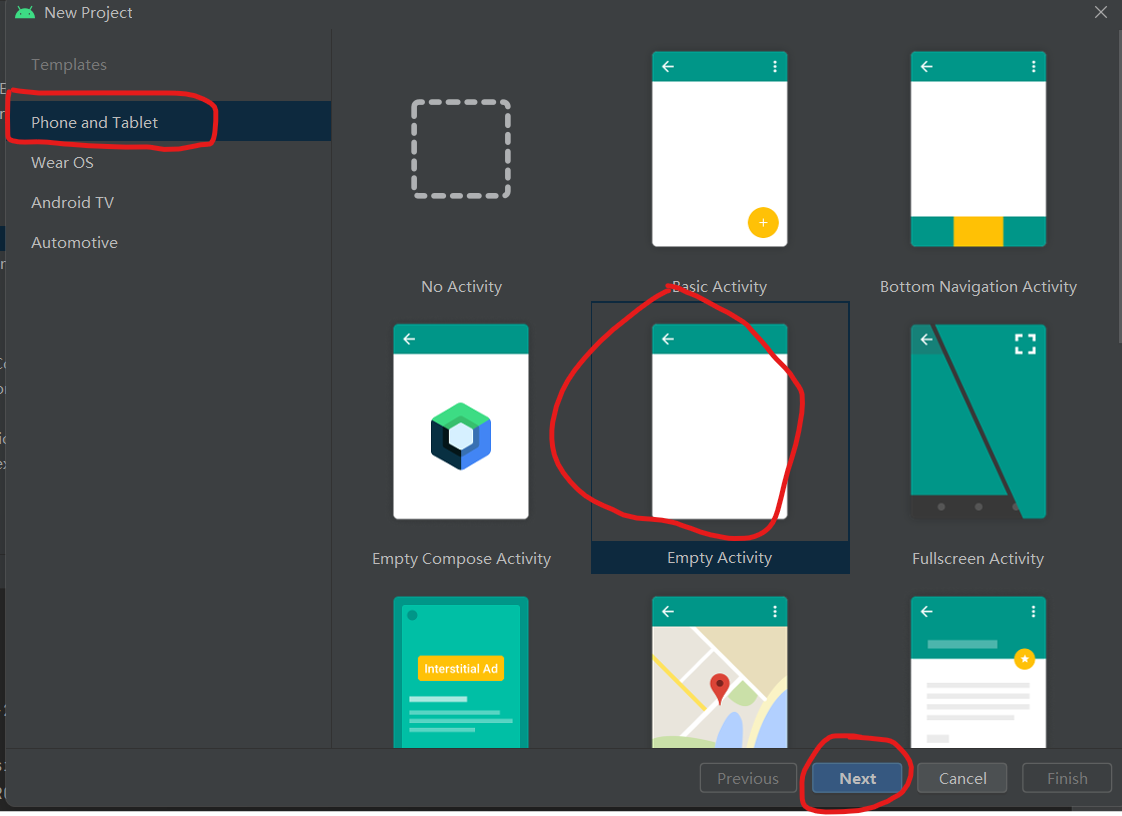
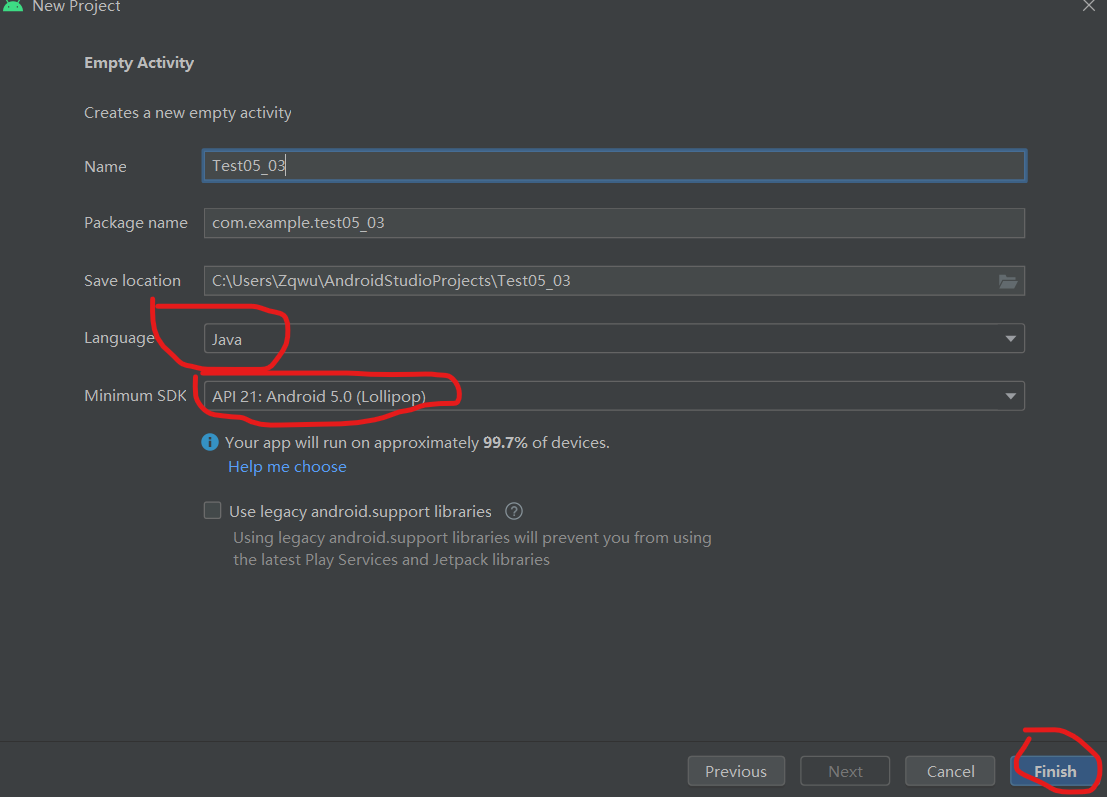
SpringInAndroid使用说明

1. 如何引用SpringInAndroid
   1. 创建项目

使用AndroidStudio创建项目。

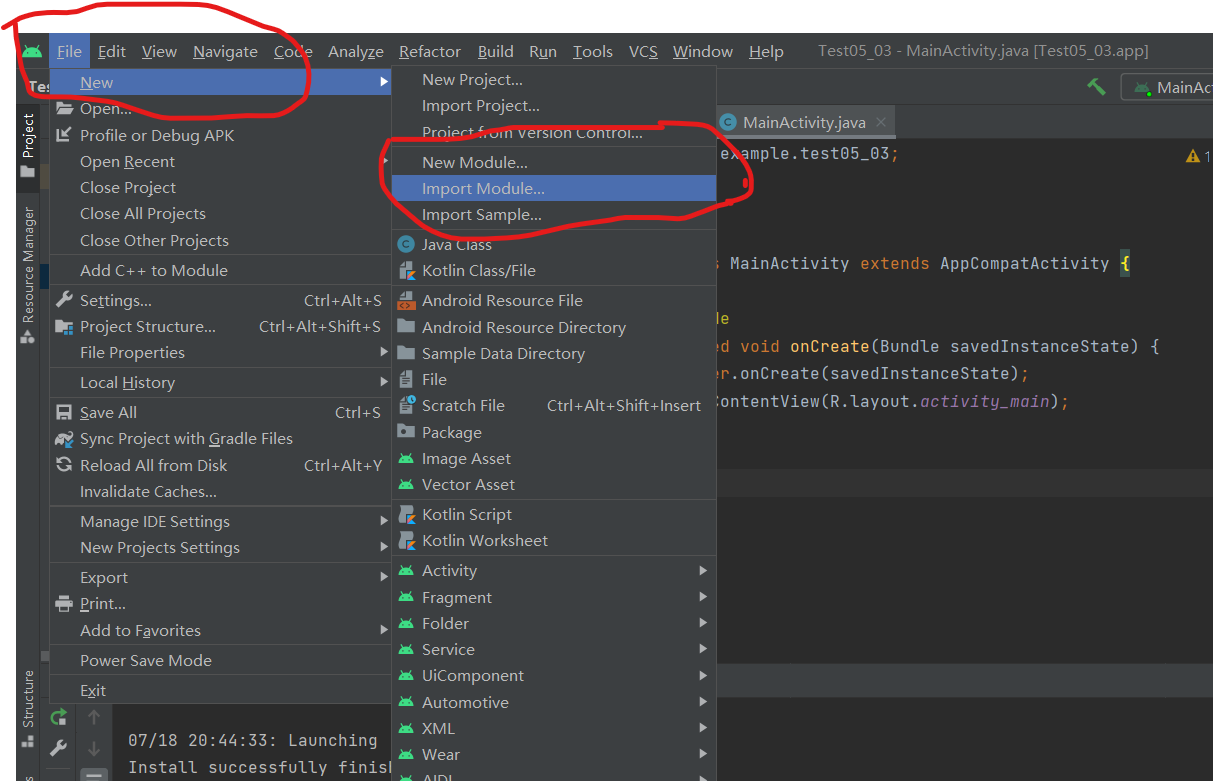


选择java语言，确定手机适配最低版本后点击finish，项目创建完毕。

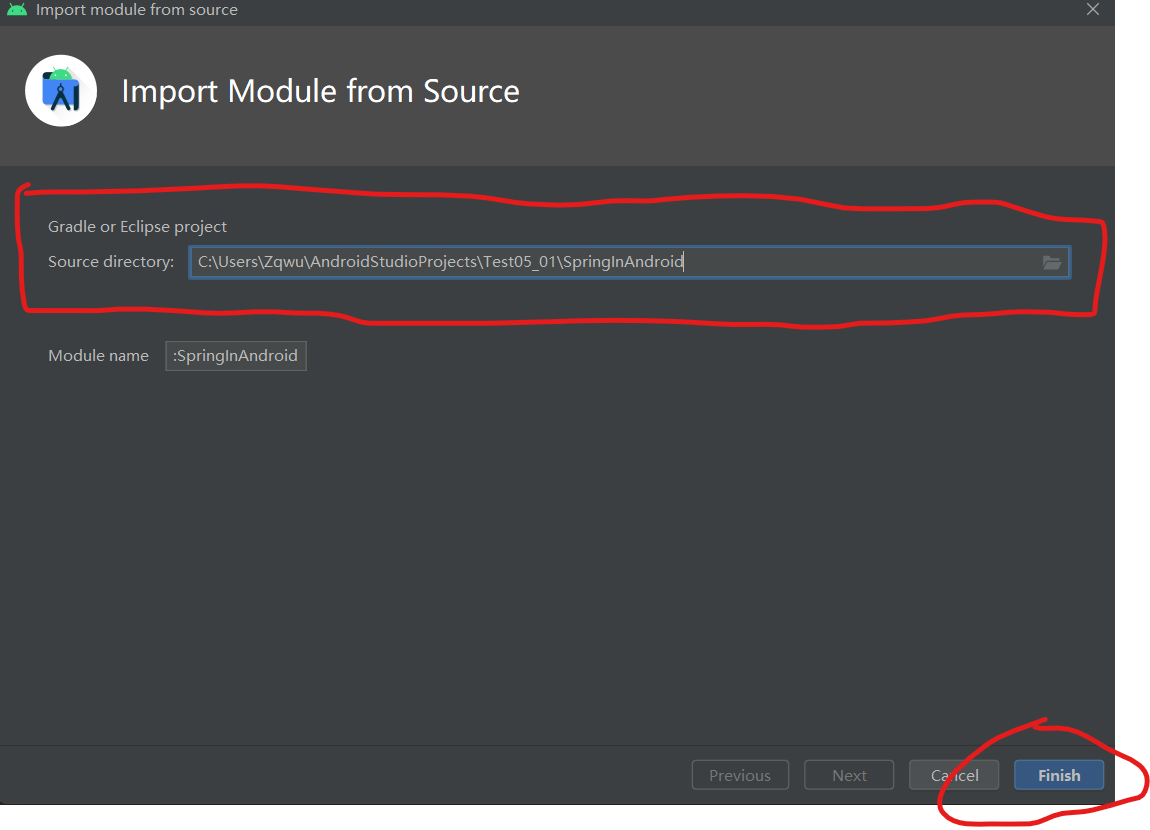


* 1. 引入依赖

提前在github上将依赖下载到本地，打开自己创建的新项目，点击import Module.



选择SpringInAndroid依赖的位置，点击finish即可。



1. SpringyInAndroid简单说明

2.1基于注解+xml生成Bean（推荐）

在xml文件中配置扫描路径，对项目中相应的类、字段、方法用相应的注解进行标记。

2.2基于xml文档生成Bean

在xml文件中配置Bean信息，并设置Bean依赖。

1. 项目实战（这里使用了注解+xml配置的方式解决）

3.1、面临的困境

假设我们有一个从各种来源获取数据的应用程序，我们必须解析不同类型的文件，比如解析CSV文件和JSON文件。

Android传统的写法：

定义一个解析的接口。

public interface Parser {  
 List parse(Reader r);  
}

根据不同文件类型有不同的实现类。

public class JSONParser implements Parser {  
 @Override  
 public List parse(Reader r) {  
 Log.*d*("test\_\_", "JSONParser解析");  
 return null;  
 }  
}

public class CSVParser implements Parser{  
 @Override  
 public List parse(Reader r) {  
 Log.*d*("test","CSVParser解析");  
 return null;  
 }  
}

最后写一个调用的客户端。

public class Client {  
 private Parser csvParser, jsonParser;  
  
 public Client(Parser csvParser, Parser jsonParser) {  
 this.csvParser = csvParser;  
 this.jsonParser = jsonParser;  
 }  
  
 Reader reader;  
  
 public List getAll(String contentType) {  
  
 switch (contentType) {  
 case "CSV":  
 return csvParser.parse(reader);  
 case "JSON":  
 return jsonParser.parse(reader);  
 }  
  
 return null;  
 }  
  
}

可以看到，通过switch case根据不同的类型调用不同的实现。

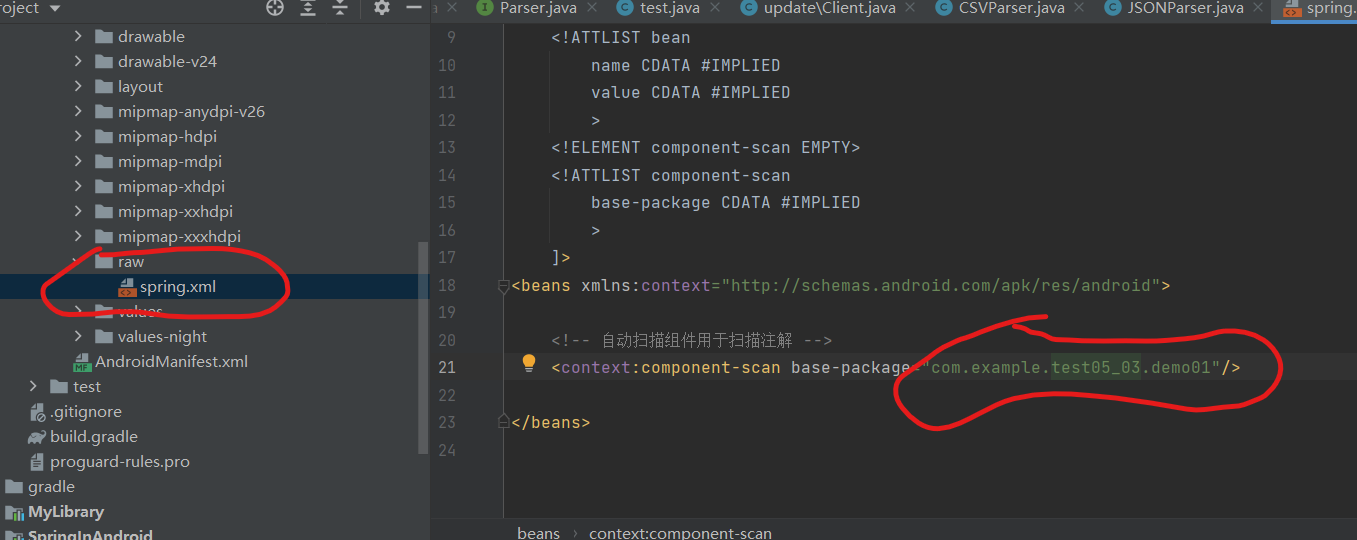
但是当我们出现新需求时，比如需要新增一个对xml文件的解析时，我们需要更改客户端的代码，这就导致了不同解析器和客户端之间的耦合了。

3.2、如何使用SpringInAndroid解耦

在依赖引入的基础上，我们对每个解析器使用@Component修饰，客户端代码更改如下：

@Component  
public class Client {  
  
 @Autowired  
 Map<String, Parser> parserMap;  
  
 Reader reader;  
  
 public List getAll(String contentType) {  
 return parserMap.get(contentType).parse(reader);  
 }  
  
}

在spring.xml中配置扫描路径。



在MainActivity中进行测试，测试结果如预期一致，解耦成功。

