Spring3MVC + MyBatis + ExtJs3整合开发系列

作者: KimHo http://kimho.iteye.com

我的博客文章精选

目 录

1. Spring3MVC

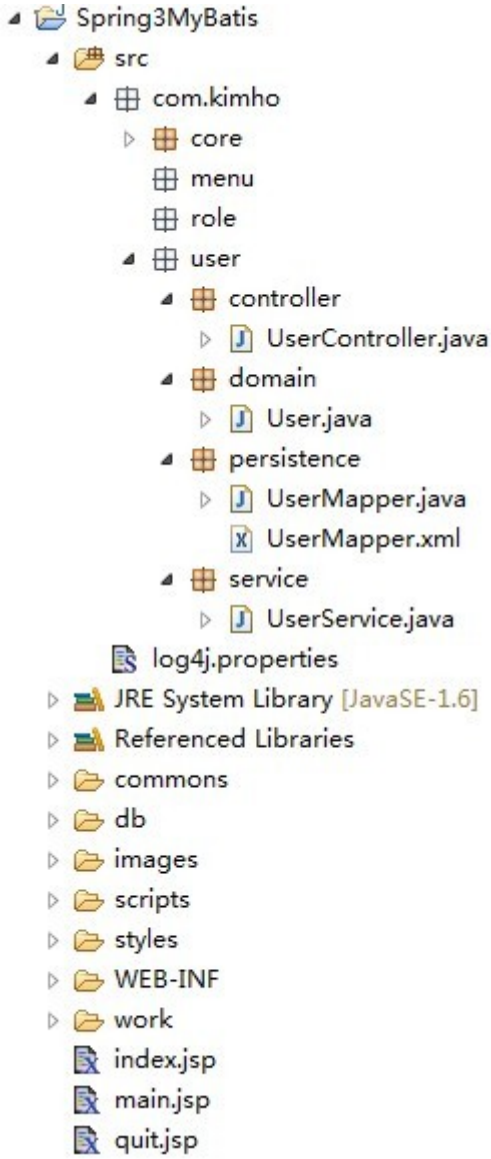
1.1 Spring3MVC+MyBatis+ExtJs3整合开发系列之一：登录模块演示

发表时间: 2011-08-02 关键字: Spring3MVC, MyBatis, ExtJs3

之前写过一篇Spring3MVC+Hibernate的整合，这套架构使用了一段时间后，发现自己对Hibernate的应用程度实在不如人意，看着项目中一堆的sql和hql，决定还是放弃hibernate，转投MyBatis，似乎更适合我这种喜欢灵活控制，习惯了写sql的我。

这次打算写一系列关于Spring3MVC+MyBatis+ExtJs3应用开发的文章，这篇算是开头篇吧，还是从咋们最熟悉的登录系统那一幕开始。

eclipse项目结构图：



简单描述下项目的结构：

代码那块:

controller：web层，用于服务客户端请求的服务端

domain：领域模型层，可以设计成简单的POJO风格

persistence：持久层，其中XXMapper.java为interface，XXMapper.xml为sql mapping配置，两者结合形成了我们传统意义上的dao

service：业务层，这块也是一个系统中最核心最重要的一层

配置那块：

root-context.xml：spring核心配置文件，这里配置了dataSource，sqlSessionFactory和transactionManager等关键组件

servlet-context.xml：springMVC核心配置文件

controllers.xml：springMVC业务组件配置文件

web.xml：这个没啥好说的了

jdbc.properties：数据库连接配置文件

页面那块：

index.jsp:登录界面

main.jsp：核心主框架页

quit.jsp：退出系统

业务核心js：\scripts\modules目录下的那堆js

这个项目目前完成度，仅仅是完成了登录功能，后续还有人员角色菜单管理等功能，就让我慢慢完善吧，有兴趣的coder，也可以自己尝试完善下系统的功能，然后发帖出来共享交流下。

注：在eclipse3.6+mysql5.5下开发。

下篇：Spring3MVC+MyBatis+ExtJs3整合开发系列之二：菜单模块演示

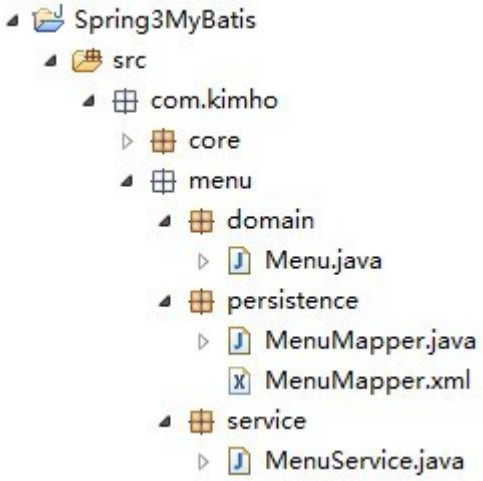
1.2 Spring3MVC+MyBatis+ExtJs3整合开发系列之二：菜单模块演示

发表时间: 2011-08-06 关键字: spring3mvc, mybatis, extjs3

我又回来了，带来了Spring3MVC+MyBatis+ExtJs3整合开发系列之第二篇：菜单模块演示。

承接上篇Spring3MVC+MyBatis+ExtJs3整合开发系列之一：登录模块演示，本篇增加了菜单加载的功能到本项目中。

菜单模块结构图：



domain层

```
public class Menu implements Serializable {
        private static final long serialVersionUID = -2726709540069876682L;


        private Long id;
        private Long parent_id;
        private String name;
        private String image;
        private String url;
        private String qtip;
        private Integer sortNum;
        private String description;
        /**
         * true:默认为叶子结点，即子菜单
         */
```

```
        private boolean leaf = true;
        private List<Menu> children;
```

其中，**leaf**用来标识主子菜单（同时也是为了配合前端extjs treePanel控件所需的标识），这里true表示为子菜单，false表示为主菜单;**children**用来存放主菜单下的所有子菜单。

service层

```
@Service
public class MenuService {
        @Autowired
        private MenuMapper menuMapper;

        @Transactional
        public List<Menu> getMenuListByUserId(Long userId) {
                Map<String,Object> param = new HashMap<String,Object>();
                List<Menu> mainMenuList = menuMapper.getMainMenuList(userId);
                Iterator<Menu> it = mainMenuList.iterator();
                //装载主菜单下所有的子菜单
                while(it.hasNext()) {
                        Menu menu = it.next();
                        //false:表示为主菜单
                        menu.setLeaf(false);
                        Long parentId = menu.getId();
                        param.put("userId", userId);
                        param.put("parentId", parentId);
                        List<Menu> subMenuList = menuMapper.getSubMenuList(param);
                        menu.setChildren(subMenuList);
                }
                return mainMenuList;
        }
}
```

简单说下两级菜单的加载原理：我这里只假设了只有主菜单和子菜单两级菜单（多级菜单的实现思想类似）。首先，根据用户的菜单权限，加载所有的主菜单，然后根据每个主菜单的菜单id去逐个获取该主菜单下的

所有子菜单，所有的子菜单加载完后那么也就完成了两级菜单的加载了。

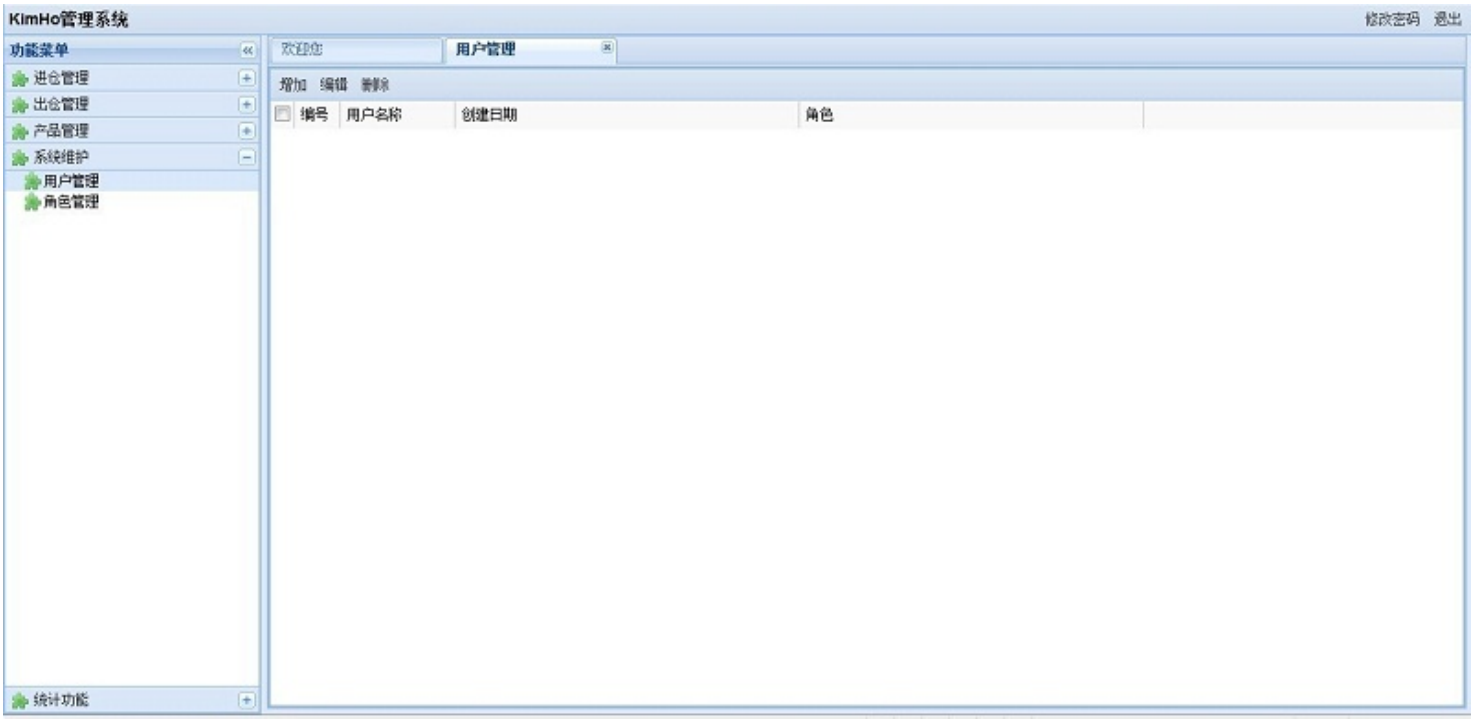
controller层

```
@Controller
@RequestMapping("/user")
public class UserController {
        @Autowired
        private UserService userService;
        @Autowired
        private MenuService menuService;
```

```
/**
         * 获取所有菜单
         * @param request
         * @param response
         * @return
         */
        @RequestMapping(value="/Menus",method=RequestMethod.POST)
        public @ResponseBody Map<String,Object> getTopMenus(HttpServletRequest request,
            HttpServletResponse response){
                Map<String,Object> result = new HashMap<String,Object>();
                User user = (User)request.getSession().getAttribute("user");
                List<Menu> list = menuService.getMenuListByUserId(user.getId());
                result.put("success", "true");
                result.put("data", list);
                return result;
        }
```

spring3mvc封装了json的自动转换，使用@ResponseBody标记下该方法，这样return对象的时候，将自动帮你把普通的pojo对象转成json格式的对象.

菜单效果图：



下一篇预告：Spring3MVC+MyBatis+ExtJs3整合开发系列之三-人员管理模块

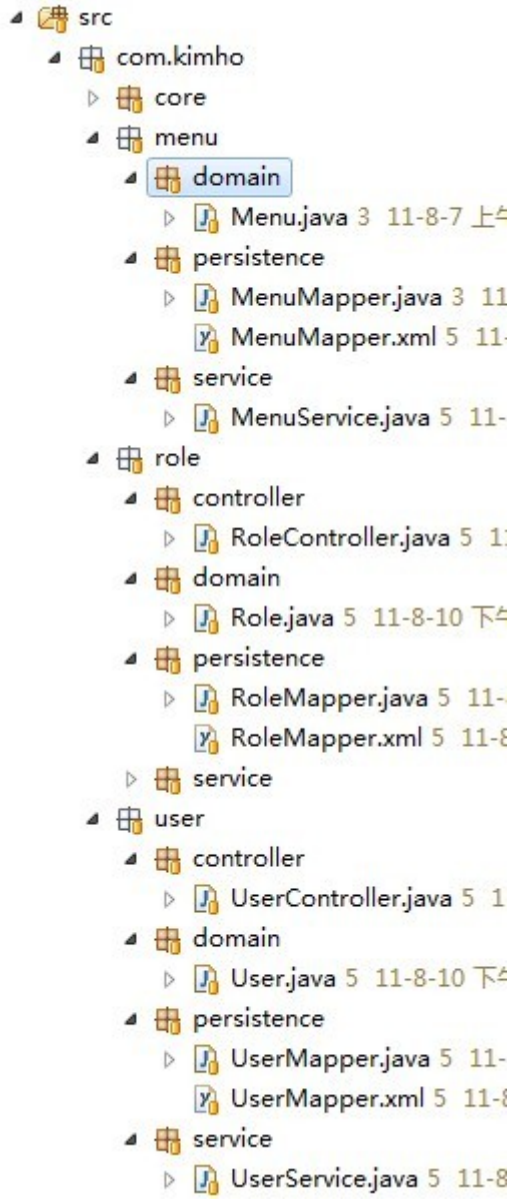
1.3 Spring3MVC+MyBatis+ExtJs3整合开发系列之三：人员管理模块

发表时间: 2011-08-11 关键字: spring3mvc

hi,boys and girls,i come back again!

承接上篇：Spring3MVC+MyBatis+ExtJs3整合开发系列之二：菜单模块，这次带来了Spring3MVC+MyBatis+ExtJs3整合开发系列之第三篇：人员管理模块。

这次改动比较大，下面是模块结构图，总算把之前搭建的package都填满了：



界面预览：

人员管理模块主界面：

新增/修改人员信息界面：

修改密码界面：

代码预览：

persistence层

```
public interface UserMapper {
        Long getId();
        User login(Map<String,Object> param);
        User getUser(Map<String,Object> param);
        List<User> getUserList();
        void insertUser(User user);
        void insertUserRole(Map<String,Object> param);
        void updateUser(User user);
        void updateUserRole(Map<String,Object> param);
        void deleteUser(Map<String,Object> param);
        void deleteUserRole(Map<String,Object> param);
        void changeUserPassword(Map<String,Object> param);
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.kimho.user.persistence.UserMapper">
```

```
<cache />

<select id="getId" resultType="long">
      SELECT max(id)+1 FROM t_user
</select>

<select id="login" parameterType="map" resultType="User">
  SELECT
      id,
      name,
              password,
      create_date
  FROM t_user
  WHERE name = #{name} AND password = #{password}
</select>

<select id="getUser" parameterType="map" resultType="User">
  SELECT
      id,
      name,
              password,
      create_date
  FROM t_user
  <where>
      <if test="id != null">
              id = #{id}
      </if>
      <if test="name != null">
              AND name = #{name}
      </if>
      <if test="password != null">
              AND password = #{password}
      </if>
  </where>
</select>

<select id="getUserList" resultMap="userResult">
```

```
  SELECT
      u.id as user_id,
      u.name as user_name,
              u.password,
      u.create_date,
      r.id as role_id,
      r.name as role_name,
      r.description
  FROM t_user u
  LEFT JOIN t_user_role ur ON(u.id=ur.user_id)
  LEFT JOIN t_role r ON(r.id=ur.role_id)
</select>

<resultMap id="userResult"  type="User">
              <id property="id"  column="user_id" />
              <result property="name" column="user_name"/>
              <result property="createDate" column="create_date"/>
              <collection property="roles"  ofType="Role">
                      <id property="id"  column="role_id"/>
                      <result property="name"  column="role_name"/>
              </collection>
</resultMap>

<insert id="insertUser" parameterType="User" >
      INSERT INTO t_user (id,name, password, create_date)
  VALUES (#{id},#{name}, #{password}, #{createDate,jdbcType=TIMESTAMP})
</insert>

<insert id="insertUserRole" parameterType="map">
      INSERT INTO t_user_role (user_id,role_id)
  VALUES (#{user_id}, #{role_id})
</insert>

<update id="updateUser"  parameterType="User">
      UPDATE t_user
      <set>
              <if test="name != null">name=#{name},</if>
```

```
                <if test="password != null">password=#{password}</if>
        </set>
        WHERE id=#{id}
  </update>

  <update id="updateUserRole"  parameterType="map">
        UPDATE t_user_role SET
        role_id=#{role_id}
        WHERE user_id=#{user_id}
  </update>

  <delete id="deleteUser"  parameterType="map">
        DELETE FROM t_user WHERE id=#{id}
  </delete>

  <delete id="deleteUserRole"  parameterType="map">
        DELETE FROM t_user_role WHERE user_id=#{user_id}
  </delete>

  <update id="changeUserPassword"  parameterType="map">
        UPDATE t_user SET password=#{newPassword} WHERE id=#{id}
  </update>

</mapper>
```

这里特别说明下xml配置中的"**<cache />**"缓存配置项，如果开启了缓存（这里就开启了），需要注意数据的刷新问题；我之前的菜单模块，就开启了缓存机制，导致角色变更后，菜单权限没相应变化，这就是缓存导致的。

service层

```
@Service
public class UserService {
        @Autowired
        private UserMapper userMapper;
        /**
```

```
     * 获取主键id
     * @return
     */
    public Long getId() {
            return userMapper.getId();
    }
    /**
     * 登录验证
     * @param param
     * @return
     */
    public User login(Map<String,Object> param) {
            return userMapper.login(param);
    }
    /**
     * 获取某个用户
     * @param param
     * @return
     */
    public User getUser(Map<String,Object> param) {
            return userMapper.getUser(param);
    }
    /**
     * 获取用户列表
     * @return
     */
    public List<User> getUserList() {
            return userMapper.getUserList();
    }
    /**
     * 新增用户
     * @param user
     * @param param
     */
    @Transactional
    public void insertUser(User user,Map<String,Object> param) {
            //新增用户
```

```
		userMapper.insertUser(user);
		String roleList = (String)param.get("roleList");
		if(!"".equals(roleList)) {
			//新增用户角色关联
			insertUserRole(param);
		}
	}
	/**
	 * 编辑用户
	 * @param user
	 * @param param
	 */
	@Transactional
	public void updateUser(User user,Map<String,Object> param) {
		//编辑用户
		userMapper.updateUser(user);
		//删除原有的用户角色关联
		userMapper.deleteUserRole(param);
		//新增用户角色关联
		insertUserRole(param);
	}
	/**
	 * 删除用户
	 * @param param
	 */
	@Transactional
	public void deleteUser(Map<String,Object> param) {
		String ids = (String)param.get("ids");
		String[] idsArray = ids.split(",");
		for(int i=0;i<idsArray.length;i++) {
			param.put("id", idsArray[i]);
			param.put("user_id", idsArray[i]);
			userMapper.deleteUser(param);
			userMapper.deleteUserRole(param);
		}
	}
	/**
```

```
         * 用户角色关联
         * @param param
         */
        private void insertUserRole(Map<String,Object> param) {
                String roleList = (String)param.get("roleList");
                String[] roleArray = roleList.split(",");
                for(int i=0;i<roleArray.length;i++) {
                        param.put("role_id", roleArray[i]);
                        userMapper.insertUserRole(param);
                }
        }
        /**
         * 修改密码
         * @param param
         */
        public void changeUserPassword(Map<String,Object> param) {
                userMapper.changeUserPassword(param);
        }
}
```

controller层

```
@Controller
@RequestMapping("/user")
public class UserController {
        @Autowired
        private UserService userService;
        @Autowired
        private MenuService menuService;
        /**
         * 登录验证
         * @param request
         * @param response
         * @param user
         * @return
```

```
     */
    @RequestMapping(value="/login",method=RequestMethod.POST)
    public @ResponseBody Map<String,String> loginCheck(HttpServletRequest request,
        HttpServletResponse response){
            //spring会利用jackson自动将返回值封装成JSON对象
            Map<String,String> responseMap = new HashMap<String,String>();
            String captcha = request.getParameter("captcha");
            String name = request.getParameter("name");
            String password = request.getParameter("password");
            String vcode = (String)request.getSession().getAttribute("vcode");
            if(vcode==null || !captcha.equals(vcode)) {
                    responseMap.put("success", "false");
                    responseMap.put("info", "验证码错误！");
                    return responseMap;
            }
            try {
                    Map<String,Object> param = new HashMap<String,Object>();
                    param.put("name", name);
                    param.put("password", password);
                    User user = userService.login(param);
                    if(user!=null) {
                            responseMap.put("success", "true");
                            responseMap.put("info", "登录成功！");
                            request.getSession().setAttribute("user", user);
                            request.getSession().setAttribute("loginUserName", user.getName
                            return responseMap;
                    }else {
                            responseMap.put("success", "false");
                            responseMap.put("info", "用户名或密码错误！");
                            return responseMap;
                    }
            }catch(Exception e) {
                    e.printStackTrace();
                    responseMap.put("info", e.getClass()+":"+e.getMessage());
                    return responseMap;
            }
    }
```

```
/**
 * 获取所有菜单
 * @param request
 * @param response
 * @return
 */
@RequestMapping(value="/menus",method=RequestMethod.POST)
public @ResponseBody Map<String,Object> getMenus(HttpServletRequest request,
    HttpServletResponse response){
        Map<String,Object> responseMap = new HashMap<String,Object>();
        User user = (User)request.getSession().getAttribute("user");
        List<Menu> list = menuService.getMenuListByUserId(user.getId());
        responseMap.put("success", "true");
        responseMap.put("data", list);
        return responseMap;
}

/**
 * 获取用户列表
 * @param request
 * @param response
 * @return
 */
@RequestMapping(value="/users",method=RequestMethod.POST)
public @ResponseBody Map<String,Object> getUsers(HttpServletRequest request,
    HttpServletResponse response){
        Map<String,Object> responseMap = new HashMap<String,Object>();
        List<User> userList = userService.getUserList();
        responseMap.put("totalCount", userList.size());
        responseMap.put("rows", userList);
        return responseMap;
}

/**
 * 保存用户信息（新增用户or编辑用户）
 * @param request
```

```
     * @param response
     * @param user
     * @return
     */
    @RequestMapping(value="/save",method=RequestMethod.POST)
    public @ResponseBody Map<String,Object> saveOrUpdate(HttpServletRequest request,
        HttpServletResponse response){
            Map<String,Object> responseMap = new HashMap<String,Object>();
            String id = request.getParameter("id");
            String name = request.getParameter("name");
            String password = request.getParameter("password");
            String roleList = request.getParameter("roleList");
            try {
                    //编辑用户信息
                    if(!"".equals(id)) {
                            Map<String,Object> param = new HashMap<String,Object>();
                            param.put("id", id);
                            User user = userService.getUser(param);
                            user.setName(name);
                            //表示修改了密码
                            if(!"".equals(password)) {
                                    user.setPassword(password);
                            }
                            param.put("user_id", id);
                            param.put("roleList", roleList);
                            userService.updateUser(user, param);
                            responseMap.put("success", "true");
                            responseMap.put("info", "编辑成功！");
                    }
                    //新增用户信息
                    else {
                            User user = new User();
                            Long newId = userService.getId();
                            user.setId(newId);
                            user.setCreateDate(new Date());
                            user.setName(name);
                            user.setPassword(password);
```

```
                        Map<String,Object> param = new HashMap<String,Object>();
                        param.put("user_id", newId);
                        param.put("roleList", roleList);
                        userService.insertUser(user, param);
                        responseMap.put("method", "Create");
                        responseMap.put("success", "true");
                        responseMap.put("info", "新增成功！");
                }
                return responseMap;
        }catch(Exception e) {
                e.printStackTrace();
                responseMap.put("info", e.getClass()+":"+e.getMessage());
                return responseMap;
        }
}
/**
 * 删除用户
 * @param request
 * @param response
 * @return
 */
@RequestMapping(value="/remove",method=RequestMethod.POST)
public @ResponseBody Map<String,Object> remove(HttpServletRequest request,
    HttpServletResponse response){
        Map<String,Object> responseMap = new HashMap<String,Object>();
        String ids = request.getParameter("ids");
        Map<String,Object> param = new HashMap<String,Object>();
        param.put("ids", ids);
        try {
                userService.deleteUser(param);
                responseMap.put("success", "true");
                responseMap.put("info", "删除成功！");
                return responseMap;
        }catch(Exception e) {
                e.printStackTrace();
                responseMap.put("info", e.getClass()+":"+e.getMessage());
                return responseMap;
```

```
        }
    }
    /**
     * 修改密码
     * @param request
     * @param response
     * @return
     */
    @RequestMapping(value="/changePassword",method=RequestMethod.POST)
    public @ResponseBody Map<String,Object> changePassword(HttpServletRequest request,
        HttpServletResponse response){
            Map<String,Object> responseMap = new HashMap<String,Object>();
            String oldPassword = request.getParameter("oldPassword");
            String newPassword = request.getParameter("newPassword");
            User user =(User)request.getSession().getAttribute("user");
            try {
                    //验证通过，允许修改密码
                    if(oldPassword.equals(user.getPassword())) {
                            Map<String,Object> param = new HashMap<String,Object>();
                            param.put("id", user.getId());
                            param.put("newPassword", newPassword);
                            userService.changeUserPassword(param);
                            user.setPassword(newPassword);
                            //更新session中的user密码信息
                            request.getSession().setAttribute("user", user);
                            responseMap.put("success", "true");
                            responseMap.put("info", "修改密码成功！");
                    }else {
                            responseMap.put("success", "false");
                            responseMap.put("info", "原密码错误！");
                    }
                    return responseMap;
            }catch(Exception e) {
                    e.printStackTrace();
                    responseMap.put("info", e.getClass()+":"+e.getMessage());
                    return responseMap;
            }
```

```
        }
}
```

好了，一个完整的人员管理模块就完成了，测试下模块功能吧。

……

到了这里，是不是发现少了点什么？

没错，还缺少角色菜单管理，那么继续下篇预告吧：Spring3MVC+MyBatis+ExtJs3整合开发系列之四：角色管理模块。

1.4 Spring3MVC+MyBatis+ExtJs3整合开发系列之四：角色管理模块

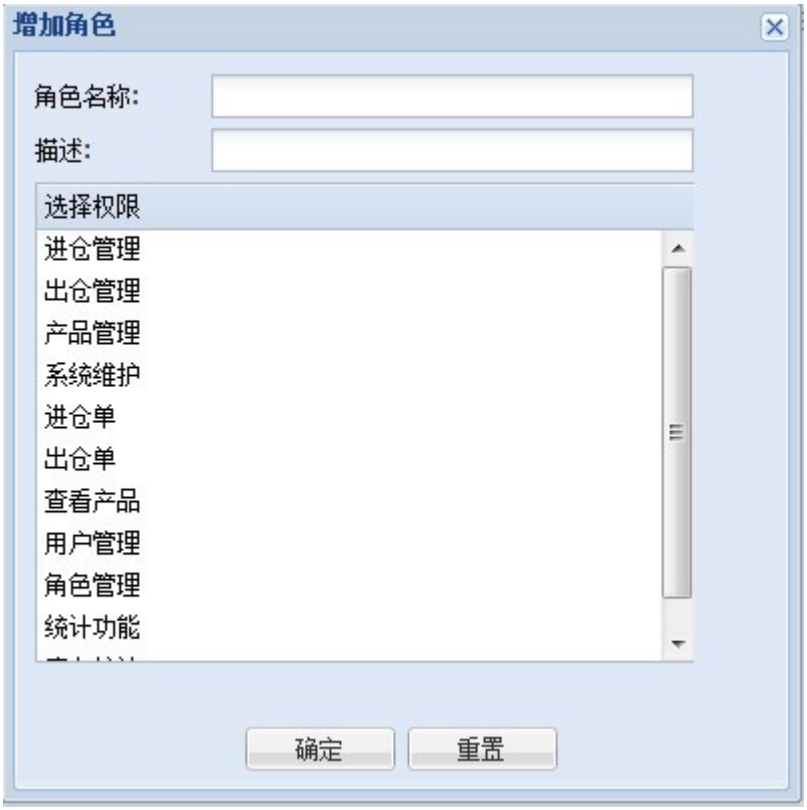
发表时间: 2011-08-13 关键字: spring3mvc, mybatis


iteye coders，周末好！

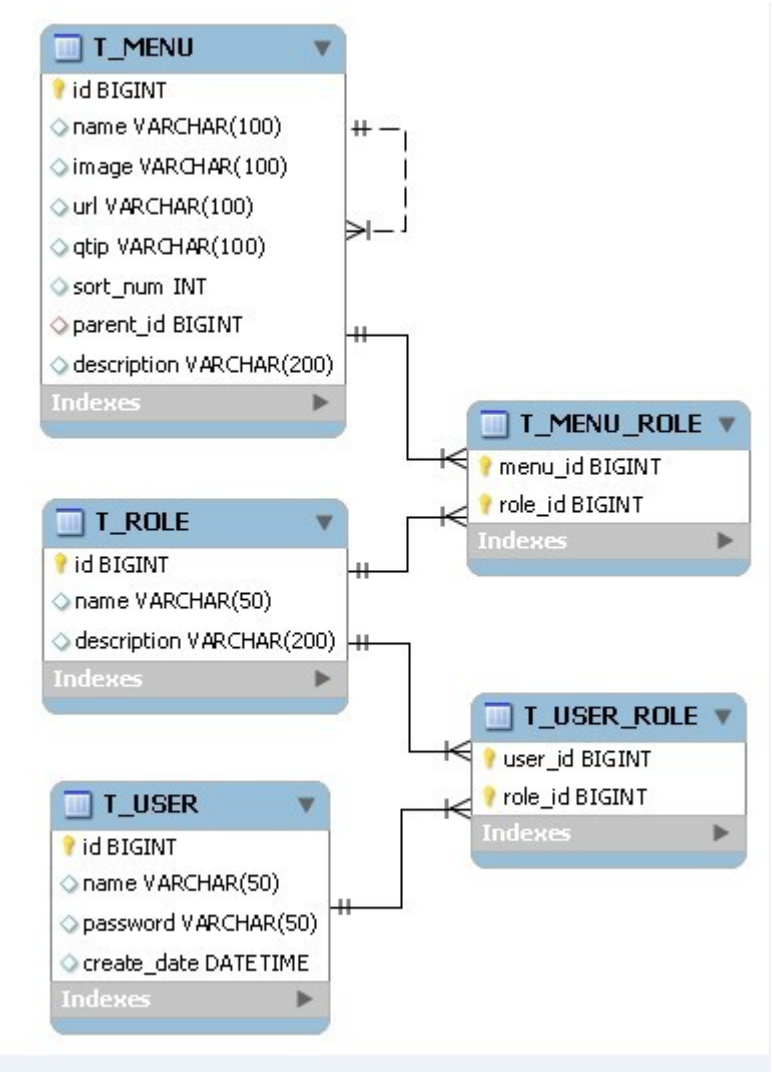
承接上篇：Spring3MVC+MyBatis+ExtJs3整合开发系列之三：人员管理模块，本次带来了整合开发系列之第四篇：角色管理。

角色管理这块的实现，跟人员管理的实现，几乎一模一样，所以这块的开发并需要太多的时间，照套就是了，这也得益于extjs面向组件化开发的最大优势：重用性很高。

界面预览：



整套系统数据库ER图：



到目前为止，我们已经实现了一个业务系统最基本的人员权限菜单管理功能了。

回顾下整个开发系列，发现仍存在几个不足的地方：

1.**事务的控制**：经测试，发现事务回滚这块是失效的，目前还没找到具体的解决方案。

2.**persistence层的代码冗余**：对比人员管理和角色管理这两个模块，会发现代码冗余度比较高，目前的解决方案是：搞个泛型dao层？然后通过不同的业务模型去继承泛型dao，顺便注入自身的业务模型pojo。

3.**缓存**：利用好缓存这块，可以提高系统的性能和并发吞吐量。

Spring3MVC+MyBatis+ExtJs3整合开发系列到此告一段落，以后有时间，将继续增加业务模块进去，希望该开发系列能给各位带来一些系统开发架构的参考价值。

参考资料：

1.spring3mvc：

MasteringSpringMVC3.pdf

spring官网的示例（mvc-showcase等）

2.mybatis:

MyBatis-Spring Reference Simplied Chinese.pdf

MyBatis 3 User Guide Simplified Chinese.pdf

官网示例：jpetstore

3.extjs：

extjs高级程序设计这本书

注：Spring3MvcMyBatisExtjs3.rar已包含项目的所有源码和数据库脚本。

附件下载:

- MyBatis_3_User_Guide_Simplified_Chinese.pdf (1.1 MB)
- dl.iteye.com/topics/download/24b4ed31-d132-3cec-bee7-8030e08b2dfe
- MyBatis-Spring_Reference_Simplied_Chinese.pdf (552.2 KB)
- dl.iteye.com/topics/download/cf65e189-a164-3d09-9b4b-b84ea7776b08
- MasteringSpringMVC3.pdf (929.4 KB)
- dl.iteye.com/topics/download/49c6825d-21c8-344f-b14d-50291206f14c
- Spring3MvcMyBatisExtjs3.rar (7.3 MB)
- dl.iteye.com/topics/download/d9a25134-a386-3ffd-b1aa-edc1ad263b91