课程目标:

1. 实现开发者后台的app上架和下架操作
2. 扩展:使用spirngmvc拦截器实现系统安全性

>>知识点:会springmvc的拦截器

>>业务:实现系统的安全性

1. 总结 ：app管理平台

实现app管理的业务流程

1. 实现开发者后台的app上架和下架操作
2. 分析需求:

当app状态为”通过审核”,”下架”时可以实现上架

当app状态为”上架时” 可以实下架

1. 实现上架和下架时就是修改app状态分别为4(上架)和5(下架)
2. 修改app信息列表页，根据app状态显示对应的下架和上架操作菜单

|  |
| --- |
| <**c:if test="${**appInfo.status==2 || appInfo.status==5**}"**>  <**li**><**a name="upJia" appid="${**appInfo.id**}"**  **class="addVersion" data-toggle="tooltip" data-placement="top" title=""**>上架</**a**></**li**> </**c:if**> <**c:if test="${** appInfo.status==4**}"**>  <**li**><**a name="upJia" appid="${**appInfo.id**}"**  **class="addVersion" data-toggle="tooltip" data-placement="top" title=""**>下架</**a**></**li**> </**c:if**> |

1. 实现上架功能: 采用异步技术(网页不刷新)
   1. 上架和下架的业务无需编写

存在一个业务修改app的状态

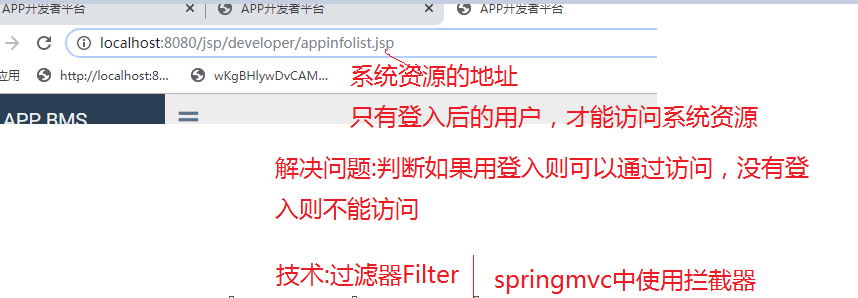
* 1. 编写控制器请求:处理上架和下架

|  |
| --- |
| *//实现app的上架和下架操作 //上架 传状态为4 下架传状态为5* @RequestMapping(**"/upDownApp"**) @ResponseBody *//返回json* **public** String upDownApp(Long id,Long state){  *//调用业务实现* **int** temp=**this**.**appInfoService**.updateAppInfoState(id,state);  **return "{\"result\":"**+temp+**"}"**; *//如果temp>0 成功 否则失败* } |

* 1. 使用jquery实现异步请求:实现上架下架

|  |
| --- |
| //实现上架 $("a[name='upJia']").click(function(){  //获取a标签  var anode=$(this);  //获取显示状态的单元格,将a标签所在的表单的单元格设置名称<td name=“td9”>  var stateNode=$(this).parents("[name='td9']").prev().prev().prev();  //获取app的id  var appid=$(this).attr("appid");  if(anode.html()=="上架")  {  //发送异步请求实现上架(4)  .post("${pageContext.request.contextPath}/upDownApp",{"appid":appid,"state":4},function(data){  if(data.result==1){ //成功  //修改显示状为"已上架"  stateNode.html("已上架");  //修改操作为下架  anode.html("下架");  }else{  alert("失败");  }  },"json");  }else{  //发送异步请求完成下载(5) $.post("${pageContext.request.contextPath}/upDownApp",{"appid":appid,"state":5},function(data){  if(data.result==1){ //成功  //修改显示状为"已下架"  stateNode.html("已下架");  //修改操作为下架  anode.html("上架");  }else{  alert("失败");  }  },"json");  } }); |

引言:



1. SpringMVC拦截器(Interceptor)
2. 什么是SpringMVC拦截器:

SpringMVC 中的Interceptor 拦截器也是相当重要的，它的主要作用是拦截用户的请求并进行相应的处理。一般用于通过它来进行权限验证，或者是来判断用户*是否*登陆等。

注意:springmvc拦截器是针对springmvc控制器请求进行拦截的.

1. 使用（创建）springmvc拦截器
   1. 创建拦截器类实现HandlerInterceptor接口:

org.springframework.web.servlet.HandlerInterceptor;

|  |
| --- |
| **public class** LoginInterceptror **implements** HandlerInterceptor {  */\*  \* preHandle,拦截请求时执行的方法(进入控制器之前)  \* 注意：该方法接收返回值，如果返回true,则表示放行可以进入控制器  \* 反之返回false表示不放行，不能进入控制器  \* \*/* **public boolean** preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) **throws** Exception {  System.***out***.println(**"preHandle......."**);  **return true**;  }   @Override  **public void** postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) **throws** Exception {  *//postHandle在业务处理器处理请求执行完成后(控制器执行完后),生成视图之前执行;* System.***out***.println(**"postHandle......."**);  }   @Override  **public void** afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex) **throws** Exception {  *//afterCompletion在DispatcherServlet完全处理完请求后被调用,可用于清理资源等 afterCompletion()执行完成后开始渲染页面* System.***out***.println(**"afterCompletion"**);  } } |

* 1. 在springmvc的配置文件中添加拦截器配置

|  |
| --- |
| *<!--配置sringmvc的拦截器-->* <**mvc:interceptors**>  <**mvc:interceptor**>  *<!--mvc:mapping指定拦截的控制请求 /\*\* 表示根目录下所有控制器请求-->* <**mvc:mapping path="/\*\*"**/>  *<!—排除哪些请求不拦截，他必需写在拦截请求的后面*  <**mvc:exclude-mapping path="/getAppInfo"**></**mvc:exclude-mapping**>  *-->*  *<!--定义拦截器对象-->* <**bean class="com.team.appinfo.interceptor.LoginInterceptror"**></**bean**>  </**mvc:interceptor**>  </**mvc:interceptors**> |

1. 使用SpringMVC拦截器保护项目系统资源安全(防止未登入的用户访问系统资源)
2. 使用拦截器对项目控制器请求的资源进行安全保护.

1.1 创建拦截器对所有控制器的请求进行拦截（ 只有登入成功后，才能访问控制器请求），但是需要对象登入页面的控制器进行放行，及登入控制器请求放行。

|  |
| --- |
| **public class** LoginInterceptror **implements** HandlerInterceptor {  */\*  \* preHandle,拦截请求时执行的方法(进入控制器之前)  \* 注意：该方法接收返回值，如果返回true,则表示放行可以进入控制器  \* 反之返回false表示不放行，不能进入控制器  \* \*/* **public boolean** preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) **throws** Exception {  *//判断session是否存在  //如果存在则表示登入，放行。如果不存在session或者session过期则无限访问* HttpSession session=request.getSession();  Object o=session.getAttribute(**"loginInfo"**);  **if**(o!=**null**){  **return true**; *//放行* }**else** {  *//如果没有session  //跳转到登入页  //*response.sendRedirect(**"/jsp/devlogin.jsp"**);  response.sendRedirect(**"/index"**); //控制器请求进入登入页  **return false**;  }  }  @Override  **public void** postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) **throws** Exception {  *//postHandle在业务处理器处理请求执行完成后(控制器执行完后),生成视图之前执行;  // System.out.println("postHandle.......");* }  @Override  **public void** afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex) **throws** Exception {  *//afterCompletion在DispatcherServlet完全处理完请求后被调用,可用于清理资源等 afterCompletion()执行完成后开始渲染页面  // System.out.println("afterCompletion");* } } |

* 1. 在springmvc配置文件中添加拦截器的配置:

|  |
| --- |
| *<!--配置sringmvc的拦截器-->* <**mvc:interceptors**>  <**mvc:interceptor**>  *<!--mvc:mapping指定拦截的控制请求 /\*\* 表示根目录下所有控制器请求-->* <**mvc:mapping path="/\*\*"**/>  *<!--哪些请求不拦，他要写在拦截请求的后面-->*  *<!--访问登入页面的请求放行-->* <**mvc:exclude-mapping path="/index"**></**mvc:exclude-mapping**> *<!--对处理登入的控制器放行-->* <**mvc:exclude-mapping path="/login"**></**mvc:exclude-mapping**>  <**mvc:exclude-mapping path="/code"**></**mvc:exclude-mapping**>  *<!--定义拦截器对象-->* <**bean class="com.team.appinfo.interceptor.LoginInterceptror"**></**bean**>  </**mvc:interceptor**> </**mvc:interceptors**> |

1. 实现springmvc项目的jsp资源保护

注意：spring mvc的拦截器只拦截controller不拦截jsp文件，如果不拦截jsp文件也会给系统带安全性问题。

解决jsp资源安全问题:将所有的jsp文件放入到WEB-INF文件夹下，这样用户是直接不能访问WEB-INF文件夹下的jsp文件的。spring mvc的理念也是通过controller里的@RequestMapping来请求相关jsp页面，而非用户直接访问jsp页面。

1. 将项目中所有jsp页面放到WEB-INF目录中,对jsp页面进行保存，因为WEB-INF下的目录资源本身是无法对外直接访问的.

注意:修改springmvc的视图解析器

<**property name="prefix" value="/WEB-INF/jsp/"**></**property**>

1. 利用控制器进行页面的跳转访问.

注意:配置视图解析器的地址: /WEB-INF/jsp/

|  |
| --- |
| @Controller **public class** LinkController {  @RequestMapping(**"/index"**)  **public** String index(){  **return "devlogin"**; *//跳转到jsp文件* } } |

注意：如果实现以上步骤实现对jsp资源保存

注意:/index的控制器请求也需要放行

总结 ：app管理平台=========实现app管理的业务流程

