**南京航空航天大学**

**软件设计模式与体**

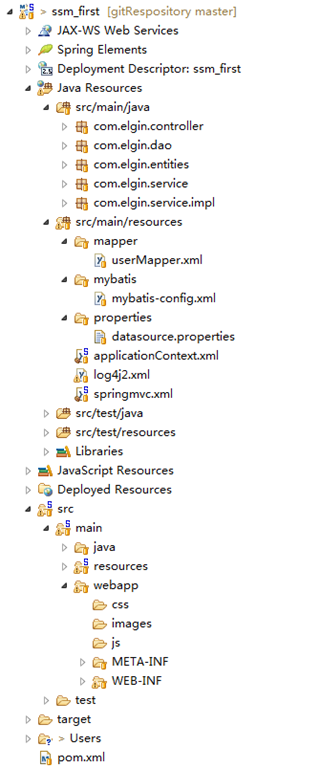
**系结构课程设计**

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
| 学生姓名 | 赵维康 |
| 学 号 | 161630220 |
| 学 院 | 计算机科学与技术学院 |
| 专 业 | 软件工程 |

二〇一八年六月

1. **spring框架架构：整个项目的框架架构如图**



**目前项目开发的框架主要是 spring ，最近学习了一下Mybatis、Maven这些内容，所以尝试使用 springMVC + Spring + Mybatis + Maven来搭建一个简单的demo ，功能很简单，主要是对于一个表的CRUD。**

1. **MVC的特点：**

**1.多个视图可以对应一个模型。按MVC设计模式，一个模型对应多个视图，可以减少代码的维护量，一旦模型发生改变，也易于维护。**

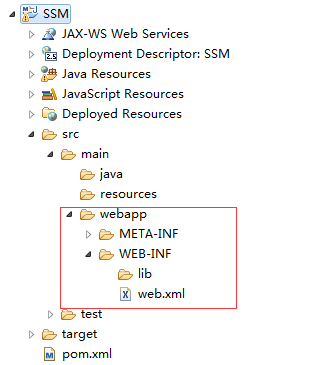
**2.模型返回的数据与显示逻辑分离。模型数据可以应用任何的显示技术，例如，JSP，FreeMarker，Velocity。**

**3.应用被分隔为三层，降低了各层之间的耦合，提供了应用的可扩展性。**

**4.控制层的概念也很有效，由于它把不同的模型和不同的视图组合在一起，完成不同的请求，因此，控制层可以说是包含了用户请求权限的概念。**

**5.MVC更符合软件工程化管理的精神。不同的层各司其职，每一层的组件具有相同的特征，有利于通过工程化和工具化产生管理程序代码。**

1. **一个简单的实例应用开发介绍**
2. **新建一个Maven工程，并改项目为WEB项目，创建出相应的项目结构，如图**



1. **配置相关的配置文件（注：本实验报告的代码只是部分截图，完整代码见项目工程文件）**
2. **配置Maven的pom.xml文件，**

**2）、配置 web.xml 文件：**

****

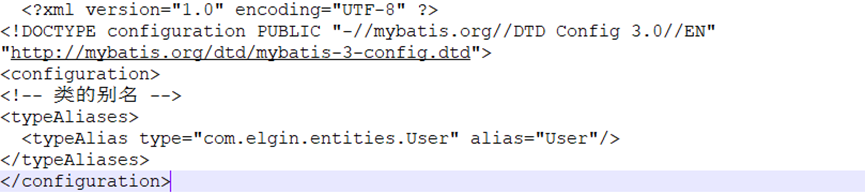
**3）配置spring的配置文件applicationContext.xml**

****

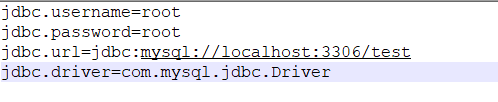
**4）配置springMVC的配置文件：**

****

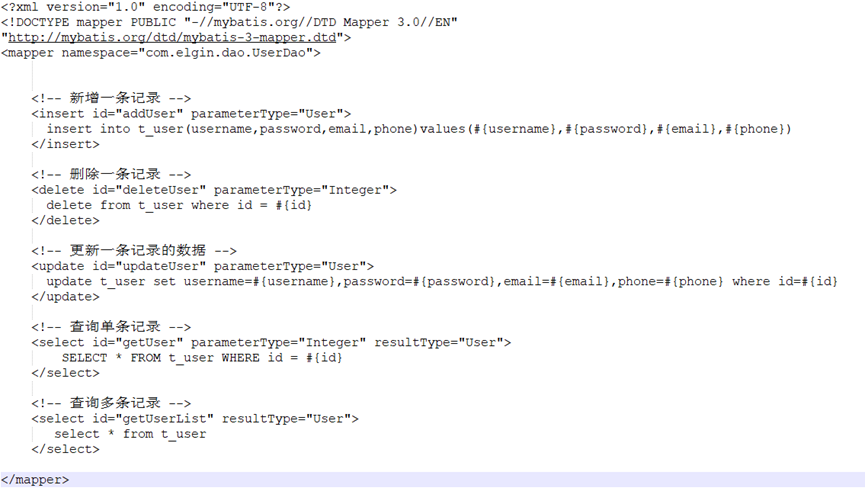
**5）配置mybatis的配置文件：在源码包src/main/resources下建package：mybatis ，然后在mybatis包下新建mybatis的配置文件：mybatis-config.xml, 内容如下**

****

**6）数据库相关配置：在源码包src/main/resourecs下新建package：properties ，然后在properties包下新建：datasource.properties ，内容如下**



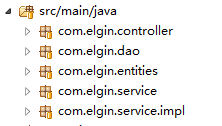
**7）、mybatis的dao接口映射文件：在源码包src/main/resourecs下新建package：mapper，然后在mapper包下新建：userMapper.xml，内容如下**

****

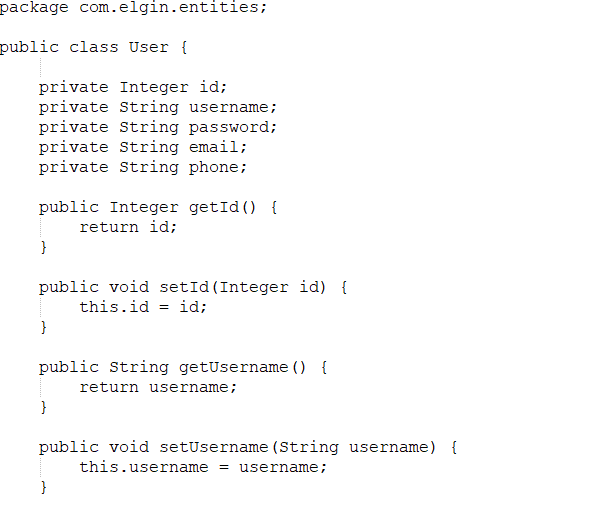
**8）配置log4j2配置文件：在源码包src/main/resourecs下新建xml配置文件：log4j2.xml ,内容如下**

****

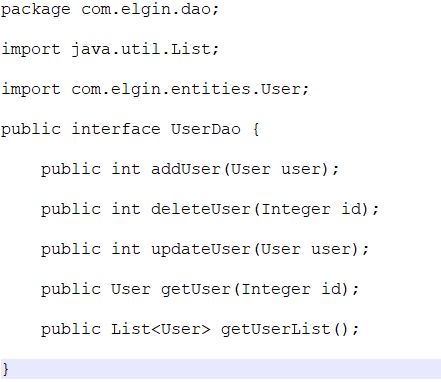
**9）开发dao，service，controller，实体entitis相关代码：在源码包src/main/java中新建如下各个包**



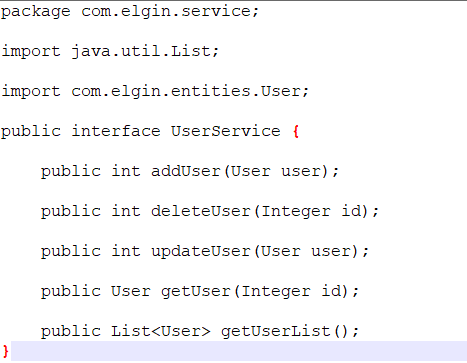
**实体类User：**



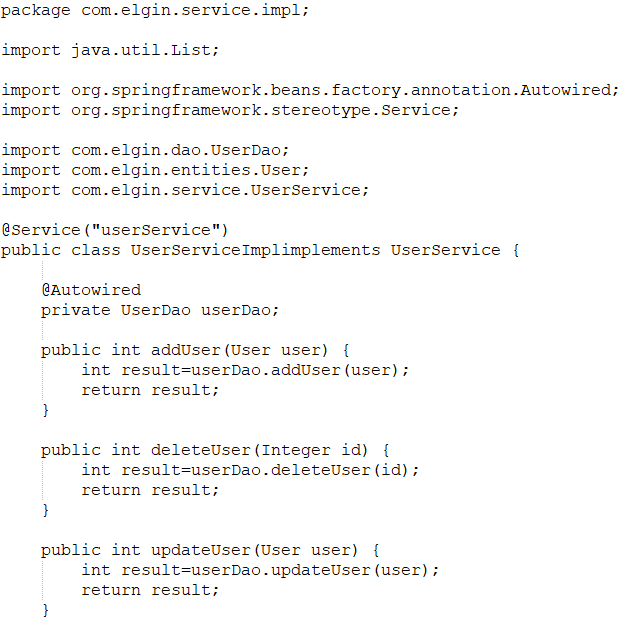
**dao接口类：UserDao**



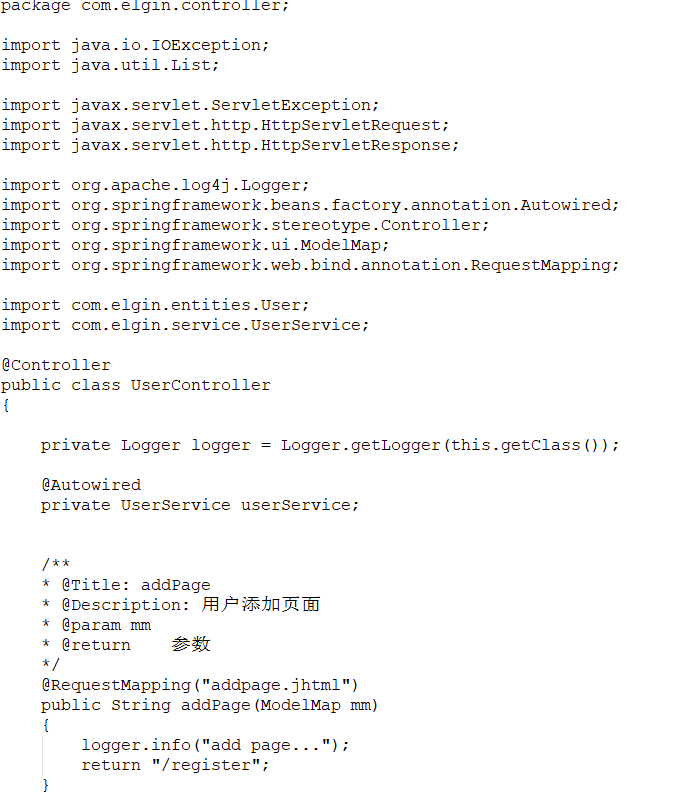
**service接口类: UserService**



**service实现类: UserServiceImpl**



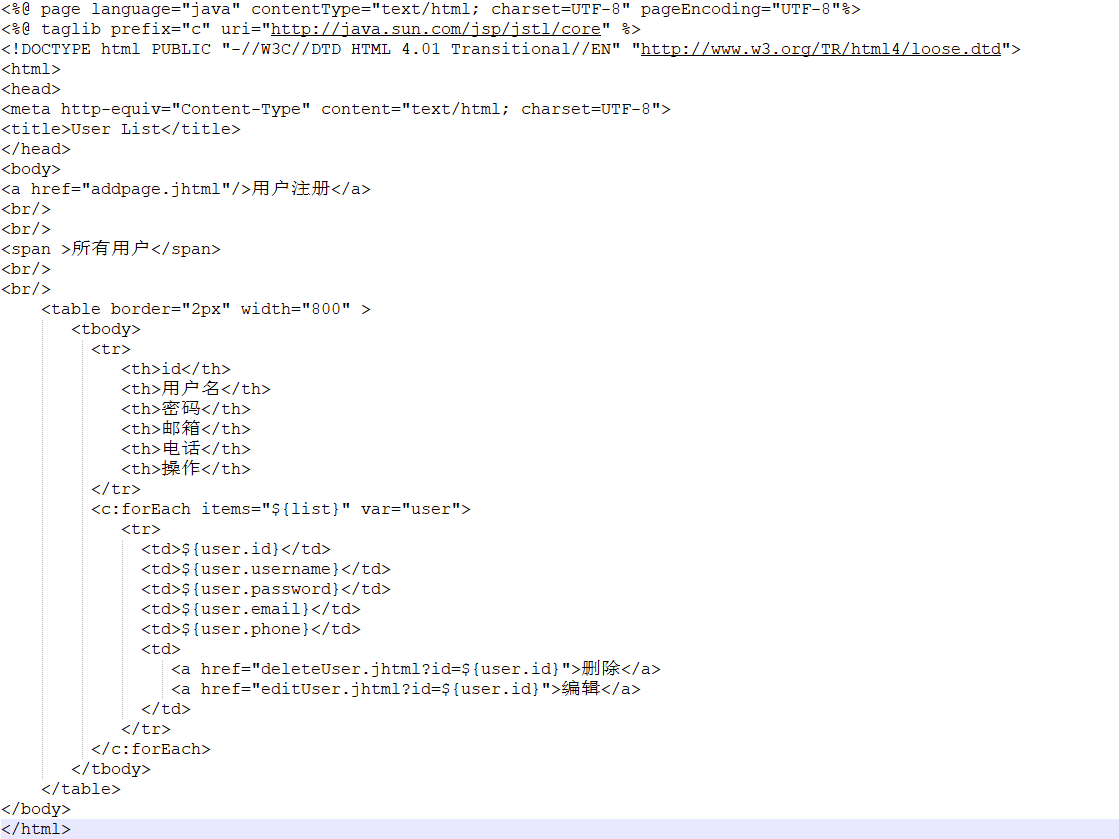
**controller类: UserController**



**10）测试页面：根据视图解析器的配置，在webapp/WEB-INF下新建jsp文件夹，用来保存相关页面，register.jsp的内容如图**

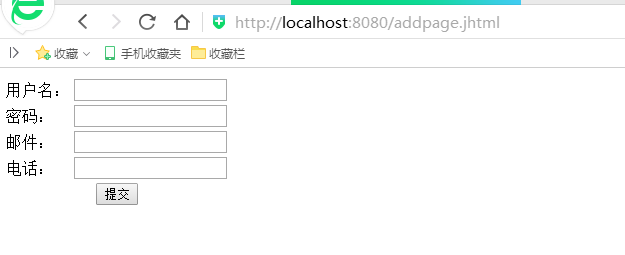


**userlist.jsp :**



**3、测试结果：将项目加载到tomcat，并启动。**

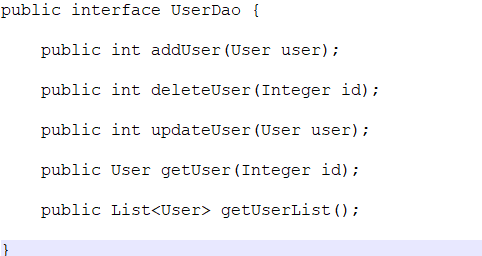
**访问：http://localhost:8080/addpage.jhtml ，进入编辑页面**



**输入数据点提交，数据保存成功并出现用户列表页面（编辑删除同理）：**



1. **相关设计模式的使用及对MVC的认识**
2. **观察者模式：在dao的接口类UserDao中，如图**



**如果修改了updateUser()中的内容，那么getUser()中的内容相应的也会发生改变，很好的体现了观察者模式。观察者模式用于定义对象间的一种依赖关系，当一个对象的状态发生变化时，所有依赖于它的对象都得到通知并被自动更新。**

1. **桥梁模式：service的接口类UserService与service的实现类UserServiceImpl就是桥梁模式很好的体现。service的接口类UserService与service的实现类UserServiceImpl将抽象部分与它的实现部分分离，使它们都可以独立的变化。这样做的好处是：****①当我们不希望在抽象和它的实现部分之间有一个固定的绑定关系；②类的抽象以及它的实现都应该可以通过生成子类的方法加以扩充；③对一个抽象的实现部分的修改应对客户不产生影响，即客户的代码不必重新编译**
2. **装饰模式：controller的类UserController中，设计了用户添加页面、用户添加操作、列表显示所有用户、用户信息编辑页面、更新用户信息、删除一条用户记录等功能就是装饰模式很好的体现。所谓装饰模式就是动态地给一些对象添加一些职责。就增加功能来说，装饰模式相比生成子类更为灵活。**
3. **工厂方法模式：定义的dao接口类UserDao和service接口类UserService就是工厂方法模式很好的体现。所谓工厂方法模式，是指定义一个用于创建对象的接口，让子类决定实例化哪一个类。工厂方法模式使一个类的实例化延迟到其子类。**
4. **对MVC的认识：MVC(Model-View-Controller)的核心思想和 Observer 模式是完全类似的。数据模型 （业务逻辑）和表现逻辑相互隔离，Controller 则负责与客户交互，根据请求调 用 M 和 V 的功能。通常情况下 Controller 并不单独存在，而是和 M 或 V 结合（JFC 中便是如此）。但在分布式系统中，客户端不可能直接与 M 或 V 进行交互，此时 Controller 将独立出来并发挥重要作用。访问协议、请求分发、安全认证、日志 记录和异常处理等都由其承担。总体上看，Observer 和 MVC 在本质上并没有太多的区别，只是应用范围的不同引起差异。MVC 通常会用在分布式系统中，而交互式 的应用程序一般使用 Observer。**