Spring Boot

Spring Boot 是一个快速开发框架，可以迅速搭建出一套基于 Spring 框架体系的应用，是 Spring Cloud 的基础。

Spring Boot 开启了各种自动装配，从而简化代码的开发，不需要编写各种配置文件，只需要引入相关依赖就可以迅速搭建一个应用。

- 特点

1、不需要 web.xml

2、不需要 springmvc.xml

3、不需要 tomcat，Spring Boot 内嵌了 tomcat

4、不需要配置 JSON 解析，支持 REST 架构

5、个性化配置非常简单

- 如何使用

1、创建 Maven 工程，导入相关依赖。

```xml
<!-- 继承父包 -->
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.7.RELEASE</version>
</parent>

<dependencies>
  <!-- web启动jar -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.6</version>
    <scope>provided</scope>
  </dependency>
</dependencies>
```

2、创建 Student 实体类

```java
package com.southwind.entity;

import lombok.Data;

@Data
public class Student {
    private long id;
    private String name;
    private int age;
}
```

3、StudentRepository

```java
package com.southwind.repository;

import com.southwind.entity.Student;

import java.util.Collection;

public interface StudentRepository {
    public Collection<Student> findAll();
    public Student findById(long id);
    public void saveOrUpdate(Student student);
    public void deleteById(long id);
}
```

4、StudentRepositoryImpl

```java
package com.southwind.repository.impl;

import com.southwind.entity.Student;
import com.southwind.repository.StudentRepository;
import org.springframework.stereotype.Repository;

import java.util.Collection;
import java.util.HashMap;
import java.util.Map;

@Repository
public class StudentRepositoryImpl implements StudentRepository {

    private static Map<Long,Student> studentMap;

    static{
        studentMap = new HashMap<>();
        studentMap.put(1L,new Student(1L,"张三",22));
        studentMap.put(2L,new Student(2L,"李四",23));
        studentMap.put(3L,new Student(3L,"王五",24));
    }

    @Override
    public Collection<Student> findAll() {
        return studentMap.values();
    }

    @Override
    public Student findById(long id) {
        return studentMap.get(id);
    }

    @Override
    public void saveOrUpdate(Student student) {
        studentMap.put(student.getId(),student);
    }

    @Override
    public void deleteById(long id) {
        studentMap.remove(id);
    }
}
```

5、StudentHandler

```
package com.southwind.controller;

import com.southwind.entity.Student;
import com.southwind.repository.StudentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.Collection;

@RestController
@RequestMapping("/student")
public class StudentHandler {

    @Autowired
    private StudentRepository studentRepository;

    @GetMapping("/findAll")
    public Collection<Student> findAll(){
        return studentRepository.findAll();
    }

    @GetMapping("/findById/{id}")
    public Student findById(@PathVariable("id") long id){
        return studentRepository.findById(id);
    }

    @PostMapping("/save")
    public void save(@RequestBody Student student){
        studentRepository.saveOrUpdate(student);
    }

    @PutMapping("/update")
    public void update(@RequestBody Student student){
        studentRepository.saveOrUpdate(student);
    }

    @DeleteMapping("/deleteById/{id}")
    public void deleteById(@PathVariable("id") long id){
        studentRepository.deleteById(id);
    }
}
```

6、application.yml

```yaml
server:
  port: 9090
```

7、启动类

```java
package com.southwind;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class,args);
    }
}
```

`@SpringBootApplication` 表示当前类是 Spring Boot 的入口，Application 类的存放位置必须是其他相关业务类的存放位置的父级。

Spring Boot 整合 JSP

- pom.xml

```xml
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.0.7.RELEASE</version>
```

```
</parent>

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

  <!-- 整合JSP -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
  </dependency>
  <dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
  </dependency>

  <!-- JSTL -->
  <dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
  </dependency>

  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.6</version>
    <scope>provided</scope>
  </dependency>
</dependencies>
```

- 创建配置文件 application.yml

```
server:
  port: 8181
spring:
  mvc:
    view:
      prefix: /
      suffix: .jsp
```

- 创建 Handler

```
package com.southwind.controller;
```

```java
import com.southwind.entity.Student;
import com.southwind.repository.StudentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

@Controller
@RequestMapping("/hello")
public class HelloHandler {

    @Autowired
    private StudentRepository studentRepository;

    @GetMapping("/index")
    public ModelAndView index(){
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("index");
        modelAndView.addObject("list",studentRepository.findAll());
        return modelAndView;
    }

    @GetMapping("/deleteById/{id}")
    public String deleteById(@PathVariable("id") long id){
        studentRepository.deleteById(id);
        return "redirect:/hello/index";
    }

    @PostMapping("/save")
    public String save(Student student){
        studentRepository.saveOrUpdate(student);
        return "redirect:/hello/index";
    }

    @PostMapping("/update")
    public String update(Student student){
        studentRepository.saveOrUpdate(student);
        return "redirect:/hello/index";
    }

    @GetMapping("/findById/{id}")
    public ModelAndView findById(@PathVariable("id") long id){
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("update");
        modelAndView.addObject("student",studentRepository.findById(id));
        return modelAndView;
    }
}
```

- JSP

```
<%--
  Created by IntelliJ IDEA.
  User: southwind
  Date: 2019-03-21
  Time: 12:02
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page isELIgnored="false" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h1>学生信息</h1>
    <table>
        <tr>
            <th>学生编号</th>
            <th>学生姓名</th>
            <th>学生年龄</th>
            <th>操作</th>
        </tr>
        <c:forEach items="${list}" var="student">
            <tr>
                <td>${student.id}</td>
                <td>${student.name}</td>
                <td>${student.age}</td>
                <td>
                    <a href="/hello/findById/${student.id}">修改</a>
                    <a href="/hello/deleteById/${student.id}">删除</a>
                </td>
            </tr>
        </c:forEach>
    </table>
    <a href="/save.jsp">添加学生</a>
</body>
</html>
```

```
<%--
  Created by IntelliJ IDEA.
  User: southwind
  Date: 2019-03-21
  Time: 12:09
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <form action="/hello/save" method="post">
        ID:<input type="text" name="id"/><br/>
        name:<input type="text" name="name"/><br/>
        age:<input type="text" name="age"/><br/>
        <input type="submit" value="提交"/>
    </form>
</body>
</html>
```

```
<%--
  Created by IntelliJ IDEA.
  User: southwind
  Date: 2019-03-21
  Time: 12:09
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <form action="/hello/update" method="post">
        ID:<input type="text" name="id" value="${student.id}" readonly/><br/>
        name:<input type="text" name="name" value="${student.name}"/><br/>
        age:<input type="text" name="age" value="${student.age}"/><br/>
        <input type="submit" value="提交"/>
    </form>
</body>
</html>
```

Spring Boot HTML

Spring Boot 可以结合 Thymeleaf 模版来整合 HTML，使用原生的 HTML 作为视图。

Thymeleaf 模版是面向 Web 和独立环境的 Java 模版引擎，能够处理 HTML、XML、JavaScript、CSS 等。

```
<p th:text="${message}"></p>
```

- pom.xml

```xml
<!-- 继承父包 -->
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.0.7.RELEASE</version>
</parent>

<dependencies>
  <!-- web启动jar -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.6</version>
    <scope>provided</scope>
  </dependency>

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

- appliction.yml

```yaml
server:
  port: 9090
spring:
  thymeleaf:
    prefix: classpath:/templates/
    suffix: .html
    mode: HTML5
    encoding: UTF-8
```

- Handler

```
package com.southwind.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/index")
public class IndexHandler {

    @GetMapping("/index")
    public String index(){
        System.out.println("index...");
        return "index";
    }
}
```

- HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <h1>Hello World</h1>
</body>
</html>
```

如果希望客户端可以直接访问 HTML 资源，将这些资源放置在 static 路径下即可，否则必须通过 Handler 的后台映射才可以访问静态资源。

Thymeleaf 常用语法

- 赋值、拼接

```
@GetMapping("/index2")
public String index2(Map<String,String> map){
  map.put("name","张三");
  return "index";
}
```

```
<p th:text="${name}"></p>
<p th:text="'学生姓名是'+${name}+2"></p>
<p th:text="|学生姓名是,${name}|"></p>
```

- 条件判断：if/unless

th:if 表示条件成立时显示内容，th:unless 表示条件不成立时显示内容

```
@GetMapping("/if")
public String index3(Map<String,Boolean> map){
    map.put("flag",true);
    return "index";
}
```

```
<p th:if="${flag == true}" th:text="if判断成立"></p>
<p th:unless="${flag != true}" th:text="unless判断成立"></p>
```

- 循环

```
@GetMapping("/index")
public String index(Model model){
    System.out.println("index...");
    List<Student> list = new ArrayList<>();
    list.add(new Student(1L,"张三",22));
    list.add(new Student(2L,"李四",23));
    list.add(new Student(3L,"王五",24));
    model.addAttribute("list",list);
    return "index";
}
```

```
<table>
  <tr>
    <th>index</th>
    <th>count</th>
    <th>学生ID</th>
    <th>学生姓名</th>
    <th>学生年龄</th>
  </tr>
  <tr th:each="student,stat:${list}" th:style="'background-
color:'+@{${stat.odd}?'#F2F2F2'}">
    <td th:text="${stat.index}"></td>
    <td th:text="${stat.count}"></td>
    <td th:text="${student.id}"></td>
    <td th:text="${student.name}"></td>
    <td th:text="${student.age}"></td>
  </tr>
</table>
```

stat 是状态变量，属性：

- index 集合中元素的index（从0开始）
- count 集合中元素的count（从1开始）

- size 集合的大小
- current 当前迭代变量
- even/odd 当前迭代是否为偶数/奇数（从0开始计算）
- first 当前迭代的元素是否是第一个
- last 当前迭代的元素是否是最后一个

- URL

Thymeleaf 对于 URL 的处理是通过 `@{...}` 进行处理，结合 th:href 、th:src

```
<h1>Hello World</h1>
<a th:href="@{http://www.baidu.com}">跳转</a>
<a th:href="@{http://localhost:9090/index/url/{na}(na=${name})}">跳转2</a>
<img th:src="${src}">
<div th:style="'background:url('+ @{${src}} +');'">
<br/>
<br/>
<br/>
</div>
```

- 三元运算

```
@GetMapping("/eq")
public String eq(Model model){
    model.addAttribute("age",30);
    return "test";
}
```

```
<input th:value="${age gt 30?'中年':'青年'}"/>
```

- gt great than 大于
- ge great equal 大于等于
- eq equal 等于
- lt less than 小于
- le less equal 小于等于
- ne not equal 不等于

- switch

```
@GetMapping("/switch")
public String switchTest(Model model){
    model.addAttribute("gender","女");
    return "test";
}
```

```
<div th:switch="${gender}">
  <p th:case="女">女</p>
  <p th:case="男">男</p>
  <p th:case="*">未知</p>
</div>
```

- 基本对象
  - `#ctx`：上下文对象
  - `#vars`：上下文变量
  - `#locale`：区域对象
  - `#request`：HttpServletRequest 对象
  - `#response`：HttpServletResponse 对象
  - `#session`：HttpSession 对象
  - `#servletContext`：ServletContext 对象

```
@GetMapping("/object")
public String object(HttpServletRequest request){
    request.setAttribute("request","request对象");
    request.getSession().setAttribute("session","session对象");
    return "test";
}
```

```
<p th:text="${#request.getAttribute('request')}"></p>
<p th:text="${#session.getAttribute('session')}"></p>
<p th:text="${#locale.country}"></p>
```

- 内嵌对象

可以直接通过 # 访问。

1、dates：java.util.Date 的功能方法

2、calendars：java.util.Calendar 的功能方法

3、numbers：格式化数字

4、strings：java.lang.String 的功能方法

5、objects：Object 的功能方法

6、bools：对布尔求值的方法

7、arrays：操作数组的功能方法

8、lists：操作集合的功能方法

9、sets：操作集合的功能方法

10、maps：操作集合的功能方法

```
@GetMapping("/util")
public String util(Model model){
    model.addAttribute("name","zhangsan");
    model.addAttribute("users",new ArrayList<>());
    model.addAttribute("count",22);
    model.addAttribute("date",new Date());
    return "test";
}
```

```
<!-- 格式化时间 -->
<p th:text="${#dates.format(date,'yyyy-MM-dd HH:mm:sss')}"></p>
<!-- 创建当前时间，精确到天 -->
<p th:text="${#dates.createToday()}"></p>
<!-- 创建当前时间，精确到秒 -->
<p th:text="${#dates.createNow()}"></p>
<!-- 判断是否为空 -->
<p th:text="${#strings.isEmpty(name)}"></p>
<!-- 判断List是否为空 -->
<p th:text="${#lists.isEmpty(users)}"></p>
<!-- 输出字符串长度 -->
<p th:text="${#strings.length(name)}"></p>
<!-- 拼接字符串 -->
<p th:text="${#strings.concat(name,name,name)}"></p>
<!-- 创建自定义字符串 -->
<p th:text="${#strings.randomAlphanumeric(count)}"></p>
```

Spring Boot 数据校验

```
package com.southwind.entity;

import lombok.Data;
import org.hibernate.validator.constraints.Length;

import javax.validation.constraints.Min;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;

@Data
public class User {
    @NotNull(message = "id不能为空")
    private Long id;
    @NotEmpty(message = "姓名不能为空")
    @Length(min = 2,message = "姓名长度不能小于2位")
    private String name;
    @Min(value = 16,message = "年龄必须大于16岁")
    private int age;
}
```

```
@GetMapping("/validator")
public void validatorUser(@Valid User user,BindingResult bindingResult){
  System.out.println(user);
  if(bindingResult.hasErrors()){
    List<ObjectError> list = bindingResult.getAllErrors();
    for(ObjectError objectError:list){
      System.out.println(objectError.getCode()+"-
"+objectError.getDefaultMessage());
    }
  }
}
```

Spring Boot 整合 JDBC

- pom.xml

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

<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.11</version>
</dependency>
```

- application.yml

```
server:
  port: 9090
spring:
  thymeleaf:
    prefix: classpath:/templates/
    suffix: .html
    mode: HTML5
    encoding: UTF-8
  datasource:
    url: jdbc:mysql://localhost:3306/test?
useUnicode=true&characterEncoding=UTF-8
    username: root
    password: root
    driver-class-name: com.mysql.cj.jdbc.Driver
```

- User

```
package com.southwind.entity;

import lombok.Data;
import org.hibernate.validator.constraints.Length;

import javax.validation.constraints.Min;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;

@Data
public class User {
    @NotNull(message = "id不能为空")
    private Long id;
    @NotEmpty(message = "姓名不能为空")
    @Length(min = 2,message = "姓名长度不能小于2位")
    private String name;
    @Min(value = 60,message = "成绩必须大于60分")
    private double score;
}
```

- UserRepository

```
package com.southwind.repository;

import com.southwind.entity.User;

import java.util.List;

public interface UserRepository {
    public List<User> findAll();
    public User findById(long id);
    public void save(User user);
    public void update(User user);
    public void deleteById(long id);
}
```

- UserRepositoryImpl

```
package com.southwind.repository.impl;

import com.southwind.entity.User;
import com.southwind.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public class UserRepositoryImpl implements UserRepository {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    @Override
    public List<User> findAll() {
        return jdbcTemplate.query("select * from user",new
BeanPropertyRowMapper<>(User.class));
    }

    @Override
    public User findById(long id) {
        return jdbcTemplate.queryForObject("select * from user where id = ?",new
Object[]{id},new BeanPropertyRowMapper<>(User.class));
    }

    @Override
    public void save(User user) {
        jdbcTemplate.update("insert into user(name,score)
values(?,?)",user.getName(),user.getScore());
    }

    @Override
    public void update(User user) {
        jdbcTemplate.update("update user set name = ?,score = ? where id =
?",user.getName(),user.getScore(),user.getId());
    }

    @Override
    public void deleteById(long id) {
        jdbcTemplate.update("delete from user where id = ?",id);
    }
}
```

- Handler

```
package com.southwind.controller;

import com.southwind.entity.User;
import com.southwind.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/user")
public class UserHandler {

    @Autowired
    private UserRepository userRepository;

    @GetMapping("/findAll")
    public List<User> findAll(){
        return userRepository.findAll();
    }

    @GetMapping("/findById/{id}")
    public User findById(@PathVariable("id") long id){
        return userRepository.findById(id);
    }

    @PostMapping("/save")
    public void save(@RequestBody User user){
        userRepository.save(user);
    }

    @PutMapping("/update")
    public void update(@RequestBody User user){
        userRepository.update(user);
    }

    @DeleteMapping("/deleteById/{id}")
    public void deleteById(@PathVariable("id") long id){
        userRepository.deleteById(id);
    }
}
```

Spring Boot 整合 MyBatis

- pom.xml

```
<dependency>
  <groupId>org.mybatis.spring.boot</groupId>
  <artifactId>mybatis-spring-boot-starter</artifactId>
  <version>1.3.1</version>
</dependency>
```

- application.yml

```
server:
  port: 9090
spring:
  thymeleaf:
    prefix: classpath:/templates/
    suffix: .html
    mode: HTML5
    encoding: UTF-8
  datasource:
    url: jdbc:mysql://localhost:3306/test?
useUnicode=true&characterEncoding=UTF-8
    username: root
    password: root
    driver-class-name: com.mysql.cj.jdbc.Driver
mybatis:
  mapper-locations: classpath:/mapping/*.xml
  type-aliases-package: com.southwind.entity
```

- UserRepository

```
package com.southwind.mapper;

import com.southwind.entity.User;
import java.util.List;

public interface UserRepository {
    public List<User> findAll(int index,int limit);
    public User findById(long id);
    public void save(User user);
    public void update(User user);
    public void deleteById(long id);
    public int count();
}
```

- UserRepository.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.southwind.mapper.UserRepository">

    <select id="findAll" resultType="User">
        select * from user limit #{param1},#{param2}
    </select>

    <select id="count" resultType="int">
        select count(id) from user
    </select>

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

    <insert id="save" parameterType="User">
        insert into user(name,score) values(#{name},#{score})
    </insert>

    <update id="update" parameterType="User">
        update user set name = #{name},score = #{score} where id = #{id}
    </update>

    <delete id="deleteById" parameterType="long">
        delete from user where id = #{id}
    </delete>
</mapper>
```

- User

```
package com.southwind.entity;

import lombok.Data;
import org.hibernate.validator.constraints.Length;

import javax.validation.constraints.Min;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;

@Data
public class User {
    @NotNull(message = "id不能为空")
    private Long id;
    @NotEmpty(message = "姓名不能为空")
    @Length(min = 2,message = "姓名长度不能小于2位")
    private String name;
    @Min(value = 60,message = "成绩必须大于60分")
    private double score;
}
```

- Handler

```
package com.southwind.controller;
import com.southwind.entity.User;
import com.southwind.mapper.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.ModelAndView;

@Controller
@RequestMapping("/mapper")
public class UserMapperHandler {

    @Autowired
    private UserRepository userRepository;
    private int limit = 10;

    @GetMapping("/findAll/{page}")
    public ModelAndView findAll(@PathVariable("page") int page){
        ModelAndView modelAndView = new ModelAndView();
        int index = (page-1)*limit;
        modelAndView.setViewName("show");
        modelAndView.addObject("list",userRepository.findAll(index,limit));
        modelAndView.addObject("page",page);
        //计算总页数
        int count = userRepository.count();
        int pages = 0;
        if(count%limit == 0){
            pages = count/limit;
        }else{
            pages = count/limit+1;
        }
        modelAndView.addObject("pages",pages);
        return modelAndView;
    }

    @GetMapping("/deleteById/{id}")
    public String deleteById(@PathVariable("id") long id){
        userRepository.deleteById(id);
        return "redirect:/mapper/findAll/1";
    }

    @GetMapping("/findById")
    public ModelAndView findById(@RequestParam("id") long id){
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject("user",userRepository.findById(id));
        modelAndView.setViewName("update");
        return modelAndView;
    }

    @PostMapping("/update")
    public String update(User user){
        userRepository.update(user);
        return "redirect:/mapper/findAll/1";
    }

    @PostMapping("/save")
    public String save(User user){
        userRepository.save(user);
        return "redirect:/mapper/findAll/1";
    }

    @GetMapping("/redirect/{name}")
    public String redirect(@PathVariable("name") String name){
        return name;
    }
}
```

- HTML

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <form action="/mapper/save" method="post">
        用户姓名: <input type="text" name="name" /><br/>
        用户成绩: <input type="text" name="score" /><br/>
        <input type="submit" value="提交"/>
    </form>
</body>
</html>
```

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <form action="/mapper/update" method="post">
        用户ID: <input type="text" name="id" th:value="${user.id}" readonly/>
<br/>
        用户姓名: <input type="text" name="name" th:value="${user.name}" /><br/>
        用户成绩: <input type="text" name="score" th:value="${user.score}" /><br/>
        <input type="submit" value="提交"/>
    </form>
</body>
</html>
```

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script type="text/javascript" th:src="@{/jquery-3.3.1.min.js}"></script>
    <script type="text/javascript">
        $(function(){
            $("#first").click(function(){
                var page = $("#page").text();
                page = parseInt(page);
                if(page == 1){
                    return false;
                }
                window.location.href="/mapper/findAll/1";
            });
            $("#previous").click(function(){
                var page = $("#page").text();
                page = parseInt(page);
                if(page == 1){
                    return false;
                }
                page = page-1;
                window.location.href="/mapper/findAll/"+page;
            });
            $("#next").click(function(){
                var page = $("#page").text();
                var pages = $("#pages").text();
                if(page == pages){
                    return false;
                }
                page = parseInt(page);
                page = page+1;
                window.location.href="/mapper/findAll/"+page;
            });
            $("#last").click(function(){
                var page = $("#page").text();
                var pages = $("#pages").text();
                if(page == pages){
                    return false;
                }
                window.location.href="/mapper/findAll/"+pages;
            });
        });
    </script>
</head>
<body>
    <h1>用户信息</h1>
    <table>
        <tr>
            <th>用户ID</th>
            <th>用户名</th>
            <th>成绩</th>
            <th>操作</th>
        </tr>
        <tr th:each="user:${list}">
            <td th:text="${user.id}"></td>
            <td th:text="${user.name}"></td>
            <td th:text="${user.score}"></td>
            <td>
                <a th:href="@{/mapper/deleteById/{id}(id=${user.id})}">删除</a>
                <a th:href="@{/mapper/findById(id=${user.id})}">修改</a>
            </td>
        </tr>
    </table>
    <a id="first" href="javascript:void(0)">首页</a>
    <a id="previous" href="javascript:void(0)">上一页</a>
    <span id="page" th:text="${page}"></span>/<span id="pages"
```

```
th:text="${pages}"></span>
    <a id="next" href="javascript:void(0)">下一页</a>
    <a id="last" href="javascript:void(0)">尾页</a><br/>
    <a href="/mapper/redirect/save">添加用户</a>
</body>
</html>
```