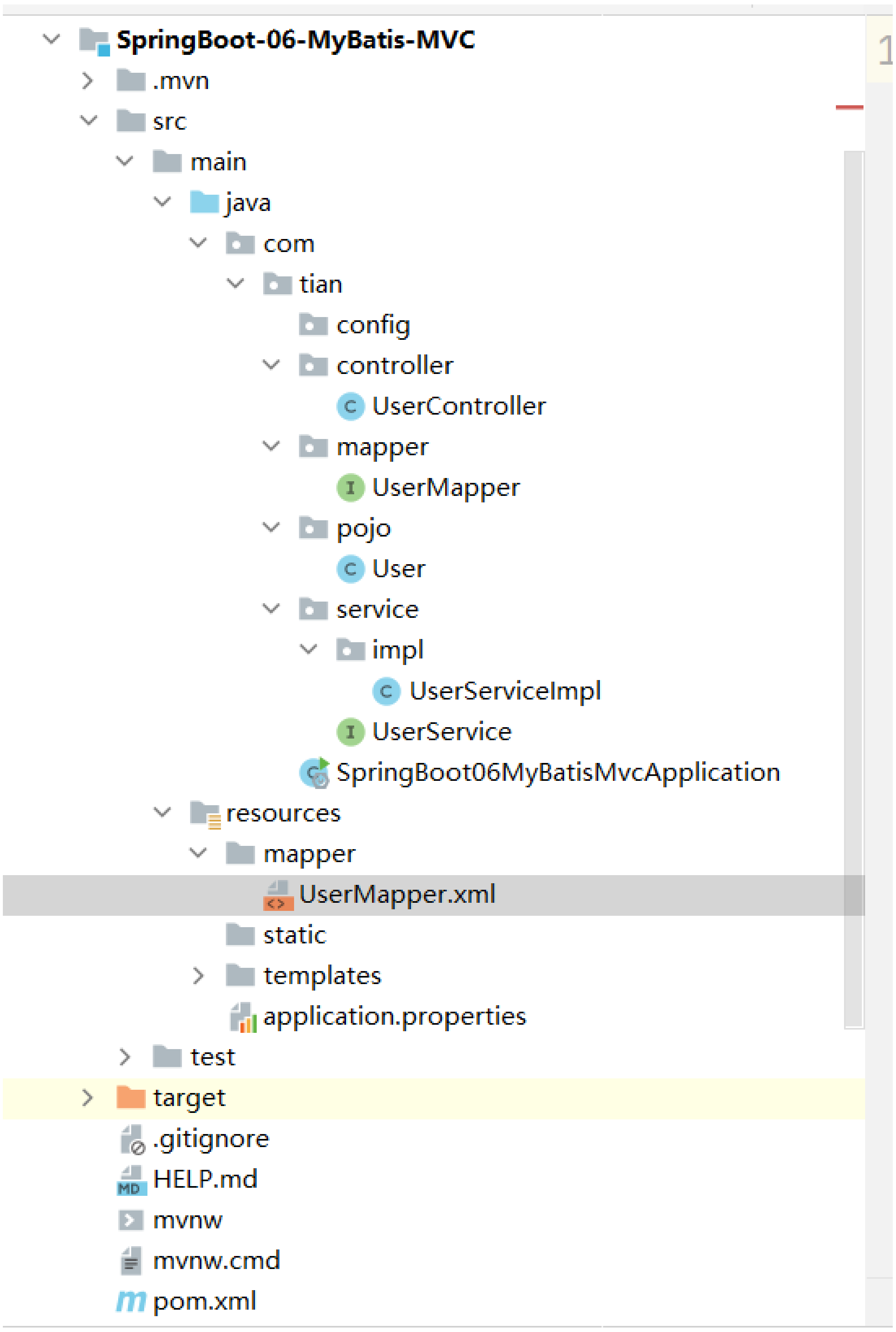
简单的ssm框架学习（Spring+SpringBoot+MyBatis）

项目结构如下



```
SpringBoot-06-MyBatis-MVC
  .mvn
  src
    main
      java
        com
          tian
            config
            controller
              UserController
            mapper
              UserMapper
            pojo
              User
            service
              impl
                UserServiceImpl
              UserService
            SpringBoot06MyBatisMvcApplication
      resources
        mapper
          UserMapper.xml
        static
        templates
        application.properties
    test
  target
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```

第一步：建立数据库

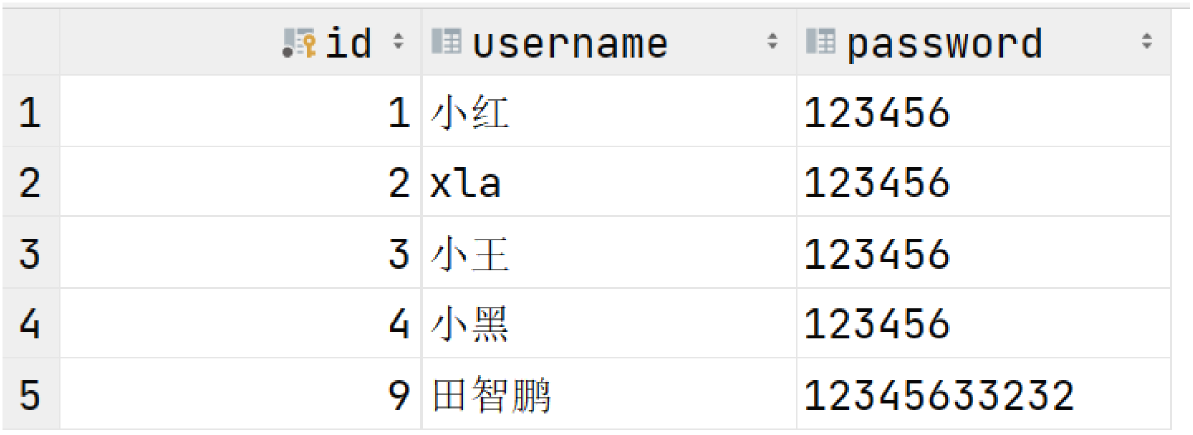
(1) 数据库代码（mysql）：

```
create table my_learn.user
(
    id       bigint auto_increment
        primary key,
    username varchar(255) null,
    password varchar(255) null
);
```

(2) 插入数据:

```
insert into user values (1,'小红','123456');
insert into user values (2,'xla','123456');
insert into user values (3,'小王','123456');
insert into user values (4,'小黑','123456');
insert into user values (9,'田智鹏','12345633232');
```

数据库建立完成！（一下为数据库的数据）



	id	username	password
1	1	小红	123456
2	2	xla	123456
3	3	小王	123456
4	4	小黑	123456
5	9	田智鹏	12345633232

第二步：创建一个简单的SpringBoot项目

（1）以下是所用到的依赖:

```
 <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-jdbc</artifactId>
        </dependency>
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.4.5</version>
        </dependency>
        <dependency>
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-starter</artifactId>
            <version>2.0.0</version>
        </dependency>

        <!--增加thymeleaf支持,本次项目为restful风格，所以次依赖可有可无! -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>

    </dependencies>
```

(2）配置application.propertis文件:

```
#项目端口和项目名配置
server.port=8081
server.servlet.context-path=/ems

#spring.datasource.type=com.alibaba.druid.pool.DruidDataSource
#数据库配置
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/my_learn?
useUnicode=true&characterEncoding=UTF-8&useSSL=true&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=123456

#mybatis的文件配置
mybatis.mapper-locations=classpath:mapper/*.xml
mybatis.type-aliases-package=com.tian.pojo
mybatis.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl
#静态资源的配置
spring.resources.static-locations=classpath:/templates/,classpath:/static/
spring.thymeleaf.cache=false
```

(3）开发实体类（对应数据库的字段）：

```
package com.tian.pojo;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
```

```
@NoArgsConstructor
@AllArgsConstructor
public class User {
    private int id;
    private String username;
    private String password;
}
```

(4) 开发dao层（数据持久层，主要实现对数据的增删改查操作！）：

```
package com.tian.mapper;

import com.tian.pojo.User;
import org.apache.ibatis.annotations.*;
import org.springframework.stereotype.Component;

import java.util.List;
@Component
@Mapper
public interface UserMapper {

    //分页查询全部用户
    //注解开发不用写Mapper文件！@Select("select * from user")
    List<User> listAll();

    //根据id查询
    //注解开发不用写Mapper文件！@Select("select * from user where id = #{id}")
    User selectById(@Param("id") int id);

    //添加用户
    //注解开发不用写Mapper文件！@Insert("insert into user value (#{id},#{username},#
{password})")
    int add(User user);

    //根据id删除用户
    //注解开发不用写Mapper文件！@Delete("delete from user where id = #{id}")
    int delete(@Param("id") int id);

    //更新用户信息
    //注解开发不用写Mapper文件！@Update("update user set username=#
{username},password=#{password} where id = #{id}")
    int updata(User user);
}
```

（5）开发Mapper文件（MyBatis框架）：

```
<?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.tian.mapper.UserMapper">

    <select id="list" resultMap="UserMapper">
        select * from user
    </select>

    <resultMap id="UserMapper" type="User">
        <id column="id" property="id"></id>
        <result column="username" property="username"></result>
        <result column="password" property="password"></result>
    </resultMap>

    <select id="selectById" resultType="User">
        select * from user where id = #{id}
    </select>

   <insert id="insert" parameterType="int">
        insert into user value (#{id},#{username},#{password})
    </insert>

    <delete id="deleteById" parameterType="int">
        delete from user where id = #{id}
    </delete>

    <update id="update" parameterType="User">
        update user set username=#{username},password=#{password} where id=#{id}
    </update>
</mapper>
```

（6）开发Service层（调用Dao层）：

```
package com.tian.service;

import com.tian.pojo.User;
import com.tian.pojo.Usertest;

import java.util.List;

public interface UserService {

    List<User> list();

    User selectById(int id);

    int insert(User user);

    int add(Usertest usertest);

    int update(Usertest usertest);

    int delete(int id);
}
```

(7) 开发ServiceImpl实现类:

```
package com.tian.service.impl;

import com.tian.mapper.UserMapper;
import com.tian.pojo.User;
import com.tian.pojo.Usertest;
import com.tian.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class UserServiceImpl implements UserService {

    @Autowired
    UserMapper userMapper;

    @Override
    public List<User> list() {
        return userMapper.list();
    }

    @Override
    public User selectById(int id) {
        return userMapper.selectById(id);
    }

    @Override
    public int insert(User user) {
        return userMapper.insert(user);
    }


    @Override
    public int update(User user) {
        return userMapper.update(user);
    }

    @Override
    public int delete(int id) {
        return userMapper.delete(id);
    }
}
```

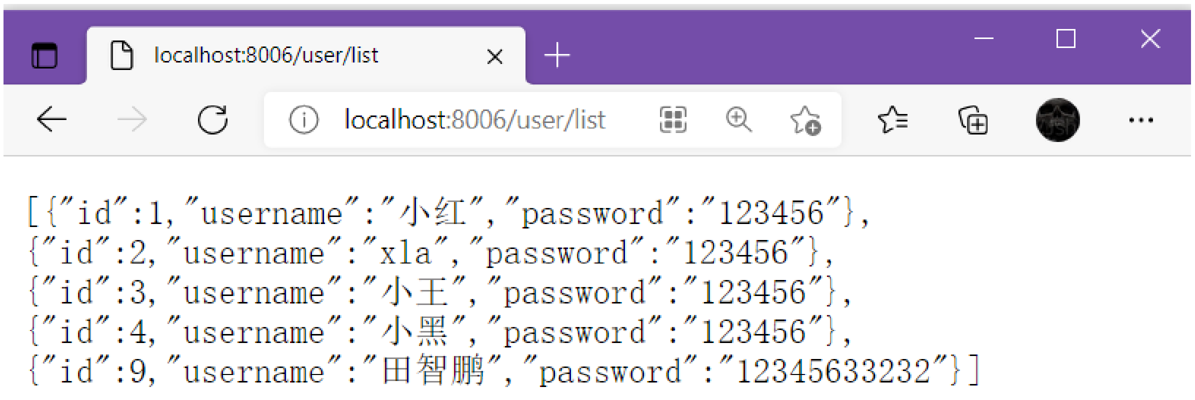
(8) 开发controller层（restful风格！）：

```
package com.tian.controller;

import com.tian.mapper.UserMapper;
import com.tian.pojo.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequestMapping("/user")
public class UserController {
    @Autowired
    private UserMapper userMapper;

    @RequestMapping("/list")
    public List<User> list(){
        return userMapper.listAll();
    }

    @RequestMapping("/selectById")
    public User selectbyid(int id){
        return userMapper.selectById(id);
    }

    @RequestMapping("/add")
    String add(User user) {
        return userMapper.add(user) == 1 ? "success" : "failed";
    }

    @RequestMapping("/updatebyid")
    String updateById(User user) {
        return userMapper.updata(user) == 1 ? "success" : "failed";
    }

    @RequestMapping("/delbyid")
    String delById(int id) {
        return userMapper.delete(id) == 1 ? "success" : "failed";
    }

}
```

(9) 简单的ssm项目开发完成！（打完手工！）

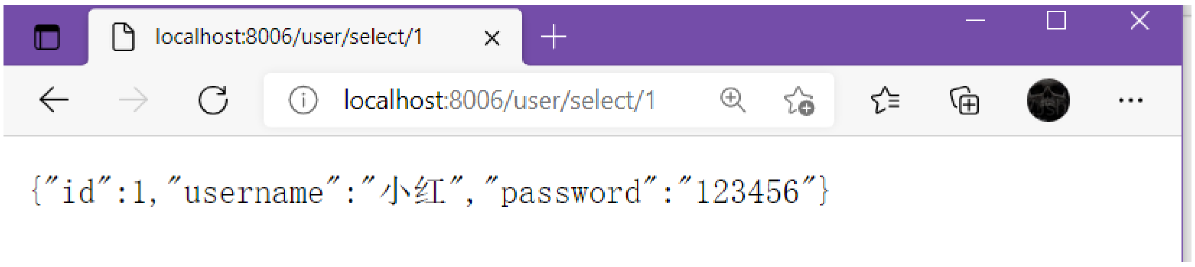
第三步： 测试ssm项目

（1）测试查寻全部用户！



```
[{"id":1,"username":"小红","password":"123456"},
{"id":2,"username":"xla","password":"123456"},
{"id":3,"username":"小王","password":"123456"},
{"id":4,"username":"小黑","password":"123456"},
{"id":9,"username":"田智鹏","password":"12345633232"}]
```

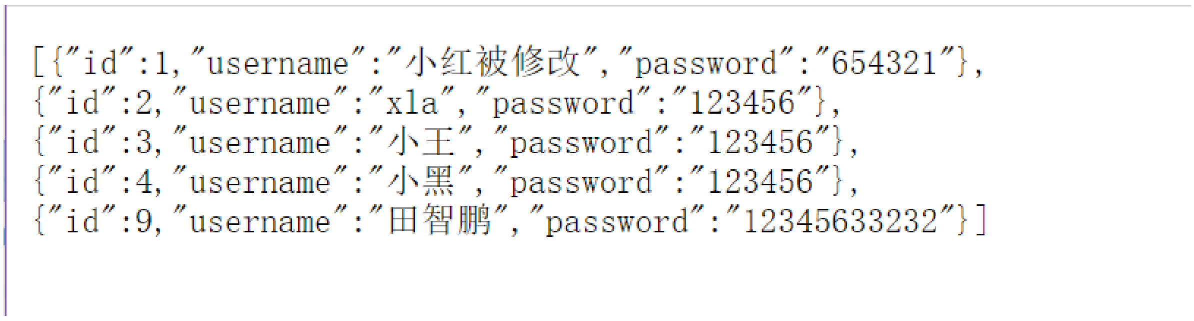
（2）测试查寻指定用户！



```
{"id":1,"username":"小红","password":"123456"}
```

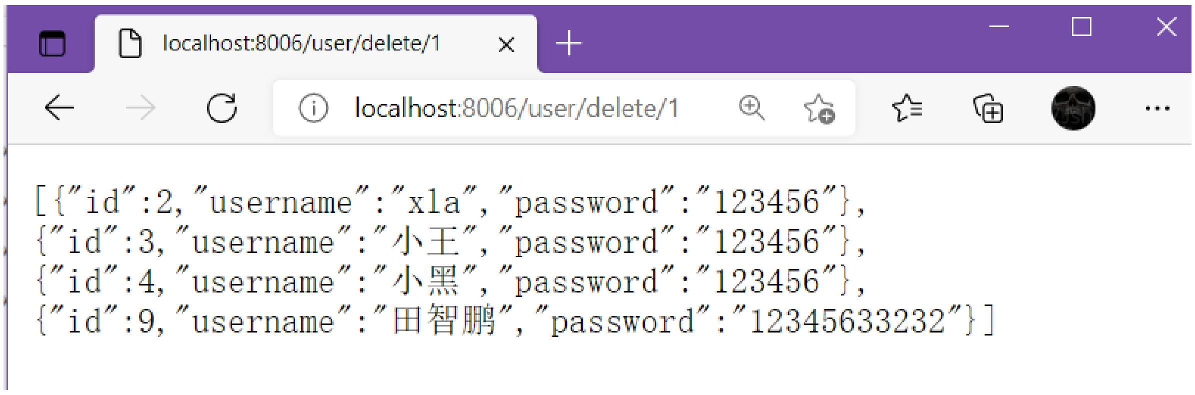
(3) 测试修改指定用户!

```
http://localhost:8006/user/update?id=1,username="小红被修改",password="654321"
```



```
[{"id":1,"username":"小红被修改","password":"654321"},
{"id":2,"username":"xla","password":"123456"},
{"id":3,"username":"小王","password":"123456"},
{"id":4,"username":"小黑","password":"123456"},
{"id":9,"username":"田智鹏","password":"12345633232"}]
```

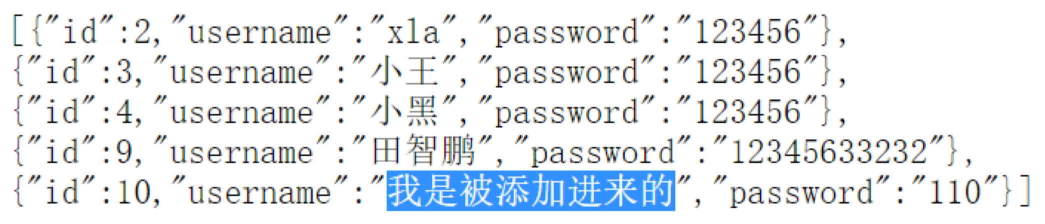
(4) 测试删除一个用户!



```
[{"id":2,"username":"xla","password":"123456"},
{"id":3,"username":"小王","password":"123456"},
{"id":4,"username":"小黑","password":"123456"},
{"id":9,"username":"田智鹏","password":"12345633232"}]
```

(5) 测试增加一个用户!

```
http://localhost:8006/user/add?id=10,username="我是被添加进来的",password="654321"
```



```
[{"id":2,"username":"xla","password":"123456"},
{"id":3,"username":"小王","password":"123456"},
{"id":4,"username":"小黑","password":"123456"},
{"id":9,"username":"田智鹏","password":"12345633232"},
{"id":10,"username":"我是被添加进来的","password":"110"}]
```

作者：SmartDragon

邮箱：2455404279@qq.com

欢迎留言询问哈！