集成Mybatis

创建数据库

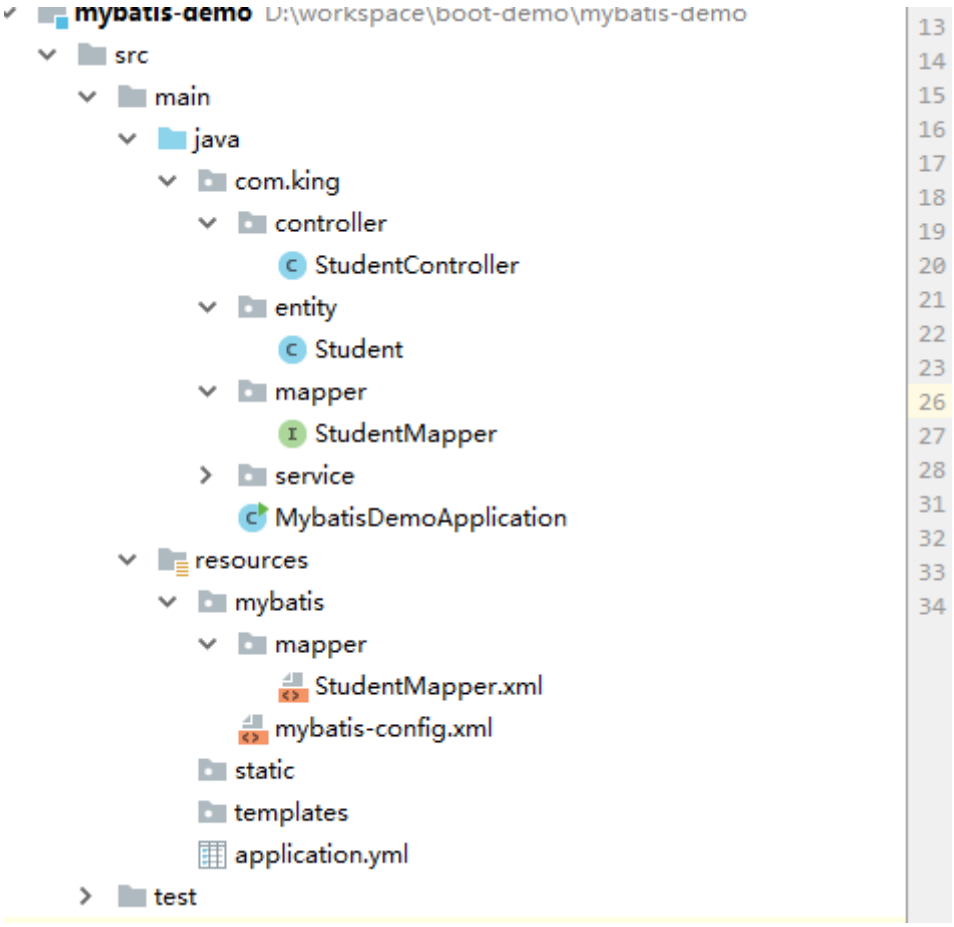
```
-- ----------------------------
-- Table structure for tb_user
-- ----------------------------
DROP TABLE IF EXISTS `tb_user`;
CREATE TABLE `student` (
  `id` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `age` int(11),
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------
-- Records of tb_user
-- ----------------------------
INSERT INTO `student` VALUES ('1', 'laowang', 22);
INSERT INTO `student` VALUES ('2', 'laoli', 22);
```

引入依赖

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<!--springboot整合mybatis的依赖-->
<dependency>
    <groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-starter</artifactId>
    <version>1.3.1</version>
</dependency>
```

创建文件



YML文件配置

```yaml
server:
  port: 8088

spring:
  http:
    encoding.charset: UTF-8
    encoding.force: true
    encoding.enabled: true
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://127.0.0.1:3306/test?
serverTimezone=CTT&characterEncoding=utf8&useUnicode=true&useSSL=false
    username: root
    password: 123456

mybatis:
  #指定实体类地址
  typeAliasesPackage: com.king.entity
   #指定mysql-config配置文件
  config-localtion: classpath:mybatis/mybatis-config.xml
  #指定SqlMap映射文件位置
```

```
  mapper-localtions: classpath:mybatis/mapper/*.xml
```

编写实体类

```
package com.king.entity;

public class Student {
    /**
     * 唯一标识id
     */
    private String id;
    /**
     * 姓名
     */
    private String name;
    /**
     * 年龄
     */
    private Integer age;


    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }
```

```
}
```

编写接口

```
package com.king.mapper;

import com.king.entity.Student;
import org.apache.ibatis.annotations.Mapper;


@Mapper
public interface StudentMapper {

//    List<Student> findAll();

    Student getStudentById(String id);
}
```

mybatis-cofnig.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>
    <properties>
        <property name="dialect" value="mysql" />
    </properties>
    <settings>
        <!-- 这个配置使全局的映射器启用或禁用缓存。系统默认值是true，设置只是为了展示出来 -->
        <setting name="cacheEnabled" value="true" />
        <!-- 全局启用或禁用延迟加载。当禁用时，所有关联对象都会即时加载。系统默认值是true，设置只是为了展示出来 -->
        <setting name="lazyLoadingEnabled" value="true" />
        <!-- 允许或不允许多种结果集从一个单独的语句中返回（需要适合的驱动）。 系统默认值是true，设置只是为了展示出来 -->
        <setting name="multipleResultSetsEnabled" value="true" />
        <!--使用列标签代替列名。不同的驱动在这方便表现不同。参考驱动文档或充分测试两种方法来决定所使用的驱动。 系统默认值是true，设置只是为了展示出来 -->
```

```
        <setting name="useColumnLabel" value="true" />
        <!--允许 JDBC 支持生成的键。需要适合的驱动。如果设置为 true 则这个设置强制生成的键被使用，尽管一些驱动拒绝兼容但仍然有效（比如
            Derby）。 系统默认值是false，设置只是为了展示出来 -->
        <setting name="useGeneratedKeys" value="false" />
        <!--配置默认的执行器。SIMPLE 执行器没有什么特别之处。REUSE 执行器重用预处理语句。BATCH 执行器重用语句和批量更新 系统默认值是SIMPLE，设置只是为了展示出来 -->
        <setting name="defaultExecutorType" value="SIMPLE" />
        <!--设置超时时间，它决定驱动等待一个数据库响应的时间。 系统默认值是null，设置只是为了展示出来 -->
        <setting name="defaultStatementTimeout" value="25000" />
    </settings>
</configuration>
```

StudentMapper.xml

```
<?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" >

<mapper namespace="com.king.mapper.StudentMapper" >
    <!--<resultMap id="BaseResultMap" type="com.king.entity.Student" >-->
        <!--<id column="id" property="id" jdbcType="VARCHAR" />-->
        <!--<result column="name" property="name" jdbcType="VARCHAR" />-->
        <!--<result column="age" property="age" jdbcType="BIGINT" />-->
    <!--</resultMap>-->

    <!--<sql id="Base_Column_List" >-->
        <!--id, name, age-->
    <!--</sql>-->

    <!--<select id="findAll" resultType="com.king.entity.Student" >-->
        <!--SELECT-->
        <!--*-->
        <!--FROM student-->
    <!--</select>-->

    <select id="getStudentById" resultType="com.king.entity.Student" >
        SELECT
        *
        FROM student where id = #{id}
    </select>
```

```
    <!--<select id="getOne" parameterType="java.lang.Long"
resultMap="BaseResultMap">-->
        <!--SELECT-->
        <!--<include refid="Base_Column_List" />-->
        <!--FROM student-->
        <!--WHERE id = #{id}-->
    <!--</select>-->

    <!--<insert id="insert" parameterType="com.king.entity.Student" >-->
        <!--INSERT INTO-->
        <!--student-->
        <!--(id, name, age)-->
        <!--VALUES-->
        <!--(#{id}, #{name}, #{age})-->
    <!--</insert>-->

    <!--<update id="update" parameterType="com.king.entity.Student" >-->
        <!--UPDATE-->
        <!--student-->
        <!--SET-->
        <!--<if test="name != null">userName = #{userName},</if>-->
        <!--passWord = #{passWord},-->
        <!--nick_name = #{nickName}-->
        <!--WHERE-->
        <!--id = #{id}-->
    <!--</update>-->

    <!--<delete id="delete" parameterType="java.lang.String" >-->
        <!--DELETE FROM-->
        <!--student-->
        <!--WHERE-->
        <!--id =#{id}-->
    <!--</delete>-->
</mapper>
```

注意：

1.namespace中需要与使用@Mapper的接口对应

2.UserMapper.xml文件名称必须与使用@Mapper的接口一致

3.标签中的id必须与@Mapper的接口中的方法名一致，且参数一致

编写接口

```
package com.king.service;

import com.king.entity.Student;
import org.apache.ibatis.annotations.Param;

import java.util.List;


public interface StudentService {

    /**
     * 通过ID查询单条数据
     *
     * @param id 主键
     * @return 实例对象
     */
    Student queryById(String id);

    /**
     * 查询指定行数据
     *
     * @param offset 查询起始位置
     * @param limit 查询条数
     * @return 对象列表
     */
    List<Student> queryAllByLimit(@Param("offset") int offset,
@Param("limit") int limit);


    /**
     * 通过实体作为筛选条件查询
     *
     * @return 对象列表
     */
    List<Student> queryAll();

    /**
     * 新增数据
     *
     * @param student 实例对象
     * @return 影响行数
     */
    int insert(Student student);
```

```
    /**
     * 修改数据
     *
     * @param student 实例对象
     * @return 影响行数
     */
    int update(Student student);

    /**
     * 通过主键删除数据
     *
     * @param id 主键
     * @return 影响行数
     */
    int deleteById(Integer id);
}
```

```
package com.king.service.impl;

import com.king.service.StudentService;
import com.king.entity.Student;
import com.king.mapper.StudentMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class StudentServiceImpl implements StudentService {

    @Autowired
    StudentMapper studentMapper;

    @Override
    public Student queryById(String id) {
        return studentMapper.getStudentById(id);
    }

    @Override
    public List<Student> queryAllByLimit(int offset, int limit) {
        return null;
    }
```

```
    @Override
    public List<Student> queryAll() {
        return null;
    }

    @Override
    public int insert(Student student) {
        return 0;
    }

    @Override
    public int update(Student student) {
        return 0;
    }

    @Override
    public int deleteById(Integer id) {
        return 0;
    }
}
```

编写api接口

```
package com.king.controller;


import com.king.service.impl.StudentServiceImpl;
import com.king.entity.Student;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import java.util.List;

@Controller
@ResponseBody
@RequestMapping("/student")
public class StudentController {

    @Autowired
```

```
    StudentServiceImpl studentServiceimpl;

    @RequestMapping("/getAll")
    public List<Student> getStudents() {
        return studentServiceimpl.queryAll();
    }

    @RequestMapping("/get/{id}")
    public Student getStudent(@PathVariable String id) {
        return studentServiceimpl.queryById(id);
    }


}
```

在启动类中添加对@MapperScan的扫描

```
package com.king;

import org.mybatis.spring.annotation.MapperScan;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
@MapperScan("com.king.mapper")
public class MybatisDemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(MybatisDemoApplication.class, args);
    }

}
```