南京航空航天大学

软件设计模式与体系结构

课程报告

姓名：刘浩

班级：1620304

学号：162030227

专业：软件工程

一．实验目的

理解框架的概念，理解配置在企业级开发中的重要性，了解 Spring 框架的基本结构，特点，并以 Spring 框架为基础，构件一个简单的 web 应用系统，包含基本的用户管理，界面展示功能，体现 MVC 特征，领会多种设计模式的复合使用，重点是观察者模式在 MVC 中的使用。

下载 jdk1.8，IntelliJ IDEA，VSCode 等相关软件及数据库

二．Spring 框架架构及 MVC 特点

1. Spring 框架架构、

springmvc 是 spring 框架的一个模块，springmvc 和 spring 无需通过中间整合层进行整合。springmvc 是一个基于 mvc 的 web 框架，而 mvc 是一个设计模式。图 1 为 springmvc 框架架构的原理分析:

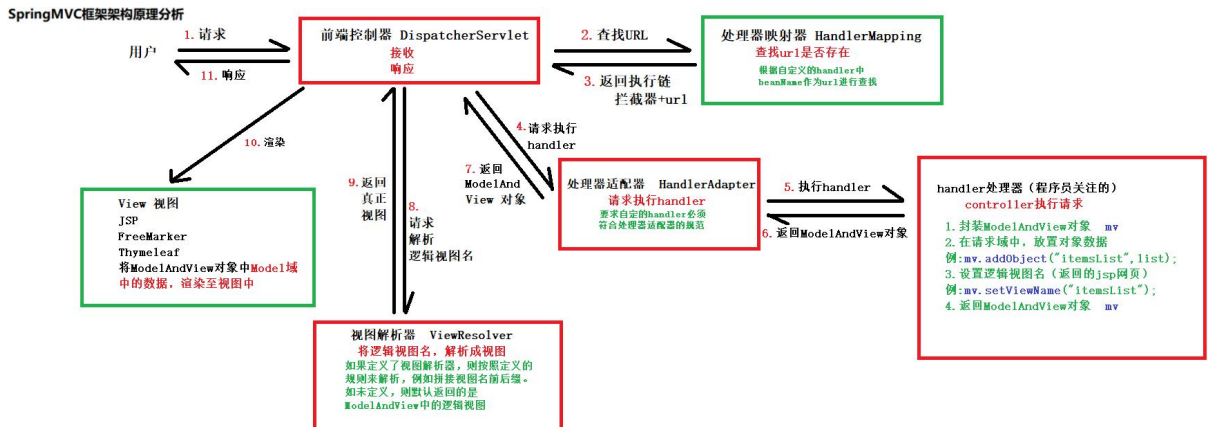


图 1 springmvc 框架架构原理分析

第一步：发起请求到前端控制器（DispatcherServlet）

第二步：前端控制器请求 HandlerMapping 查找 Handler，可以通过 xml 配置和注解进行查找

第三步：处理器映射器 HandlerMapping 向前端控制器返回 Handler

第四步：前端控制器调用处理器适配器去执行 Handler

第五步：处理器适配器去执行 Handler

第六步：Handler 执行完成给适配器返回 ModelAndView

第七步：处理器适配器向前端控制器返回 ModelAndView。ModelAndView 是 springmvc 提供的一个底层对象，包括 Model 和 View

第八步：前端控制器请求视图解析器去进行视图解析。根据逻辑视图名解析成真正的视图

第九步：视图解析器向前端控制器返回 View

第十步：前端控制器进行视图渲染。视图渲染将模型数据填充到 request 域

第十一步：前端控制器向用户响应结果

2. Mvc 及其特点

java SpringMVC 的工程结构一般来说分为三层，自下而上是 Modle 层（模型，数据访问层）、Cotroller 层（控制，逻辑控制层）、View 层（视图，页面显示层），其中 Modle 层分为两层：dao 层、service 层，MVC 架构分层的主要作用是解耦。采用分层架构的好处，普遍接受的是系统分层有利于系统的维护，系统的扩展。就是增强系统的可维护性和可扩展性。

对于 Spring 这样的框架，（View\Web）表示层调用控制层（Controller），控制层调用业务层（Service），业务层调用数据访问层（Dao）。图 2 即为三层架构的具体实现过程：

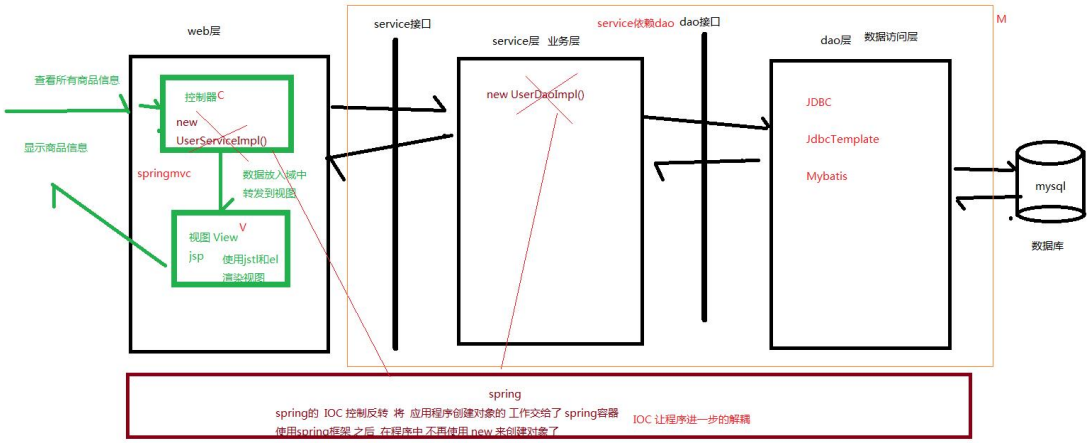


图 2　三层架构图

service 层：业务层，用来实现业务逻辑。能调用 dao 层或者 service 层，返回数据对象 DO 或者业务对象 BO，BO 通常由 DO 转化、整合而来，可以包含多个 DO 的属性，也可以是只包含一个 DO 的部分属性。通常为了简便，如果无需转化，service 也可以直接返回 DO。外部调用（HTTP、RPC）方法也在这一层，对于外部调用来说，service 一般会将外部调用返回的 DTO 转化为 BO。是专注业务逻辑，对于其中需要的数据库操作，都通过 Dao 去实现。主要去负责一些业务处理，比如取得连接、关闭数据库连接、事务回滚，一些复杂的逻辑业务处理就放到 service 层。

DAO 层：负责访问数据库进行数据的操作，取得结果集，之后将结果集中的数据取出封装到 VO 类对象之后返回给 service 层。数据层，直接进行数据库的读写操作，返回数据对象 DO，DO 与数据库表一一对应。Dao 的作

用是封装对数据库的访问：增删改查，不涉及业务逻辑，只是达到按某个条件获得指定数据的要求。

Cotroller 层：叫做控制层，主要的功能是处理用户发送的请求。主要处理外部请求。调用 service 层，将 service 层返回的 BO/DO 转化为 DTO/VO 并封装成统一返回对象返回给调用方。如果返回数据用于前端模版渲染则返回 VO，否则一般返回 DTO。不论是 DTO 还是 VO，一般都会对 BO/DO 中的数据进行一些转化和整合。

View 层：叫做显示层，主要是负责现实数据。

在实际开发中 dao 层要先定义出自己的操作标准即标准接口，就是为了解耦合。

3. SpringMvc 的优点

(1) 基于 MVC 架构，功能分工明确。解耦合。

(2) 容易理解，上手快；使用简单。就可以开发一个注解的 SpringMVC 项目，SpringMVC 也是轻量级的，jar 很小。不依赖的特定的接口和类。

(3) 作为 Spring 框架一部分，能够使用 Spring 的 IoC 和 Aop。方便整合 Strtus，MyBatis，Hiberate, JPA 等其他框架。

(4) SpringMVC 强化注解的使用，在控制器，Service，Dao 都可以使用注解。方便灵活。使用@Controller 创建处理器对象,@Service 创建业务对象，@Autowired 或者@Resource 在控制器类中注入 Service, Service 类中注入 Dao。

三．简单实例应用开发介绍

1. 需求分析:

随着高校的扩招，需要处理的学生信息日趋加大，不仅花费大量的教师资源，处理的效率还十分低下。为提高学生管理的管理水平，优化资源，尽可能降低管理成本成为学生管理的新课题，学生管理系统是从学生管理现状触发，根据学生管理的新要求进行开发设计的，它解决了学生管理数据信息量大，修改不方便，对一系列数据进行分析时花费时间长等问题，帮助学生管理人员有效管理学生信息，成为管理高校中必不可少的管理工具。

2. 功能:

（1）注册：用户注册信息，输入包括用户名，账号和密码

（2）登录：注册后可直接登陆，通过用户名和密码进行登陆

（3）重新登陆：在进入管理界面后，点击右上角的重新登陆可回到登陆界面

（4）学生管理：可以实现新增学生，查看全部学生，修改学生信息，删除学生，以及搜索学生的功能

（5）显示时间：在网页上显示当时的系统时间，年月日形式

3. 项目设计

（1） 框架

使用了 springboot 框架，前端代码使用 freemarker+jQuery+css 实现，后端则采用 java 实现，数据库采用 mysql。

（2） 业务流程

首先管理员进行登录输入个人的账号信息，系统将用户或系统管理员输入的信息和后台数据库作比对，如果不合法提示用户名或密码错误，需要重新登录，若合法进入系统界面。同时如果没有账号，可以在当前页面进行注册，注册时需输入账号，用户名，密码，同时和后台数据库进行比对，若没用相同的账号即创建成功，回到刚才界面。进入主界面后，可以通过左边的标题进入不同的页面设置，比如首页，删除学生，查询学生等。

（3） 数据库设计

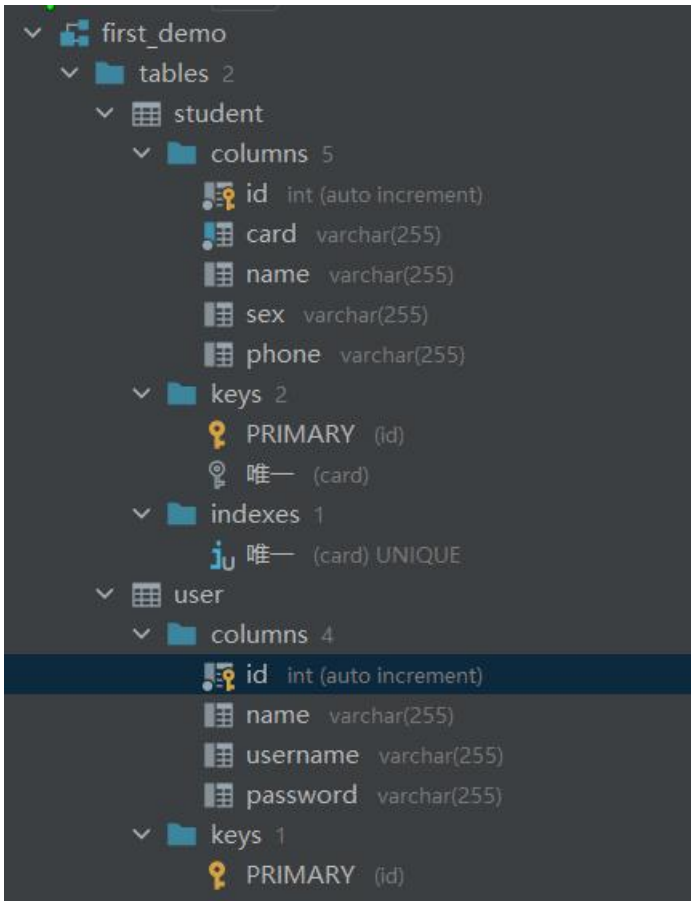


表 1 管理员表

列名	类型	KEY	可否为空	注释
id	int	PRI	否	管理员 id
admin	varchar(50)		否	管理员账号
password	varchar(50)		否	管理员密码
realname	varchar(50)		否	管理员性别
phone	varchar(50)		否	管理员电话

表 2 学生用户表

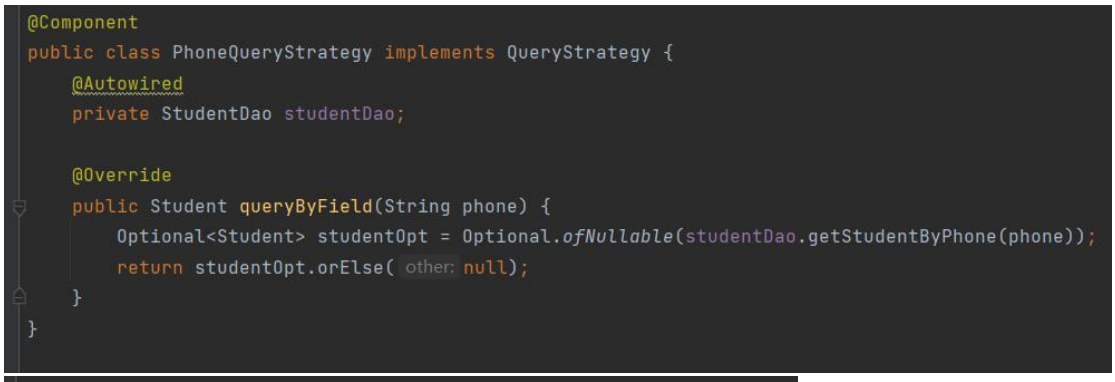
列名	类型	KEY	可否为空	注释
id	int	PRI	否	学生 id
user	varchar(50)		否	学生学号
password	varchar(50)		否	学生密码
phone	varchar(50)		否	学生电话

4. 设计模式:

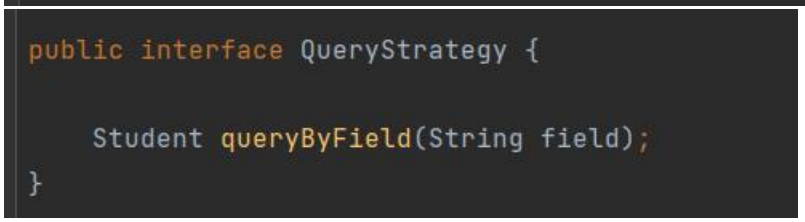
（1） 策略模式的代码分析

```java
@Component
public class CardQueryStrategy implements QueryStrategy {

    @Autowired
    private StudentDao studentDao;

    @Override
    public Student queryByField(String card) {

        Optional<Student> cardOpt = Optional.ofNullable(studentDao.getByCard(card));
        return cardOpt.orElse( other: null);
    }
}
```

```java
@Component
public class NameQueryStrategy implements QueryStrategy {

    @Autowired
    private StudentDao studentDao;

    @Override
    public Student queryByField(String name) {
        Optional<Student> studentOpt = Optional.ofNullable(studentDao.getStudentByName(name));
        return studentOpt.orElse( other: null);
    }
}
```



```java
@Component
public class PhoneQueryStrategy implements QueryStrategy {
    @Autowired
    private StudentDao studentDao;

    @Override
    public Student queryByField(String phone) {
        Optional<Student> studentOpt = Optional.ofNullable(studentDao.getStudentByPhone(phone));
        return studentOpt.orElse( other: null);
    }
}
```



```java
public interface QueryStrategy {

    Student queryByField(String field);
}
```

使用场景：此项目在查询学生信息详情处，根据用户选择不同选项(学号,姓名,手机号)查询对应的学生详情信息，出于后期扩展出发点，可能还会有别的查询项，采用策略模式是比较好的出发点。

代码实现方式：此处定义一个 QueryStrategy 的策略查询接口，里面定义了一个 queryByField()方法，方式实现 QueryStrategy 接口的类去实现具体的查询业务方法，列如 CardQueryStrategy 策略类根据学号查询的学生信息，NameQueryStrategy 策略类根据姓名查询学生信息，PhoneQueryStrategy 策略类根据手机号查询学习信息。此处的使用的策略模式管理是借助 spring 的 @Autowired Map<String, QueryStrategy> 该 Map 集合 key 值是对应的策略列如

(phoneQueryStrategy,nameQueryStrategy,cardQueryStrategy)根据接口的传入的对应策略，获取 Map 对应的策略方法去查询学生信息

（2） 单例模式代码分析

```
public class TimeKit {

    1 usage
    private final String DATA_FORMAT_YYYY_MM_DD_HH_MM_SS = "yyyy-MM-dd HH:mm:ss";

    1 usage
    private final String DATA_FORMAT_YYYY_MM_DD = "yyyy-MM-dd";

    1 usage
    private TimeKit() {
    }

    2 usages
    public static TimeKit getInstance() {
        return new TimeKit();
    }
```

```
public class TimeKit {
    private final String DATA_FORMAT_YYYY_MM_DD_HH_MM_SS = "yyyy-MM-dd HH:mm:ss";
    private final String DATA_FORMAT_YYYY_MM_DD = "yyyy-MM-dd";
    private TimeKit() {
    }
    public static TimeKit getInstance() { return new TimeKit(); }

    //年月日时分秒 转成 年月日
    public LocalDate transition(LocalDateTime time) { return time.toLocalDate(); }

    //年月日时分秒转字符串
    public String transitionStr(LocalDateTime time) {
        if (null == time) {
            return "";
        } else {
            return time.format(DateTimeFormatter.ofPattern(DATA_FORMAT_YYYY_MM_DD_HH_MM_SS));
        }
    }

    //年月日转字符串
    public String transitionStr(LocalDate time) {
        if (null == time) {
            return "";
        } else {
            return time.format(DateTimeFormatter.ofPattern(DATA_FORMAT_YYYY_MM_DD));
        }
    }
}
```

项目里面时间处理类这边采用了单例设计模式，保障了项目里面使用的时间处理对象 TimeKit 获取的是同一份对象。代码实现方式，将 TimeKit 的构造器私有化，对外提供一个静态的方法 getInstance 方法，获取到真实的 TimeKit 对象。

（3） mvc 模式

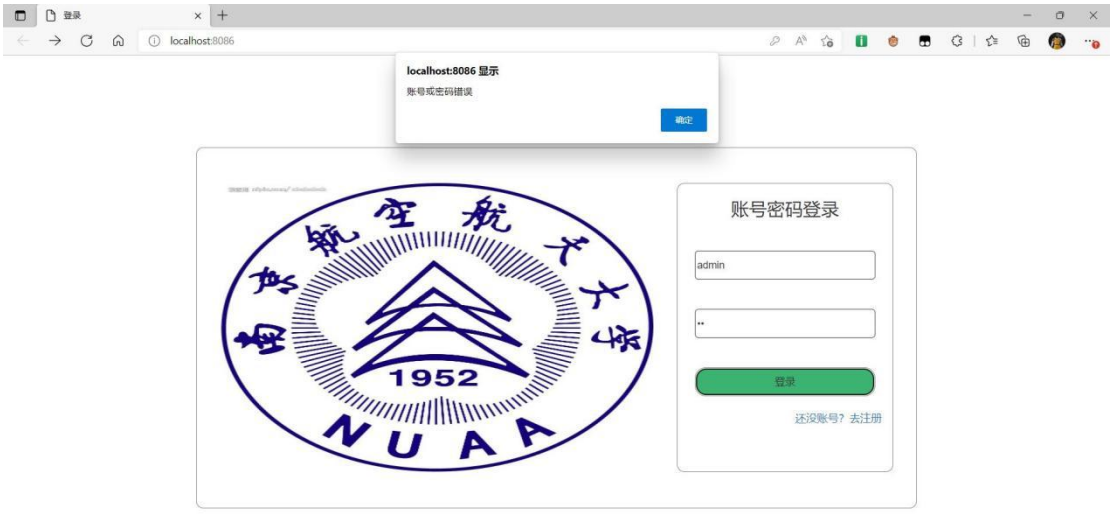
这是由 springmvc 框架自身体现的特点。

4. 功能实现即截图：


全部学生信息
localhost:8086/message
欢迎访问学生管理系统
欢迎您，admin 重新登录
当前系统日期: 2022-06-25
个人中心
首页
学生信息
人员管理
学生注册
删除学生
修改学生
学生评情
学生信息
学号 姓名 性别 联系电话 操作
162030227 刘洁 1 1 删除
162030226 程建 男 151 删除


首页
localhost:8086/notifications
欢迎访问学生管理系统
欢迎您，admin 重新登录
当前系统日期: 2022-06-25
个人中心
首页
学生信息
人员管理
学生注册
删除学生
修改学生
学生评情


管理员注册
localhost:8086/register
学生管理系统
1952
NUAA
注册账号
用户名
帐号
密码
注册
已有帐号？点此登录



四．心得体会

在做这一次的 java 设计模式课程设计之前，我既没有使用过 java 进行编程，也没有学习过前端的相关知识，整个人就是一个小白。得知这个题目之后，根本不知道从哪开始，先干什么，要干什么，怎么干。不得已，只能去 b 站上面找一些视频进行学习，开始想先看 springmvc 的相关内容，发现没有 java 的基础学这个和看天书一样，所以买了本 java 编程相关的书籍想着能够速成一下。学了几个星期学会了使用 java 之后再去学习 springmvc 相关的知识，发现没有 web

的经验还是不会写程序，而且没有学过前端，更是前进路上的绊脚石。只能在 b 站上面先找一些具体项目实践的视频同时网上搜索一些项目的源码，研究一下那些博主是怎样搭建这样的一个项目的，一步步跟着他们实现一个小项目，再去类比到自己要实现的功能上面。最后也是经过好久的努力完成了本次 java 设计模式，在这次课程设计中，我真的付出了我自己好多血和泪，多少个深夜在电脑前还在学习相关知识。也算是从小白加上了一些斑点吧，了解了 web 服务基本方法，还有前后端的一些用法收获真的很多。