Provider 结合 fetcher 搭建方式

一、前言

1. 这种方式的特点如下：
2. 不能使用 graphql.scalars.ExtendedScalars 来进行类型扩展（其实无伤大雅，用自定义的复杂类型扩展不香吗），只能使用自定义的复杂类型扩展，建议参考 graphql.scalars.ExtendedScalars 源码进行扩展，没有的只能自己写了。
3. 可以使用 playground 插件（见下面官网的例子）
4. 可以利用 provider 等结合 controller 等来进行前后端交互
5. Graphqls 必须只有一个根文件，其他的 query 或者 mutation 必须继承根文件的类型才可以正常使用
6. 总体来说，相对于 resolver 和 provider **实用性较差**，因为 fetcher 是面向每一个接口的，当 graphqls 的接口较多时，会产生很多的类，不像 resolver 一样，是面向 graohqls 文件的。

二、搭建方式

1. 搭建方式有两个教程，一个是参考

https://blog.csdn.net/qq_40794266/article/details/102972273

另一个是参考官网

https://www.graphql-java.com/tutorials/getting-started-with-spring-boot

---

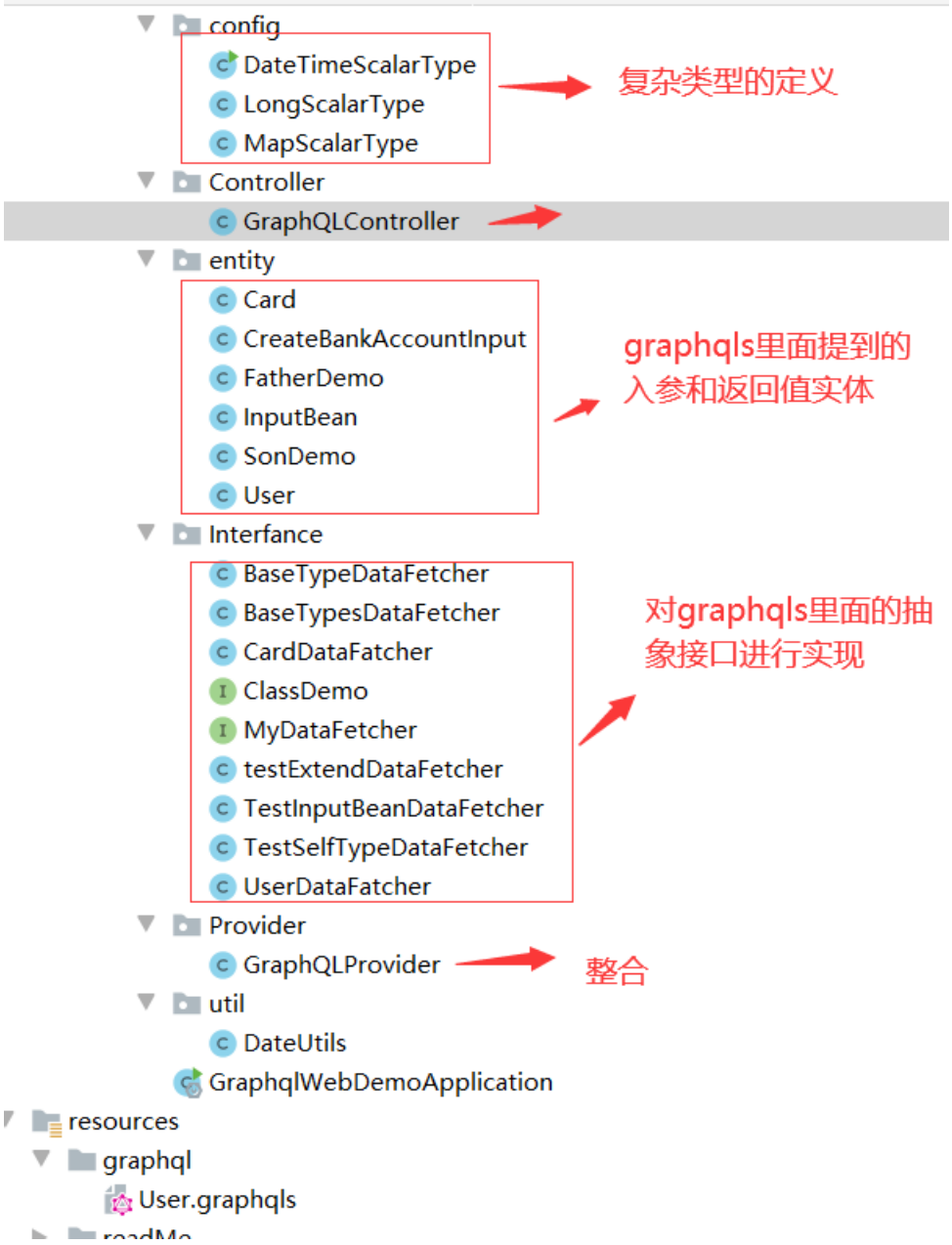
Getting started with Spring Boot

This is a tutorial for people who want to create a GraphQL server in Java. It requires some Spring Bo Java knowledge and while we give a brief introduction into GraphQL, the focus of this tutorial is on a GraphQL server in Java.

GraphQL in 3 minutes

GraphQL is a query language to retrieve data from a server. It is an alternative to REST, SOAP or gRF way.

2. 下面是按照第一个进行结构实例讲解



3. 因为 controller 里面的注释比较重要，所以特地考出来，防止因为编码的问题看不清，这里面保存了很重要的访问参数。
4. 技术升级：技术升级参照官网及视频教程，官网的话比较全面，但是是全英文文档，视频的话讲解的比较详细，大概包括以下内容：

视频选集 (1/17)　　自动连播

P1	01-课程说明和环境需求	16:49
P2	02-创建Schema和Query	22:56
P3	03-maxQueryDepth配置演示	14:03
P4	04-Playground配置演示	12:07
P5	05-Schema可视化Voyager演示	06:08
P6	06-数据解析器Resolvers	13:43
P7	07-定制异常处理Spring ExceptionHandler	11:00
P8	08-定制异常处理GraphQLErrorHandler	08:08
P9	09-通过DataFetcherResult返回部分数据	05:01
P10	10-异步数据解析(Async Resolvers)	09:52
P11	11-改变数据(Mutations)	11:46
P12	12-文件上传(File Upload)	12:28
P13	13-DataFetchingEnvironment_Selection...	09:48
P14	14-定制标量(Custom Scalas)	13:03
P15	15-定制日期标量(Date Type Scalars)	09:00
P16	16-输入校验(Input Validation)	05:29
P17	17-请求监听器(Request Listener)	10:16

13-DataFetchingEnv

5.

```
@GetMapping("/graphql")
 @ResponseBody
// public Map<String,Object> graphql(@RequestParam("query") String
query){
    public Map<String,Object> graphql(){
    String query = null;
    //注意  这里的query和background  插件里的query是一样的!!!!!!!!!!

    //根据query  的json(类似json 而已)内容  决定访问  是card方法  还是user方
法  以及对应的返回值
    //比如  第一个访问的就是  query方法  第二个访问的就是  user方法
    //内嵌的如果是实体或者数组  都按照下面的方式进行
    //query = "{card(id:1){id,address}}";
    //query =
"{user(id:1){id,name,age,card{id,address},cards{id,address}}}";
    //query = "{baseType}";//测试回参是基本类型
    //query = "{testExtend{id,name,address,firstName,IS_LAST}}";// 继承关
系
    //query = "{testInputBean(input: {id:\"1\",name:\"王五\"})}";//测试入
参是bean类型
    query = "{testSelfType(num:5,myDateTime:\"2021-01-01 01:01\")}";//测
试graphql里面的自定义类型
    return graphQL.execute(query).toSpecification();
 }
```