MyBatis

MyBatis是什么

MyBatis是一个半自动的ORM框架，封装了jdbc,加载驱动，创建连接，创建statement等繁杂的过程，开发者开发时只需要关注如何编写sql语句，可以严格控制sql的执行性能，灵活度高。

作为一个半自动的ORM框架，MyBatis可以使用XML或者注解来配置和映射原生信息，将pojo类映射成数据库中的记录，避免了几乎所有的jdbc代码，手动设置参数和获取结果集

通过xml文件或者注解的方式将要执行的各种statement配置起来，并通过java对象和statement中sql的动态参数进行映射生成最终的sql语句，最后由mybatis框架执行sql语句并将返回结果映射成为java对象

MyBatis专注于sql本身，灵活度高，适合性能要求高，需求变化多的项目

Mybatis优缺点

优点：

基于sql语句编程，相当灵活，sql写在xml文件中，降低与代码的耦合，支持动态sql，可以重用

与jdbc相比，减少50%以上的代码，不需要手动开关连接

很好的与各种数据库兼容，使用jdbc连接数据库，支持所有jdbc支持的数据库

能够与spring很好的集成

提供映射标签，方便的进行对象关系映射

缺点：

sql语句编写量大，需要一定的sql基础

sql语句依赖数据库,数据库移植性差

为什么说MyBatis是半自动的ORM框架

Mybatis在查询关系对象或者关系对象集合时，需要手写sql来完成，全自动框架可以根据对象关系模型直接获取

Hibernate和Mybatis的区别

相同点：都是持久层的框架，都是对jdbc的封装，用于dao层的开发

不同点

1.映射关系

Mybatis是半自动的ORM框架，配置的是java对象与sql语句执行结果之间的关系，多表关联关系配置简单

Hibernate是全自动的ORM框架，配置的是java对象与数据库表之间的关系，多表关联关系配置复杂

2.sql优化和移植

Mybatis需要手动编写sql，支持动态sql,直接使用sql操作数据库，不支持数据库无关性，sql语句容易优化

Hibernate是对sql语句的封装，提供了HQL操作数据库，支持数据库无关性，sql语句优化困难

3.开发难易程度

MyBatis是轻量级框架，适用于需求复杂，大型项目，学习成本低

Hibernate是重量级框架，适用于需求简单，中小型项目，学习成本高

JDBC的不足之处，MyBatis如何解决

1.jdbc频繁创建释放连接，造成资源浪费

Mybatis配置数据库连接池，管理连接

2.jdbc sql语句写在代码中不易维护

Mybatis sql语句写在xml配置文件中

3.jdbc 向sql语句传参麻烦，占位符需要和参数一一对应

Mybatis自动将java对象映射到sql语句

4.jdbc对结果集解析麻烦

Mybatis自动将sql语句的执行结果映射到java对象

MyBatis的编程步骤

创建sqlSessionFactory

通过sqlSessionFactory创建sqlSession

通过sqlSession执行数据库操作

通过sqlSession.commit提交事务

通过sqlSession.close关闭会话

MyBatis常见标签

select insert update delete resultMap parameterMap sql include selecteKey 和动态标签

#{}和${}的区别

#{}是参数占位符，作用相当于变量值替换，${}是字符串替换符，作用相当于字符串拼接

#{}使用预编译的方式，可以有效防止sql注入，${}是字符串拼接的方式，可能导致sql注入。

参数是字符串时${}需要手动添加引号，#{}不需要

与xml文件对应的dao接口的工作原理

da接口即mapper接口，接口的全类名就是mapper文件中的namespace值，接口的方法名就是mapper文件中的statement的id值，mapper接口是没有实现类的，当调用接口方法时，接口的全类名加方法名拼接成的字符串作为key，可以唯一确定一个mapper statement

dao接口的工作原理是jdk动态代理，MyBatis运行时使用JDK动态代理为DAO接口生成代理对象，代理对象拦截接口方法，转而执行mapper statement对应的sql，然后将sql执行结果返回

dao方法可以重载，在mapper中使用动态sql

在Mapper中如何传递多个参数

1.若DAO层有多个参数，那么对应的xml文件中，#{0}代表接收的是第一个参数，#{1}代表接收的是第二个参数

arg0/param1

2.使用@Param注解，注解中的参数名为传递到mapper中的参数名

3.多个参数封装成map，以hashmap的形式传递到mapper中。直接去map的key

4.多个参数封装成对象，直接取属性名

MyBatis如何将sql执行结果封装为目标对象并返回

resultMap标签

sql别名

Mybatis动态sql原理

根据sql参数对象计算表达式的值完成逻辑判断，并动态拼接sql

trim where set foreach if choose when otherwise bind

xml映射文件中，不同的xml映射文件的id是否可以重复

如果配置了namespace，可以重复，否则不可以重复

MyBatis实现一对多有几种方式

collection

```xml
<mapper namespace="com.wdlm.mybatis.mapper.DepartmentMapper">
    <resultMap id="departmentMap" type="com.wdlm.mybatis.pojo.Department">
        <id column="did" property="did"></id>
        <result column="dname" property="dname"></result>
        <collection property="employeeList" ofType="com.wdlm.mybatis.pojo.Employee">
            <id column="eid" property="eid"></id>
            <result column="ename" property="ename"></result>
        </collection>
    </resultMap>
    <select id="getDepartmentAndEmployeeByDid" resultMap="departmentMap">
        select t_dept.*,t_emp.* from t_dept left join t_emp on t_dept.did = t_emp.did where t_dept.did = #{did}
    </select>
</mapper>
```

分步查询

```xml
<mapper namespace="com.wdlm.mybatis.mapper.DepartmentMapper">
    <resultMap id="departmentMap" type="com.wdlm.mybatis.pojo.Department">
        <id column="did" property="did"></id>
        <result column="dname" property="dname"></result>
        <collection property="employeeList" select="com.wdlm.mybatis.mapper.EmployeeMapper.getEmployeeByEid" column="did"></collection>
    </resultMap>
    <select id="getDepartmentAndEmployeeByDid" resultMap="departmentMap">
        select *from t_dept where t_dept.did = #{did}
    </select>
</mapper>
```

```xml
<mapper namespace="com.wdlm.mybatis.mapper.EmployeeMapper">
    <select id="getEmployeeByEid" resultType="com.wdlm.mybatis.pojo.Employee">
        select * from t_emp where eid = #{eid}
    </select>
</mapper>
```

MyBatis实现多对一有几种方式

级联方式

```xml
<mapper namespace="com.wdlm.mybatis.mapper.EmployeeMapper">
    <resultMap id="employeeMap" type="com.wdlm.mybatis.pojo.Employee">
        <id column="eid" property="eid"></id>
        <result column="ename" property="ename"></result>
        <result column="did" property="dept.did"></result>
        <result column="dname" property="dept.dname"></result>
    </resultMap>
    <select id="getEmployeeAndDepartmentByEid" resultMap="employeeMap">
        select t_emp.*,t_dept.* from t_emp left join t_dept on t_emp.did = t_dept.did where t_emp.eid=#{eid}
    </select>
</mapper>
```

association

```xml
<mapper namespace="com.wdlm.mybatis.mapper.EmployeeMapper">
    <resultMap id="employeeMap" type="com.wdlm.mybatis.pojo.Employee">
        <id column="eid" property="eid"></id>
        <result column="ename" property="ename"></result>
        <association property="dept" javaType="com.wdlm.mybatis.pojo.Department">
            <id column="did" property="did"></id>
            <result column="dname" property="dname"></result>
        </association>
    </resultMap>
    <select id="getEmployeeAndDepartmentByEid" resultMap="employeeMap">
        select t_emp.*,t_dept.* from t_emp left join t_dept on t_emp.did = t_dept.did where t_emp.eid=#{eid}
    </select>
</mapper>
```

分步查询

```xml
<mapper namespace="com.wdlm.mybatis.mapper.EmployeeMapper">
    <resultMap id="employeeMap" type="com.wdlm.mybatis.pojo.Employee">
        <id column="eid" property="eid"></id>
        <result column="ename" property="ename"></result>
        <association property="dept" select="com.wdlm.mybatis.mapper.DepartmentMapper.getDepartmentByDid" column="did"></association>
    </resultMap>
    <select id="getEmployeeAndDepartmentByEid" resultMap="employeeMap">
        select * from t_emp where eid=#{eid}
    </select>
</mapper>
```

```xml
<mapper namespace="com.wdlm.mybatis.mapper.DepartmentMapper">
    <select id="getDepartmentByDid" resultType="com.wdlm.mybatis.pojo.Department">
        select * from t_dept where did = #{did}
    </select>
</mapper>
```

分步查询可以实现延迟加载

MyBatis的一二级缓存

一级缓存是sqlSession级别的，同一个sqlSession级别的查询数据会被缓存

一级缓存失效

不同的sqlSession

同一sqlSession但是查询条件不同

两次查询期间执行了增删改操作

两次查询期间手动清空缓存

一级缓存是sqlSessionFactry级别的，同一个sqlSessionFactory创建的sqlSession查询数据会被缓存

二级缓存需要手动开启，查询数据转化成的实体类类型必须实现序列化接口

二级缓存失效

两次查询之间的增删改操作

先查询二级缓存，未命中查询一级缓存，sqlSession关闭后，一级缓存写入二级缓存

Mapper接口的调用要求

mapper接口的类路径是mapper.xml的namespace

mapper接口的方法名是mapper.xml的id

mapper接口的输入参数和mapper.xml中sql的parameterType想提供

mapper接口的返回值和mapper.xml中的sql的resultType相同

MyBatis如何实现分页

MyBatis使用RowBounds对象进行分页，针对结果集进行内存分页，而非物理分页

可以直接在sql内使用limit进行物理分页，也可以使用插件进行物理分页

使用插件进行物理分页拦截sql进行重写

MyBatis插件的运行原理

MyBatis 仅可以编写针对 ParameterHandler 、 ResultSetHandler 、 StatementHandler 、 Executor 这 4 种接口的插件，MyBatis 使用 JDK 的动态代理，为需要拦截的接口生成代理对象以 实现接口方法拦截功能，每当执行这 4 种接口对象的方法时，就会进入拦截方法，具体就是 InvocationHandler 的 invoke() 方法，当然，只会拦截那些你指定需要拦截的方法。 实现 MyBatis 的 Interceptor 接口并复写 intercept() 方法，然后在给插件编写注解，指定要拦截哪一个接口的哪些方法即可，记住，别忘了在配置文件中配置你编写的插件。

MyBatis延迟加载的原理

配置文件中手动开启延迟加载，仅支持association和collection分步查询的延迟加载

并不预先装配A对象中的B对象，使用CGLIb创建目标对象的代理对象，调用目标方法，进入拦截器，进行关联查询，然后装配。

MyBatis Executor执行器

SimpleExecutor:每次查询或者更新，开启Statement对象，用完立刻关闭

ReuseExecutor：以sql为key，statement为值，存在map中，复用

BatchExecutor：执行更新，所有的sql都添加到批处理中，统一执行。