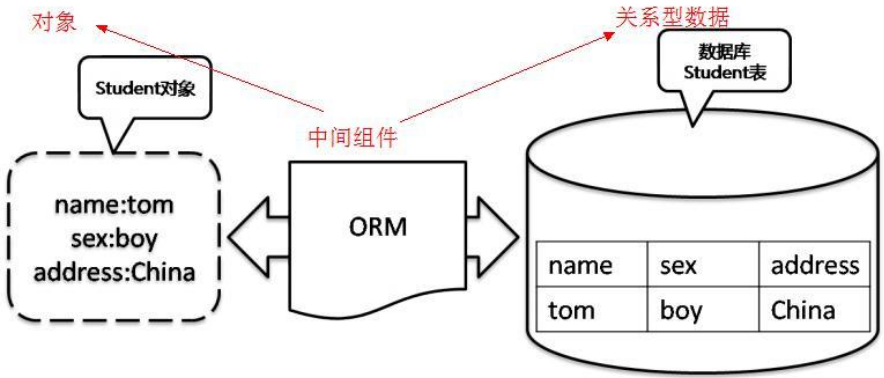
MybatisPlus快速上手

教学内容

- 第一节 ORM介绍
- 第二节 MyBatis-Plus介绍
- 第三节 MyBatis-Plus CRUD操作

ORM介绍

- ORM（Object Relational Mapping，对象关系映射）是为了解决面向对象与关系数据库存在的互不匹配现象的一种技术。
- ORM通过使用描述对象和数据库之间映射的元数据将程序中的对象自动持久化到关系数据库中。
- ORM框架的本质是简化编程中操作数据库的编码。



MyBatis-Plus介绍

- MyBatis是一款优秀的数据持久层ORM框架，被广泛地应用于应用系统。
- MyBatis能够非常灵活地实现动态SQL，可以使用XML或注解来配置和映射原生信息，能够轻松地将Java的POJO（Plain Ordinary Java Object，普通的Java对象）与数据库中的表和字段进行映射关联。
- MyBatis-Plus是一个 MyBatis 的增强工具，在 MyBatis 的基础上做了增强，简化了开发。

添加依赖

```
<!--  MyBatisPlus依赖  -->
<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-boot-starter</artifactId>
    <version>3.4.2</version>
</dependency>
<!-- mysql驱动依赖  -->
<dependency>
     <groupId>mysql</groupId>
     <artifactId>mysql-connector-java</artifactId>
     <version>5.1.47</version>
</dependency>
<!--    数据连接池 druid    -->
<dependency>
     <groupId>com.alibaba</groupId>
     <artifactId>druid-spring-boot-starter</artifactId>
     <version>1.1.20</version>
</dependency>
```

全局配置

- 配置数据库相关信息。

```
spring.datasource.type=com.alibaba.druid.pool.DruidDataSource
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/mydb?useSSL=false
spring.datasource.username=root
spring.datasource.password=root
mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl
```

- 添加@MapperScan注解

```
@SpringBootApplication
@MapperScan("com.xx.mapper")
public class MybatisplusDemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(MybatisplusDemoApplication.class, args);
    }
}
```

Mybatis CRUD注解

注解	功能
@Insert	实现插入
@Update	实现更新
@Delete	实现删除
@Select	实现查询
@Result	实现结果集封装
@Results	可以与@Result 一起使用，封装多个结果集
@One	实现一对一结果集封装
@Many	实现一对多结果集封装

CRUD操作

```java
@Mapper
public interface UserMapper {
    @Insert("insert into user values(#{id},#{username},#{password},#{birthday})")
    int add(User user);

    @Update("update user set username=#{username},password=#{password},birthday=#{birthday} where id=#{id}")
    int update(User user);

    @Delete("delete from user where id=#{id}")
    int delete(int id);

    @Select("select * from user where id=#{id}")
    User findById(int id);

    @Select("select * from user")
    List<User> getAll();
}
```

MybatisPlus注解

- @TableName，当表名与实体类名称不一致时，可以使用@TableName注解进行关联。
- @TableField，当表中字段名称与实体类属性不一致时，使用@TableField进行关联
- @TableId，用于标记表中的主键字段，MybatisPlus也提供了主键生成策略。

值	描述
AUTO	数据库 ID 自增
NONE	无状态，该类型为未设置主键类型（注解里等于跟随全局，全局里约等于 INPUT）
INPUT	insert 前自行 set 主键值
ASSIGN_ID	分配 ID(主键类型为 Number(Long 和 Integer)或 String)(since 3.3.0),使用接口 `IdentifierGenerator` 的方法 `nextId`(默认实现类为 `DefaultIdentifierGenerator` 雪花算法)
ASSIGN_UUID	分配 UUID,主键类型为 String(since 3.3.0),使用接口 `IdentifierGenerator` 的方法 `nextUUID`(默认 default 方法)