivity;  
  
 public MainActivityModule(MainActivity mainActivity) {  
 mMainActivity = mainActivity;  
 }  
  
 @Provides：标识需要创建的对象  
 MainActivityPresenter providerMainActivityPresenter(){  
 return new MainActivityPresenter(mMainActivity);  
 }  
}

@Component：将创建好的对象与需要注入的对象组合到一起

@Component(modules = MainActivityModule.class)  
public interface MainActivityComponent {  
 void in(MainActivity mainActivity);  
}

@Inject  
MainActivityPresenter mMainActivityPresenter;

最后赋值即可：

DaggerMainActivityComponent.builder().mainActivityModule(new MainActivityModule(this)).build().in(this);

4.源码分析原理（最后一天课讲）

**首页UI的处理**

### 1.首页布局分析





### 2.状态栏设置样式

需要处理4.4以下版本、4.4、5.0以上版本，创建：values-v19,values-v21

Values:

<**style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar"**>  
</**style**>

values-v21:

<**style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar"**>  
 <**item name="android:windowTranslucentNavigation"**>true</**item**>  
 *<!--Android 5.x开始需要把颜色设置透明，否则导航栏会呈现系统默认的浅灰色-->* <**item name="android:statusBarColor"**>@android:color/transparent</**item**>  
</**style**>

*/\*\*  
 \* 获取状态栏的高度  
 \** ***@return*** *\*/***protected int** getStatusBarHeight(){  
 **try** {  
 Class<?> c=Class.*forName*(**"com.android.internal.R$dimen"**);  
 Object obj=c.newInstance();  
 Field field=c.getField(**"status\_bar\_height"**);  
 **int** x=Integer.*parseInt*(field.get(obj).toString());  
 **return** getResources().getDimensionPixelSize(x);  
 }**catch**(Exception e){  
 e.printStackTrace();  
 }  
 **return** 0;  
}

### 3.底部标签切换

private void initBottomBar() {  
 int childCount = mMainBottomeSwitcherContainer.getChildCount();  
 for (int i = 0; i < childCount; i++) {  
 final View child = mMainBottomeSwitcherContainer.getChildAt(i);  
 child.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 int index = mMainBottomeSwitcherContainer.indexOfChild(child);  
 selectTab(index);  
 }  
 });  
 }  
}  
  
private void selectTab(int selectIndex) {  
 int childCount = mMainBottomeSwitcherContainer.getChildCount();  
 for (int i = 0; i < childCount; i++) {  
 View child = mMainBottomeSwitcherContainer.getChildAt(i);  
 if (i == selectIndex) {  
 setEnable(child, false);  
 } else {  
 setEnable(child, true);  
 }  
 }  
}  
  
private void setEnable(View child, boolean isEnable) {  
 child.setEnabled(isEnable);   
 if (child instanceof ViewGroup) {  
 ViewGroup viewGroup = (ViewGroup) child;  
 for (int i = 0; i < viewGroup.getChildCount(); i++) {  
 View item = viewGroup.getChildAt(i);  
 **setEnable(item, isEnable); //递归调用**  
 }  
 }  
}

### 4.标题部分透明度渐变处理

需求：依据RecyclerView滚动的距离进行处理。滚动距离设置在0到150dp之间，透明度由0x553190E8变动到0xFF3190E8

分析：

1. 需要对RecyclerView的滚动进行监听（add….）
2. 获取滚动的方向及滚动的总距离
3. 利用距离所在（0-150）之间的比例，使用透明度颜色计算器进行对应颜色计算
4. 将计算的值设置给背景

注意边界处理

**if**(**sumY**<0){  
 color = 0x553190E8;  
}**else if**(**sumY** > 150){  
 color = 0XFF3190E8;  
}**else** {

}

**首页RecycleView的使用**

1.创建首页Fragment并切换

getFragmentManager().beginTransaction().replace(R.id.main\_fragment\_container, mFragmentList.get(selectIndex)).commit();

2.布局管理器

mRvHome.setLayoutManager(new LinearLayoutManager(getActivity()));

3.RecyclerView.Adapter和BasicAdapter一样

@Override  
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
 View homeItemView = View.inflate(mContext, R.layout.item\_home\_common, null);  
 return new ViewHolder(homeItemView);  
}  
  
@Override  
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {  
 ViewHolder viewHolder = (ViewHolder) holder;  
 viewHolder.setData(mDatas.get(position));  
}

**首页多类型条目**

1. 定义并支持多类型

public static final int TYPE\_TITLE = 0;  
public static final int TYPE\_NORMAL = 1;  
  
@Override  
public int getItemViewType(int position) {  
 if (position == 0) {  
 return TYPE\_TITLE;  
 } else {  
 return TYPE\_NORMAL;  
 }  
}

2. 两种Holder的实现

@Override  
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
 if (viewType == TYPE\_TITLE) {  
 View homeItemView = View.inflate(mContext, R.layout.item\_home\_common, null);  
 return new TitleHolder(homeItemView);  
 } else {  
 View homeItemView = View.inflate(mContext, R.layout.item\_home\_common, null);  
 return new SellerHolder(homeItemView);  
 }  
}  
  
@Override  
public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {  
 int viewType = getItemViewType(position);  
 if (viewType == TYPE\_TITLE) {  
 TitleHolder titleHolder = (TitleHolder) holder;  
 titleHolder.setData("我是大哥大-----------");  
 } else {  
 SellerHolder viewHolder = (SellerHolder) holder;  
 viewHolder.setData(mDatas.get(position));  
 }  
}

3. 第三方快速实现轮播图

compile 'com.squareup.picasso:picasso:2.3.2'

compile 'com.nineoldandroids:library:2.4.0'  
compile 'com.daimajia.slider:library:1.1.5@aar'

public void setData(String data) {  
 HashMap<String, String> url\_maps = new HashMap<String, String>();  
 url\_maps.put("Hannibal", "http://static2.hypable.com/wp-content/uploads/2013/12/hannibal-season-2-release-date.jpg");  
 url\_maps.put("Big Bang Theory", "http://tvfiles.alphacoders.com/100/hdclearart-10.png");  
 url\_maps.put("House of Cards", "http://cdn3.nflximg.net/images/3093/2043093.jpg");  
 url\_maps.put("Game of Thrones", "http://images.boomsbeat.com/data/images/full/19640/game-of-thrones-season-4-jpg.jpg");  
  
 for (String desc : url\_maps.keySet()) {  
 TextSliderView textSliderView = new TextSliderView(itemView.getContext());  
 textSliderView  
 .description(desc)  
 .image(url\_maps.get(desc));  
 mSlider.addSlider(textSliderView);

}  
}

**无文档的接口调试(快速学习JavaEE的Servlet)**

1. Web.xml是服务器的清单文件

<servlet>

<description></description>

<display-name>HomeServlet</display-name>

<servlet-name>HomeServlet</servlet-name>

<servlet-class>**cn.itcast.tackout.servlet.HomeServlet**</servlet-class>

</servlet> **//接口处理Servlet类在这里**

<servlet-mapping>

<servlet-name>HomeServlet</servlet-name>

<url-pattern>**/home**</url-pattern> //接口URL在这里

2. doGet()是服务器对客服端URL请求的响应

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

Response resp = new Response();

resp.setCode("0");

3. 服务器返回的数据结构

public class Response {

//服务器开发者定义的数据结构

private String code;

private String data;

}

4. 通过HttpServletResponse对象返回gson数据

HomeInfo info = new HomeInfo(promotionList, categorieList, nearbySellerList, ortherSellerList);

res.setData(JSONObject.fromObject(info).toString());

CommonUtil.renderJson(response, res);

5. 最终HttpServletResponse通过IO流返回数据给客户端

PrintWriter writer = response.getWriter();

writer.write(contentJson);

writer.flush();

writer.close();

**使用Retrofit请求服务器**

1. 新建HomeFragmentPresenter

public class MainActivityPresenter {  
  
 private HomeFragment mHomeFragment;  
  
 public HomeFragmentPresenter(HomeFragment homeFragment) {  
 mHomeFragment = homeFragment;  
 }  
  
 public void loadHomeInfo(){  
   
 }  
}

2. 配置Retrofit

**apt插件的安装：**

1.在project的gradle中添加

buildscript {

dependencies {

classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'

}

}

2.apt插件的使用

在modle的gradle中添加

apply plugin: 'com.neenbedankt.android-apt'

compile 'com.google.code.gson:gson:2.2.4'  
compile 'com.squareup.retrofit2:retrofit:2.1.0'  
compile 'com.squareup.retrofit2:converter-gson:2.1.0'

3. 创建外卖Service,封装所有接口

public interface TakeoutService {  
 //封装所有接口，方便代码共享、调用  
 @GET("home")  
 Call<ResponseInfo> getHomeInfo();  
  
 @GET("order")  
 Call<ResponseInfo> getOrderList(@Query("userId") String userId);  
}

4. 使用retrofit调用接口，获取Request

public void loadHomeInfo() {  
 Retrofit retrofit = new Retrofit.Builder().baseUrl(Constant.HOST).addConverterFactory(GsonConverter Factory.create()).build();  
 TakeoutService takeoutService = retrofit.create(TakeoutService.class);  
 Call<ResponseInfo> call = takeoutService.getHomeInfo();

5. Presenter异步请求，Activity回调处理结果

call.enqueue(new Callback<ResponseInfo>() {  
 @Override  
 public void onResponse(Call<ResponseInfo> call, Response<ResponseInfo> response) {  
 if (response.isSuccessful()) {  
 mHomeFragment.onHomeSuccess();  
 } else {  
 mHomeFragment.onHomeFailed();  
 }  
 }  
  
 @Override  
 public void onFailure(Call<ResponseInfo> call, Throwable t) {  
 mHomeFragment.onHomeError();  
 }  
});

## 总结MVP特征：

## Presenter抽取activity或fragment所有业务代码

## 业务执行完后，让activity或fragment刷新View

## 封装NetPresenter基类，完成首页功能

1.定义抽象方法对应成功、失败、异常

public abstract class NetPresenter {

protected abstract void onNetError(Throwable t);  
 protected abstract void onFailed(int code);  
 protected abstract void onSuccess(String json);

2.手动拆分解析json

JSONObject jsonObject = new JSONObject(json);  
String nearby = jsonObject.getString("nearbySellerList");  
//使用匿名子类  
List<Seller> nearbyList = gson.fromJson(nearby, new TypeToken<List<Seller>>(){}.getType());  
  
String other = jsonObject.getString("otherSellerList");  
List<Seller> otherList = gson.fromJson(other, new TypeToken<List<Seller>>(){}.getType());

3.fragment回调Adapter

mHomeFragment.onHomeSuccess(nearbyList, otherList);

public void onHomeSuccess(List<Seller> nearbyList, List<Seller> otherList) {  
 mHomeRvAdapter.setDatas(nearbyList, otherList);  
 }

4.首页item使用真实布局

public void setDatas(List<Seller> nearbyList, List<Seller> otherList) {  
 this.nearbyList = nearbyList;  
 this.otherList = otherList;  
 mDatas.clear();  
 mDatas.addAll(nearbyList);  
 mDatas.addAll(otherList);  
 notifyDataSetChanged();  
}

@Override  
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
 ...  
 View homeItemView = View.inflate(mContext, **R.layout.item\_seller**, null);  
 return new SellerHolder(homeItemView);  
 }  
}

## 每日作业

1.了解MVP与MVC的区别  
2.会使用dagger2的四个注解  
3.会使用Retrofit请求数据  
4.会封装NetPresenter基类