不依赖GsonFormat，手动写复杂Json数据的实体类

1. 学习原因：

因为有些时候公司的后台API返回的Json数据不符合GsonFormat自动生成实体类的标准，此时就只能手动写Json解析的实体类Bean，解析调用Gson.fromJson()即可，实体类首选GsonFormat, 不符合标准就手动写实体类

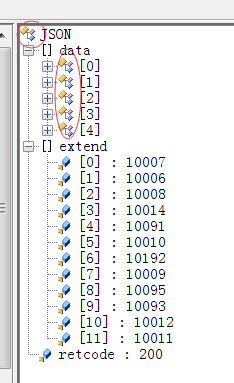
1. 不能解析的情况：

Key是数字，然后后面是字段........，要学习一遍Json

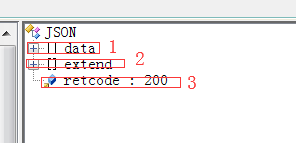
1. 访问本地主机的Tomcat的三种方式，不一定要查ip地址

（1）192.168.1.199:8080/web\_home

1. localhost:8080/web\_home(手机模拟器不可访问)
2. 127.0.0.1:8080/web\_home（手机模拟器不可访问，一般选择真机调试）
3. 手动写Json解析实体类的流程
4. 打开HiJson
5. 复制后台返回的全部Json数据，粘贴到HiJson
6. 点击格式化Json字符串
7. 黄色摄像头代表是一个对象



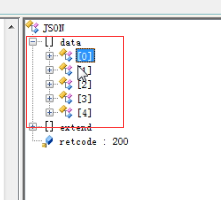
1. 解析流程： 对象---》属性---》再看属性是否为对象（是）---》属性.....一层层解析即可

属性1,2,3（一层层并列解析）

所以对应：

**public class** Bean {  
 **private** List **data**; //注意，data,extend,retcode都是json数据的键名，不能打错  
 **private** List **extend**; //打错一个键名，就得不到数据  
 **private int retcode**; //最好的方式是：Hijson右键复制键名  
  
}

\*data属性里又有对象，需要下一层解析



\*data里面有对象，所以List里面要指定此对象的类泛型，所以要写这个泛型类

而且是写成内部静态类public static class A{}的形式（内部静态类的好处？补一下）

\*然后打开对象0,1,2,3,4的所有字段（对应类里面的属性，Json里面称为字段），补充写到classA里面作为A的属性

\*然后写完所有的属性和静态内部类（泛型类）之后，开始写Getter,Setter

\*之后开始写:toString()，方便打印和debug数据

\*从里到外的类写Getter,Setter,toString()，这样不会出错

\*\*\*完整代码如下

**package** com.example.myapplication;  
**import** java.util.List;  
**public class** Bean {  
  
 **private** List<DetailPagerData> **data**;  
 **private** List **extend**;  
 **private int retcode**;  
  
 *//3写Getter,Setter,toString()* **public** List<DetailPagerData> getData() {  
 **return data**;  
 }  
  
 **public** List getExtend() {  
 **return extend**;  
 }  
  
 **public int** getRetcode() {  
 **return retcode**;  
 }  
  
 **public void** setData(List<DetailPagerData> data) {  
 **this**.**data** = data;  
 }  
  
 **public void** setExtend(List extend) {  
 **this**.**extend** = extend;  
 }  
  
 **public void** setRetcode(**int** retcode) {  
 **this**.**retcode** = retcode;  
 }  
  
  
 @Override  
 **public** String toString() {  
 **return "Bean{"** +  
 **"data="** + **data** +  
 **", extend="** + **extend** +  
 **", retcode="** + **retcode** +  
 **'}'**;  
 }  
  
 **public static class** DetailPagerData{  
  
 **private int id**;  
 **private** String **title**;  
 **private int type**;  
 **private** String **url**;  
 **private** String **url1**;  
 **private** String **dayurl**;  
 **private** String **excurl**;  
 **private** String **weekurl**;  
 **private** List<ChildrenData> **children**;  
 *//2写Getter,Setter,toString()* **public int** getId() {  
 **return id**;  
 }  
  
 **public** String getTitle() {  
 **return title**;  
 }  
  
 **public int** getType() {  
 **return type**;  
 }  
  
 **public** String getUrl() {  
 **return url**;  
 }  
  
 **public** String getUrl1() {  
 **return url1**;  
 }  
  
 **public** String getDayurl() {  
 **return dayurl**;  
 }  
  
 **public** String getExcurl() {  
 **return excurl**;  
 }  
  
 **public** String getWeekurl() {  
 **return weekurl**;  
 }  
  
 **public** List<ChildrenData> getChildren() {  
 **return children**;  
 }  
  
 **public void** setId(**int** id) {  
 **this**.**id** = id;  
 }  
  
 **public void** setTitle(String title) {  
 **this**.**title** = title;  
 }  
  
 **public void** setType(**int** type) {  
 **this**.**type** = type;  
 }  
  
 **public void** setUrl(String url) {  
 **this**.**url** = url;  
 }  
  
 **public void** setUrl1(String url1) {  
 **this**.**url1** = url1;  
 }  
  
 **public void** setDayurl(String dayurl) {  
 **this**.**dayurl** = dayurl;  
 }  
  
 **public void** setExcurl(String excurl) {  
 **this**.**excurl** = excurl;  
 }  
  
 **public void** setWeekurl(String weekurl) {  
 **this**.**weekurl** = weekurl;  
 }  
  
 **public void** setChildren(List<ChildrenData> children) {  
 **this**.**children** = children;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "DetailPagerData{"** +  
 **"id="** + **id** +  
 **", title='"** + **title** + **'\''** +  
 **", type="** + **type** +  
 **", url='"** + **url** + **'\''** +  
 **", url1='"** + **url1** + **'\''** +  
 **", dayurl='"** + **dayurl** + **'\''** +  
 **", excurl='"** + **excurl** + **'\''** +  
 **", weekurl='"** + **weekurl** + **'\''** +  
 **", children="** + **children** +  
 **'}'**;  
 }  
  
 *//内部类再嵌套内部类: 1写Getter,Setter,toString()* **public static class** ChildrenData{  
  
 **private int id**;  
 **private** String **title**;  
 **private int type**;  
 **private** String **url**;  
  
 **public void** setId(**int** id) {  
 **this**.**id** = id;  
 }  
  
 **public void** setTitle(String title) {  
 **this**.**title** = title;  
 }  
  
 **public void** setType(**int** type) {  
 **this**.**type** = type;  
 }  
  
 **public void** setUrl(String url) {  
 **this**.**url** = url;  
 }  
  
 **public int** getId() {  
 **return id**;  
 }  
  
 **public** String getTitle() {  
 **return title**;  
 }  
  
 **public int** getType() {  
 **return type**;  
 }  
  
 **public** String getUrl() {  
 **return url**;  
 }  
 //可以打印日志，跟踪返回数据的解析情况  
 @Override  
 **public** String toString() {  
 **return "ChildrenData{"** +  
 **"id="** + **id** +  
 **", title='"** + **title** + **'\''** +  
 **", type="** + **type** +  
 **", url='"** + **url** + **'\''** +  
 **'}'**;  
 }  
 }  
 }  
  
}

5、写完实体类后，开始真正解析Json数据

（1）使用JSONObject, JSONArray + for循环

String **responseData**=response.body().string();  
parseJson(**responseData**);  
  
**private void** parseJson(String jsonData){  
  
 JSONArray jsonArray=**new** JSONArray(jsonData);  
 **try** {  
 **for** (**int** i = 0; i < jsonArray.length(); i++) {  
  
 JSONObject jsonObject = jsonArray.getJSONObject(i);  
 String id = jsonObject.getString(**"id"**);  
 String name = jsonObject.getString(**"id"**);  
 String version = jsonObject.getString(**"id"**);  
  
 }  
 }**catch** (Exception e){  
 e.printStackTrace();  
 }  
}

（2）使用Gson解析

\*build.gradle: compile ‘com.google.code.gson:gson:2.7’

\*简单Gson解析

// 2 解析JSON数据

Gson gson = new Gson();

ShopInfo shopInfo = gson.fromJson(json, ShopInfo.class);

// 3 展示数据

tv\_gson\_last.setText(shopInfo.toString())；

\*复杂Gson解析：解析Json数组

private void parseJson(String jsonData){

Gson gson=new Gson();

List<App> appList=gson.fromJson(jsonData,new TypeToken<List<App>>(){}.getType());

for(App app: appList){

Log.d(“ManActivity”,”id is”+ app.getId());

}

}

1、添加依赖

implementation **'com.google.code.gson:gson:2.7'**

1. Gson解析的示例

*//初始化Data，填充Data到view显示出来：解析返回的json数据***private void** processData(String result) {  
  
 *//调用象方：解析json* NewsCenterPagerBean bean=parsedJson(result);  
 *//日志打印检验是否成功* String title=bean.getData().get(0).getChildren().get(1).getTitle();  
 LogUtil.*e*(**"解析标题成功---------"**+title);  
  
  
  
 *//调用象方：初始化Data，得到左侧菜单的所有数据* **data**=bean.getData();  
 *//构造对象：将context转回Activity* MainActivity mainActivity= (MainActivity) **context**;  
 *//构造对象：得到左侧菜单的实例* LeftMenuFragment leftMenuFragment=mainActivity.getLeftMenuFragment();  
  
  
 *//构造对象：初始化DataView，创建了4个view配合左侧菜单切换显示* **detailBasePagers**=**new** ArrayList<>();  
 **detailBasePagers**.add(**new** NewsMenuDetailPager(**context**,**data**.get(0)));  
 **detailBasePagers**.add(**new** TopicMenuDetailPager(**context**,**data**.get(0)));  
 **detailBasePagers**.add(**new** PhotoMenuDetailPager(**context**,**data**.get(2)));  
 **detailBasePagers**.add(**new** InteracMenuDetailPager(**context**,**data**.get(2)));  
  
 *//调用象方：初始化Data，传递左侧菜单数据给左侧菜单，左侧菜单用setter接收* leftMenuFragment.setData(**data**);  
}  
*//调用象方解析：Gson解析效率高***private** NewsCenterPagerBean parsedJson(String result) {  
 **return new** Gson().fromJson(result,NewsCenterPagerBean.**class**);  
}