**BadmintonCourt项目文档**

目录

**一、项目结构1**

**二、各层具体方法介绍2**

1.Domain层3

2.Dao层3

3.Service层4

4.Controller层4

5.BadmintoncourtApplication入口类 5

6.ApplicationStart类5

7.Mapper层6

8.测试类6

**三、样例输出结果6**

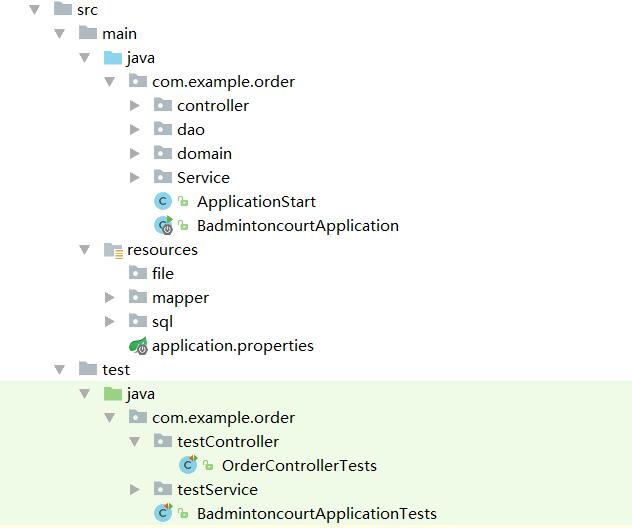
1. **项目结构**

本项目主要使用Springboot、Mybatis、Junit、ojdbc第三方库，使用的数据库为Oracle

注：本项目虽然使用Web后端框架及其分层形式，但本项目中的各层并非指Web工程中的各层，仅是根据Web工程中的划分层次依据对各层进行具体区分

在项目导入、数据库建表后，在application.properties文件中修改数据库配置

1. 项目代码段结构图



main/java包中为项目的代码段

其中BadmintoncourtApplication类为Springboot入口类

main/resources包中为项目的配置文件

mapper包中为Dao接口类的映射文件

sql包中为项目的sql脚本（即数据库配置及索引配置）

application.properties文件记录Springboot配置、Mybatis及数据库链接配置

file包中存储了收入总计打印到的文件，但真正打印到的地方在target/classes/file包中

test/java包中为进行Junit单元测试的代码部分

1. Springboot编译后代码结构图



编译后的代码结构与项目本身结构基本一一对应

需要注意的是，只有target/file/generalIncome文件中记录了收入汇总的清单表

1. **各层具体方法介绍**
2. **Domain层**

定义了订单对象，其中定义了属性userId、orderDate、startTime、endTime、court、mark、fee，并与数据库中的表项一一对应；

重写了toString()方法，以便获取收入汇总时对Order对象的打印。

1. **Dao层**

主要包含最基础的数据库操作相关操作的方法，与Mapper相映射

（1）insertOrder 添加订单对象

（2）cancelOrder 取消订单

（3）getByOrder 通过订单对象获取订单

（4）getAll 获取所有订单

（5）getByOrderDate 通过日期场地获取订单

（6）getByCourt 通过场地获取订单

1. **Service层**

在Dao层基础上对数据进行逻辑处理，其中包括判断对象是否与数据库对象冲突，在数据库中是否含有此对象，根据日期判断对应的星期，根据星期及时间范围计算出相对应的费用

1. insertOrder

先判断数据是否冲突，然后进行添加

1. cancelOrder

先判断数据库是否含有此条订单，然后进行取消

1. CalculateFee

先获得日期对应的星期，然后根据星期，起止时间，判断费用

1. getDayOfWeek

通过日期类型，获得对应的星期

1. isConflict

获取对应场地、日期的所有订单，遍历判断起止时间是否冲突

1. isExist

根据Order对象查询数据库，判断是否含有此订单且还未取消

1. getAll

获取数据库中所有订单对象

1. getByCourt

根据场地获取订单对象

1. **Controller层**

在Service层基础上，每次对从控制台获取的字符串的合法性进行判断

1. addOrder

判断字符串合法性后，将字符串解析赋值给Order对象，调用Service层的添加

1. cancelOrder

判断字符串合法性后，将字符串解析赋值给Order对象，调用Service层的取消

1. getByCourt

通过场地获取所有Order对象

1. **BadmintoncourtApplication入口类**

由于本项目非Web工程，故继承CommandLineRunner接口，实现run（）方法，在Main()函数中调用run()方法启动项目

注：在入口类的main（）方法中无法使用@Autowired自动注入的类，但在run（）方法中可以使用自动注入的类

1. **ApplicationStart类**
2. 为什么要使用ApplicationStart类

答：由于此项目使用的Scanner进行控制台输入，且将结束条件判断为hasNextLine（）。若放在入口类的run（）方法中，在每次单元测试时，[由于不可避免地使用了@RunWith(SpringRunner.class)](mailto:由于不可避免地使用了@RunWith(SpringRunner.class))注解，每次测试时会从入口类创建工程，调用Scanner控制台输入（由于本人能力有限，未找出在测试类中关闭入口类Scanner的方法），使得测试进入死循环，导致无法进行测试。这也是为什么在单元测试时需要注释main.start();的原因。

（2）ApplicationStart类在Controller层基础上，添加了收入汇总的方法，并打印到target/classes/file/generalIncome文件中

1. **Mapper层**

Mapper层中主要写了关于Dao层接口中方法的映射

问：为什么在Mapper中有to\_date(#{orderDate} || ' ' || #{startTime} ,'yyyy-mm-dd HH24:mI')？

答：因为本项目要求收回汇总根据日期时间排序，故，采用此种方法对数据库中的日期项进行拼接赋值和查找，然后在获取Order对象集合时，对其进行日期排序。

1. **测试类**

参照第5页说明，故无法对ApplicationStart类进行测试，对入口类及Dao接口测试的意义不大，故仅对Controller层和Service层进行测试

1. **样例输出结果**

注：由于自己本地数据库中数据内容多，故收入汇总的输出结果比样例多

