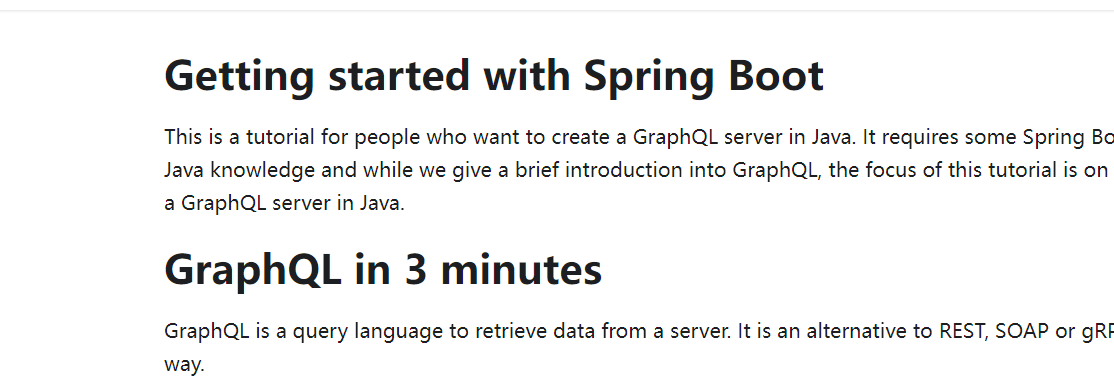
Provider结合fetcher搭建方式

1. 前言
2. 这种方式的特点如下：
3. 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。
4. 可以使用playground插件（见下面官网的例子）
5. 可以利用provider等结合controller等来进行前后端交互
6. 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
7. 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。
8. 搭建方式
9. 搭建方式有两个教程，一个是参考

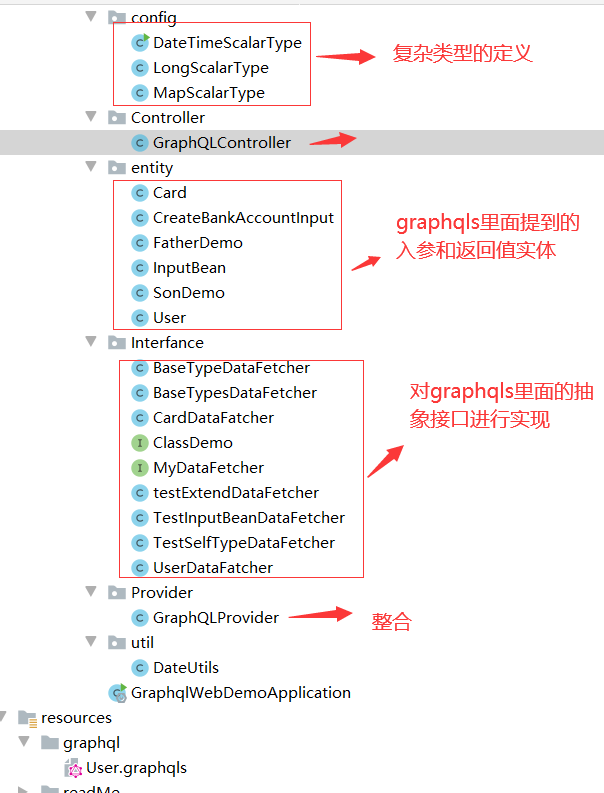
<https://blog.csdn.net/qq_40794266/article/details/102972273>

另一个是参考官网

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



1. 下面是按照第一个进行结构实例讲解



1. 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。
2. 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：



@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();  
 }