## dao层实现

1. 在任务二创建的dao层接口MallUserMapper接口中，新增方法：根据用户名和密码查询一个用户

MallUser selectByLoginNameAndPasswd(@Param("loginName") String loginName, @Param("password") String password);

1. 在resources-》mapper中的MallUserMapper.xml 实现上一步声明的接口方法

<select id="selectByLoginNameAndPasswd" resultMap="BaseResultMap">  
 select  
 <include refid="Base\_Column\_List"/>  
 from tb\_qst\_mall\_user  
 where login\_name = #{loginName} and password\_md5 = #{password} and is\_deleted = 0  
</select>

## Service层实现

1. 在任务二中创建service层接口，QSTMallUserService接口中，新增方法：

String login(String loginName, String passwordMD5, HttpSession httpSession);

1. 在QSTMallUserService实现类QSTMallUserServiceImpl中实现上一步创建的方法

@Override  
public String login(String loginName, String passwordMD5, HttpSession httpSession) {  
 MallUser user = mallUserMapper.selectByLoginNameAndPasswd(loginName, passwordMD5);  
 if (user != null && httpSession != null) {  
 if (user.getLockedFlag() == 1) {  
 return ServiceResultEnum.*LOGIN\_USER\_LOCKED*.getResult();  
 }  
 QstMallUserVO qstMallUserVO = new QstMallUserVO();  
 BeanUtil.*copyProperties*(user, qstMallUserVO);  
 *//设置购物车中的数量* httpSession.setAttribute(Constants.*MALL\_USER\_SESSION\_KEY*, qstMallUserVO);  
 return ServiceResultEnum.*SUCCESS*.getResult();  
 }  
 return ServiceResultEnum.*LOGIN\_ERROR*.getResult();  
}

## Controller层实现

1. 在controller包中的MallUserMapper中新增处理请求方法：

@GetMapping({"/login", "login.html"})  
public String loginPage() {  
 return "mall/login";  
}  
  
@PostMapping("/login")  
@ResponseBody  
public Result login(@RequestParam("loginName") String loginName,  
 @RequestParam("password") String password,  
 HttpSession httpSession) {  
 *//todo 清verifyCode* String loginResult = qstMallUserService.login(loginName, MD5Util.*MD5Encode*(password), httpSession);  
 *//登录成功* if (ServiceResultEnum.*SUCCESS*.getResult().equals(loginResult)) {  
 return ResultGenerator.*genSuccessResult*();  
 }  
 *//登录失败* return ResultGenerator.*genFailResult*(loginResult);  
}

1. 在com.qst.qstmall.controller包中新建子包vo，在com.qst.qstmall.controller.vo包中存放业务层实体类
2. 在com.qst.qstmall.controller.vo中新建业务层用户实体类QSTMallUserVO.java

public class QstMallUserVO implements Serializable {  
 private Long userId;  
 private String nickName;  
 private String loginName;  
 private String introduceSign;  
 private String address;  
 private int shopCartItemCount;

。。。。省略get和set方法

## 视图层

在resources-》templates-》mall目录下新建登录页面login.html：

参考代码见github仓库：https://github.com/[Henry0227/Qstmall](https://github.com/Henry0227/Qstmall)

## 工具类

在utils包中新建实体类转换工具类BeanUtil.java，该类用于数据层和业务层实体类之间的转换

public abstract class BeanUtil {  
  
 public static Object copyProperties(Object source, Object target, String... ignoreProperties) {  
 if (source == null) {  
 return target;  
 }  
 BeanUtils.*copyProperties*(source, target, ignoreProperties);  
 return target;  
 }  
  
 public static <T> List<T> copyList(List sources, Class<T> clazz) {  
 return *copyList*(sources, clazz, null);  
 }  
  
 public static <T> List<T> copyList(List sources, Class<T> clazz, Callback<T> callback) {  
 List<T> targetList = new ArrayList<>();  
 if (sources != null) {  
 try {  
 for (Object source : sources) {  
 T target = clazz.newInstance();  
 *copyProperties*(source, target);  
 if (callback != null) {  
 callback.set(source, target);  
 }  
 targetList.add(target);  
 }  
 } catch (InstantiationException e) {  
 e.printStackTrace();  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 }  
 }  
 return targetList;  
 }  
  
 public static Map<String, Object> toMap(Object bean, String... ignoreProperties) {  
 Map<String, Object> map = new LinkedHashMap<>();  
 List<String> ignoreList = new ArrayList<>(Arrays.*asList*(ignoreProperties));  
 ignoreList.add("class");  
 BeanWrapper beanWrapper = PropertyAccessorFactory.*forBeanPropertyAccess*(bean);  
 for (PropertyDescriptor pd : beanWrapper.getPropertyDescriptors()) {  
 if (!ignoreList.contains(pd.getName()) && beanWrapper.isReadableProperty(pd.getName())) {  
 Object propertyValue = beanWrapper.getPropertyValue(pd.getName());  
 map.put(pd.getName(), propertyValue);  
 }  
 }  
 return map;  
 }  
  
 public static <T> T toBean(Map<String, Object> map, Class<T> beanType) {  
 BeanWrapper beanWrapper = new BeanWrapperImpl(beanType);  
 map.forEach((key, value) -> {  
 if (beanWrapper.isWritableProperty(key)) {  
 beanWrapper.setPropertyValue(key, value);  
 }  
 });  
 return (T) beanWrapper.getWrappedInstance();  
 }  
  
 public static interface Callback<T> {  
 void set(Object source, T target);  
 }  
  
 *//检查Pojo对象是否有null字段* public static boolean checkPojoNullField(Object o, Class<?> clz) {  
 try {  
 Field[] fields = clz.getDeclaredFields();  
 for (Field field : fields) {  
 field.setAccessible(true);  
 if (field.get(o) == null) {  
 return false;  
 }  
 }  
 if (clz.getSuperclass() != Object.class) {  
 return *checkPojoNullField*(o, clz.getSuperclass());  
 }  
 return true;  
 } catch (IllegalAccessException e) {  
 return false;  
 }  
 }  
}