07 SpringBoot3和MyBatis查询结果返回Map或List示例

一、Map

1、单个Map

AuthorMapper.xml

```
<select id="getAuthorMap" resultType="java.util.Map">
SELECT *
FROM author
LIMIT 1
</select>
```

AuthorMapper.java

```
@MapKey("id")
Map getAuthorMap();
```

service和controller省略
返回结果：

```
{
"17596572231704509 48": {
"birthday": "1881-09-25",
"nationality": "中国",
"name": "鲁迅",
```

```
”id”: 17596572231704509 48,
”deathday”: ”1936-10-19”
}
}
```

2、单个HashMap

AuthorMapper.xml

```
<select id=”getAuthorHashMap” resultType=”java.util.HashMap”>
SELECT *
FROM author
LIMIT 1
</select>
```

AuthorMapper.java

```
HashMap getAuthorHashMap();
```

service和controller省略
返回结果：

```
{
”birthday”: ”1881-09-25”,
”nationality”: ”中国”,
”name”: ”鲁迅”,
”id”: 17596572231704509 48,
”deathday”: ”1936-10-19”
}
```

3、单个LinkedHashMap

AuthorMapper.xml

```
    <select id="getAuthorLinkedHashMap"
resultType="java.util.LinkedHashMap">
        SELECT *
        FROM author
        LIMIT 1
    </select>
```

AuthorMapper.java

```
LinkedHashMap getAuthorLinkedHashMap();
```

service和controller省略
返回结果：

```
{
"id": 1759657223170450948,
"name": "鲁迅",
"nationality": "中国",
"birthday": "1881-09-25",
"deathday": "1936-10-19"
}
```

4、MapKey注解

4.1、作用

示例1中的Mapper方法中有使用注解`@MapKey`，在使用Map作为返回类型时，如果不增加这个注解，会出现提醒`@MapKey is required` 。

@MapKey注解用于指定一个Map类型的属性中作为键的字段。
如果将示例1的注解改为`@MapKey(”name”)`，返回的结果是：

```
{
”鲁迅”: {
”birthday”: ”1881-09-25”,
”nationality”: ”中国”,
”name”: ”鲁迅”,
”id”: 17596572231704509 48,
”deathday”: ”1936-10-19”
}
}
```

4.2、使用场景

@MapKey注解通常与@OneToMany和@MapKeyJoinColumn注解一起使用，用于描述一对多关系中的映射关系，将数据库中的一对多关系映射到Java对象中的Map类型属性。
使用@MapKey注解的优势包括：

* 提供了一种简洁的方式来指定Map类型属性中作为键的字段或方法。
* 可以在一对多关系中更灵活地定义映射关系，使代码更加清晰和易于理解。
* 可以帮助开发人员更好地管理和维护数据之间的关系。
当需要在Java对象中表示一对多关系，并且希望将多的一方映射为一个Map类型的属性，并且指定Map中的键时，可以使用@MapKey注解。
示例： 假设有两个实体类User和Phone，一个用户可以拥有多个电话号码，可以使用@OneToMany和@MapKey注解来描述这种一对多关系，如下所示：

```
@Entity
public class User {
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @OneToMany
    @MapKey(name = "phoneType")
    private Map<String, Phone> phones = new HashMap<>();

    // 省略其他属性和方法
}
@Entity
public class Phone {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String phoneNumber;

    private String phoneType;

    // 省略其他属性和方法
}
```

在上面的示例中，User类中有一个Map类型的属性phones，使用@OneToMany和@MapKey注解来描述用户拥有多个电话号码的关系，并指定phoneType字段作为Map中的键。这样就可以通过User对象的phones属性来获取用户拥有的电话号码，以phoneType作为键

5、Map、HashMap、LinkedHashMap异同

5.1、概念

* Map是Java中的一个接口，用来表示键值对的映射关系。
* HashMap是Map接口的一个实现类，它使用哈希表来存储键值对，具有快速的查找和插入性能。
* LinkedHashMap也是Map接口的一个实现类，它在HashMap的基础上使用双向链表来维护键值对的插入顺序。

5.2、区别

* HashMap不保证键值对的顺序，而LinkedHashMap保证键值对的插入顺序

。**示例3的返回结果键值对顺序是和数据库表的顺序一样的**，而示例1、2则不同。
* LinkedHashMap在HashMap的基础上增加了一些额外的空间和时间开销来维护键值对的插入顺序。

5.3、项目使用

* 项目中，由于大部分接口对返回的键值对顺序并没有要求，所以使用HashMap更多一些，毕竟比起使用LinkedHashMap性能更好一些。
* 在实际开发中，接口通常返回类型为`HashMap<String, Object>`，它表示查询结果将以HashMap<String, Object>的形式返回，其中键为String类型，值为Object类型。这种方式更明确地指定了键的类型为String，更方便直观地操作键值对数据。

二、List

ArrayList<Map>

AuthorMapper.xml

```
<select id="getAuthorArrayListMap" resultType="java.util.Map">
SELECT *
FROM author
 LIMIT 2
</select>
```

AuthorMapper.java

```
@MapKey("id")
List<Map> getAuthorArrayListMap();
```

service和controller省略
返回结果：

```
[
{
”birthday”: ”1881-09-25”,
”nationality”: ”中国”,
”name”: ”鲁迅”,
”id”: 17596572231704509 48,
”deathday”: ”1936-10-19”
},
{
”birthday”: ”1874-01-25”,
”nationality”: ”英国”,
”name”: ”威廉·萨默塞特·毛姆”,
”id”: 17606885125990504 32,
”deathday”: ”1965-12-16”
}
]
```