




















mybatis的学习

第一步（创建数据库的表）

-  t_clazz
-  t_course
-  t_course_stuc
-  t_forum
-  t_post
-  t_rank
-  t_student
-  t_teacher
-  t_topic

第二步（根据数据库创建实体类）

-  Clazz
-  ClazzVo
-  Course
-  CourseStudent
-  Forum
-  MatterDto
-  MatterVo
-  Music
-  Student
-  Teacher

第三步（根据所需要的功能在mapper层写出方法）

```

@Transactional(rollbackFor = Exception.class)
public interface StudentMapper {

    /**
     * 新增student，并返回自增主键
     * @param student
     */
    void insert(Student student);

    /**
     * 根据studentId删除Student
     * @param studentId
     */
    void delete(int studentId);

    /**
     * 更新学生
     * @param student
     */
    void update(Student student);
}

```

第四步（在xml文件里写出数据库方法）

- 单表增删改查
- 单表批量增删改
- 单表指定条件查询（结合动态SQL）
- 多表联查（返回ResultType）
- 多表联查（返回ResultMap）
- 多表联查（结合动态SQL）
- 一对一
- 一对多
- 多对一
- 多对多

```

<!-- 批量增加学生-->
<insert id="batchInsert" parameterType="com.soft1851.spring.mybatis.entity.Student">
    INSERT INTO t_student VALUES
    <foreach collection='students' item='item' index='index' separator=','>
        ({item.studentId},{item.clazzId},{item.studentName},{item.hometown},{item.birthday})
    </foreach>
</insert>

```

```

<!-- 批量删除-->
<delete id="batchDelete" parameterType="int">
    DELETE FROM t_student
    WHERE student_id IN
    <foreach collection="idList" item="item" index="index" open="(" separator="," close=")">
        #{item}
    </foreach>
</delete>

```

```

<!-- 批量修改-->
<update id="batchUpdate" parameterType="java.util.List">
    <foreach collection="students" item="item" open="" close="" separator=";">
        UPDATE t_student t
        <set>
            t.student_name = #{item.studentName}
        </set>
        WHERE t.student_id=${item.studentId}
    </foreach>
</update>

```

```

<!-- 根据学生的id查询学生-->
<select id="getStudentById" parameterType="int" resultType="Student">
    SELECT *
    FROM t_student
    WHERE student_id = ${studentId}
</select>

```

```

<!-- 多表指定条件查询（结合动态SQL）-->
<select id="selectLimitFromTeacherInClassByDynamicSql"
    parameterType="MatterDto" resultType="MatterVo">
    SELECT t.teacher_name,c.clazz_name,s.student_name,s.hometown,s.birthday
    FROM t_teacher t,t_clazz c,t_student s
    WHERE t.clazz_id = c.clazz_id AND c.clazz_id = s.clazz_id
    <if test="teacherName != null and clazzName != null and studentName != null">
        AND teacher_name like "%#{teacherName}%"
        AND clazz_name like "%#{clazzName}%"
        AND student_name like "%#{studentName}%"
    </if>
</select>

```

第四步（测试类实现以下）

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = {"spring-mybatis.xml"})
public class StudentMapperTest {
    @Resource
    private StudentMapper studentMapper;

    @Test
    public void insert() {
        Student student = Student.builder()
            .clazzId(1)
            .studentName("席光平")
            .hometown("安徽阜阳")
            .birthday(LocalDate.of(year: 1999, month: 9, dayOfMonth: 28))
            .build();
        studentMapper.insert(student);
        System.out.println(student.getStudentId());
    }
}
```

补充：数据库连接，以及包的配置以及类型别名的配置也要配置好

