**0217**

1-实现mybatis步骤：

1.添加两个jar

2.创建数据库表

3.写conf.xml

4.创建对应表的实体类

5.写Mapper.xml

6.写对应Mapper.xml中的增删改查方法的接口

7.写测试类测试方法

2-配置文件属性

（1）mybatis核心配置文件 sqlmapconfig.xml配置文件中

首先要写根元素 （根元素叫：configuration）

environments （default属性 对应下面多个environment标签中id值），environment (id属性，用来给当前连接数据库环境取个名字)

transactionManager （type 属性值如果为jdbc话，代表当前事务管理交由开发人员自己进行管理；属性值如果是managed 代表mybatis把事务管理交给第三方进行管理）

dataSource (type属性，用来数据库连接池 如果写UNPOOLED代表不需要数据库连接池；如果写POOLED代表需要数据库连接池；如果值为JNDI的话，代表数据库连接池功能交给其他应用服务)

mapper （resource属性 跟随映射mapper配置文件所在相对路径地址）

（2）mapper映射文件：

mapper（namespace命名空间属性 该属性值 对应当前mapper配置文件对应的接口的路径，包括该接口的包路径+接口路径）

（id属性对应接口中方法名字 id值是唯一的 不可以重复（上述四个标签一定包含的属性）

parameterType方法入参类型，如果是八大基本类型则直接写类型名 ，例如int float String等都可以 如果是一个引用类型的，这种需要写全路径

resultType：方法返回值类型 ，其内容同parameterType 。值得注意的是无论一条还是多条数据，返回值类型都按照一条记录的类型去书写）

3-操作实例

1、实现模糊查询有两种方式：

1. 使用#{}占位符 传递的占位符内容中 自行传入% %

where name like #{name} queryLikeName(‘%e%’)

1. 使用${}使用该方式 直接拼接SQL 直接写入要查询的模糊词条

Where name like ‘%${value}%’ queryLikeName(‘e’)

2、返回刚添加数据记录的主键ID值（前提条件是当前表的主键设置为自增长的）

方法1、selectKey标签配合使用 查询刚添加数据的主键值 适用绝大多数数据库 甚至一些不带主键自增功能的数据库

selectKey(keyColumn属性:查询主键的字段名，keyProperty属性:实体类对应的属性名，order属性:AFTER/BEFORE,resultType属性:返回类型)

select LAST\_INSERT\_ID()-----为mysql自带的函数，其他数据库要变化

方法2、在insert标签里添加useGenerateKey="true" keyProperty="对应属性名称" 适用于当前表已经设置好主键自增 仅限于sqlserver以及mysql数据库两种带有主键自增功能的数据里

order：BEFORE/AFTER

AFTER:先选择主键，设置keyProperty，执行插入，再执行selectKey

BEFORE：先执行selectKey，再执行插入---------返回结果为0

4-mybatis核心对象彼此创建关系

1、SqlSessionFactoryBuilder > SqlSessionFactory > SqlSession

SqlSessionFactoryBuilder创建SqlSessionFactory，通过SqlSessionFactory使用openSession来获取SqlSession

2、SqlSessionFactory 注意事项：（面试可能会考）

SqlSessionFactory是线程安全的，建议一个项目中创建一个该对象即可

每个SqlSession 对象是单线程的，SqlSession是线程不安全的，不要把他设置为全局变量，在某一个方法中使用，用完了释放掉

**0218**

1-resultMap的使用

1、如果类中属性和数据库属性类型相同可合理识别用resultType 否则用resultMap

2、如果类中属性和数据库字段名相同可合理识别用resultType 否则用resultMap

主要使用resultmap进行手动属性匹配 -->

<resultMap type="student" id="selectStudentMap">

<id property="sid" column="id"/>

<result property="sname" column="name"/>

<result property="sage" column="age"/>

</resultMap>

<select id="selectStudentResultMap" resultMap="selectStudentMap">

select \* from student

</select>

id->cid name->sname age->sage

1. selectsql语句的属性

1.useCache:二级缓存,默认对select开启

一级缓存不跨sqlsession,默认开启;二级缓存跨sqlsession,需手动开启

数据库开启了一级缓存,第一次查询回答结果放在和缓存中,如果下次执行同一过程,直接从缓存中提取,不经过数据库

2.timeout:设置定时器，超时异常，默认不设置 -->

3-设置别名

设置别名可以减少代码冗余有以下两种方式

1.设置单个或者多个别名

<typeAliases>

<typeAlias type="org.pro.mybatis.pojo.Student" alias="student"/>

</typeAliases>

org.pro.mybatis.pojo.Student----->student

2.批量设置别名,映射相应的文件包，将对应保重类名作为别名 无视大小写，所以在类创建的时候需要注意命名

</typeAliases>-->

<package name="org.pro.mybatis.pojo"/>

</typeAliases>

org.pro.mybatis.pojo.Student----->student/Student

4-配置加载映射文件，加载映射文件bookMapper.xml

1.通过mapper.xml引入,没有限制条件

<mapper resource="org/pro/mybatis/mapper/studentMapper.xml"/>

2.通过mapper映射引入,需要注意配置文件和映射文件名称相同且在同一个文件夹

<mapper class="org.pro.mybatis.mapper.StudentMapper"/>

3.引入映射包需要注意配置文件和映射文件名称相同切在同一个文件夹内

<package name="org.pro.mybatis.mapper"/>

1. 实现一对多关联查询
2. 创建一对象，创建多对象，里面包含一个一对象的属性比如一对象的id
3. 创建相应pojo,再一对象添加属性list表示对应一对多链表接收多对象
4. 编写sql查询mapper.xml

Collection表示接受对应的多对象

<resultMap type="teacher" id="selectTeacher">

<id property="tid" column="tid"/>

<result property="tname" column="tname"/>

<result property="tage" column="tage"/>

<collection property="studentlist" ofType="student">

<id property="sid" column="sid"/>

<result property="sname" column="sname"/>

<result property="sage" column="sage"/>

<result property="tid" column="tid"/>

</collection>

</resultMap>

<select id="selectAll" resultMap="selectTeacher">

select \* from teacher t,student s where t.tid=s.tid

</select>

1. 编写映射文件mapper.java
2. 编写测试类test.java
3. 实现一对一关联查询,基本和一对多差不多

差别在于

<association property="card" javaType="card">

<id property="cid" column="cid"/>

<result property="ccontent" column="ccontent"/>

<result property="tid" column="tid"/>

</association>

1. 实现一对一嵌套查询

主查询是查询老师信息

<select id="queryBookWithNumberO2OLater" resultMap="teacher">

select \* from teacher

</select>

<resultMap id="teacher" type="teacher">

<id property="tid" column="tid"/>

<result property="tname" column="tname"/>

嵌套查询是查询对应tid的card信息

根据主查询得到的tid,来调用另一个sql查询来获得对应tid的card对象

<association property="card" javaType="card" select="org.pro.mybatis.mapper.CardMapper.queryCard" column="tid">

</association>

</resultMap>

此处的“org.pro.mybatis.mapper.CardMapper.queryCard”是另一个sql查询需要事先写好

1. 当入参为多个值的处理方法
2. 将多个值，封装在一个类里面，将此类作为入参传入
3. 入参设置为HashMap形式
4. 直接多值传入但会存在引用参数的版本限制 如下，arg0,arg1,arg2

select \* from teacher where tname like #{arg0} limit #{arg1},#{arg2}

1. select \* from teacher limit n , m

Sql语句中limit的用法，n表示查询的位置，初始为0，比数据库位置少1，m为查询记录条数

1. 动态sql语句

基本结构：where

<select id="" parameterType="" resultType="">

.....

<where>

<if test="tname !=null and tname !=''"> </if>

<if test="tage !=null and tage !=0"> </if>

</where>

</select>

Where标签当下方存在时自动填补，且可以自动处理第一个and

If标签对传值进行判断，用于在多值搜索的时候存在空值

基本结构：foreach

<select id="" parameterType="" resultType="">

.....

<foreach collection="" item="" open=" and tid in (" separator="," close=")" >

.....

</foreach>

</select>

**0224**

1-springboot核心配置类需要@SpringBootApplication注解

主配置类包含

@SpringBootConfiguration:包含@Configuration表示配置类，会将此类自动纳入springioc容器

@EnableAutoConfiguration:

@AutoConfigurationPackage自动配置,找到主配置类对应的包，相当于扫描仪

2-全局配置文件

1)application.properties

格式:server.port=8081

2)application.yml非标记文档

格式：server:

port: 8081 层次关系通过垂直对齐,冒号后面有空格

两者都以#注释

1. 配置文件为pojo注入值

1.在pojo加入注解

@Controller

@ConfigurationProperties(prefix="person") -- 引入配置文件，进行配置文件赋值

2-1.在application.yml中注入赋值

person:

age: 12

name: "帅锋"

boss: false

birthday: 1998/07/25

maps: {height: 170,weight: 100}

list: ["one","two","three"]

pet: {name: "旺旺",age: 2}

其中hashmap,arraylist，对象有两种赋值格式

1、行内赋值

maps: {height: 170,weight: 100}

list: ["one","two","three"]

pet: {name: "旺旺",age: 2}

2、行外赋值

maps:

height: 170

weight: 100

list:

- "one"

- "two"

- "three"

pet:

name: "旺旺"

age: 2

2-2.在application.properties中注入赋值

person.age=12

person.name="\u6797\u67AB"

person.boss=false

person.birthday=1998/07/25

person.list: "one","two","three"

person.maps.height=120

person.maps.weight=100

person.pet.name="aa"

person.pet.age=2

2-3.使用额外的\*\*\*.properties文件

在对应pojo对象加上注解

@ConfigurationProperties(prefix="person")

@PropertySource(value= {"classpath:application\_person.properties"})

1. 通过自动注入从配置中获取测试
2. 配置文件占位符

随机数使用方式

${random.int},${random.value},${random.long}

1. 额外配置文件切换
2. 编写额外配置文件注意命名方式为application-\*\*.properties
3. 主配置文件中设置spring.profiles.active=\*\*来调用额外作用的配置文件
4. 测试
5. 多环境切换(profiles)

1)properties方式

默认读取application.properties中环境

自定义环境资源名：application-\*\*.properties

通过在默认文件中配置spring.profiles.active=环境名

2)yml多文档块方式

spring:

profiles:

active: dev

---

server:

port: 8091

spring:

profiles: dev

3)properties优先级>yml

1. xml与config

@ImportResource(locations={"classpath:spring.xml"})

默认不支持自己编写配置，但可用@ImportResource指明

不推荐自己编写，推荐使用注解@Configuration @Bean

1. freemaker模板样式
2. 引入依赖

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-freemarker</artifactId>

</dependency>

1. 文件位置为resources的templates的当中，且文件名称为\*\*\*.ftl才可被拦截起作用
2. 语法

1-接收一般传值${username}，类似于ER表达式

2-if判断

<#if age lt 10> 儿童

<#elseif age gt 18>中年

<#else> 青年

</#if>

1. 链表类型

<#list stlist?reverse as stu>

<td>${stu.name}</td>

</#list>

**0225**

1-thymeleaf使用

1.添加依赖

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

2.thymeleaf编写需要注意严格按照前端写法编写

3.语法

一般取值

<h3 th:text="${title}"></h3>

链表取值

<tr align="center" th:each="student:${stlist}">

<td th:text="${student.name}"></td>

</tr>

对象取值

<form th:object="${stu}" action="">

<input th:value="\*{name}" />

</form>

1. springboot整合mybatis

1.添加依赖

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.mybatis.spring.boot</groupId>

<artifactId>mybatis-spring-boot-starter</artifactId>

<version>2.1.1</version>

</dependency>

2.在配置文件配置数据源

spring.datasource.url=jdbc:mysql://127.0.0.1/\*\*\*?serverTimezone=GMT%2B8

spring.datasource.username=\*\*\*

spring.datasource.password=\*\*\*

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

3.新建核心配置类\*Mapper.xml,放在resources的mappers，和映射类\*Mapper.java

4.配置mapper映射

mybatis.mapper-locations=classpath:/mappers/\*Mapper.xml

mybatis.type-aliases-package=\*\*\*.mapper

对于简单的sql语句可以在mapper接口方法上写上@Insert,@Select,@Delete,@Update注解，而省去mapper.xml和相关配置

5.@MapperScan(value="org.pro.boot.mapper")--批量扫描mapper映射文件

3-springboot整合Druid数据库连接池

1.引入依赖:

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>druid</artifactId>

<version>1.1.10</version>

</dependency>

2.在配置文件进行数据库连接池相关配置

spring.datasource.type=com.alibaba.druid.pool.DruidDataSource

spring.datasource.initialSize=5

spring.datasource.minIdle=5

spring.datasource.maxActive=20

spring.datasource.maxWait=60000

spring.datasource.timeBetweenEvictionRunsMillis=60000

spring.datasource.minEvictableIdleTimeMillis=300000

spring.datasource.validationQuery=SELECT 1 FROM DUAL

spring.datasource.testWhileIdle=true

spring.datasource.testOnBorrow=false

spring.datasource.testOnReturn=false

spring.datasource.poolPreparedStatements=true

spring.datasource.maxPoolPreparedStatementPerConnectionSize=20

spring.datasource.filters=stat,wall,log4j

spring.datasource.connectionProperties=druid.stat.mergeSql=true;druid.stat.slowSqlMillis=5000

spring.datasource.useGlobalDataSourceStat=true

3.编写Druid配置config

编写一个ServletRegistrationBean和FilterRegistrationBean

[特别注意需要加上@ConfigurationProperties(prefix="spring.datasource")注解](mailto:特别注意需要加上@ConfigurationProperties(prefix=\"spring.datasource\")注解)，使其效果

4.测试访问

[http://localhost:8080/druid/webapp.html](http://localhost:8080/myboot/druid/webapp.html)