### JAVA第三阶段—DAY09-JAVA案例

1. MyBatis基本使用

通过案例了解MyBatis的基本配置和使用方法

1）导入依赖

<dependency>  
  <groupId>org.mybatis</groupId>  
  <artifactId>mybatis</artifactId>  
  <version>3.5.5</version>  
</dependency>

2）添加配置文件mybatis-config.xml

<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE configuration  
      PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  
      "http://mybatis.org/dtd/mybatis-3-config.dtd">  
<configuration>  
  <!--多种环境：开发、测试、生产-->  
  <environments default="development">  
      <environment id="development">  
          <!--事务管理-->  
          <transactionManager type="JDBC"/>  
          <!--数据源 POOLED代表连接池-->  
          <dataSource type="POOLED">  
              <!--JDBC的配置-->  
              <property name="driver" value="com.mysql.jdbc.Driver"/>  
              <property name="url" value="jdbc:mysql://localhost:3306/bookms\_db?useUnicode=true&amp;characterEncoding=UTF-8&amp;serverTimezone=UTC&amp;useSSL=false"/>  
              <property name="username" value="root"/>  
              <property name="password" value="123456"/>  
          </dataSource>  
      </environment>  
  </environments>  
  <!--配置映射文件的位置-->  
  <mappers>  
      <mapper resource="mappers/\*.xml"></mapper>  
  </mappers>  
</configuration>

3）定义mapper接口

只定义一个insert方法

/\*\*  
\* 书籍表的映射接口  
\*/  
public interface BookMapper {  
​  
  void insert(Book book);  
}

4） 映射文件

在resources的mappers目录下添加映射文件

<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE mapper  
      PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
      "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  
<!--命名空间指定接口类型-->  
<mapper namespace="com.blb.mybatis\_demo.dao.BookMapper">  
  <!--insert配置插入操作 id是接口中方法的名称 parameterType是参数类型-->  
  <insert id="insert" parameterType="com.blb.mybatis\_demo.entity.Book">  
      insert into tb\_book(book\_name, price, type\_id, author, publish\_org, publish\_time, state,book\_image)  
      values(#{bookName},#{price},#{typeId},#{author},#{publishOrg},#{publishTime},#{state},#{bookImage});  
  </insert>  
</mapper>

5) 单元测试

public class TestBookMapper {  
​  
  @Test  
  public void testInsert(){  
      //创建SQLSessionFactoryBuilder对象，用于创建SQLSessionFactory  
      SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();  
      //创建SqlSessionFactory  
      try {  
          SqlSessionFactory factory = builder.build(Resources.getResourceAsStream("mybatis-config.xml"));  
          //SqlSessionFactory用于创建SqlSession会话  
          try(SqlSession session = factory.openSession()){  
              //通过session获得Mapper对象  
              BookMapper mapper = session.getMapper(BookMapper.class);  
              //执行操作  
              mapper.insert(new Book(0L,"test-11",88,1,"xx","xx","2011-1-1",0,"xx.jpg"));  
              //提交修改  
              session.commit();  
          }  
      } catch (IOException e) {  
          e.printStackTrace();  
      }  
  }  
}

1. 查询案例

通过案例了解基本的查询配置，了解resultType和resultMap的作用

1） 给接口加方法

  List<Book> findAll();  
​  
  Book findById(int id);

2） 返回值的配置

需要给select标签配置返回值的类型resultType

  <!--select配置查询方法，resultType配置返回类型-->  
  <select id="findAll" resultType="com.blb.mybatis\_demo.entity.Book">  
      select \* from tb\_book  
  </select>  
​  
  <select id="findById" resultType="com.blb.mybatis\_demo.entity.Book">  
      select \* from tb\_book where id = #{id}  
  </select>

3） 测试

  @Test  
  public void testSelectAll(){  
      //创建SQLSessionFactoryBuilder对象，用于创建SQLSessionFactory  
      SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();  
      //创建SqlSessionFactory  
      try {  
          SqlSessionFactory factory = builder.build(Resources.getResourceAsStream("mybatis-config.xml"));  
          //SqlSessionFactory用于创建SqlSession会话  
          try(SqlSession session = factory.openSession()){  
              //通过session获得Mapper对象  
              BookMapper mapper = session.getMapper(BookMapper.class);  
              //执行操作  
              List<Book> books = mapper.findAll();  
              books.forEach(book -> System.out.println(book));  
          }  
      } catch (IOException e) {  
          e.printStackTrace();  
      }  
  }

两种解决方法：

1） 添加resultMap，实现数据库字段和属性的映射，把resultType改为resultMap

  <!--实现Book到tb\_book映射-->  
  <resultMap id="bookMap" type="com.blb.mybatis\_demo.entity.Book">  
      <!--配置主键 property是java属性名 column是表字段名-->  
      <id property="id" column="id"></id>  
      <!--普通字段-->  
      <result property="bookName" column="book\_name"></result>  
      <result property="price" column="price"></result>  
      <result property="typeId" column="type\_id"></result>  
      <result property="author" column="author"></result>  
      <result property="publishOrg" column="publish\_org"></result>  
      <result property="publishTime" column="publish\_time"></result>  
      <result property="state" column="state"></result>  
      <result property="bookImage" column="book\_image"></result>  
  </resultMap>

2） 在mybatis配置文件里，配置下划线转换为驼峰命名风格

  <settings>  
      <setting name="mapUnderscoreToCamelCase" value="true"/>  
  </settings>