第 3 讲_JSON 解析

讲师：尚硅谷-大海哥
谷粉第 47 群：285047793

1_JSON 简介

1.1_简介

JSON 的全称是 JavaScript Object Notation，是一种轻量级的数据交换格式。

1.2_特点

（1）JSON 比 XML 数据传输的有效性要高出很多
（2）JSON 完全独立于编程语言。
（3）本质就是具有特定格式的字符串

2_JSON 数据格式

2.1_整体结构

```
String json1 = "{"id":12,"name":"Tom"}"
String json2 = "[{"id":12,"name":"Tom"},{"id":12,"name":"Tom"}]"
```

2.2_Json 数组 ： [ ]

（1) Json 数组的结构: [value1, value2, value3]

（2）例子：

[1, “ab”,[], {“n”:123, “b”:”abc”}]　　正确
[1, “a”:3]　　错误

2.2_Json 对象: {}

（1）Json 对象的结构: {key1:value1, key2:value2, key3:value3}
（2）key 的数据类型: 字符串
（3）value 的数据类型: 数值、字符串、null、json 数组 []、json 对象 {}
（4）例子:
{"name":"TOM", "age":12} 正确
{"aa":"a", 3} 错误

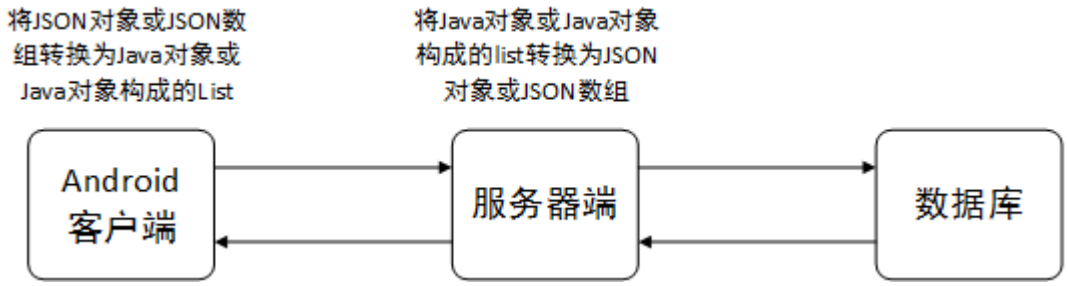
3_JSON 解析方向

3.1_将 java 对象(包含集合)转换为 json 格式字符串

在服务器端应用。

3.2_将 json 格式字符串转换为 java 对象（包含集合）

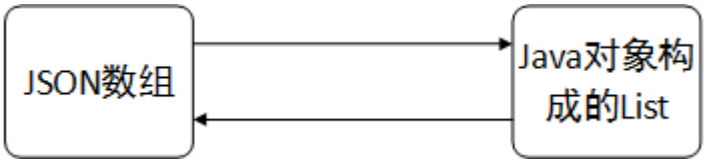
在客户端应用。



3.3_Json 和 Java 之间转换关系

（1）JSON 中的对象对应着 Java 中的对象

（2）Json 中的数组，对应着 Java 中的集合



4_JSON 解析技术

4.1_Android 原生技术

1）特点：编程相对麻烦

4.1.1_将 json 格式的字符串{}转换为 Java 对象

1）API：JsonObject

JSONObject(String json)：将 json 字符串解析为 json 对象

Xxx getXxx(String name)：根据 name，在 json 对象中得到对应的 Value

2）测试数据

```
{
    "id":2, "name":"大虾",
    "price":12.3,
    "imagePath":"http://192.168.10.165:8080/L05_Server/images/f1.jpg"
}
```

3）例子

```
// 将 json 格式的字符串{}转换为 Java 对象
    private void jsonToJavaObjectByNative() {

        // 获取或创建 JSON 数据
        String json = "{\n" +
                "\t\"id\":2, \"name\":\"大虾\", \n" +
                "\t\"price\":12.3, \n" +

"\t\"imagePath\":\"http://192.168.10.165:8080/L05_Server/images/f1.jpg\
"\n" +"}\n";

        ShopInfo shopInfo = null;
        // 解析 json
        try {
            JSONObject jsonObject = new JSONObject(json);
//            int id = jsonObject.getInt("id");
            int id1 = jsonObject.optInt("id");

            String name = jsonObject.optString("name");

            double price = jsonObject.optDouble("price");
```

```
            String imagePath = jsonObject.optString("imagePath");

            // 封装 Java 对象
            shopInfo = new ShopInfo(id1, name, price, imagePath);

        } catch (JSONException e) {
            e.printStackTrace();
        }

        // 显示 JSON 数据
        tv_native_orignal.setText(json);
        tv_native_last.setText(shopInfo.toString());
    }
```

4.1.2_将 json 格式的字符串[]转换为 Java 对象的 List

1）API:JSONArray

JSONArray(String json)：将 json 字符串解析为 json 数组

int length()：得到 json 数组中元素的个数

Xxx getXxx(int index)：根据下标得到 json 数组中对应的元素数据

2）测试数据

```
[
    {
     "id":1, "name":"大虾1",
     "price":12.3,
     "imagePath":"http://192.168.10.165:8080/f1.jpg"
    },
    {
     "id":2, "name":"大虾2",
     "price":12.5,
     "imagePath":"http://192.168.10.165:8080/f2.jpg"
    }
]
```

3）例子

```
// 将 json 格式的字符串[]转换为 Java 对象的 List
private void jsonToJavaListByNative() {

    // 获取或创建 JSON 数据
```

```
    String json = "[\n" +
            "    {\n" +
            "        \"id\": 1,\n" +
            "        \"imagePath\":
\"http://192.168.10.165:8080/f1.jpg\",\n" +
            "        \"name\": \"大虾 1\",\n" +
            "        \"price\": 12.3\n" +
            "    },\n" +
            "    {\n" +
            "        \"id\": 2,\n" +
            "        \"imagePath\":
\"http://192.168.10.165:8080/f2.jpg\",\n" +
            "        \"name\": \"大虾 2\",\n" +
            "        \"price\": 12.5\n" +
            "    }\n" +
            "]";

    List<ShopInfo> shops = new ArrayList<>();

    // 解析 json
    try {
        JSONArray jsonArray = new JSONArray(json);

        for (int i = 0; i < jsonArray.length(); i++) {
            JSONObject jsonObject = jsonArray.getJSONObject(i);

            if (jsonObject != null) {
                int id = jsonObject.optInt("id");

                String name = jsonObject.optString("name");

                double price = jsonObject.optDouble("price");

                String imagePath = jsonObject.optString("imagePath");

                // 封装 Java 对象
                ShopInfo shopInfo = new ShopInfo(id, name, price, imagePath);

                shops.add(shopInfo);
            }
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
```

```
    // 显示JSON 数据
    tv_native_orignal.setText(json);
    tv_native_last.setText(shops.toString());
}
```

4.1.3_复杂 json 数据解析

1）测试数据

```
{
    "data": {
        "count": 5,
        "items": [
            {
                "id": 45,
                "title": "坚果"
            },
            {
                "id": 132,
                "title": "炒货"
            },
            {
                "id": 166,
                "title": "蜜饯"
            },
            {
                "id": 195,
                "title": "果脯"
            },
            {
                "id": 196,
                "title": "礼盒"
            }
        ]
    },
    "rs_code": "1000",
    "rs_msg": "success"
}
```

2）例子

```
// 复杂json 数据解析
private void jsonToJavaOfComplex() {
```

```
// 获取或创建 JSON 数据
String json = "{\n" +
        "    \"data\": {\n" +
        "        \"count\": 5,\n" +
        "        \"items\": [\n" +
        "            {\n" +
        "                \"id\": 45,\n" +
        "                \"title\": \"坚果\"\n" +
        "            },\n" +
        "            {\n" +
        "                \"id\": 132,\n" +
        "                \"title\": \"炒货\"\n" +
        "            },\n" +
        "            {\n" +
        "                \"id\": 166,\n" +
        "                \"title\": \"蜜饯\"\n" +
        "            },\n" +
        "            {\n" +
        "                \"id\": 195,\n" +
        "                \"title\": \"果脯\"\n" +
        "            },\n" +
        "            {\n" +
        "                \"id\": 196,\n" +
        "                \"title\": \"礼盒\"\n" +
        "            }\n" +
        "        ]\n" +
        "    },\n" +
        "    \"rs_code\": \"1000\",\n" +
        "    \"rs_msg\": \"success\"\n" +
        "}";

// 封装 Java 对象
DataInfo dataInfo = new DataInfo();

// 解析 json
try {
    JSONObject jsonObject = new JSONObject(json);

    // 第一层解析
    JSONObject data = jsonObject.optJSONObject("data");
    String rs_code = jsonObject.optString("rs_code");
    String rs_msg = jsonObject.optString("rs_msg");
```

```
        // 第一层封装
        dataInfo.setRs_code(rs_code);
        dataInfo.setRs_msg(rs_msg);
        DataInfo.DataBean dataBean = new DataInfo.DataBean();
        dataInfo.setData(dataBean);

        // 第二层解析
        int count = data.optInt("count");
        JSONArray items = data.optJSONArray("items");

        // 第二层数据的封装
        dataBean.setCount(count);

        List<DataInfo.DataBean.ItemsBean> itemsBean = new ArrayList<>();
        dataBean.setItems(itemsBean);

        // 第三层解析
        for (int i = 0; i < items.length(); i++) {
            JSONObject jsonObject1 = items.optJSONObject(i);

            if (jsonObject1 != null) {
                int id = jsonObject1.optInt("id");

                String title = jsonObject1.optString("title");

                // 第三层数据的封装
                DataInfo.DataBean.ItemsBean bean = new DataInfo.DataBean.ItemsBean();
                bean.setId(id);
                bean.setTitle(title);

                itemsBean.add(bean);
            }
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }

    // 显示 JSON 数据
    tv_native_orignal.setText(json);
    tv_native_last.setText(dataInfo.toString());
}
```

4.1.4_特殊 json 数据解析

1）测试数据

```
{
    "code": 0,
    "list": {
        "0": {
            "aid": "6008965",
            "author": "哔哩哔哩番剧",
            "coins": 170,
            "copyright": "Copy",
            "create": "2016-08-25 21:34"
        },
        "1": {
            "aid": "6008938",
            "author": "哔哩哔哩番剧",
            "coins": 404,
            "copyright": "Copy",
            "create": "2016-08-25 21:33"
        }
    }
}
```

2）例子

```
// (4)特殊 json 数据解析
private void jsonToJavaOfSpecial() {

    // 1 获取或创建 JSON 数据
    String json = "{\n" +
            "    \"code\": 0,\n" +
            "    \"list\": {\n" +
            "        \"0\": {\n" +
            "            \"aid\": \"6008965\",\n" +
            "            \"author\": \"哔哩哔哩番剧\",\n" +
            "            \"coins\": 170,\n" +
            "            \"copyright\": \"Copy\",\n" +
            "            \"create\": \"2016-08-25 21:34\"\n" +
            "        },\n" +
            "        \"1\": {\n" +
            "            \"aid\": \"6008938\",\n" +
            "            \"author\": \"哔哩哔哩番剧\",\n" +
            "            \"coins\": 404,\n" +
            "            \"copyright\": \"Copy\",\n" +
```

```
            "            \"create\": \"2016-08-25 21:33\"\n" +
            "        }\n" +
            "    }\n" +
            "}";

    // 创建封装的Java 对象
    FilmInfo filmInfo = new FilmInfo();

    // 2 解析json
    try {
        JSONObject jsonObject = new JSONObject(json);

        // 第一层解析
        int code = jsonObject.optInt("code");
        JSONObject list = jsonObject.optJSONObject("list");

        // 第一层封装
        filmInfo.setCode(code);
        List<FilmInfo.FilmBean> lists = new ArrayList<>();
        filmInfo.setList(lists);

        // 第二层解析
        for (int i = 0; i < list.length(); i++) {
            JSONObject jsonObject1 = list.optJSONObject(i + "");

            if(jsonObject1 != null) {
                String aid = jsonObject1.optString("aid");

                String author = jsonObject1.optString("author");

                int coins = jsonObject1.optInt("coins");

                String copyright = jsonObject1.optString("copyright");

                String create = jsonObject1.optString("create");

                // 第二层数据封装
                FilmInfo.FilmBean filmBean = new FilmInfo.FilmBean();
                filmBean.setAid(aid);
                filmBean.setAuthor(author);
                filmBean.setCoins(coins);
                filmBean.setCopyright(copyright);
                filmBean.setCreate(create);
```

```
                lists.add(filmBean);
            }
        }

    } catch (JSONException e) {
        e.printStackTrace();
    }

    // 3 显示 JSON 数据
    tv_native_orignal.setText(json);
    tv_native_last.setText(filmInfo.toString());
}
```

4.2_GSON 框架技术

1）特点：编码简洁，谷歌官方推荐
2）下载地址：https://mvnrepository.com/artifact/com.google.code.gson/gson

4.2.1_将 json 格式的字符串{}转换为 Java 对象

1）用到的 API
<T> T fromJson(String json, Class<T> classOfT);//将 json 对象转换为 Java 对象的方法
注意：要求 json 对象中的 key 的名称与 java 对象对应的类中的属性名要相同

2）使用步骤
（1）将 Gson 的 jar 包导入到项目中
（2）创建Gson对象 ： Gson gson = new Gson();
（3）通过创建的Gson对象调用fromJson()方法，返回该JSON数据对应的Java对象
ShopInfo shopInfo = gson.fromJson(json, ShopInfo.class);

3）测试数据

```
{
    "id":2, "name":"大虾",
    "price":12.3,
    "imagePath":"http://192.168.10.165:8080/L05_Server/images/f1.jpg"
}
```

4）例子

```
// (1)将 json 格式的字符串{}转换为 Java 对象
private void jsonToJavaObject() {
```

```
    // 1 获取或创建 json
    String json = "{\n" +
            "\t\"id\":2, \"name\":\"大虾\", \n" +
            "\t\"price\":12.3, \n" +

"\t\"imagePath\":\"http://192.168.10.165:8080/L05_Server/images/f1.jpg\
"\n" +
            "}\n";

    // 2 解析 json
    Gson gson = new Gson();

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

    // 3 显示 JSON 数据
    tv_native_orignal.setText(json);
    tv_native_last.setText(shopInfo.toString());
}
```

4.2.2_将 json 格式的字符串[]转换为 Java 对象的 List

1）用到的 API

T fromJson(String json, Type typeOfT);//将 json 数组转换为 Java 对象的 list

注意：要求 json 对象中的 key 的名称与 java 对象对应的类中的属性名要相同

2）使用步骤

（1）将 Gson 的 jar 包导入到项目中

（2）创建Gson对象 ： Gson gson = new Gson();

（3）通过创建的Gson对象调用fromJson()方法，返回该JSON数据对应的Java集合：

List<ShopInfo> shops = gson.fromJson(json, new TypeToken<List<ShopInfo>>() {}.getType());

3）测试数据

```
[
    {
        "id": 1,
        "imagePath": "http://192.168.10.165:8080/f1.jpg",
        "name": "大虾1",
        "price": 12.3
    },
```

```
    {
        "id": 2,
        "imagePath": "http://192.168.10.165:8080/f2.jpg",
        "name": "大虾2",
        "price": 12.5
    }
]
```

4）例子

```
//(2)将 json 格式的字符串[]转换为 Java 对象的 List
private void jsonToJavaList() {

    // 1 获取或创建 json
    String json = "[\n" +
            "    {\n" +
            "        \"id\": 1,\n" +
            "        \"imagePath\":
\"http://192.168.10.165:8080/f1.jpg\",\n" +
            "        \"name\": \"大虾 1\",\n" +
            "        \"price\": 12.3\n" +
            "    },\n" +
            "    {\n" +
            "        \"id\": 2,\n" +
            "        \"imagePath\":
\"http://192.168.10.165:8080/f2.jpg\",\n" +
            "        \"name\": \"大虾 2\",\n" +
            "        \"price\": 12.5\n" +
            "    }\n" +
            "]";
    // 2 解析 json
    Gson gson = new Gson();

    List<ShopInfo> shops = gson.fromJson(json, new
TypeToken<List<ShopInfo>>() {
    }.getType());

    // 3 显示 JSON 数据
    tv_native_orignal.setText(json);
    tv_native_last.setText(shops.toString());
}
```

4.2.3_将 Java 对象转换为 json 字符串{}

1）用到的 API

String toJson(Object src);

2）使用步骤

（1）将 Gson 的 jar 包导入到项目中

（2）创建Gson对象 ： Gson gson = new Gson();

（3）通过创建的Gson对象调用toJson()方法，返回json数据：

ShopInfo shop = new ShopInfo(1, "鲍鱼", 250.0, "");

String json = gson.toJson(shop);

3）例子

```
// (3)将 Java 对象转换为 json 字符串{}
private void javaToJsonObject() {
    // 1 获取或创建 Java 对象
    ShopInfo shop = new ShopInfo(1, "鲍鱼", 250.0, "");

    // 2 生成 JSON 数据
    Gson gson = new Gson();
    String json = gson.toJson(shop);

    // 3 展示 json 数据
    tv_native_orignal.setText(shop.toString());
    tv_native_last.setText(json);
}
```

4.2.3_将 Java 对象的 List 转换为 json 字符串[]

1）用到的 API

String toJson(Object src);

2）使用步骤

（1）将 Gson 的 jar 包导入到项目中

（2）创建Gson对象 ： Gson gson = new Gson();

（3）通过创建的Gson对象调用toJson()方法，返回json数据：

List<ShopInfo> shops = new ArrayList<>();

String json = gson.toJson(shops);

3）例子

```
// (4) 将 Java 对象的 List 转换为 json 字符串[]
```

```
private void javaToJsonList() {
    // 1 获取或创建 Java 集合
    List<ShopInfo> shops = new ArrayList<>();
    ShopInfo baoyu = new ShopInfo(1, "鲍鱼", 250, "baoyu");
    ShopInfo haisen = new ShopInfo(2, "海参", 251, "haisen");
    shops.add(baoyu);
    shops.add(haisen);

    // 2 生成 JSON 数据
    Gson gson = new Gson();

    String json = gson.toJson(shops);

    // 3 展示 json 数据
    tv_native_orignal.setText(shops.toString());
    tv_native_last.setText(json);
}
```

4.3_FastJson 框架技术

1）特点：Fastjson 是一个 Java 语言编写的高性能功能完善的 JSON 库。它采用一种"假定有序快速匹配"的算法，把 JSON Parse 的性能提升到极致，是目前 Java 语言中最快的 JSON 库。

2）下载地址：https://github.com/alibaba/fastjson/wiki

4.3.1_将 json 格式的字符串{}转换为 Java 对象

1）用到的 API

< T > T parseObject(String json, Class<T> classOfT); //将 json 对象转换为 Java 对象的方法

注意：要求 json 对象中的 key 的名称与 java 对象对应的类中的属性名要相同

2）使用步骤

（1）导入 fastjson 的 jar 包

（2）JSON调用parseObject()方法，获取转换后的Java对象

例如：ShopInfo shopInfo = JSON.parseObject(json, ShopInfo.class);

3）测试数据

```
{
    "id":2, "name":"大虾",
```

```
    "price":12.3,
    "imagePath":"http://192.168.10.165:8080/L05_Server/images/f1.jpg"
}
```

4）例子

```
// （1）将 json 格式的字符串{}转换为 Java 对象
private void jsonToJavaObjectByFastJson() {

    // 1 获取或创建 JSON 数据
    String json = "{\n" +
            "\t\"id\":2, \"name\":\"大虾\", \n" +
            "\t\"price\":12.3, \n" +

"\t\"imagePath\":\"http://192.168.10.165:8080/L05_Server/images/f1.jpg\
"\n" +
            "}\n";

    // 2 解析 JSON 数据
    ShopInfo shopInfo = JSON.parseObject(json, ShopInfo.class);

    // 3 显示数据
    tv_fastjson_orignal.setText(json);
    tv_fastjson_last.setText(shopInfo.toString());
}
```

4.3.2_将 json 格式的字符串[]转换为 Java 对象的 List

1）用到的 API

List<T> parseArray(String json,Class<T> classOfT);//将 json 数组转换为 Java 对象的 list

注意：要求 json 对象中的 key 的名称与 java 对象对应的类中的属性名要相同

2）使用步骤

（1）导入 fastjson 的 jar 包

（2）JSON调用parseArray()方法，获取转换后的Java集合

例如：List<ShopInfo> shopInfos = JSON.parseArray(json, ShopInfo.class);

3）测试数据

```
[
    {
        "id": 1,
        "imagePath": "http://192.168.10.165:8080/f1.jpg",
```

```
        "name": "大虾1",
        "price": 12.3
    },
    {
        "id": 2,
        "imagePath": "http://192.168.10.165:8080/f2.jpg",
        "name": "大虾2",
        "price": 12.5
    }
]
```

4）例子

```
// （2）将 json 格式的字符串[]转换为 Java 对象的 List
private void jsonToJavaListByFastJson() {

    // 1 获取或创建 JSON 数据
    String json = "[\n" +
            "    {\n" +
            "        \"id\": 1,\n" +
            "        \"imagePath\":
\"http://192.168.10.165:8080/f1.jpg\",\n" +
            "        \"name\": \"大虾 1\",\n" +
            "        \"price\": 12.3\n" +
            "    },\n" +
            "    {\n" +
            "        \"id\": 2,\n" +
            "        \"imagePath\":
\"http://192.168.10.165:8080/f2.jpg\",\n" +
            "        \"name\": \"大虾 2\",\n" +
            "        \"price\": 12.5\n" +
            "    }\n" +
            "]";

    // 2 解析 JSON 数据
    List<ShopInfo> shopInfos = JSON.parseArray(json, ShopInfo.class);

    // 3 显示数据
    tv_fastjson_orignal.setText(json);
    tv_fastjson_last.setText(shopInfos.toString());
}
```

4.3.3_将 Java 对象转换为 json 字符串{}

1）用到的 API

String toJSONString(Object object);

2）使用步骤

（1）导入 fastjson 的 jar 包

（2）JSON调用toJSONString()方法，获取转换后的json数据

例如：

```
ShopInfo shopInfo = new ShopInfo(1, "鲍鱼", 250.0, "baoyu");
String json = JSON.toJSONString(shopInfo);
```

3）例子

```
// （3）将 Java 对象转换为 json 字符串{}
private void javaToJsonObjectByFastJson() {

    // 1 获取 Java 对象
    ShopInfo shopInfo = new ShopInfo(1, "鲍鱼", 250.0, "baoyu");

    // 2 生成 JSON 数据
    String json = JSON.toJSONString(shopInfo);

    // 3 数据显示
    tv_fastjson_orignal.setText(shopInfo.toString());
    tv_fastjson_last.setText(json);
}
```

4.3.4_将 Java 对象的 List 转换为 json 字符串[]

1）用到的 API

String toJSONString(Object object);

2）使用步骤

（1）导入 fastjson 的 jar 包

（2）JSON调用toJSONString()方法，获取转换后的json数据

例如：

```
List<ShopInfo> shops = new ArrayList<>();
ShopInfo baoyu = new ShopInfo(1, "鲍鱼", 250.0, "baoyu");
ShopInfo longxia = new ShopInfo(2, "龙虾", 251.0, "longxia");

shops.add(baoyu);
```

```
shops.add(longxia);

String json = JSON.toJSONString(shops);
```

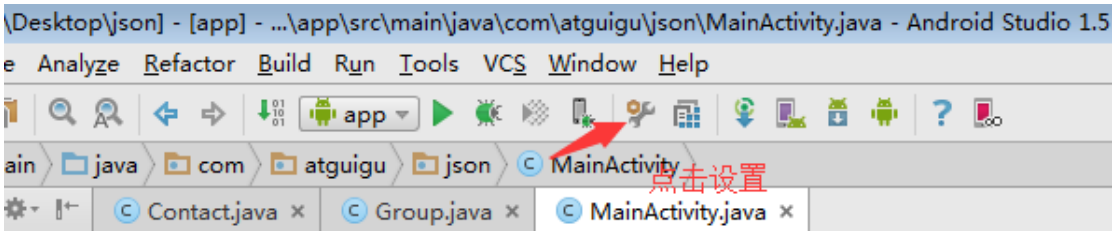
3）例子

```
// （4）将 Java 对象的 List 转换为 json 字符串[]
private void javaToJsonArrayByFastJson() {

    // 1 获取 Java 集合
    List<ShopInfo> shops = new ArrayList<>();
    ShopInfo baoyu = new ShopInfo(1, "鲍鱼", 250.0, "baoyu");
    ShopInfo longxia = new ShopInfo(2, "龙虾", 251.0, "longxia");

    shops.add(baoyu);
    shops.add(longxia);

    // 2 生成 JSON 数据
    String json = JSON.toJSONString(shops);

    // 3 数据显示
    tv_fastjson_orignal.setText(shops.toString());
    tv_fastjson_last.setText(json);
}
```

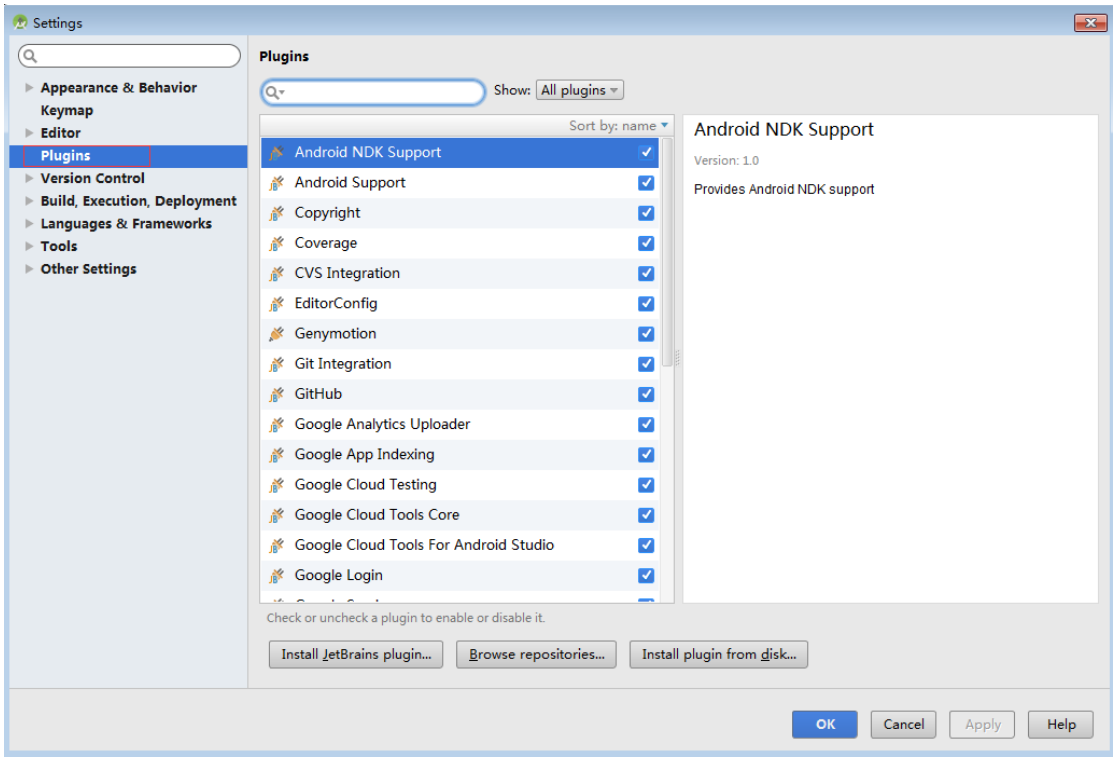
5_工具使用

5.1_GsonFormat

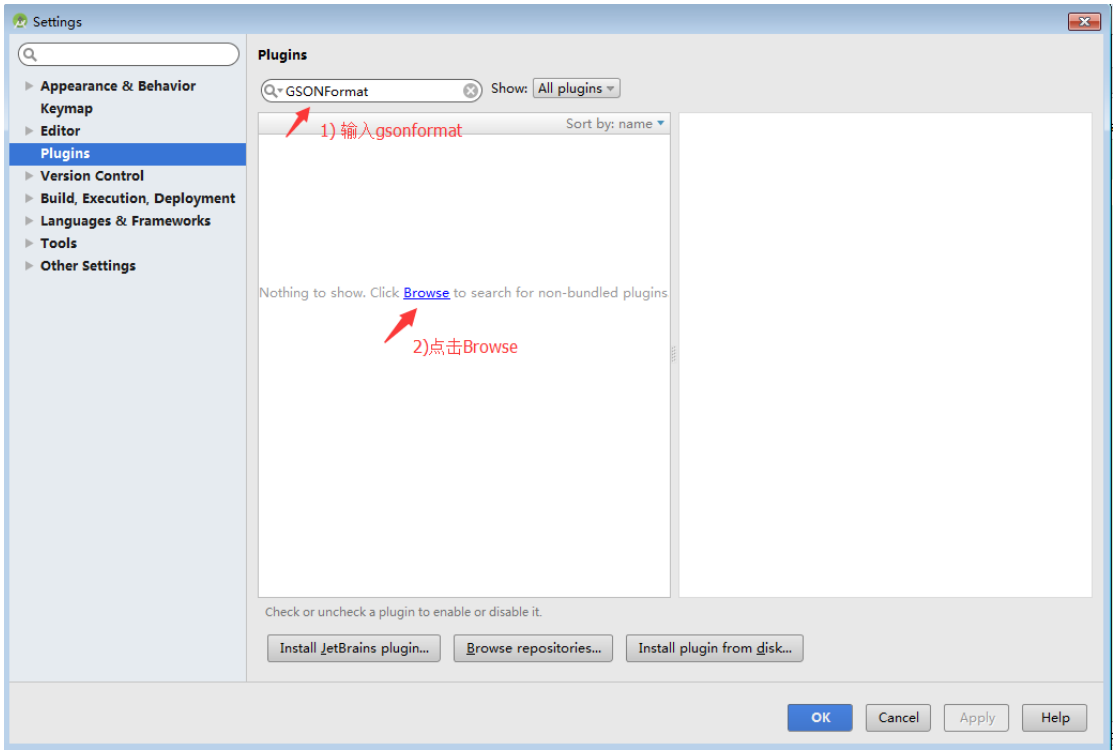
1）打开 Android studio 页面，点击设置按钮。



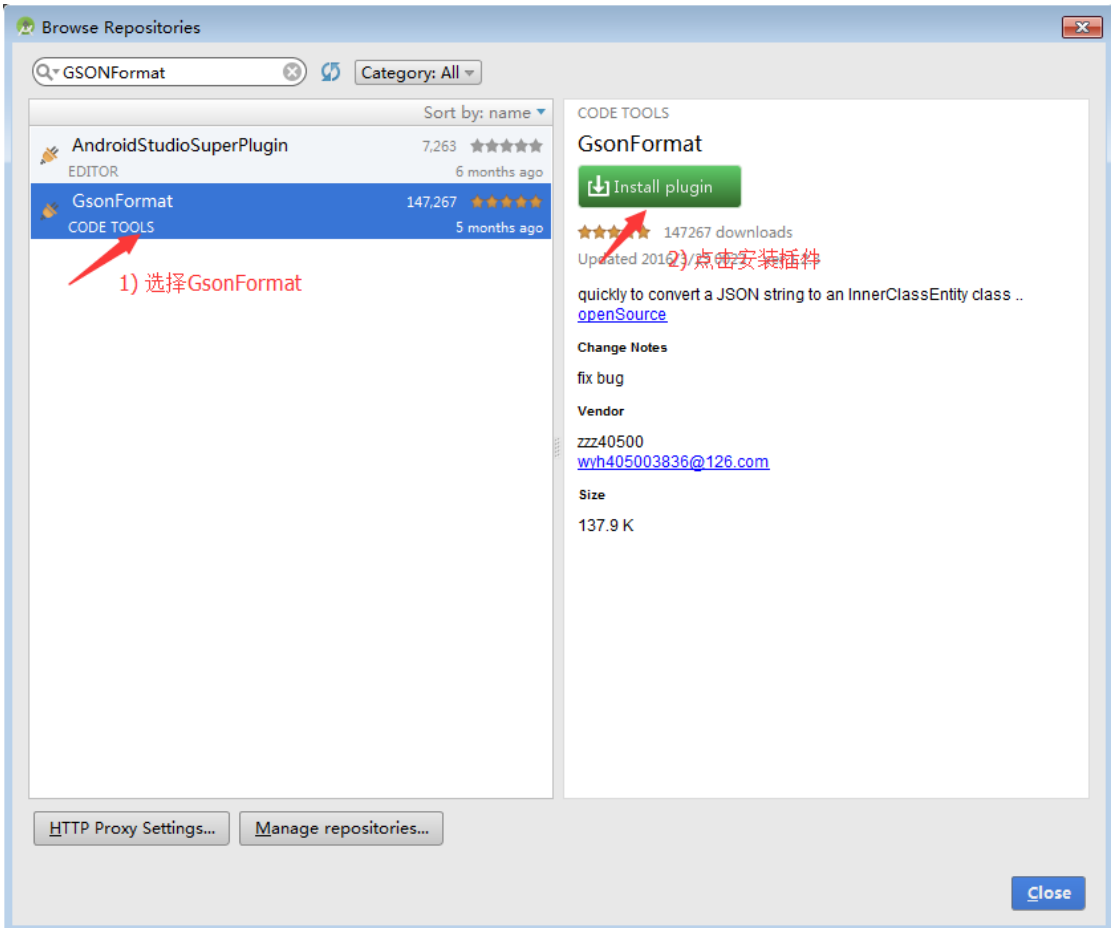
2）点击 Plugins 按钮



3）在右侧输入框中输入 gsonformat,然后点击中间部位的 Browse(必须在联网情况下点击)

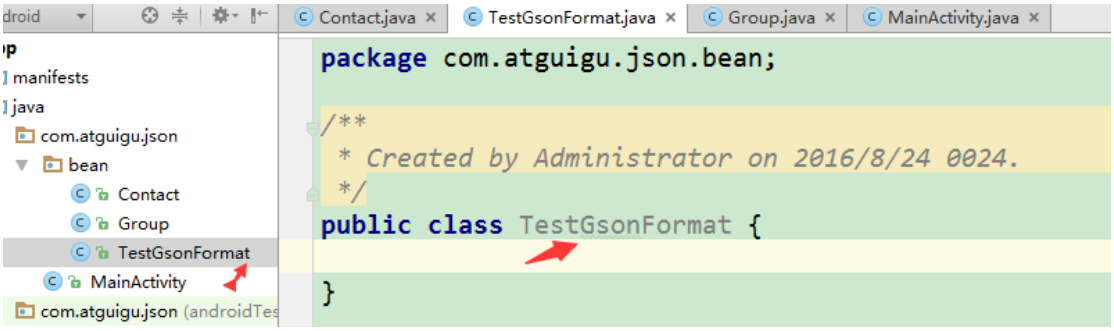


4）选择 GsonFormat，点击右侧的安装插件

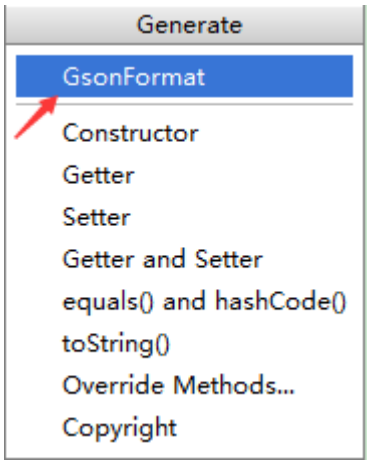


5）重启一下 Android studio

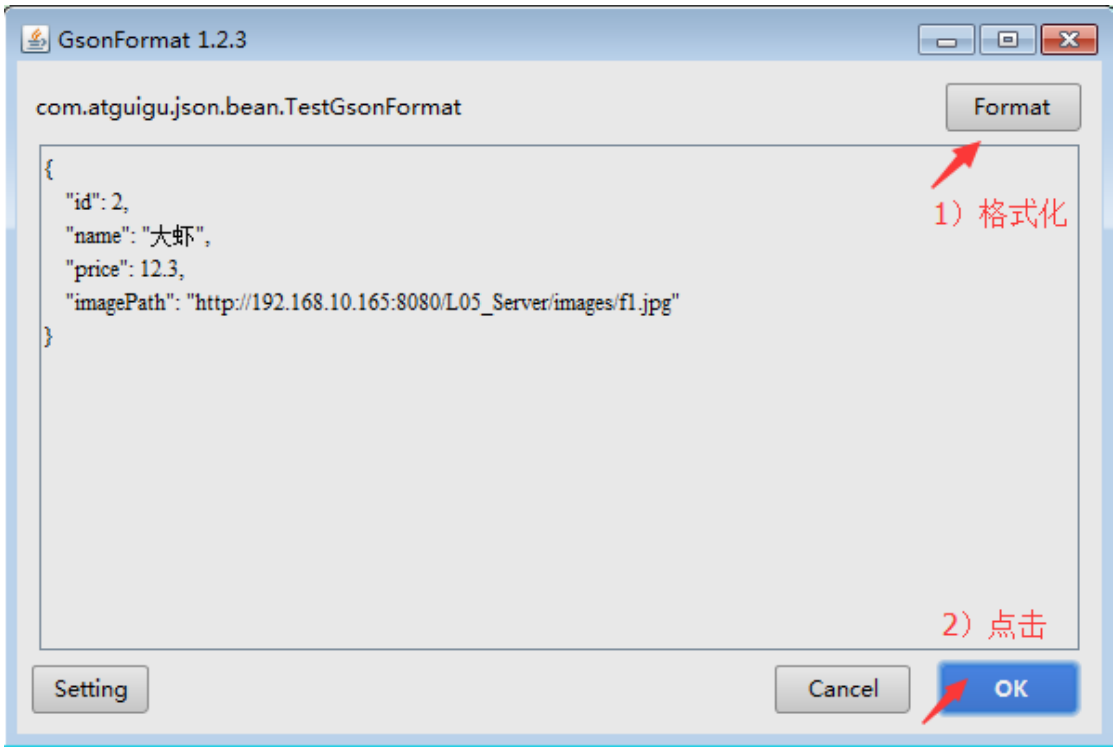
6）在 Android studio 中创建一个类



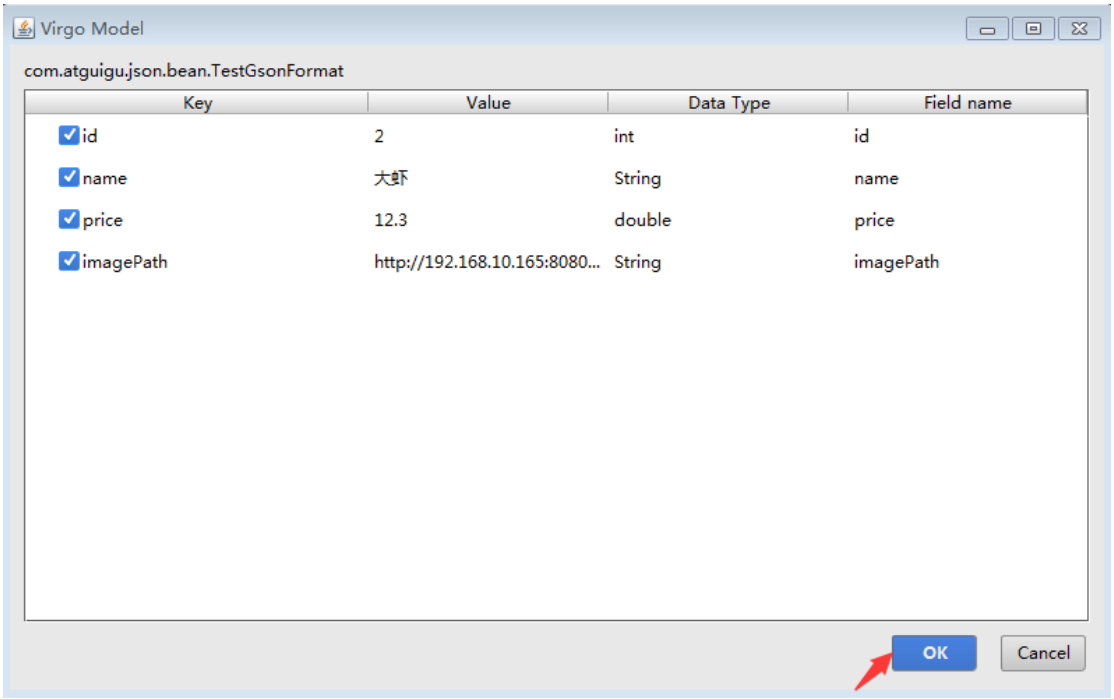
7）在该类中同时按下 alt+shift+s，并点击 GsonFormat



8）将要解析的 JSON 字符串粘贴到 GsonFormat 中



9）点击 OK



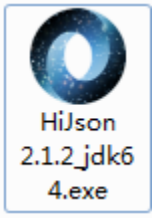
10）这样就将输入的 JSON 数据转换为了 bean 对象

```
public class TestGsonFormat {
    /**
     * id : 2
     * name : 大虾
     * price : 12.3
     * imagePath : http://192.168.10.165:8080/L05_Server/images/f1.jpg
     */
    private int id;
    private String name;
    private double price;
    private String imagePath;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
```

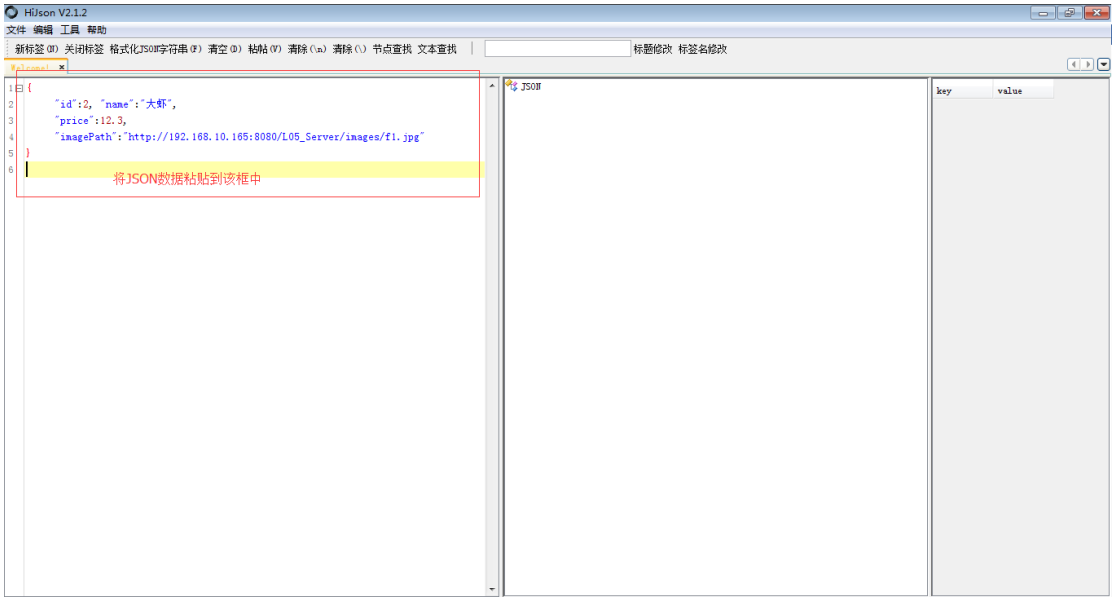
```
        this.name = name;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public String getImagePath() {
        return imagePath;
    }

    public void setImagePath(String imagePath) {
        this.imagePath = imagePath;
    }
}
```

5.2_HiJson

1）双击图标



2）将要解析的 JSON 数据粘贴到左侧页面



3）点击格式化 JSON 字符串，在右侧就会方便查看 JSON 数据