用户、分享、点赞、评论、回复、收藏、关注、管理员、文件

### 用户相关接口

@RestController  
@RequestMapping(value = "/user")

注册  
 @RequestMapping(value = "/register",method = RequestMethod.POST)  
 public boolean registerUser(@RequestBody SystemUser user)

{

    "name":"xuan",

    "password":"123"

}

登录  
 @RequestMapping(value = "/signin",method = RequestMethod.POST)  
 public boolean signIn(@RequestBody SystemUser user, HttpSession session)

{

    "name":"xuan",

    "password":"123"

}

通过id获取用户基本信息  
 @RequestMapping(value = "/getuser",method = RequestMethod.GET)  
 public SystemUser getUser(@RequestParam(value = "userId", required = true) long userId){  
 return userService.getUser(userId);  
 }

通过id获取用户详细信息  
 @RequestMapping(value = "/getuserinfo",method = RequestMethod.GET)  
 public UserInfo getUserInfo(@RequestParam(value = "userId", required = true) long userId){  
 return userInfoService.getUserInfo(userId);  
 }

获取当前用户基本信息  
 @RequestMapping(value = "/getcurrentuser",method = RequestMethod.GET)  
 public SystemUser getUser(HttpSession session){  
 return userService.getUser(((SystemUser) session.getAttribute("user")).getId());  
 }

不用传参数

获取当前用户详细信息  
 @RequestMapping(value = "/getcurrentuserinfo",method = RequestMethod.GET)  
 public UserInfo getUserInfo(HttpSession session){  
 return userInfoService.getUserInfo(((SystemUser) session.getAttribute("user")).getId());  
 }  
不用传参数

更新当前用户用户名  
 @RequestMapping(value = "/updateusername",method = RequestMethod.POST)  
public boolean updateUsername(@RequestBody SystemUser user,HttpSession session){  
 return userService.updateUsername(((SystemUser)session.getAttribute("user")).getId(), user.getName());  
}

{

    "name":"peng"

}

更新当前用户密码  
@RequestMapping(value = "/updateuserpassword",method = RequestMethod.POST)  
public boolean updateUserPassword(@RequestBody SystemUser user,HttpSession session){  
 return userService.updateUserPassword(((SystemUser)session.getAttribute("user")).getId(), user.getPassword());  
}

{

    "password":"456"

}

更新当前用户详细信息  
 @RequestMapping(value = "/updateuserinfo",method = RequestMethod.POST)  
 public boolean updateUserInfo(@RequestBody UserInfo userInfo){  
 return userInfoService.updateUserInfo(userInfo);  
 }  
{

    "gender":"M",

    "birthday":"2000-1-1",

    "selfIntroduction":"I am peng"

}

搜索用户  
 @RequestMapping(value = "/searchuser" ,method = RequestMethod.GET)  
 public List<UserListInfo> searchUser(@RequestParam String username){  
 return userService.searchUser(username);  
 }

127.0.0.1:8080/user/searchuser?username=xuan

### 分享相关接口

@RestController  
@RequestMapping(value = "/share")  
public class ShareController {

发布分享，返回值为分享的id，获得的id在上传图片时用到  
 @RequestMapping(value = "/publish", method = RequestMethod.POST)  
 public long publish(@RequestBody Share share, HttpSession session){  
 share.setUserId(((SystemUser)session.getAttribute("user")).getId());  
 return shareService.publish(share);  
 }

{

        "content": "my share",

        "state": "public"

}

根据shareId获取分享  
 @RequestMapping(value = "/get", method = RequestMethod.GET)  
 public ShareListInfo get(@RequestParam(value = "shareId", required = true) long shareId){  
 return shareService.getShare(shareId);  
 }

按分页的方式获取公开的分享，当前页起始分享为第begin个分享，每页length个(begin从0开始)  
 @RequestMapping(value = "/getpublicsharelist", method = RequestMethod.GET)  
 public List<ShareListInfo> getPublicShareList(@RequestParam long begin, @RequestParam long length){  
 return shareService.getPublicShareList(begin,length);  
 }

按分页的方式获取当前用户的分享，当前页起始分享为第begin个分享，每页length个(begin从0开始)  
 @RequestMapping(value = "/getmysharelist", method = RequestMethod.GET)  
 public List<ShareListInfo> getMyShareList(HttpSession session,@RequestParam long begin, @RequestParam long length){  
 return shareService.getMyShareList(((SystemUser)session.getAttribute("user")).getId(),begin,length);  
 }

分页的方式获取最新分享  
 @RequestMapping(value = "/getnewestshare", method = RequestMethod.GET)  
 public List<ShareListInfo> getNewestShare(@RequestParam long begin, @RequestParam long length){  
 return shareService.getNewestShare(begin,length);  
 }

分页的方式获取高赞分享  
 @RequestMapping(value = "/getfavoriteshare", method = RequestMethod.GET)  
 public List<ShareListInfo> getFavoriteShare(@RequestParam long begin, @RequestParam long length){  
 return shareService.getFavoriteShare(begin,length);  
 }

设置分享权限  
 @RequestMapping(value = "/setsharestate", method = RequestMethod.GET)  
 public boolean setShareState(@RequestParam long shareId, @RequestParam String state){  
 return shareService.setShareState(shareId,state);  
 }

根据分享内容搜索分享  
 @RequestMapping(value = "/searchshare", method = RequestMethod.GET)  
 public List<ShareListInfo> searchShare(@RequestParam String content){  
 return shareService.searchShare(content);  
 }  
  
 @RequestMapping(value = "/deletepicture", method = RequestMethod.GET)  
 public boolean deletePicture(@RequestParam long id){  
 return pictureService.deletePicture(id);  
 }

更新分享内容  
 @RequestMapping(value = "/updatecontent", method = RequestMethod.POST)  
 public boolean updateContent(@RequestBody Share share){  
 return shareService.updateContent(share);  
 }

{

    "id":"1",

    "content":"new share"

}

}

### 点赞相关接口

@RestController  
@RequestMapping(value = "/like")  
public class LikeController {

点赞  
 @RequestMapping(value = "/add")  
public boolean add(@RequestBody Like like, HttpSession session){  
 like.setUserId(((SystemUser)session.getAttribute("user")).getId());  
 return likeService.add(like);  
}

{

        "shareId": "3"

}

}

### 评论相关接口

@RestController  
@RequestMapping(value = "/comment")  
public class CommentController {

发布评论  
 @RequestMapping(value = "/publish", method = RequestMethod.POST)  
 public boolean publish(@RequestBody Comment comment, HttpSession session){  
 comment.setUserId(((SystemUser)session.getAttribute("user")).getId());  
 return commentService.publishComment(comment);  
 }

{

    "content":"my first comment",

    "shareId": "2"

}

分页的方式获取指定分享下的评论  
 @RequestMapping(value = "/getcomment", method = RequestMethod.GET)  
 public List<CommentListInfo> getComment(@RequestParam long shareId, @RequestParam long begin, @RequestParam long length){  
 return commentService.getComment(shareId,begin,length);  
 }  
}

### 回复相关接口

@RestController  
@RequestMapping(value = "/reply")  
public class ReplyController {

发布回复  
 @RequestMapping(value = "/publish", method = RequestMethod.POST)  
 public boolean publish(@RequestBody Reply reply, HttpSession session){  
 reply.setSendUserId(((SystemUser)session.getAttribute("user")).getId());  
 return replyService.publishReply(reply);  
 }

{

    "content": "my first reply",

    "acceptUserId":"1",

    "commentId":"2"

}

获取评论下的回复  
 @RequestMapping(value = "/getreply", method = RequestMethod.GET)  
 public List<ReplyListInfo> getReply(@RequestParam long commentId, @RequestParam long begin, @RequestParam long length){  
 return replyService.getReply(commentId,begin,length);  
 }  
}

### 收藏相关接口

@RestController  
@RequestMapping(value = "collect")  
public class CollectController {

收藏分享  
 @RequestMapping(value = "/add", method = RequestMethod.POST)  
 public boolean add(@RequestBody Collect collect,HttpSession session){  
 collect.setUserId(((SystemUser)session.getAttribute("user")).getId());  
 return collectService.add(collect);  
 }

{

    "shareId": "2"

}

获取用户收藏的分享  
 @RequestMapping(value = "/get", method = RequestMethod.GET)  
 public List<Collect> get(HttpSession session){  
 return collectService.getCollectList(((SystemUser)session.getAttribute("user")).getId());  
 }  
}

不用传参

### 关注相关接口

@RestController  
@RequestMapping(value = "/follow")  
public class FollowController {

关注用户  
 @RequestMapping(value = "/follow", method = RequestMethod.POST)  
 public boolean follow(@RequestBody Follow follow, HttpSession session){  
 follow.setFromUserId(((SystemUser)session.getAttribute("user")).getId());  
 return followService.insertFollow(follow.getFromUserId(),follow.getToUserId());  
 }

{

    "toUserId":"1"

}

获取当前用户关注的用户  
 @RequestMapping(value = "/getfollowinguser", method = RequestMethod.GET)  
 public List<UserListInfo> getFollowingUser(HttpSession session){  
 return followService.getFollowingUser(((SystemUser)session.getAttribute("user")).getId());  
 }

获取当前用户的粉丝  
 @RequestMapping(value = "/getfans", method = RequestMethod.GET)  
 public List<UserListInfo> getFans(HttpSession session){  
 return followService.getFans(((SystemUser)session.getAttribute("user")).getId());  
 }  
}

### 管理员相关接口

@RestController  
@RequestMapping(value = "/admin")  
public class AdministratorController {

更新用户锁定状态  
 @RequestMapping(value = "/updateuserlockstatus",method = RequestMethod.POST)  
 public boolean updateUserLockStatus(@RequestBody SystemUser user){  
 return userService.updateUserLocked(user.getId(),user.isLocked());  
 }

{

    "id":"3",

    "locked":1

}

1为锁定，0为未锁定

添加屏蔽关键字  
 @RequestMapping(value = "/insertsensitivekeyword",method = RequestMethod.POST)  
 public boolean insertSensitiveKeyword(@RequestBody Map<String,String> keywordMap){  
 return sensitiveKeywordService.insertSensitiveKeyword(keywordMap.get("keyword"));  
 }

{

    "keyword":"sb"

}

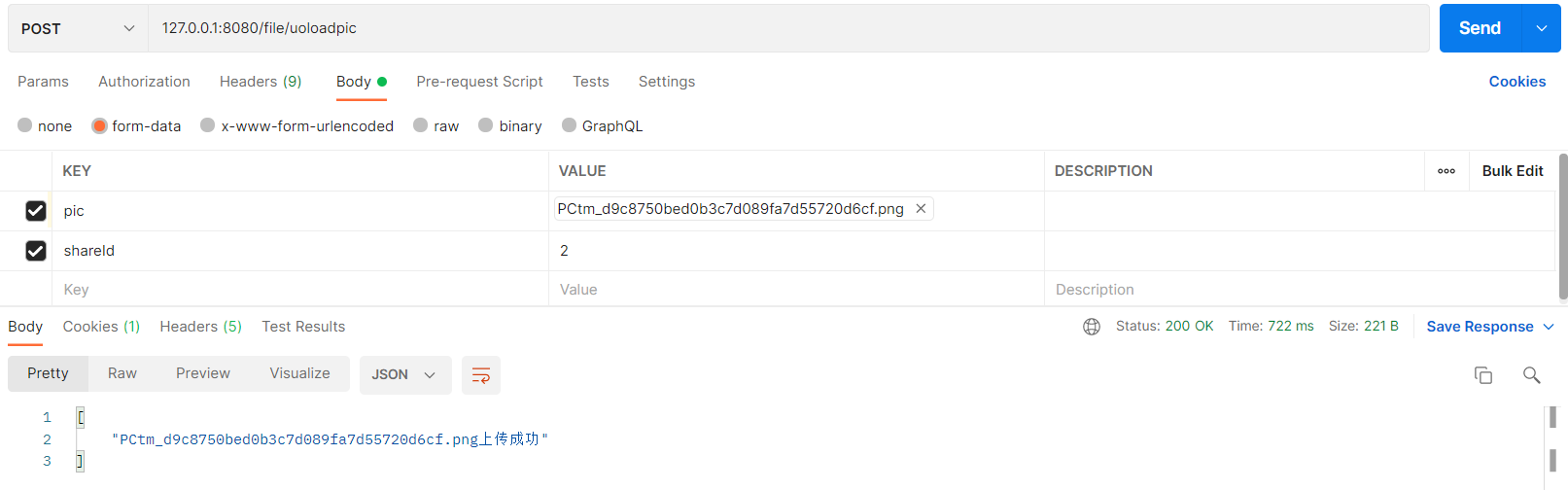
删除屏蔽关键字  
 @RequestMapping(value = "/deletesensitivekeyword",method = RequestMethod.DELETE)  
 public boolean deleteSensitiveKeyword(@RequestParam(value = "id",required = true) long id){  
 return sensitiveKeywordService.deleteSensitiveKeyword(id);  
 }  
}

显示屏蔽关键字列表

### 文件相关接口

@RestController  
@RequestMapping(value = "/file")  
public class FileController {

上传分享的图片  
 @RequestMapping(value = "/uoloadpic")  
 public List<String> uploadPicture(@RequestParam(value = "pic")MultipartFile[] multipartFiles, @RequestParam(value = "shareId")long shareId){  
 return fileService.uploadPicture(multipartFiles, shareId);  
 }



上传头像  
 @RequestMapping(value = "/uoloadavatar")  
 public boolean uploadAvatar(@RequestParam(value = "avatar")MultipartFile multipartFile, @RequestParam(value = "userId")long userId){  
 return fileService.uploadAvatar(multipartFile, userId);  
 }

获取图片  
 @RequestMapping("/get")  
 public ResponseEntity<Resource> get(String url) {  
 try {  
 return ResponseEntity.ok(resourceLoader.getResource("file:" + System.getProperty("user.dir") + "/" + url));  
 } catch (Exception e) {  
 return ResponseEntity.notFound().build();  
 }  
 }  
}