ssm练习第四天

第一章：用户认证功能完善

第一节：显示用户名功能

使用SpringSecurity框架进行操作时，SpringSecurity会产生一个上下文对象SecurityContext，该上下文对象会被自动存储到session域中,于此同时也会将该上下文对象绑定到当前线程上，通过SecurityContext可以获得认证对象Authentication，而认证对象Authentication内部封装了principal（主角）属性，该principal就是当前用户对象User，而User对象自然包含用户名等信息。

1、获得用户名方式一：

服务器端可以通过程序API的方式获得

```java
package com.itheima.controller;

import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContext;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.User;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import javax.servlet.http.HttpServletRequest;

@Controller
@RequestMapping("/test")
public class TestController {

    @RequestMapping("/showUsername")
    @ResponseBody
    public String showUsername(HttpServletRequest request){

        //通过SecurityContextHolder获得当前线程上绑定的SecurityContext对象
        SecurityContext context = SecurityContextHolder.getContext();

        //也可以通过session获得SecurityContext对象
        SecurityContext context_session = (SecurityContext)
request.getSession().getAttribute("SPRING_SECURITY_CONTEXT");

        //证明线程绑定的SecurityContext与session域中存储的SecurityContext是同一个
        System.out.println(context==context_session);

        //获得Authentication认证对象
        Authentication authentication = context_session.getAuthentication();
        //获得User对象
        User user = (User) authentication.getPrincipal();
```

```
            //获得用户名
            return user.getUsername();
        }

    }

```

2、获得用户名方式二：

在页面中通过el表达式从session域中获得

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h1>获得当前用户的名称</h1>
    ${sessionScope.SPRING_SECURITY_CONTEXT.authentication.principal.username}
</body>
</html>
```

3、获得用户名方式三：

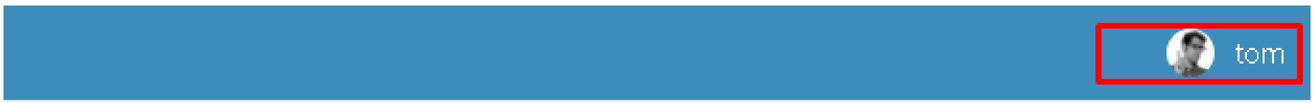
在页面中通过SpringSecurity的标签直接获得

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://www.springframework.org/security/tags" prefix="security"%>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h1>通过security的标签获得用户名名称</h1>
    <security:authentication property="principal.username"/>
</body>
</html>
```

4、在header.jsp中显示用户名

```
<%@ taglib uri="http://www.springframework.org/security/tags" prefix="security"%>

<a href="#" class="dropdown-toggle" data-toggle="dropdown">
    <img src="${pageContext.request.contextPath}/img/user2-160x160.jpg" class="user-image" alt="User Image">
    <span class="hidden-xs">
        <%--获得用户名--%>
        <security:authentication property="principal.username" />
    </span>
</a>
```

显示用户名效果

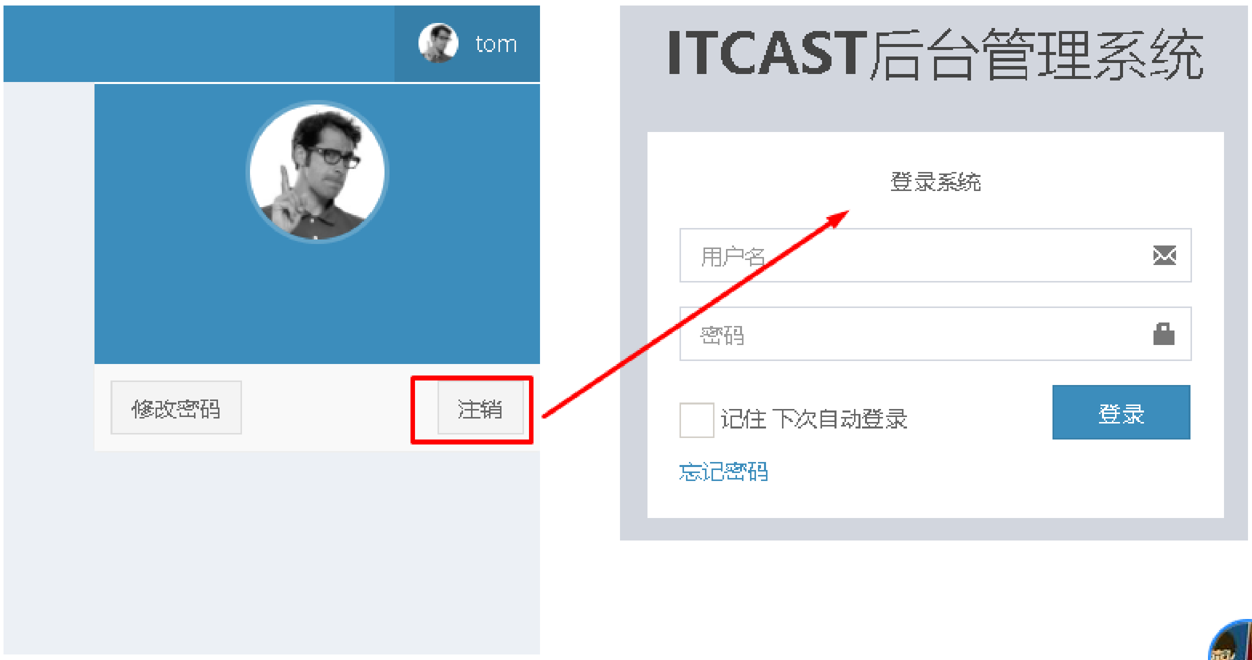


第二节：用户退出功能

在header.jsp的页面中编写超链接

```
<!-- Menu Footer-->
<li class="user-footer">
    <div class="pull-left">
        <a href="#" class="btn btn-default btn-flat">修改密码</a>
    </div>
    <div class="pull-right">
        <a href="${pageContext.request.contextPath}/logout"
           class="btn btn-default btn-flat">注销</a>
    </div>
</li>
```

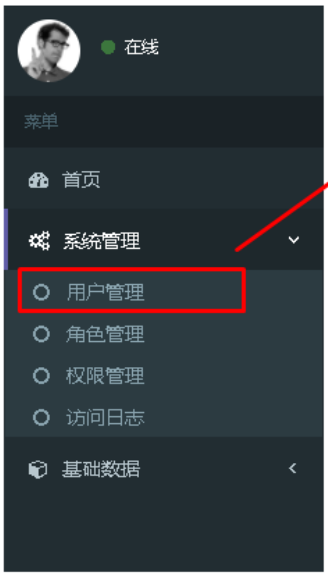
效果图

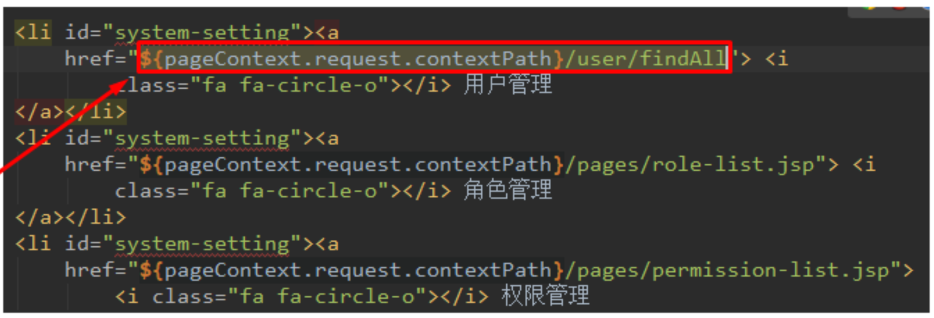


第二章：用户模块

第一节：用户列表查询功能

1、页面入口





```
<li id="system-setting"><a
    href="${pageContext.request.contextPath}/user/findAll"> <i
        class="fa fa-circle-o"></i> 用户管理
</a></li>
<li id="system-setting"><a
    href="${pageContext.request.contextPath}/pages/role-list.jsp"> <i
        class="fa fa-circle-o"></i> 角色管理
</a></li>
<li id="system-setting"><a
    href="${pageContext.request.contextPath}/pages/permission-list.jsp">
        <i class="fa fa-circle-o"></i> 权限管理
```

2、编写Controller

```
@RequestMapping("/findAll")
public ModelAndView findAll(){
    List<SysUser> userList = userService.findAll();
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.addObject("userList",userList);
    modelAndView.setViewName("user-list");
    return modelAndView;
}
```

3、编写Service

接口

```
package com.itheima.service;

import com.itheima.domain.SysUser;
import org.springframework.security.core.userdetails.UserDetailsService;

import java.util.List;

public interface UserService extends UserDetailsService {
    List<SysUser> findAll();
}

```

实现

```
package com.itheima.service.impl;

import com.itheima.dao.UserMapper;
```

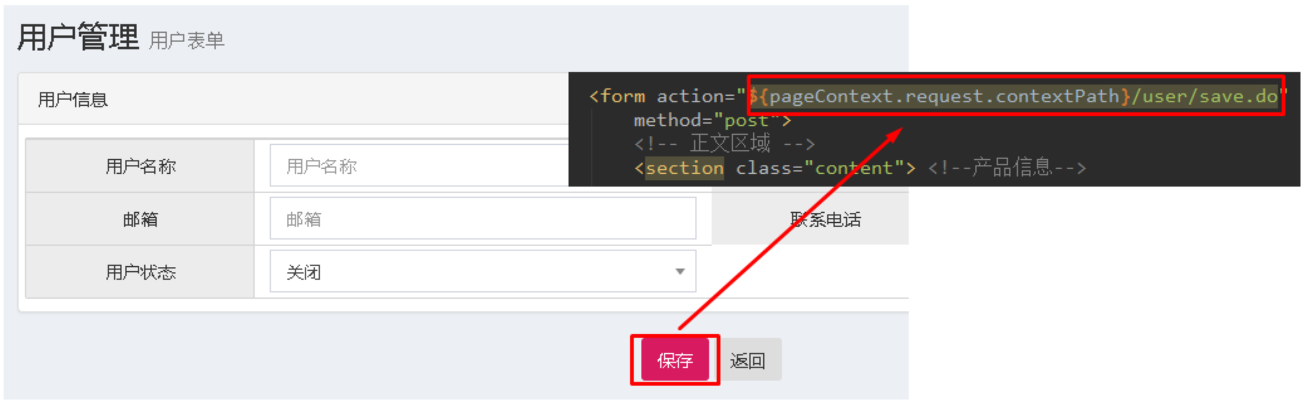
```java
import com.itheima.domain.SysUser;
import com.itheima.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;

@Service("userService")
public class UserServiceImpl implements UserService {

    @Autowired
    private UserMapper userMapper;

    @Override
    public List<SysUser> findAll() {
        return userMapper.findAll();
    }

}

```

4、编写Mapper

```java
package com.itheima.dao;

import com.itheima.domain.SysUser;
import org.apache.ibatis.annotations.Insert;
import org.apache.ibatis.annotations.Select;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface UserMapper {

    @Select("select * from sys_user")
    List<SysUser> findAll();

}

```

第二节：添加用户功能

1、页面入口


用户管理 用户表单

用户信息

用户名称	用户名称	
邮箱	邮箱	联系电话
用户状态	关闭	

```
<form action="${pageContext.request.contextPath}/user/save.do"
    method="post">
    <!-- 正文区域 -->
    <section class="content"> <!--产品信息-->
```

2、编写Controller

```java
@RequestMapping("/save")
public String findAll(SysUser sysUser){
    userService.save(sysUser);
    return "redirect:/user/findAll";
}
```

3、编写Service

接口

```java
void save(SysUser sysUser);
```

实现

```java
@Override
public void save(SysUser sysUser) {
    userMapper.save(sysUser);
}
```

4、编写Mapper

```java
@Insert("insert into sys_user (username,email,password,phoneNum,status) values (#{username},#{email},#{password},#{phoneNum},#{status})")
void save(SysUser sysUser);
```

第三节：用户密码的加密操作

为了客户账户的安全性，在数据库中不能明文显示密码，所以就需要对客户端提交的密码进行加密后在存储到数据库中。原来可以使用md5、加盐加密等工具进行密码加密，目前我们使用的SpringSecurity框架本身就提供了加密工具。

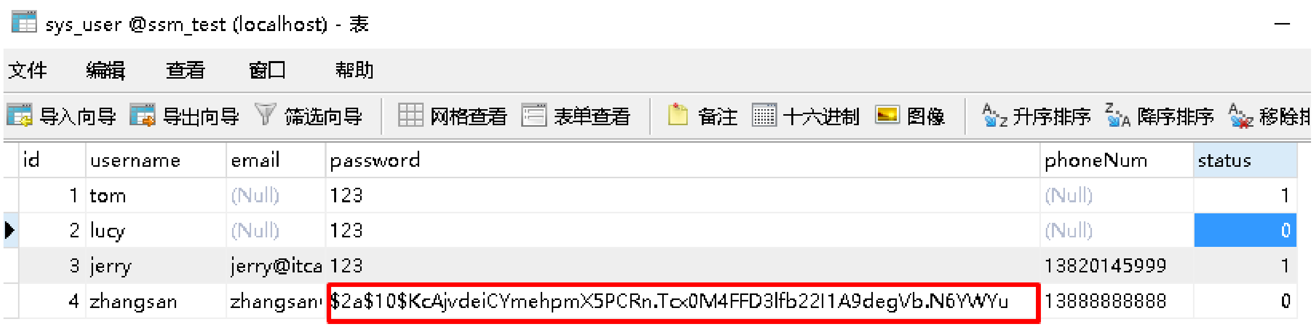
1、在spring-security.xml中配置加密类BCryptPasswordEncoder

```xml
<!--配置加密类-->
<bean id="passwordEncoder"
class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"/>
```

2、修改save业务方法

```java
@Autowired
private BCryptPasswordEncoder bCryptPasswordEncoder;

@Override
public void save(SysUser sysUser) {
    String encodePassword = bCryptPasswordEncoder.encode(sysUser.getPassword());
    sysUser.setPassword(encodePassword);
    userMapper.save(sysUser);
}
```

3、加密效果


sys_user @ssm_test (localhost) - 表
文件 编辑 查看 窗口 帮助
导入向导 导出向导 筛选向导 网格查看 表单查看 备注 十六进制 图像 升序排序 降序排序 移除排

id	username	email	password	phoneNum	status
1	tom	(Null)	123	(Null)	1
2	lucy	(Null)	123	(Null)	0
3	jerry	jerry@itca	123	13820145999	1
4	zhangsan	zhangsan	$2a$10$KcAjvdeiCYmehpmX5PCRn.Tcx0M4FFD3lfb22I1A9degVb.N6YWYu	13888888888	0

4、修改登录操作

用户在登录时的密码也需要进行加密才能与数据库中的密码进行匹配，但认证时的加密操作不需要手动编写程序实现，只需要对spring-security.xml进行配置即可。

修改spring-security.xml

```xml
<security:authentication-manager>
    <!-- 提供服务类，去数据库查询用户名和密码 -->
    <security:authentication-provider user-service-ref="userService">
        <!--指定使用spring容器中passwordEncoder进行加密操作-->
        <security:password-encoder ref="passwordEncoder"/>
    </security:authentication-provider>
</security:authentication-manager>
```

修改userServiceImpl的认证方法loadUserByUsername，把{noop}代码去掉，表示采用加密方式登录

```java
@Override
public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
    // 先设置假的权限
    List<GrantedAuthority> authorities = new ArrayList<>();
    authorities.add(new SimpleGrantedAuthority("ROLE_USER"));
    // 通过用户名查询密码
    SysUser sysUser = userMapper.findByUsername(username);
    //判断认证是否成功
    if(sysUser!=null){
        User user = new User(username,sysUser.getPassword(),authorities);
        return user;
    }
    return null;
}
```

第三章：角色模块

第一节：角色表与实体的创建

1、角色表建表语句和字段含义

```sql
CREATE TABLE sys_role(
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    roleName VARCHAR(50) ,
    roleDesc VARCHAR(50)
)
```

字段含义：

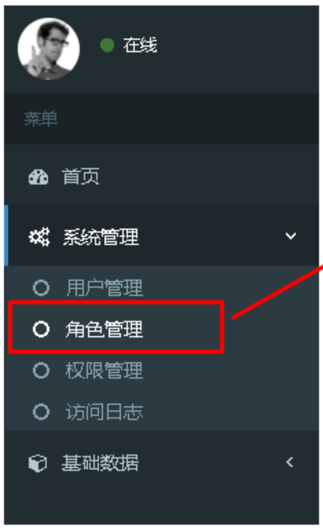
序号	字段名称	字段类型	字段描述
1	id	bigint	无意义，主键自动增长
2	roleName	varchar	角色名
3	roleDesc	varchar	角色描述

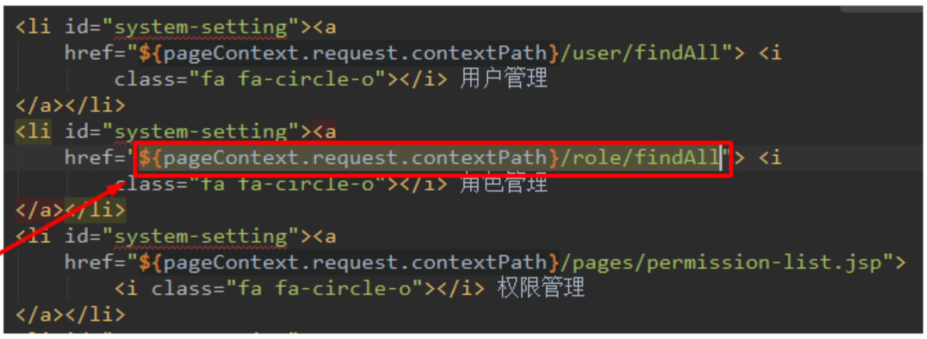
2、Role实体创建

```java
package com.itheima.domain;

public class Role {

    private Long id;
    private String roleName;
    private String roleDesc;

    //此处省略getter和setter方法... ...
}

```

第二节：角色列表查询功能

1、页面入口





```
<li id="system-setting"><a
    href="${pageContext.request.contextPath}/user/findAll"> <i
        class="fa fa-circle-o"></i> 用户管理
</a></li>
<li id="system-setting"><a
    href="${pageContext.request.contextPath}/role/findAll"> <i
        class="fa fa-circle-o"></i> 角色管理
</a></li>
<li id="system-setting"><a
    href="${pageContext.request.contextPath}/pages/permission-list.jsp">
        <i class="fa fa-circle-o"></i> 权限管理
</a></li>
```

2、编写Controller

```java
package com.itheima.controller;

import com.itheima.domain.Role;
import com.itheima.service.RoleService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import java.util.List;

@Controller
@RequestMapping("/role")
public class RoleController {

    @Autowired
    private RoleService roleService;

    @RequestMapping("/findAll")
    public ModelAndView findAll(){
        List<Role> roleList = roleService.findAll();
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject("roleList",roleList);
        modelAndView.setViewName("role-list");
        return modelAndView;
    }

}

```

3、编写Service

接口

```java
package com.itheima.service;

import com.itheima.domain.Role;

import java.util.List;

public interface RoleService {
    List<Role> findAll();
}

```

实现

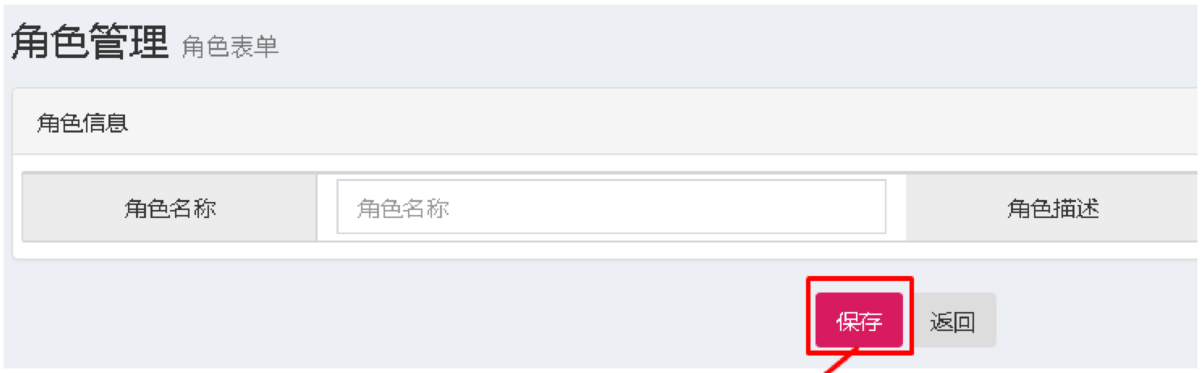
```java
package com.itheima.service.impl;

import com.itheima.dao.RoleMapper;
import com.itheima.domain.Role;
import com.itheima.service.RoleService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service("roleService")
public class RoleServiceImpl implements RoleService {

    @Autowired
    private RoleMapper roleMapper;

    @Override
    public List<Role> findAll() {
        return roleMapper.findAll();
    }
}

```

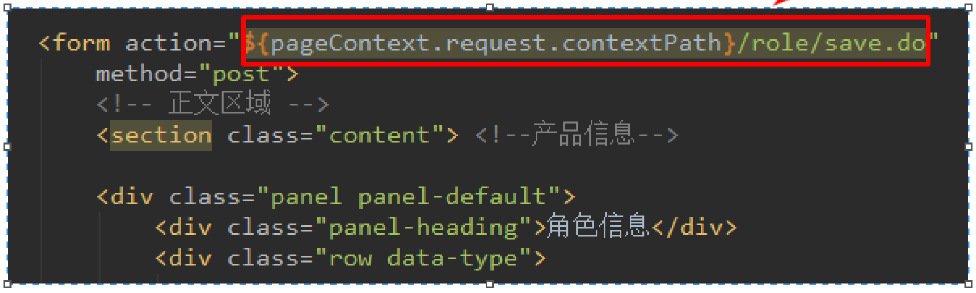
4、编写Mapper

```java
package com.itheima.dao;

import com.itheima.domain.Role;
import org.apache.ibatis.annotations.Select;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface RoleMapper {

    @Select("select * from sys_role")
    List<Role> findAll();
}

```

第三节：添加角色功能

1、页面入口





```
<form action="${pageContext.request.contextPath}/role/save.do"
    method="post">
    <!-- 正文区域 -->
    <section class="content"> <!--产品信息-->

    <div class="panel panel-default">
        <div class="panel-heading">角色信息</div>
        <div class="row data-type">
```

2、编写Controller

```

```

3、编写Service

接口

```

```

实现

```

```

4、编写Mapper

```

```

第四章：权限模块

第一节：权限表与实体的创建

1、权限表建表语句和字段含义

```sql
CREATE TABLE sys_permission(
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    permissionName VARCHAR(50) ,
    url VARCHAR(50),
    pid BIGINT
)
```

字段含义：

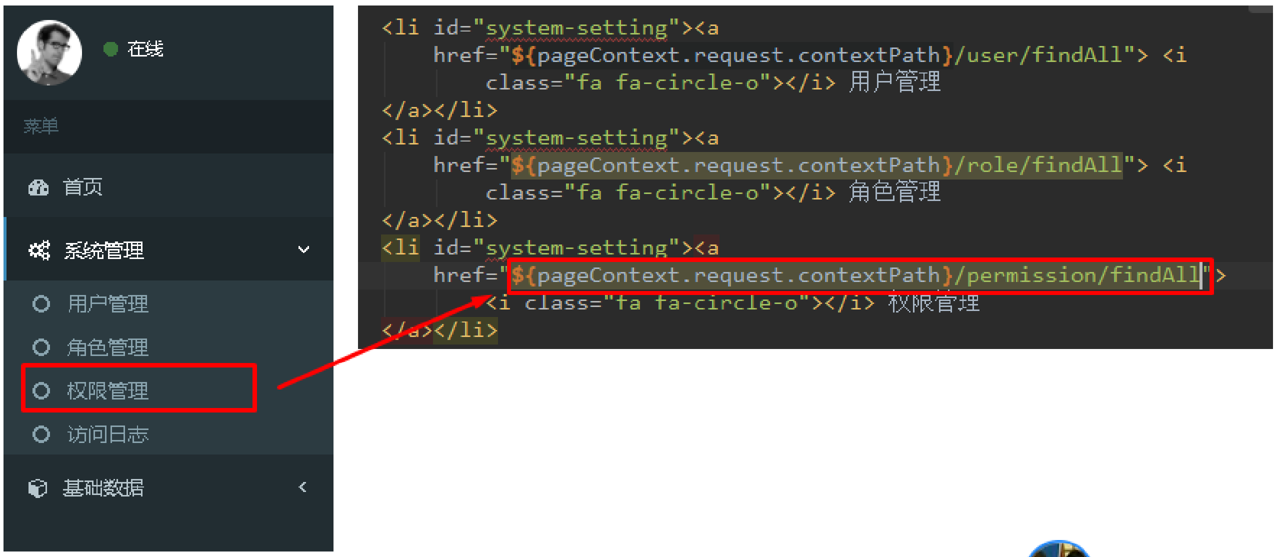
序号	字段名称	字段类型	字段描述
1	id	bigint	无意义，主键自动增长
2	permissionName	varchar	权限名
3	url	varchar	资源路径
4	pid	bigint	父菜单id

2、Permission实体创建

```java
package com.itheima.domain;

public class Permission {

    private Long id;
    private String permissionName;
    private String url;
    private Long pid;

    //此处省略getter和setter方法... ...
}

```

第二节：权限列表查询功能

1、页面入口



```
<li id="system-setting"><a
    href="${pageContext.request.contextPath}/user/findAll"> <i
        class="fa fa-circle-o"></i> 用户管理
</a></li>
<li id="system-setting"><a
    href="${pageContext.request.contextPath}/role/findAll"> <i
        class="fa fa-circle-o"></i> 角色管理
</a></li>
<li id="system-setting"><a
    href="${pageContext.request.contextPath}/permission/findAll">
        <i class="fa fa-circle-o"></i> 权限管理
</a></li>
```

2、编写Controller

```java
package com.itheima.controller;

import com.itheima.domain.Permission;
import com.itheima.service.PermissionService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

import java.util.List;

@Controller
@RequestMapping("/permission")
public class PermissionController {

    @Autowired
    private PermissionService permissionService;

    @RequestMapping("/findAll")
    public ModelAndView findAll(){
        List<Permission> permissionList = permissionService.findAll();
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.addObject("permissionList",permissionList);
        modelAndView.setViewName("permission-list");
        return modelAndView;
    }

}

```

3、编写Service

接口

```java
package com.itheima.service;

import com.itheima.domain.Permission;

import java.util.List;

public interface PermissionService {
    List<Permission> findAll();
}

```

实现

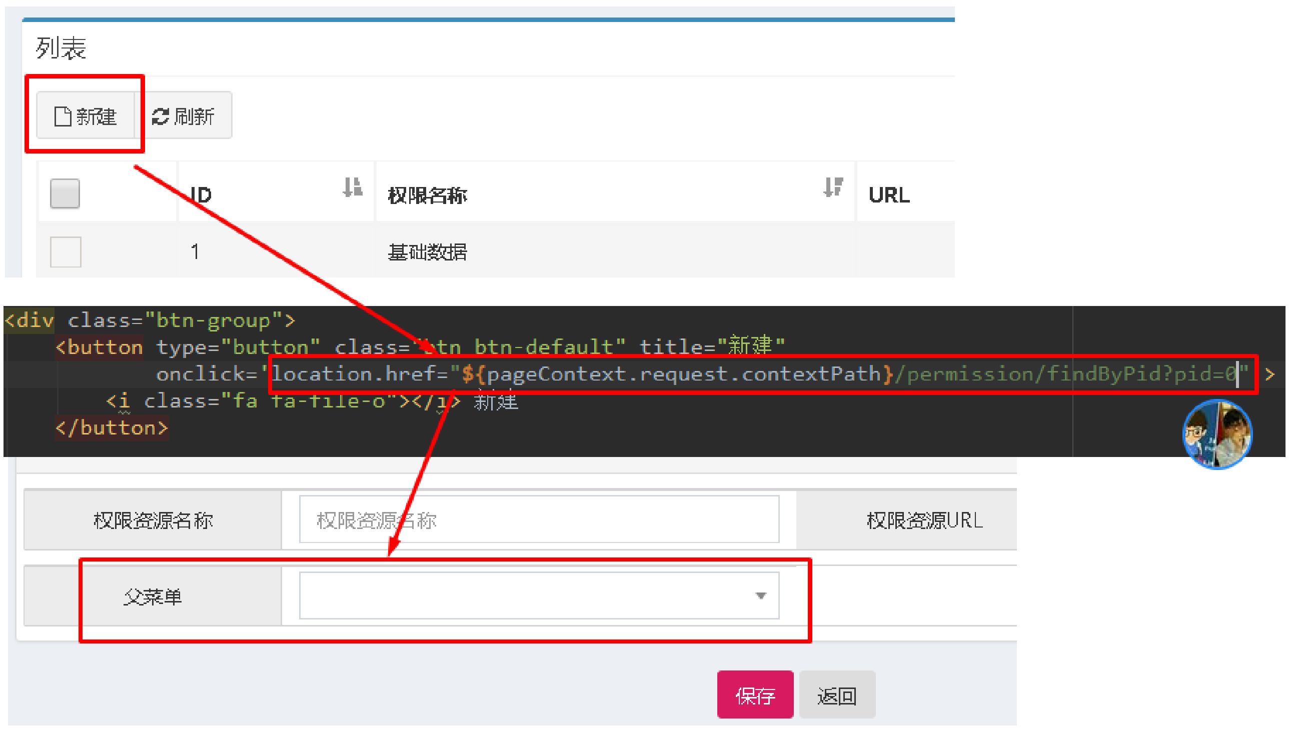
```java
package com.itheima.service.impl;

import com.itheima.dao.PermissionMapper;
import com.itheima.domain.Permission;
import com.itheima.service.PermissionService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service("permissionService")
public class PermissionServiceImpl implements PermissionService {

    @Autowired
    private PermissionMapper permissionMapper;

    @Override
    public List<Permission> findAll() {
        return permissionMapper.findAll();
    }
}

```

4、编写Mapper

```java
package com.itheima.dao;

import com.itheima.domain.Permission;
import org.apache.ibatis.annotations.Select;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface PermissionMapper {

    @Select("select * from sys_permission")
    List<Permission> findAll();
}

```

第三节：添加权限功能-回显父菜单

1、页面入口



```html
<div class="btn-group">
    <button type="button" class="btn btn-default" title="新建"
            onclick='location.href="${pageContext.request.contextPath}/permission/findByPid?pid=0"' >
        <i class="fa fa-file-o"></i> 新建
    </button>
```

2、编写Controller

```java
@RequestMapping("/findByPid")
public ModelAndView findByPid(Long pid){
    List<Permission> parentPermissionList = permissionService.findByPid(pid);
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.addObject("parentPermissionList",parentPermissionList);
    modelAndView.setViewName("permission-add");
    return modelAndView;
}
```

3、编写Service

接口

```java
List<Permission> findByPid(Long pid);
```

实现

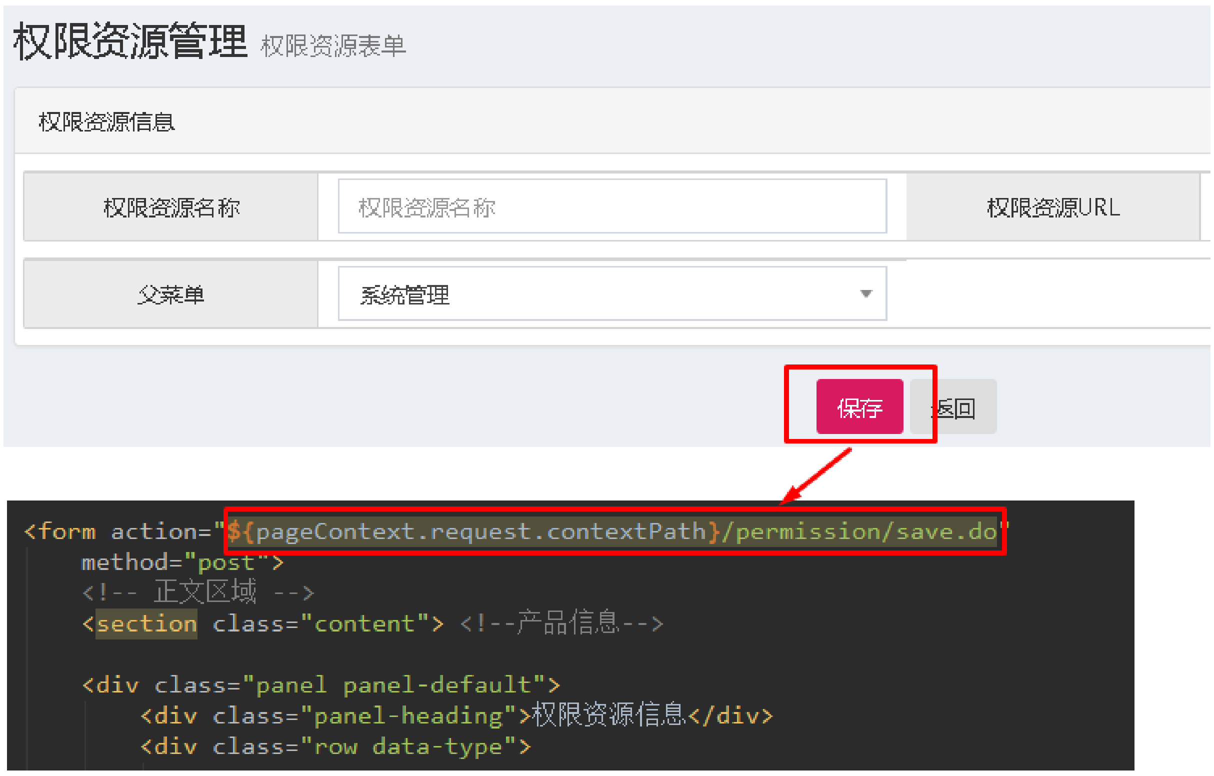
```java
@Override
public List<Permission> findByPid(Long pid) {
    return permissionMapper.findByPId(pid);
}
```

4、编写Mapper

```java
@Select("select * from sys_permission where pid=#{pid}")
List<Permission> findByPId(Long pid);
```

第四节：添加权限功能-保存到数据库

1、页面入口



```html
<form action="${pageContext.request.contextPath}/permission/save.do"
    method="post">
    <!-- 正文区域 -->
    <section class="content"> <!--产品信息-->

    <div class="panel panel-default">
        <div class="panel-heading">权限资源信息</div>
        <div class="row data-type">
```

2、编写Controller

```java
@RequestMapping("/save")
public String save(Permission permission){
    permissionService.save(permission);
    return  "redirect:/permission/findAll";
}
```

3、编写Service

接口

```java
void save(Permission permission);
```

实现

```java
@Override
public void save(Permission permission) {
    permissionMapper.save(permission);
}
```

4、编写Mapper

```java
@Insert("insert into sys_permission (permissionName,url,pid) values (#{permissionName}, #{url}, #{pid})")
void save(Permission permission);
```