

实验一 基于 Mybatis 的校级课程管理程序

一、任务目的

掌握 MyBatis 的环境搭建、核心配置文件、映射文件，学会使用基于 XML 和基于注解的 MyBatis 进行关系数据库的增删查改操作。

二、实验要求

1. 技术选型：Java + Spring + MyBatis
2. 实验结果在 Test 包中创建测试类，将实验结果直接输出到控制台
3. 提交实验报告([github地址](#))

三、实验环境

- JDK 1.8.0_171
- MySQL 8.0.31
- maven-3.9.1
- IDEA-2022

四、实验内容

1.搭建实验数据库。

创建 mybatis 数据库，并创建两个表，分别为一个课程表 c_course 和一个学院表 c_school，学院表和课程表之间是一对多的关系。课程表 c_course 表：

课程id(id)	课程名(name)	课时(hours)	开课学院(sid)
1	C语言程序设计	70	1
2	Python程序设计	70	1
3	大学英语	96	2
4	高级Web技术	32	1
5	改革开放与新时期党的历史研究	32	3
6	企业战略管理	32	4

学院表 s_school 表：

学院 id(id)	学院名称(schoolname)
1	计算机学院
2	外国语学院
3	马克思学院
4	商学院

SQL 代码如下：

```
USE mybatis;
CREATE TABLE s_school (
    id int(32) PRIMARY KEY AUTO_INCREMENT,
    schoolname varchar(40)
);
INSERT INTO s_school VALUES (1, '计算机学院');
INSERT INTO s_school VALUES (2, '外国语学院');
INSERT INTO s_school VALUES (3, '马克思学院');
INSERT INTO s_school VALUES (4, '商学院');

CREATE TABLE c_course (
    id int(32) PRIMARY KEY AUTO_INCREMENT,
    name varchar(40),
    hours int,
    sid int(32) NOT NULL,
    FOREIGN KEY(sid) REFERENCES s_school(id)
);
INSERT INTO c_course VALUES (1, 'C语言程序设计', 70,1);
INSERT INTO c_course VALUES (2, 'Python程序设计', 70,1);
INSERT INTO c_course VALUES (3, '大学英语', 96,2);
INSERT INTO c_course VALUES (4, '高级Web技术', 32,1);
INSERT INTO c_course VALUES (5, '改革开放与新时期党的历史研究', 32,3);
INSERT INTO c_course VALUES (6, '企业战略管理', 32,4);
```

2. 创建 Maven 工程，命名为 demo1，在 pom.XML 中导入了相关依赖。

```
<dependencies>
    <!--Mybatis核心包-->
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>3.5.13</version>
    </dependency>
    <!--MYSQL驱动包-->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.32</version>
    </dependency>
    <!--JUnit测试包-->
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13.2</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

3. 连接数据库，创建 db.properties 文件。

```
mysql.driver=com.mysql.cj.jdbc.Driver
mysql.url=jdbc:mysql://localhost:3306/mybatis?\n    serverTimezone=UTC&characterEncoding=utf8&useUnicode=true&useSSL=false
mysql.username=root
mysql.password=11h1314520
```

4. 编写 MyBatis 核心配置文件 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>
  <!-- 环境配置 -->
  <!-- 加载类路径下的属性文件 -->
  <properties resource="db.properties"/>
  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC"/>
      <!-- 数据库连接相关配置 ,db.properties文件中的内容-->
      <dataSource type="POOLED">
        <property name="driver" value="${mysql.driver}"/>
        <property name="url" value="${mysql.url}"/>
        <property name="username" value="${mysql.username}"/>
        <property name="password" value="${mysql.password}"/>
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="mappers/CourseMapper.xml"/>
  </mappers>
</configuration>
```

5. 在 src/main/java 目录下创建 org.example.pojo 包，并创建 Course 和 School 两个持久化类，并分别在类中定义相关属性。

```
package org.example.pojo;

public class Course {
    private int id; //课程id
    private String name; //课程名
    private int hours; //课时
    private int sid; //开课学院
    private String schoolname; //学院名称

    public int getId() {
```

```
        return id;
    }

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

    public String getName() {
        return name;
    }

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

    public int getHours() {
        return hours;
    }

    public void setHours(int hours) {
        this.hours = hours;
    }

    public int getSid() {
        return sid;
    }

    public void setSid(int sid) {
        this.sid = sid;
    }

    public String getSchoolname() {
        return schoolname;
    }

    public void setSchoolname(String schoolname) {
        this.schoolname = schoolname;
    }

    @Override
    public String toString() {
        return "Course{" +
            "id=" + id +
            ", name='" + name + '\'' +
            ", hours=" + hours +
            ", sid=" + sid +
            '}';
    }

    public String allinfo(){
        return "Course{" +
            "id=" + id +
            ", name='" + name + '\'' +
            ", hours=" + hours +
```

```
        ", sid=" + sid +
        ", schoolname='" + schoolname + '\'' +
        '}';
    }
}
```

6. 在 `src/main/resources` 目录下创建一个 `mappers` 文件夹，在 `mappers` 文件夹下创建 `Course` 的映射文件 `CourseMapper.xml`，在文件中编写配置。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "https://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="org.example.pojo.Course">
    <select id="findById" parameterType="int"
resultType="org.example.pojo.Course">
        select * from c_course where id=#{id}
    </select>
    <select id="findByName" parameterType="String"
resultType="org.example.pojo.Course">
        select * from c_course as c,s_school as s where c.sid=s.id and
s.schoolname=#{id}
    </select>
    <select id="findAll" resultType="org.example.pojo.Course">
        select c.id,name,hours,sid from c_course as c,s_school as s where sid in
        (select id from s_school) and s.id=c.sid order by c.sid
    </select>
    <update id="updateByHours" parameterType="int">
        update c_course set hours=hours+#{hours} where id=#{id}
    </update>
    <insert id="insertByCourse" parameterType="org.example.pojo.Course">
        insert into c_course(id,name,hours,sid)values(#{id},#{name},#{hours},#
{sid})
    </insert>
</mapper>
```

7. 在核心配置文件 `mybatis-config.xml` 中添加 `mapper` 子标签指定 `CourseMapper.xml` 配置文件所在的位置。

```
<mappers>
    <mapper resource="mappers/CourseMapper.xml"/>
</mappers>
```

8. **MyBatisUtils 工具类。**

```
package Utils;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import java.io.Reader;

public class MybatisUtil {
    private static SqlSessionFactory sqlSessionFactory = null;
    // 初始化SqlSessionFactory对象
    static {
        try {
            // 使用MyBatis提供的Resources类加载MyBatis的配置文件
            Reader reader =
                Resources.getResourceAsReader("mybatis-config.xml");
            // 构建SqlSessionFactory工厂
            sqlSessionFactory =
                new SqlSessionFactoryBuilder().build(reader);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    // 获取SqlSession对象的静态方法
    public static SqlSession getSession() {
        //开启一个事务
        return sqlSessionFactory.openSession(true);
    }
}
```

9. 编写测试类。

```
Scanner sc = new Scanner(System.in);
MybatisUtil mybatisUtil = new MybatisUtil();
SqlSession sqlSession = mybatisUtil.getSession();
```

① 查询 id=2 的课程信息；

```
@Test
// 通过 id 查询的课程信息
public void findById(){
    System.out.print("请输入您要查询的课程 id: ");
    int id = sc.nextInt();
    Course course = sqlSession.selectOne("findById", id);
    System.out.println("查询 id=" + id + " 的课程信息为: ");
    System.out.println(course.toString());
}
```

②查询出所有计算机学院开设的课程信息；

```
@Test
//通过学院名称查出这个学院开的所有课程信息
public void findBySchoolName() {
    System.out.print("请输入您要查询的学院名称: ");
    String str = sc.next();
    List<Course> list = sqlSession.selectList("findBySchoolName", str);
    System.out.println("查询" + str + "的所有课程信息结果为: ");
    System.out.println(list);
}
```

③将 id=4 这门课程的课时数修改为 32+8=40；

```
@Test
//通过课程id修改这门课程的课时数
public void updateByHours() {
    Course course = new Course();
    System.out.print("请输入要修改课程课时数的id: ");
    int it=sc.nextInt();
    System.out.print("请输入要增加的课程课时数数量: ");
    int num=sc.nextInt();
    course.setId(it);
    course.setHours(num);
    int result = sqlSession.update("updateByHours", course);
    if(result==1)
        System.out.println("修改成功!");
    else System.out.println("修改失败! ");
}
```

④插入一条新的课程记录： name="大数据存储", hours=32, sid =1;

```
@Test
//插入一条新的课程记录
public void insertByCourse() {
    Course course = new Course();
    System.out.print("请输入要插入的课程id: ");
    int id=sc.nextInt();
    System.out.print("请输入要插入的课程名: ");
    String str=sc.next();
    System.out.print("请输入要插入的课时: ");
    int hour=sc.nextInt();
    System.out.print("请输入要插入的开课学院id: ");
    int sid=sc.nextInt();
    course.setSid(id);
    course.setName(str);
    course.setHours(hour);
```

```
course.setSid(sid);
int result = sqlSession.insert("insertByCourse", course);
if(result==1)
    System.out.println("插入成功! ");
else System.out.println("插入失败! ");
}
```

⑤输出所有的学院开设的课程信息。

```
@Test
//输出所有的学院开设的课程信息
public void findByAll() {
    List<Course> list = sqlSession.selectList("findByAll");
    System.out.println("所有的学院开设的课程信息为");
    System.out.println(list);
}
```

五、实验结果

1. 查询 id=2 的课程信息；

```
✓ Tests passed: 1 of 1 test – 7 sec 98 ms
"D:\Program Files\Java\jdk1.8.0_171\bin\java.exe" ...
请输入您要查询的课程id: 查询id=2的课程信息为:
Course{id=2, name='Python程序设计', hours=70, sid=1}

Process finished with exit code 0
```

2. 查询出所有计算机学院开设的课程信息；

```
✓ Tests passed: 1 of 1 test – 11 sec 597 ms
"D:\Program Files\Java\jdk1.8.0_171\bin\java.exe" ...
请输入您要查询的学院名称: 查询计算机学院的所有课程信息结果为:
Course{id=1, name='C语言程序设计', hours=70, sid=1}
Course{id=2, name='Python程序设计', hours=70, sid=1}
Course{id=4, name='高级Web技术', hours=32, sid=1}

Process finished with exit code 0
```

3. 将 id=4 这门课程的课时数修改为 32+8=40;

```
> ✓ Tests passed: 1 of 1 test – 22 sec 399 ms
| "D:\Program Files\Java\jdk1.8.0_171\bin\java.exe" ...
| 请输入要修改课程课时数的id: 请输入要增加的课程课时数数量: 修改成功!
|
| Process finished with exit code 0
```

4. 插入一条新的课程记录: name="大数据存储", hours=32, id =1;

5. 输出所有的学院开设的课程信息。

```
◀ MybatisTest.findByAll ×
» ✓ Tests passed: 1 of 1 test – 413 ms
| "D:\Program Files\Java\jdk1.8.0_171\bin\java.exe" ...
| 所有的学院开设的课程信息为
| Course{id=1, name='C语言程序设计', hours=70, sid=1}
| Course{id=2, name='Python程序设计', hours=70, sid=1}
| Course{id=4, name='高级Web技术', hours=40, sid=1}
| Course{id=32, name='大数据存储', hours=32, sid=1}
| Course{id=3, name='大学英语', hours=96, sid=2}
| Course{id=5, name='改革开放与新时期党的历史研究', hours=32, sid=3}
| Course{id=6, name='企业战略管理', hours=32, sid=4}
```

六、实验心得

通过这个实验，我学会了如何通过 MyBatis 配置文件和映射文件来实现对数据库表的增删改查等操作，能够更好地理解 MyBatis 的核心组件之间的关系，如SqlSessionFactory、SqlSession 和 Mapper 接口等。

实验二、基于 SpringMVC 的应用

一、任务目的

掌握 MyBatis 的环境搭建、核心配置文件、映射文件，学会使用基于 XML 和基于注解的 MyBatis 进行关系数据库的增删查改操作。

二、实验要求

1. 技术选型: Java + Spring + MyBatis +SpringMVC
2. 实验交互及结果要以Web视图形式进行展示
3. 提交实验报告([github地址](#))

三、实验环境

- JDK 1.8.0_171
- MySQL 8.0.31
- maven-3.9.1

- IDEA-2022
- tomcat7

四、实验内容

该实验继续沿用高级Web技术实验一中的题目背景，结合自己设计的数据模型和表设计，在此基础上，结合SpringMVC框架完成。

1. 展示课程

① allCourse.jsp 核心部分。

```
<table class="table table-hover table-striped">
    <thead>
        <tr>
            <th>课程id</th>
            <th>课程名</th>
            <th>课时数</th>
            <th>课程所属学院id</th>
            <th>操作</th>
        </tr>
    </thead>
    <tbody>
        <c:forEach var="course" items="${requestScope.get('course')}">
            <tr>
                <td>${course.getId()}</td>
                <td>${course.getName()}</td>
                <td>${course.getHours()}</td>
                <td>${course.getSid()}</td>
                <td>
                    <a href="${pageContext.request.contextPath}/course/toUpdateCourse?id=${course.getId()}">更改</a> |
                    <a href="${pageContext.request.contextPath}/course/del/${course.getId()}">删除</a>
                </td>
            </tr>
        </c:forEach>
    </tbody>
</table>
```

② Dao 层映射文件 CourseMapper.XML 核心部分

```
<select id="queryAllCourse" resultType="Course">
    select * from mybatis.c_course;
</select>
```

③ Service 层

```
public List<Course> queryAllCourse() {
    return courseMapper.queryAllCourse();
}
```

④Controller 层

```
//查询全部课程信息
@RequestMapping("/allCourse")
public String list(Model model){
    List<Course> course = courseService.queryAllCourse();
    for(Course b:course){
        System.out.println(b);
    }
    model.addAttribute("course",course);
    return "allCourse";
}
```

2.新增课程

①addCourse.jsp 核心部分。

```
<form action="${pageContext.request.contextPath}/course/addCourse" method="post">
    课程名: <input type="text" name="name"><br><br><br>
    课时数: <input type="text" name="hours"><br><br><br>
    课程所属学院id: <input type="text" name="sid"><br><br><br>
    <input type="submit" value="添加">
</form>
```

②Dao 层映射文件 CourseMapper.XML 核心部分

```
<insert id="addCourse" parameterType="Course">
    insert into mybatis.c_course(name, hours, sid)
    values (#{name}, #{hours}, #{sid});
</insert>
```

③Service 层

```
public int addCourse(Course course) {
    return courseMapper.addCourse(course);
}
```

④Controller 层

```
//添加课程信息
@RequestMapping("/toAddCourse")
public String toAddPaper() {
    return "addCourse";
}
@RequestMapping("/addCourse")
public String addPaper(Course course) {
    System.out.println(course);
    courseService.addCourse(course);
    return "redirect:/course/allCourse";
}
```

3. 修改课程

①updateCourse.jsp 核心部分。

```
<form action="${pageContext.request.contextPath}/course/updateCourse"
method="post">
    <input type="hidden" name="id" value="${course.getId()}" />
    课程名: <input type="text" name="name" value="${course.getName()}" />
    课时数: <input type="text" name="hours" value="${course.getHours()}" />
    课程所属学院: <input type="text" name="sid" value="${course.getSid()}" />
    <input type="submit" value="提交" />
</form>
```

②Dao 层映射文件 CourseMapper.XML 核心部分

```
<update id="updateCourse" parameterType="Course">
    update mybatis.c_course
    set name = #{name},hours = #{hours},sid=#{sid}
    where id = #{id};
</update>
```

③Service 层

```
public int updateCourse(Course course) {
    return courseMapper.updateCourse(course);
}
```

④Controller 层

```
//修改课程信息
@RequestMapping("/toUpdateCourse")
```

```
public String toUpdateCourse(Model model, int id) {
    Course course = courseService.queryCourseById(id);
    System.out.println(course);
    model.addAttribute("course", course );
    return "updateCourse";
}
@RequestMapping("/updateCourse")
public String updateCourse(Model model, Course course) {
    System.out.println(course);
    courseService.updateCourse(course);
    Course c = courseService.queryCourseById(course.getId());
    model.addAttribute("course", c);
    return "redirect:/course/allCourse";
}
```

4. 删除课程

① Dao 层映射文件 CourseMapper.XML 核心部分

```
<delete id="deleteCourseById" parameterType="int">
    delete from mybatis.c_course where id = #{id};
</delete>
```

② Service 层

```
public int deleteCourseById(int id) {
    return courseMapper.deleteCourseById(id);
}
```

③ Controller 层

```
//删除课程信息
@RequestMapping("/del/{id}")
public String deleteCourse(@PathVariable int id) {
    courseService.deleteCourseById(id);
    return "redirect:/course/allCourse";
}
```

五、实验结果

1. 展示课程：

课程列表 —— 显示所有课程

新增	课程id	课程名	课时数	课程所属学院id	操作
	1	C语言程序设计	70	1	更改 删除
	2	Python程序设计	70	1	更改 删除
	3	大学英语	96	2	更改 删除
	4	高级Web技术	40	1	更改 删除
	5	改革开放与新时期党的历史研究	32	3	更改 删除
	6	企业战略管理	32	4	更改 删除
	7	大数据存储	32	1	更改 删除

2. 新增课程：

新增课程

课程名: 课时数: 课程所属学院id:

3. 修改课程：

六、实验心得

本次实验通过跟着网上的视频教程一起学习完成，帮助我掌握了 SpringMVC 的数据绑定和响应，学会了使用 JSP 页面视图进行交互和结果的页面展示，让我对 SpringMVC 的执行流程和工作原理有了更深入的理解，在用户提交表单时，我们可以使用数据绑定将表单数据绑定到后端 Java 对象中，然后通过业务逻辑处理后，再将处理结果返回给前端页面。在这个过程中，SpringMVC 的数据绑定是比较高效灵活的。

实验三、基于 SpringMVC 的高级应用

一、任务目的

掌握 SpringMVC 的文件上传和展示。

二、实验要求

1. 技术选型：Java + Spring + MyBatis + SpringMVC
2. 提交实验报告([github地址](#))

三、实验环境

- JDK 1.8.0_171
- MySQL 8.0.31
- maven-3.9.1
- IDEA-2022
- tomcat7

四、实验内容

1. 在 pom.XML 配置依赖 commons-fileupload 和 commons-io , Maven 将会自动从 Maven 中央库中下载这些依赖项的 jar 包，并将其添加到项目的类路径下，以供项目使用。

```
<!--文件上传所需要的依赖-->
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.11.0</version>
</dependency>
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.5</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.13.4</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.13.4.2</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-annotations</artifactId>
    <version>2.13.4</version>
</dependency>
```

2. 在 Spring-mvc.XML 中加入 MultipartFile 接口，用于接收上传的文件内容。

```
<!--配置多部件解析器-->
<bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <!--设置请求编码方式-->
    <property name="defaultEncoding" value="UTF-8"/>
    <!--设置允许上传文件的最大值-->
```

```
<property name="maxUploadSize" value="9097152"/>
</bean>
```

3. 文件上传。通过 `MultipartFile` 类型的文件对象获取上传文件的原始文件名，提取出文件后缀，使用自己编写的`JSONFileUtils`类的静态方法生成一个新的唯一的文件名来保存上传的文件。

```
@RequestMapping("/addCourse")
public String addPaper(Course course, MultipartFile file, HttpServletRequest
request) throws Exception {
    //设置图片存放的路径
    String path=request.getServletContext().getRealPath("/")+"Image/";
    ObjectMapper mapper=new ObjectMapper();
    if(!file.isEmpty()) {
        //获取上传文件的名称
        String filename = file.getOriginalFilename();
        System.out.println(filename);
        ArrayList<String> list=new ArrayList<>();
        //读取file.json文件中的文件名称
        String json= JSONFileUtils.readFile(path+"files.json");
        if(json.length()>0){
            //将files.json的内容转换为集合
            list=mapper.readValue(json, new TypeReference<ArrayList<String>>()
{
    });
        }
        for(String image:list){
            //如果上传的文件在files.json文件有同名文件，将当前上传文件重命名，以避免重名
            if(filename.equals(image)){
                String[] split=filename.split("\\.");
                filename=split[0]+(1)+"."+split[1];
            }
        }
        //文件保存的全路径
        String filepath = path + filename;
        System.out.println(filepath);
        //保存上传的文件
        file.transferTo(new File(filepath));
        list.add(filename);
        course.setImage(filename);
        System.out.println(filename);
        //将集合转换成json
        json=mapper.writeValueAsString(list);
        //将上传文件的名称保存在files.json文件中
        JSONFileUtils.writeFile(json,path+"files.json");
    }
    else {
        course.setImage("test1.jpg");
    }
    courseService.addCourse(course);
}
```

```
        return "redirect:/course/allCourse";
    }
```

五、实验结果

1. 展示课程:

课程列表 —— 显示所有课程

新增	课程图片	课程id	课程名	课时数	课程所属学院id	操作
		1	C语言程序设计	70	1	更改 删除
		2	Python程序设计	70	1	更改 删除
		3	大学英语	96	2	更改 删除
		4	高级Web技术	40	1	更改 删除
		5	改革开放与新时期党的历史研究	32	3	更改 删除
		6	企业战略管理	32	4	更改 删除
		7	大数据存储	32	1	更改 删除

2. 新增课程:

新增课程

课程名:

课时数:

课程所属学院id:

课程图片:



[更换图片](#)

[添加](#)

3. 修改课程:

修改信息

课程名: 大学英语 课时数: 96 课程所属学院: 2 提交

六、实验心得

学会了文件的上传和下载以及图片的显示。

实验四：基于 SSM 的校级课程管理系统

一、任务目的

进一步掌握 SpringMVC 的数据绑定和页面跳转技术，能利用 JSP 视图技术来完成实际系统的开发。

二、实验要求

1. 技术选型：Java + Spring + MyBatis + SpringMVC
2. 实验交互及结果要以 Web 视图形式进行展示
3. 提交实验报告([github地址](#))

三、实验环境

- JDK 1.8.0_171
- MySQL 8.0.31
- maven-3.9.1
- IDEA-2022
- tomcat7

四、实验内容

1. 用户登陆：如果用户输入正确的邮箱密码以及验证码，提交表单给后端，后端查询数据库后确认邮箱和密码匹配后则自动进入到课程管理主界面。否则，返回登陆失败的提示。页面上如果邮箱和密码有一个为空的时候点击登陆按钮时前端需要显示“不能为空”。

①login.jsp

```
<form action="${pageContext.request.contextPath}/login" name="forms" method="post">
    <div class="box2">
        <span>${msg}</span><br/>
        <h2>LOGIN</h2>
        <p></p>
        <div id="input_box">
            <input id="username" type="text" name="email" placeholder="用户名">
        </div><br>
        <div>
            <input id="pwd" type="password" name="password" placeholder="密码">
        </div><br>
        <div>
```

```
        <input id="checkcode" type="text" name="checkcode" placeholder="请输入验证码">
    </div><br>
    <div>
        
        <br>
    </div>
    <div><br>
        <button id="btn" onclick="login()">登录</button>
    </div>
</div>
</form>
```

②javascript

```
<script>
    function login() {
        var x = document.getElementById("username").value;
        var y = document.getElementById("pwd").value;
        var z = document.getElementById("checkcode").value;
        if (x == null || x == "") {
            alert("用户名不能为空, 请输入用户名");
        }
        else if (y == null || y == "") {
            alert("密码不能为空, 请输入密码");
        }
        else if (z == null || z == "") {
            alert("验证码不能为空, 请输入验证码");
        }
        else {
            document.getElementsByTagName("form").submit();
        }
    }
</script>
```

Controller 层

```
/*
 * 用户登入
 */
@RequestMapping("/tologin")
public String tologin(){
    return "login";
}
@RequestMapping("/login")
public String login(User user, HttpServletRequest request, String checkcode){
    //RandomValidateCode.RANDOMCODEKEY为工具类定义的变量, 即验证码图片内正确识别后
```

的数据

```
String vcode = (String)  
  
request.getSession().getAttribute(RandomValidateCode.RANDOMCODEKEY);  
System.out.println(vcode);  
try{  
    User u=userService.login(user);  
    /*  
     *用户名和密码是否查询用户信息  
     *是: 跳转到后台首页  
     *否: 跳转到登入页面  
    */  
    if (u != null) {  
        if (checkcode.equals(vcode)) {  
            request.getSession().setAttribute("USER_SESSION", u);  
            return "main";  
        }  
        else {  
            request.getSession().setAttribute("msg", "验证码错误!");  
            return "redirect:tologin";  
        }  
    }  
    else {  
        request.getSession().setAttribute("msg", "用户名或密码错误!");  
        return "redirect:tologin";  
    }  
}  
}catch (Exception e){  
    e.printStackTrace();  
    request.setAttribute("msg", "系统出错啦!");  
    return "redirect:tologin";  
}  
}  
}
```

2. 用户登陆成功，转向课程管理主界面main.jsp，右上角显示用户邮箱和一个“退出”按钮（或超链接）。

```
<%--  
Created by IntelliJ IDEA.  
User: lilonghua  
Date: 2023/4/24  
Time: 16:52  
To change this template use File | Settings | File Templates.  
--%>  
<%@ page contentType="text/html; charset=UTF-8" language="java" %>  
<html>  
<head>  
    <title>后台系统</title>  
</head>  
<body>  
    <li>您好: ${USER_SESSION.name}</li>  
    <li><a href="${pageContext.request.contextPath}/logout">退出</a></li>  
    <li><a href="${pageContext.request.contextPath}/course/allCourse">课程信息</a>
```

```
</li>
</body>
</html>
```

3. 定义一个请求拦截器，用于对用户请求进行拦截和处理。

在Spring MVC中配置拦截器

```
<!--配置拦截器-->
<mvc:interceptors>
    <!--拦截所有请求-->
    <bean class="org.example.interceptor.ResourcesInterceptor"/>
</mvc:interceptors>
```

创建一个ResourcesInterceptor类去继承HandlerInterceptorAdapter接口

```
public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
                           Object handler) throws Exception{
    User user=(User)request.getSession().getAttribute("USER_SESSION");
    //如果用户是已登入状态,放行
    if(user!=null){
        return true;
    }
    //获取请求路径
    String url=request.getRequestURI();
    //用户登入的相关请求,放行
    if(url.indexOf("login")>=0 || url.indexOf(".jpg")>=0)
        return true;
    //其他情况都直接跳转到登入页面
    request.setAttribute("msg","您还没有登入,请先登入!");
    request.getRequestDispatcher("/WEB-
INF/jsp/login.jsp").forward(request,response);
    return false;
}
```

4. 生成验证码图片的工具类。

```
package org.example.utils;

import javax.imageio.ImageIO;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.Random;
```

```
public class RandomValidateCode {  
    public static final String RANDOMCODEKEY = "randomcode_key"; // 放到session中的key  
    private Random random = new Random();  
    private String randString = "0123456789ABCDEFGHIJKLMNPQRSTUVWXYZ"; // 随机产生的字符串  
    private int width = 80; // 图片宽  
    private int height = 26; // 图片高  
    private int lineSize = 40; // 干扰线数量  
    private int stringNum = 4; // 随机产生字符数量  
  
    /**  
     * 生成随机图片  
     */  
    public void getRandcode(HttpServletRequest request, HttpServletResponse response) {  
        HttpSession session = request.getSession();  
        // BufferedImage类是具有缓冲区的Image类, Image类是用于描述图像信息的类  
        BufferedImage image = new BufferedImage(width, height,  
        BufferedImage.TYPE_INT_BGR);  
        // 产生Image对象的Graphics对象, 该对象可以在图像上进行各种绘制操作  
        Graphics g = image.getGraphics();  
        g.fillRect(0, 0, width, height);  
        g.setFont(new Font("Times New Roman", Font.ROMAN_BASELINE, 18));  
        g.setColor(getRandColor(160, 200));  
        // 绘制干扰线  
        for (int i = 0; i <= lineSize; i++) {  
            drawLine(g);  
        }  
        // 绘制随机字符  
        String randomString = "";  
        for (int i = 1; i <= stringNum; i++) {  
            randomString = drawString(g, randomString, i);  
        }  
        session.removeAttribute(RANDOMCODEKEY);  
        session.setAttribute(RANDOMCODEKEY, randomString);  
        g.dispose();  
        try {  
            // 将内存中的图片通过流动形式输出到客户端  
            ImageIO.write(image, "JPEG", response.getOutputStream());  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
  
    /*  
     * 获得字体  
     */  
    private Font getFont() {  
        return new Font("Fixedsys", Font.CENTER_BASELINE, 18);  
    }  
  
    /*
```

```
* 获得颜色
*/
private Color getRandColor(int fc, int bc) {
    if (fc > 255)
        fc = 255;
    if (bc > 255)
        bc = 255;
    int r = fc + random.nextInt(bc - fc - 16);
    int g = fc + random.nextInt(bc - fc - 14);
    int b = fc + random.nextInt(bc - fc - 18);
    return new Color(r, g, b);
}

/*
* 绘制字符串
*/
private String drawString(Graphics g, String randomString, int i) {
    g.setFont(getFont());
    g.setColor(new Color(random.nextInt(101), random.nextInt(111),
random.nextInt(121)));
    String rand =
String.valueOf(getRandomString(random.nextInt(randString.length())));
    randomString += rand;
    g.translate(random.nextInt(3), random.nextInt(3));
    g.drawString(rand, 13 * i, 16);
    return randomString;
}

/*
* 绘制干扰线
*/
private void drawLine(Graphics g) {
    int x = random.nextInt(width);
    int y = random.nextInt(height);
    int xl = random.nextInt(13);
    int yl = random.nextInt(15);
    g.drawLine(x, y, x + xl, y + yl);
}

/*
* 获取随机的字符
*/
public String getRandomString(int num) {
    return String.valueOf(randString.charAt(num));
}
}
```

五、实验结果

1. 前端用户登录界面



用户名或密码错误!

LOGIN

用户名

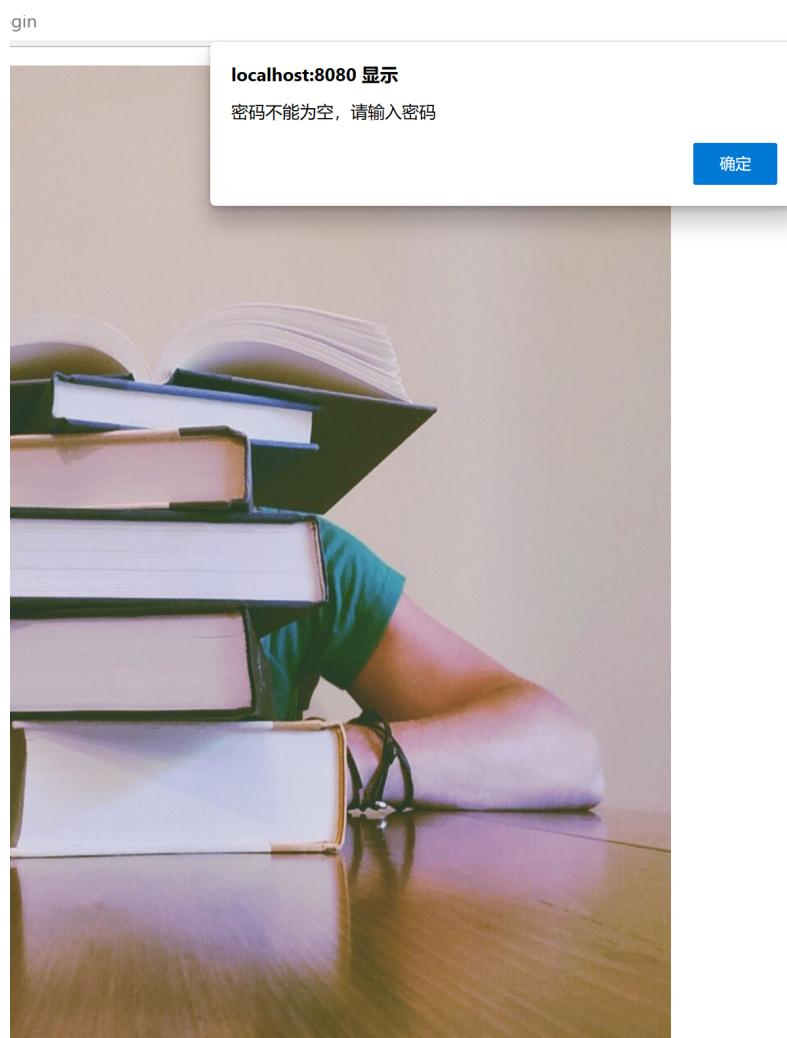
密码

请输入验证码



登录

2.用户非法登入



用户名或密码错误!

LOGIN

001

密码

请输入验证码



登录

3.用户登陆成功，转向课程管理主界面，右上角显示用户邮箱和一个“退出”按钮（或超链接）。

- 您好:llh
- [退出](#)
- [课程信息](#)

六、实验心得

进一步掌握了 SpringMVC 的数据绑定和页面跳转技术，能利用 JSP 视图技术来完成实际系统的开发，在做实验过程中通过上网查资料修改，学会了把生成验证码图片功能加入登录界面，将其与用户输入的验证码比较，提高了登录注册的安全性，整个实验过程让我对 ssm 的整合有了更深的理解，学会将业务逻辑、数据访问、控制层分离，会用注解、XML 配置完成事务管理。