SSM – Spring + Spring MVC + MyBatis

主要模块

持久层 ： DAO层 （mapper）

主要是做数据持久层的工作，负责与数据库进行联络的一些任务都封装在此。

步骤：

1. 设计DAO的接口
2. 在spring 的配置文件中定义此接口的实现类
3. 在模块中调用此接口来进行数据业务的处理
4. 在spring的配置文件中配置DAO层的数据源配置以及有关数据库连接的参数

业务层： Service层

主要是负责业务模块的逻辑应用设计

步骤：

1. 设计接口，再设计其实现的类
2. 再spring的配置文件中配置其实现的关联
3. 依靠调用已定义的DAO层的接口来实现业务逻辑

表现层： Controller层（Handler层）

负责具体的业务模块流程的控制

在此层里面要调用Service层的接口来控制业务流程

控制的配置是在spring的配置文件里面进行，针对具体的业务流程，会有不同的控制器

View层

该层与控制层结合比较紧密。主要负责前台jsp页面的表示。

整合

1. DAO层
   1. MyBatis配置文件 sqlMapConfig.xml：

配置别名（不要求）

* 1. Spring配置文件 ApplicationContext.xml：

<context:component-scan

<dataSource

<SqlSessionFactory

<MapperScannerConfigurer

* 1. 持久化类
  2. DAO层方法接口
  3. 映射文件mapper.xml

对应接口

对应持久化类

对应方法的sql语句

1. service层
   1. Service层方法接口
   2. Service层实现类
2. Controller 层
   1. Controller类
   2. Spring-mvc配置文件
      1. 扫描控制器

<context:component-scan

* + 1. 视图解析器

InternalResourceViewResolver

* + 1. 静态资源处理（可能）

<mvc:default-servlet-handler

1. web.xml
   1. 整合spring配置文件

<context-param> ContextConfigLocation

* 1. 监听器

<listener> ContextLoaderListener

* 1. 过滤器
  2. 整合SpringMVC配置文件

<servlet> DispatcherServlet

步骤

1. 完成用户登录
2. 为数据库表t\_user录入用户数据
3. 配置ssm框架
4. 配置实体类UserEntity,映射文件UserMapper,服务层UserService,UserServiceImpl
5. 配置控制器UserController
6. 编辑登录页面login.jsp,辅助js login.js

具体措施

1. 在maven的pom.xml中配置需要的包（百度springmvc-mybatis整合需要包）
2. 配置目录结构
3. 配置applicationContext文件，放在resources资源文件中,

修改数据库地址

1. 配置日志文件log4j.xml
2. 配置spring-mvc.xml
3. 在数据库创建添加表数据
4. 创建实体类

UserEntity

RoleEntity

UserRoleEntity

PermissionEntity

RolePermissionEntity

OperationEntity

PermissionOperationEntity

1. 创建LoginMapper.xml文件，编写实体类的resultMap
2. 编写LoginMapper接口，需要在applicationContext中的MapperScannerConfigurer中加入路径，定义方法：

获取系统角色

登录验证

获取权限及操作

在执行sql语句时。确保语句正确

1. 编写LoginService接口及其实现类LoginServiceImpl，添加注解@Service，
2. 编写LoginController
   1. @Controller
   2. @RequestMapper
   3. 方法返回值
   4. @RequestParam
   5. 参数传递
   6. 页面跳转
      1. 在执行页面跳转时，路径应全为小写
      2. 页面传值要和控制器接受参数一致，除非添加参数unrequired
      3. 路径一致
3. 编辑登录页面login.jsp

处理静态资源

编辑页面样式login.css

编辑脚本login.js

1. 设计管理页面main.jsp
2. 完成忘记密码操作
3. 转至填写邮箱页面，输入你的注册邮箱。
4. 提交后验证该邮箱是否存在，不存在返回错误信息，存在则向该邮箱发送验证码
5. 验证码页面，把发送的验证码存入页面，用户并输入验证码，通过对比，判断是否一致，一致则提交至后台
6. 跳转至密码修改页面，提交至后台修改

结果：邮箱被网易拉黑了

1. 完成后台菜单栏
2. 配置MainMapper接口，及MainMapper.xml配置文件，SQL语句查询所有权限及其附属操作
3. 编写Main.jsp页面，设置子窗口，并将菜单栏的连接转至js操作。
4. 用户修改密码
5. 编写toUpdatePassword.jsp页面
6. 编写修改密码的方法
7. 编写结果页面，通过子窗口调用父窗口方法实现退出
8. 学生管理
9. 完成数据库表搭建和数据填充
10. 编写实体类
11. 添加学生资料
12. 点击主页面的菜单栏中的添加学生资料
13. 添加学生资料页面toAddOneStuData.jsp,编写学生资料应包含的属性，并提交至添加页面
14. 将学生数据提交至数据库，并返回主键值；为该学生绑定用户，绑定角色；最后将在数据库存储的数据传至页面
15. 编辑oneStuData.jsp
16. 批量添加学生资料
17. 导入需要的包：poi.jar，io.jar
18. 点击主页面的菜单栏中的批量学生资料，转至添加学生页面
19. 批量添加学生资料就是通过读取存有学生资料的excel文件，文件里面的格式需要为特定格式的，符合读取规则。将读取的属性存入对象，然后批量存储.

关于文件中的属性值，存储的是id

1. 编写读取excel的类 ExcelReader.class
2. 编辑获取文件的页面toAddBatchStuData.jsp
3. 将获取的文件通过读取类读取数据ExcelReader.readExcel(file)
4. 将数据存入数据库，批量存储addBatchStuData
5. 通过数据读取数据库的数据。通过循环读取单个数据，返回list
6. 将读取的数据显示在页面 readBatchStuData.jsp
7. 学生资料统计显示
8. 点击菜单栏中的批量查看学生资料 QueryBatchStuByData
9. 返回所有条件，返回当前条件，从数据库中读取资料，编辑动态sql语句查询数据库数据
10. 将数据传入显示页面readBatchStuData.jsp
11. 查看当前学生资料

通过id查询学生资料，将数据显示至页面

1. 修改当前学生资料

通过id查询学生资料，将所有条件传入显示页面

将条件修改为当前学生资料拥有的

通过提交资料。进行更新，返回修改后的资料显示页面

1. 删除当前学生资料

通过id将删除表示改为删除状态

在页面上添加按钮：

批量添加，单个添加，根据id查询，根据名称查询（返回list）

1. 权限管理

（一） 用户管理

1. 查看所有用户
2. 添加用户
3. 用户授权：为用户添加角色
4. 用户删除
5. 用户真正删除（需要？？？）

（二） 角色管理

1. 查看所有角色
2. 编辑角色名
3. 角色授权：为角色添加权限
4. 删除角色

（三） 权限管理

1. 查看所有权限

2. 编辑权限名

3. 权限授权：为权限添加操作

4. 删除权限

（四）操作管理

1. 查看所有操作

2. 编辑操作名

3. 删除操作