ポートフォリオ説明書

Wu jianchen

一、web サービス簡単紹介

web サービスの構築についての記事をシェアーする SNS システムである．大まかにユーザーサイトと管理サイト二つに分けられた．

web サービス名は「toshare」で，すべての http リクエストは「toshare」から始まります．

マッピングの xml ファイルは「src/main/resources/com/wjc」に置きます．

データベースにすべてのパスワードは「123456」です．

不足：１．Web サービスの半ばまで，POJO とデータベースのマッピングに camel-case ネーミング規則(「mybatis.configuration.map-underscore-to-camel-case=true」)を使っていないことを意識したが，そのまま続けました．またメソッドや，ウェブページなどのネーミングも多少違和感があると思います．

２．高凝集低結合について，外部からインポートの js コードに重複しているコードがあります．

３．メソッドに対するコメントが足りません．

４．Web サービスを作りながら，MySQL の使用方法を勉強していたため，最初にデータベースの最適化を考えませんでした．一つのリクエストに対して，実際に一、二回データベースにアクセスして済ですが，何度も MySQL にアクセスするケースが多いです．時間不足の原因で，一つずつを改善しません．その点どんどん注意して，後で MySQL にアクセスするとき，できる限り最適化できるようにしました．詳細について，説明文の最後のセクション「MySQL 最適化」に書いてあります．

二、開発ツールと技術

1．ツール

- Framework: spring boot 2.3.4
- IDE: IntelliJ IDEA
- パーケッジ管理: Maven 3.0
- DBMS: MySQL Workbench 8.0
- DB mapper: mybatis 2.1.3
- テンプレート　エンジン: thymeleaf
- UI Framework: Bootstrap 4.5.3
- Log: Facade:slf4j 実装:logback
- Test: junit, postman

- 認証と許可：shiro 1.5.3
- パスワードの暗号化：MD5+salt

2. 技術

- Front end：HTML+CSS+jQuery 3.5.1
- Back end：java 8

三、機能

1. ユーザーサイトの機能

(1) Guest

① 記事一覧，詳細表示，記事検索

- 記事一覧（図１）

Package:src/main/java/com/wjc/type/controller/TypeController.java
Request:@GetMapping("/main")
Method:public String find1stType

Guest さんはアドレスバーで「toshare/main」を入力するか，またはナビゲーションバーの一番左側の「ToShare」ロゴをクリックすると，すべての記事一覧画面に移動します.

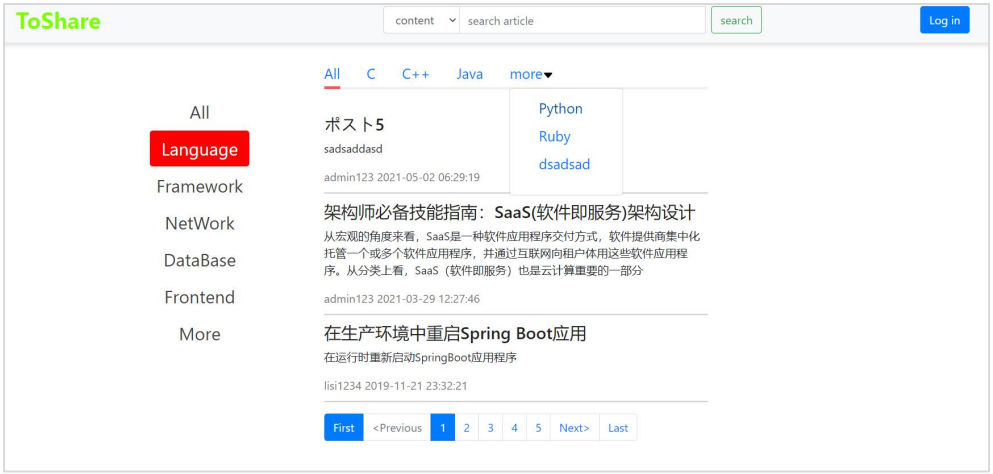

図１　記事一覧

- 記事詳細表示（図 2）

Package:src/main/java/com/wjc/post/controller/postShow.java
Request:@GetMapping("{pid}")
Method:public String showPost()

Guest さんはすべての記事一覧画面で任意の記事のタイトルをクリックする（または下側のページネーションで任意のページの番号をクリックして，記事のタイトルをクリックする），またはアドレスバーで「toshare/記事の番号」を入力すると，この記事の詳細表示画面に移動します．しかし「like」「favorite」「follow」「comment」

機能を利用する権限がありません.

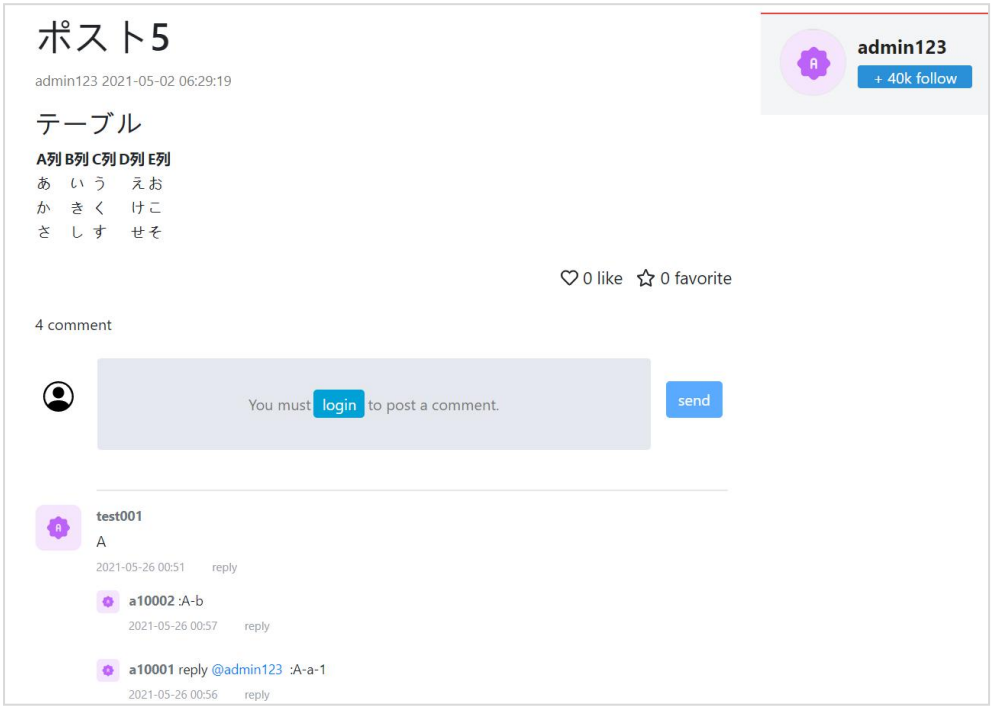

図 2　記事詳細表示

- 記事検索(図 3)

Package:src/main/java/com/wjc/post/controller/postShow.java

Request:@PostMapping("/search")

Method:public String search()

Guest さんはナビゲーションバーで「search」機能を利用することが可能です.
左側のボタンで記事の内容、タイトルと作者三つの範囲で記事を検索できます.

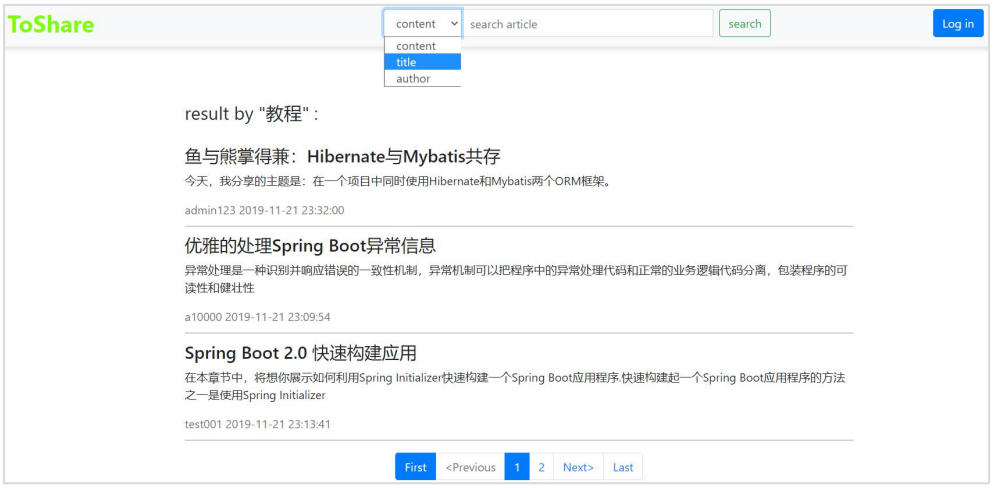

図 3　記事検索

② User のホームページへの訪問(図 4)

Package:src/main/java/com/wjc/user/controller/myPage.java

Request:@RequestMapping("/user")@PostMapping("{no}")
Method:public String personPage()
Guest さんはアドレスバーに「toshare/user/ユーザーの番号」を入力するか，記事一覧または記事詳細画面で作者の名前あるいはアバターをクリックすると，このユーザーのホームページに移動します．

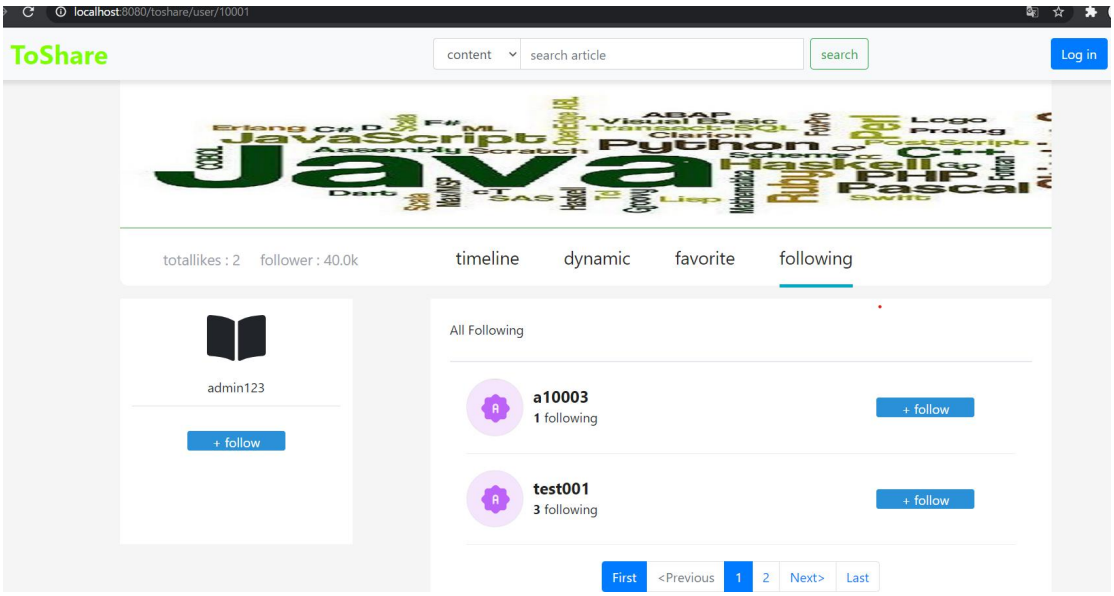

図 4　ユーザーページ訪問

③　新しいユーザー登録(図 5)

- Front end validation

Page hopping：src/main/java/com/wjc/user/controller/PageController.java
Request：@GetMapping("/registration")
Method：public String toRsgister()

Front end に jquery で各入力項目に対して，非 null、username exist、mailbox のフォーマット、長さ、パスワードと確認パスワード一致かどうかの validation を行うことがあります．すべての validation を通過する限り，フォームをサーバーに提出します．

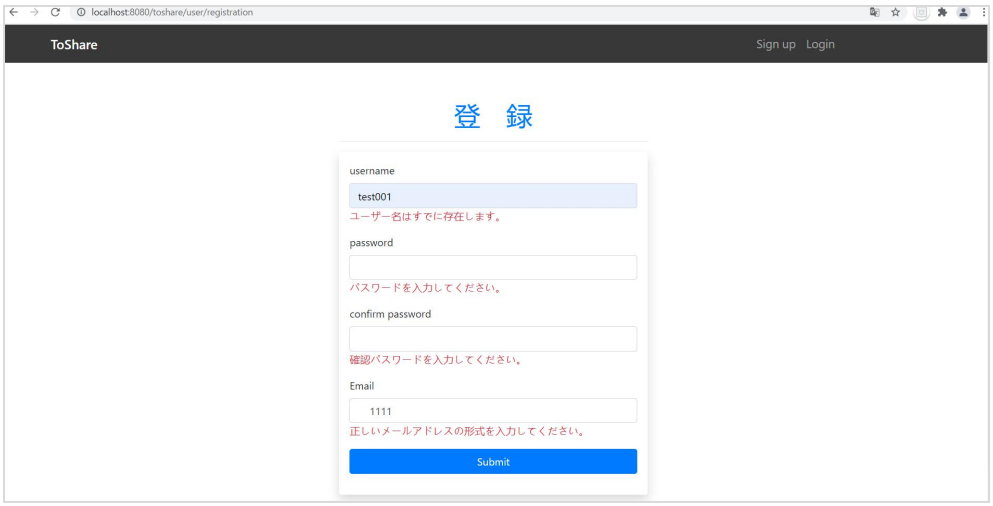

図 5　登録

- Back end validation

Package：src/main/java/com/wjc/user/controller/regist.java
Request：@RequestMapping("/user")@PostMapping("/registration")
Method：public String register()

安全性を保障する前提としてサーバーサイドにクライアントサイドと同じような validation を行います．エラーメッセージを map に格納し，登録画面に提示します．すべての validation
を通過する限り，データベースに保存します．

またデータベースの中,パスワードを MD5 と salt を 1024 回ハッシュ化された後，salt 値と一緒に保存されています(一部のパスワードを plaintext で保存されている図 6)．

登録が成功すると，記事一覧画面「toshare/main」にリダイレクトします．

username	password	email	userno	followrnum	followernum	registdate	salt	role
admin123	101010	aa@aa.cn	10001	5	40000	2021-01-01 03:41:48		user
lisi12345	123456	lisi12345@163.com	10002	6	4	2021-01-02 03:41:48		user
test001	test001	test001@gamil.com	10003	0	3	2021-01-03 03:41:48		user
lisi1234	123456	lisi1234@163.com	10004	1	3	2021-01-04 03:41:48		user
a10000	123456	a10000@1.com	10005	0	2	2021-02-02 03:41:48		user
a10001	123456	a10001@1.com	10006	1	1	2021-02-03 03:41:48		user
a10002	123456	a10002@1.com	10007	1	2	2021-02-05 23:46:46		user
a10003	123456	a10003@1.com	10008	5	1	2021-02-05 23:55:27		user
a10009	5c9ef7bd4f82b612d841dbf495e9c...	a10009@1.com	10009	0	0	2021-05-04 15:52:31	'BVDvDge	user
a10010	7353b8b9bc784c77083de91864cc1...	a10010@1.com	10010	0	0	2021-05-04 15:55:36	m8A0X...	user
a10011	3c5f35d89c349e1a678fc871b5e05...	a10011@1.com	10011	0	0	2021-05-04 15:58:43	CyKJzT2X	user
a10012	e0af841cec0ee2c4a665a93db415e...	a10012@1.com	10012	0	0	2021-05-04 16:10:52	dcA0zG7!	user
a10013	c79dac336d4a0bbc4959bf709c630...	a10013@1.com	10013	0	0	2021-05-25 05:58:14	bd2Mxxnb	user
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

図 6　MD５+salt パスワード

(2) User

① Guest のすべての機能
② ログイン・ログアウト機能

- ログイン(図 7)

Page hopping：src/main/java/com/wjc/user/controller/PageController.java
Request：@RequestMapping("/user")@GetMapping("/login")
Method：public String loginPage()

アドレスバーで「toshare/user/login」を入力するか，記事一覧画面「toshare/main」に「Login」ボタンをクリックするか，ユーザーのホームページと記事一覧画面にホップアップした login モーダルウィンドウから login 画面に移動できます．

一つブラウザで一つだけのユーザーをログインできます．ログインしたユーザーはアドレスバーで「toshare/user/login」を入力しても，記事一覧画面に移動します．

登録と同じように front end と back end 両方 validation があります．Back end の validation は shiro の login メソッドを利用しています．詳細について，説明文の「権限管理」の「認証」セクションに参照してください．成功すると，記事一覧画面にリダイレクトします．

Validation：
Package：src/main/java/com/wjc/user/controller/login.java
Request：@RequestMapping("/user")@PostMapping("/login")
Method：public String userLogin()

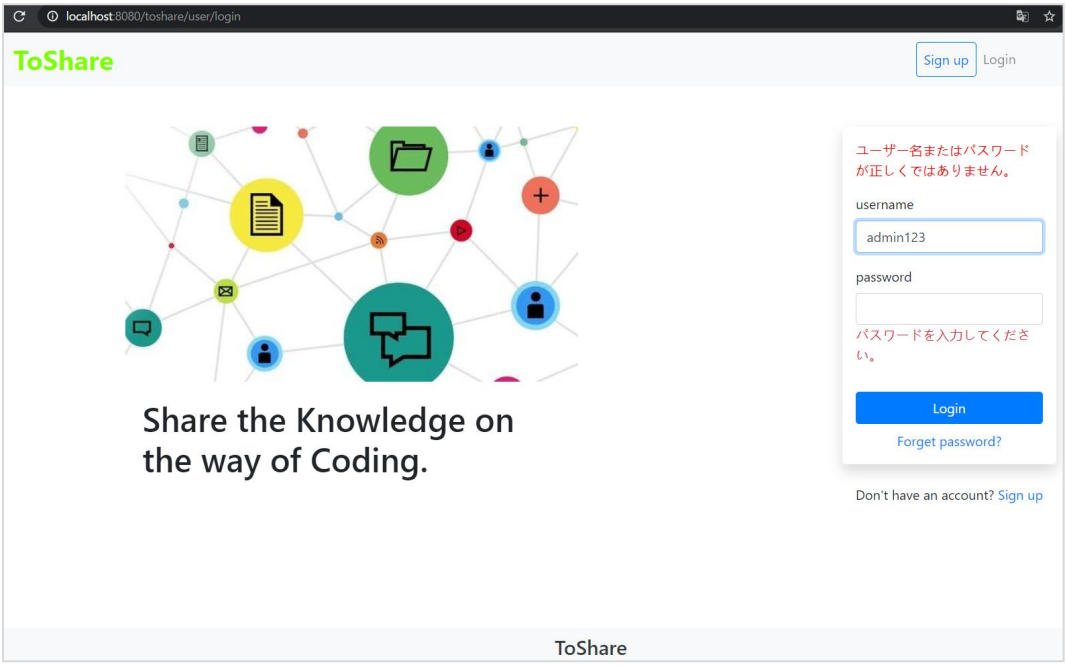

図 7　ログイン

- ログアウト(図 8)

Package：src/main/java/com/wjc/user/controller/login.java
Request：@RequestMapping("/user")@GetMapping("/logout")
Method：public String logout()

ログインした後，ナビゲーションバーの右側にユーザータグをクリックすると，現れたメニューに「Logout」をクリックして，ユーザーが session から退出して，ログイン画面にリダイレクトします．ログインのように shiro の logout メソッドを

利用しています.

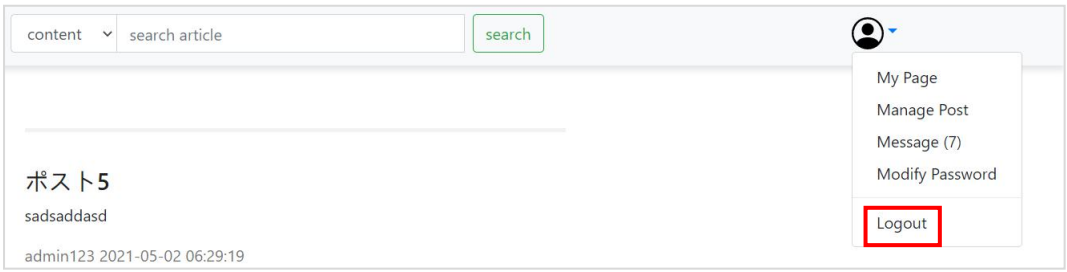

図 8　ログアウト

③　パスワード更新

Page hopping：src/main/java/com/wjc/user/controller/PageController.java
Request：@RequestMapping("/user")@GetMapping("/password")
Method：public String updatePwd()

アドレスバーで「toshare/user/password」を入力するか,上の図 8 のように「Modify Password」をクリックして，パスワード更新画面に移動します.

Package：src/main/java/com/wjc/user/controller/login.java
Request：@RequestMapping("/user")@PostMapping("/password")
Method：public String update()

現在のパスワードと更新するパスワードを入力して，現在のパスワードを間違いない場合，成功メッセージを提示されます．図 9 は失敗した場合です.

図 9　更新失敗

④　コメント投稿，「like」「favorite」「follow」機能

- コメント投稿

Package：src/main/java/com/wjc/post/controller/postShow.java
Request：@PostMapping("/comments")
Method：public String saveComments()

js：src/main/resources/static/js/article.js
function：postData

ユーザーは各記事の一番下にコメントを投稿とコメントを返事することができます(図 10).

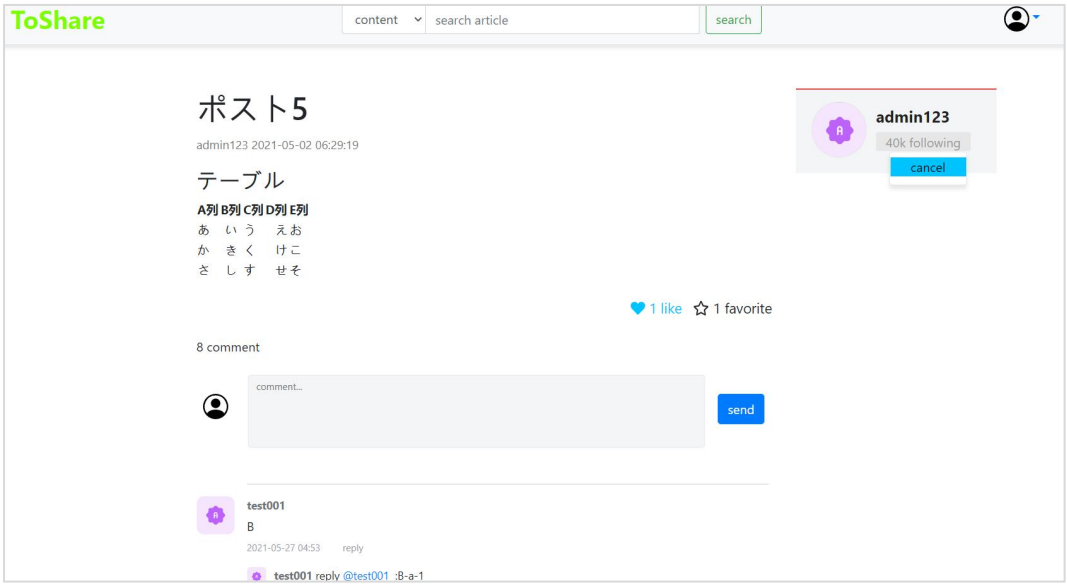

図 10　コメント

● 「like」「favorite」「follow」
Package：src/main/java/com/wjc/action/controller/changeLike.java
src/main/java/com/wjc/action/controller/changeFavorite.java
src/main/java/com/wjc/action/controller/changeFollow.java
Request：@PostMapping("/like")
@PostMapping("/favorite")
@PostMapping("/follow")

js：src/main/resources/static/js/article.js
function：changeFollow,changeFavorite,changeLike

図 10 のようにユーザーは各記事の末に「like」または「favorite」ボタンを一回クリックして，jQuery の Ajax で記事が「like」または「favorite」状態になります.もう一回クリックすると，最初の状態に戻ります．「follow」は同様に，ボタンを一回クリックして，この作者をフォローします．フォローした後，「cancel」をクリックすると，フォローしていない状態になります.

⑤　ホームページ各種記事一覧(図 11)
Package：src/main/java/com/wjc/user/controller/myPage.java
Request：@RequestMapping("/user")@GetMapping("{no}")
Method：public String personPage()

js：src/main/resources/static/js/user/myPage.js

ユーザーはアドレスバーで「toshare/user/ユーザーの番号」を入力するか，ナビゲーションバーにユーザータグに「my page」をクリックすると，ユーザー自身のホームページに移動します．

ホームページのメニューバーの各タグをクリックすると，jQuery の Ajax でタグにより記事を表示します．図 11 のように「totallikes」をクリックしたら，今まで「like」を獲得した記事を表示されています．

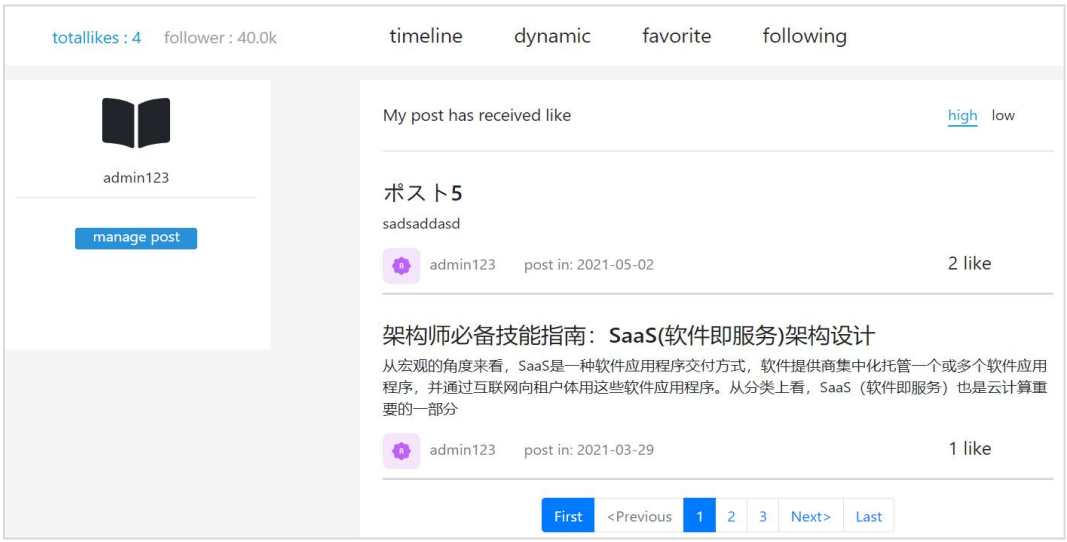

図 11　「totallikes」

左から「follower」をクリックすると，自分をフォローしているユーザーを表示されます．「timeline」をクリックすると，自分が発表した記事を表示されます．「dynamic」をクリックすると，自分がフォローしているユーザーが発表した記事を表示されます．「favorite」をクリックすると，自分が「favorite」ボタンをクリックした記事を表示されます．図 12 のように自分がフォローしているユーザーを表示されます．

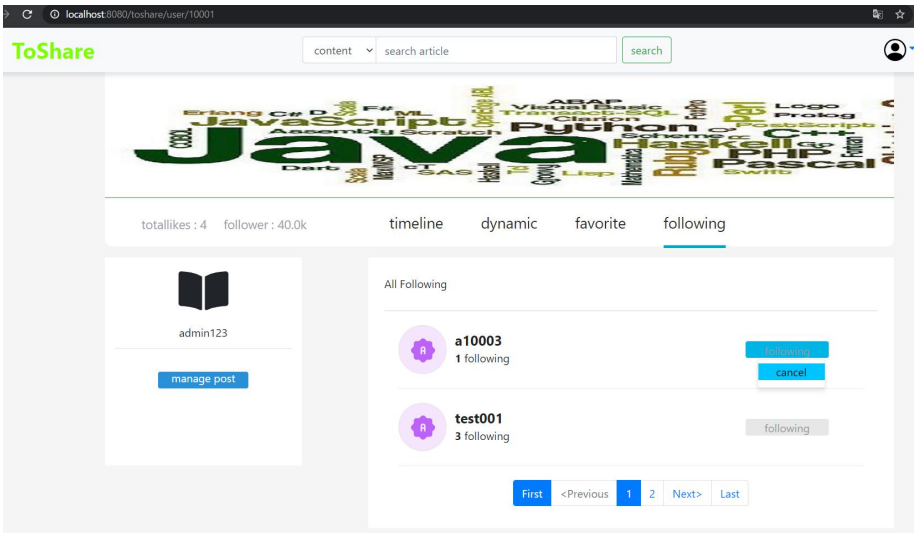

図 12　「following」

⑥　記事管理:投稿，編集，削除

Package：src/main/java/com/wjc/user/controller/managePost.java

Request：@GetMapping("/{username}/myPost")

Method：public String showMyPost()

js：src/main/resources/static/js/user/managePost.js

アドレスバーで「toshare/ユーザー名/myPost」図 13 のように「manage post」をクリックすると，ユーザーの記事管理画面に移動します.

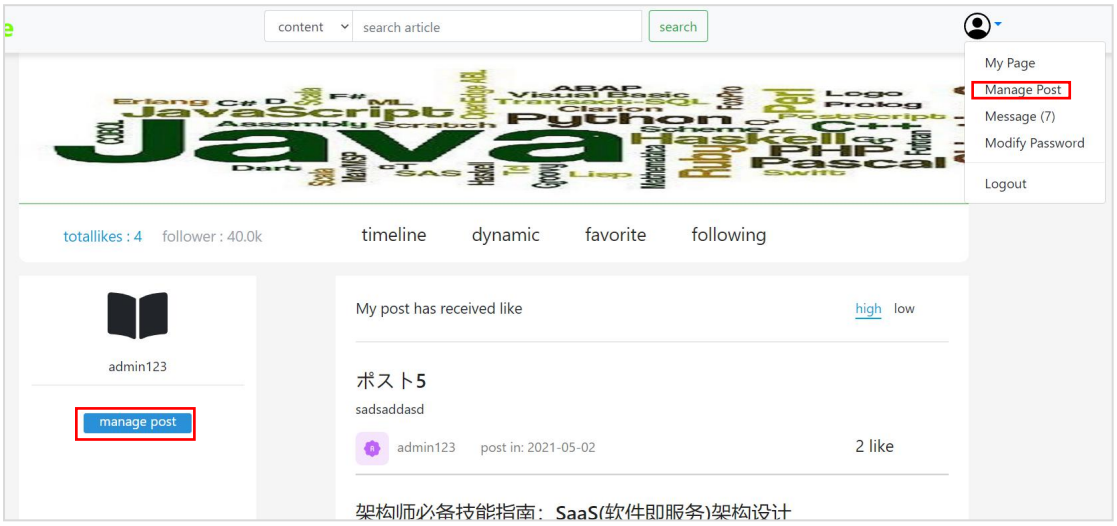

図 13　「manage post」

記事管理画面，にユーザーは審査を通過し，成功に発表した記事を表示されます.また図 14 のように，上の Multi-level Menu に super type と sub type を選択して，「Check Post」ボタンをクリックすると，この type の記事を表示されます.

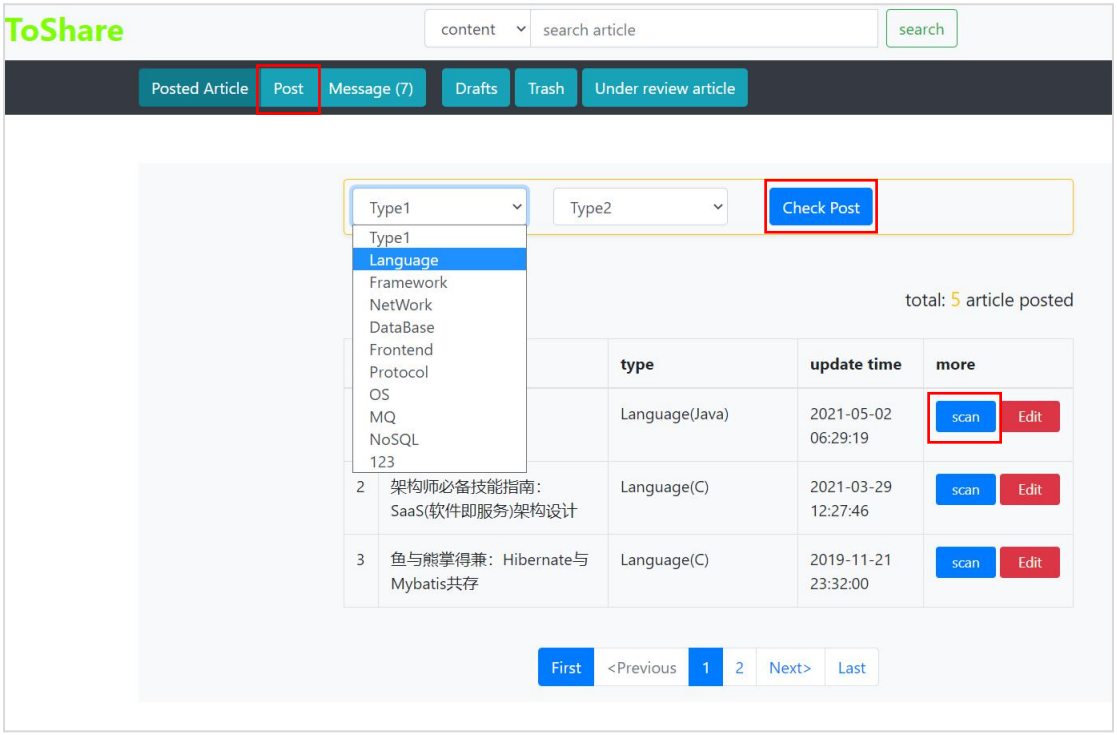

図 14　「Posted Article」

Package：src/main/java/com/wjc/user/controller/PostController.java
Request：@RequestMapping("/{username}")@GetMapping("/edit/{pid}")
Method：public String editArticle()

図 14 のように，記事テーブルの右側の「scan」ボタンをクリックすると，この記事の HTML と Markdown 二つの型で表示されます.
「scan」モードで表示される記事は編集と投稿機能を利用できません.

- 投稿

Page hopping：src/main/java/com/wjc/user/controller/PostController.java
Request：@RequestMapping("/{username}")@GetMapping("/post/new")
Method：public String post()

js：src/main/resources/static/js/user/doPost.js
Plugin：Editor.md

アドレスバーで「toshare/ユーザー名/post/new」を入力するか，図 14 のように「Post」ボタンをクリックすると，投稿画面に移動します(図 15).

Package：src/main/java/com/wjc/user/controller/managePost.java
Request：@RequestMapping("/{username}/myPost")@PostMapping("/postArticle")
Method：public String postArticle()

必要な項目を入力すると，下側に「post」、「save」二つ選択肢があり，「post」

をクリックすると，記事が審査状態になり，審査中の記事を記事管理画面に「Under review article」をクリックすると，閲覧できます．「save」をクリックすると，記事が草稿状態になり，この記事が記事管理画面に「Drafts」をクリックすると，閲覧、再編集または削除することができます．

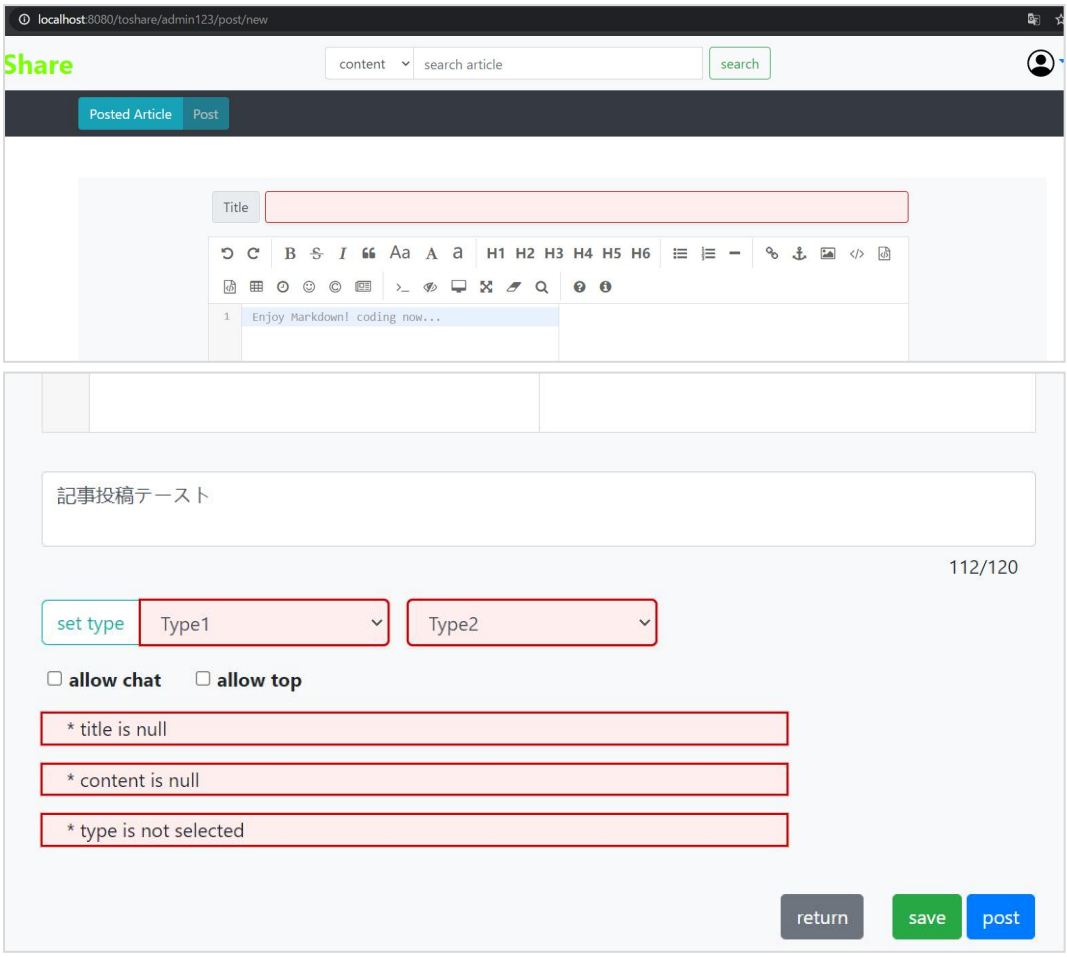

図 15　投稿

- 編集

Page hopping：src/main/java/com/wjc/user/controller/PostController.java
Request：@RequestMapping("/{username}")@GetMapping("/edit/{pid}")
Method：public String editArticle()

js：src/main/resources/static/js/user/doPost.js

図 14 のように記事管理画面に，発表した記事に「Edit」をクリックすると，再編集できるます．また「Drafts」をクリックすると，発表していない記事を編集できます(図 16).

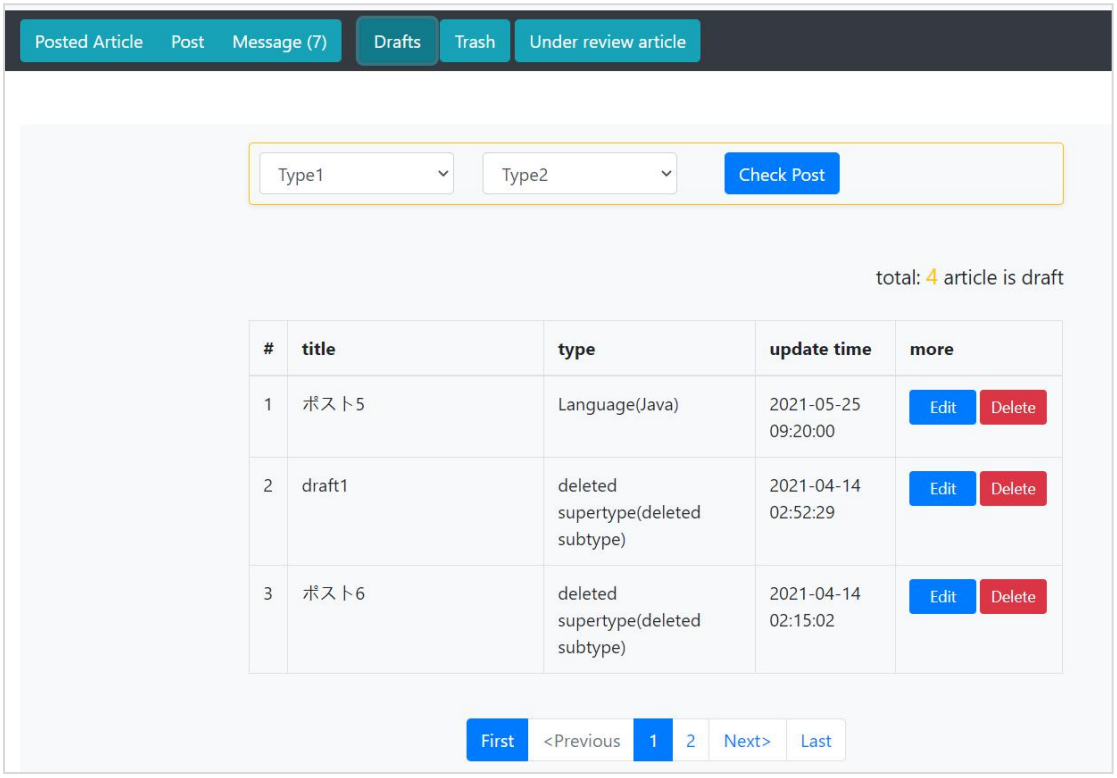

図 16　編集

- 削除

Page hopping：src/main/java/com/wjc/user/controller/PostController.java
Request：@RequestMapping("/{username}")@GetMapping("/delete/{pid}")
Method：public String deleteArticle()

js：src/main/resources/static/js/user/doPost.js

図 17 のように「Trash」をクリックすると，削除した記事を表示されます.

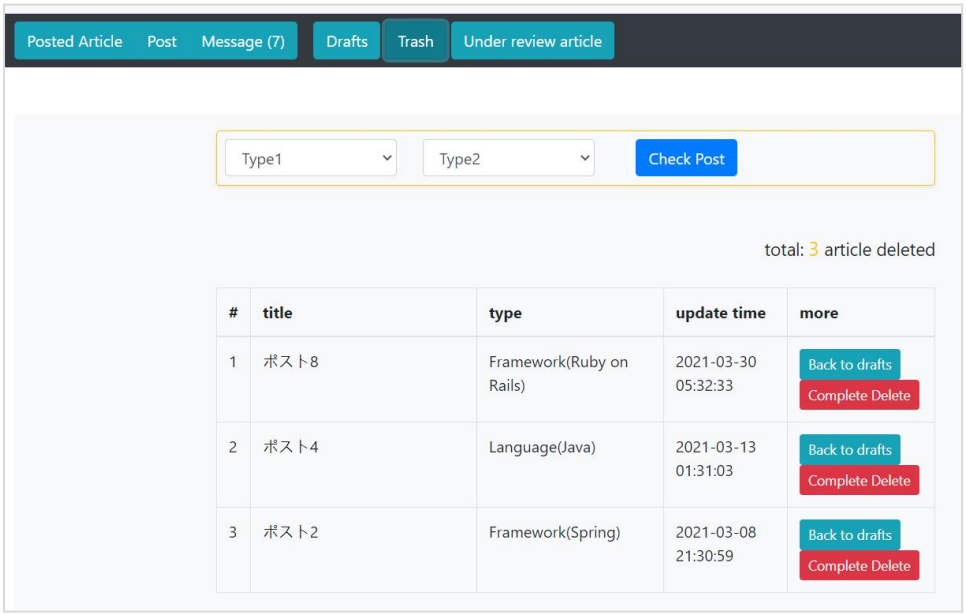

図 17　削除

⑦　記事審査のメッセージのチェック

Package：src/main/java/com/wjc/user/controller/managePost.java

Request：@GetMapping("/{username}/message")

Method：public String message()

js：src/main/resources/static/js/user/message.js

ナビゲーションバーにユーザータグに「message」をクリックするか，または記事管理画面に「Message」をクリックすると，ユーザーの message 画面に移動します(図 18)．Message 画面に入った後，「message()」かっこ内の未読メッセージの数が 0 になります．

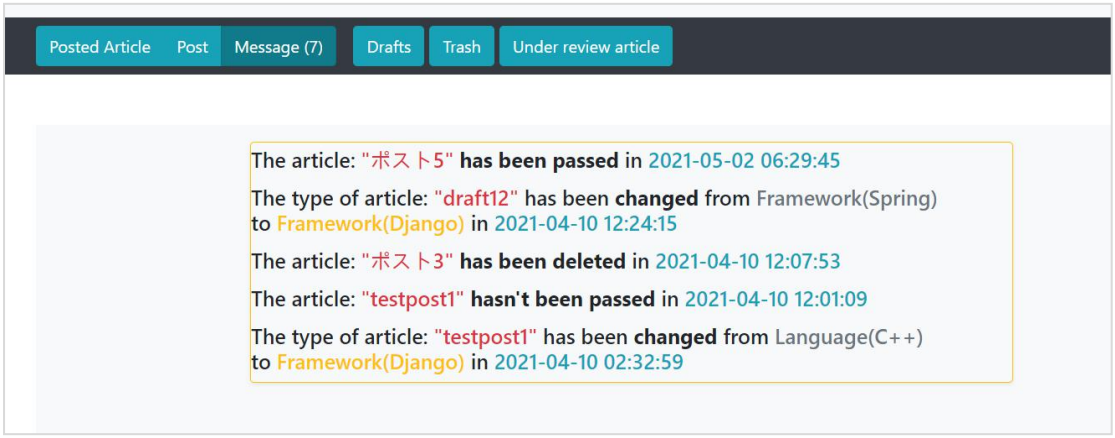

図 18　message

2.　管理サイトの機能

(1) Admin

①　ログイン

Package：src/main/java/com/wjc/admin/controller/LoginController.java

Page hopping：@RequestMapping("/admin")@GetMapping("/login")

Method：public String loginPage()

Request：@RequestMapping("/admin")@PostMapping("/login")

Method：public String login()

アドレスバーで「toshare/admin/login」を入力して，管理サイトのログイン画面に移動します(図 19)．一つブラウザで一つだけの admin(管理者)をログインできます．ログインしたユーザーはアドレスバーで「toshare/admin/login」を入力しても，管理者サイトのホームページ「dashboard」に移動します．

User と同じように back end に validation があります．Back end の validation は shiro の login メソッドを利用しています．成功すると，「dashboard」画面にリダイレクトします．

管理者のアカウントは２つがあります.「admin001 123456」「admin002 123456」

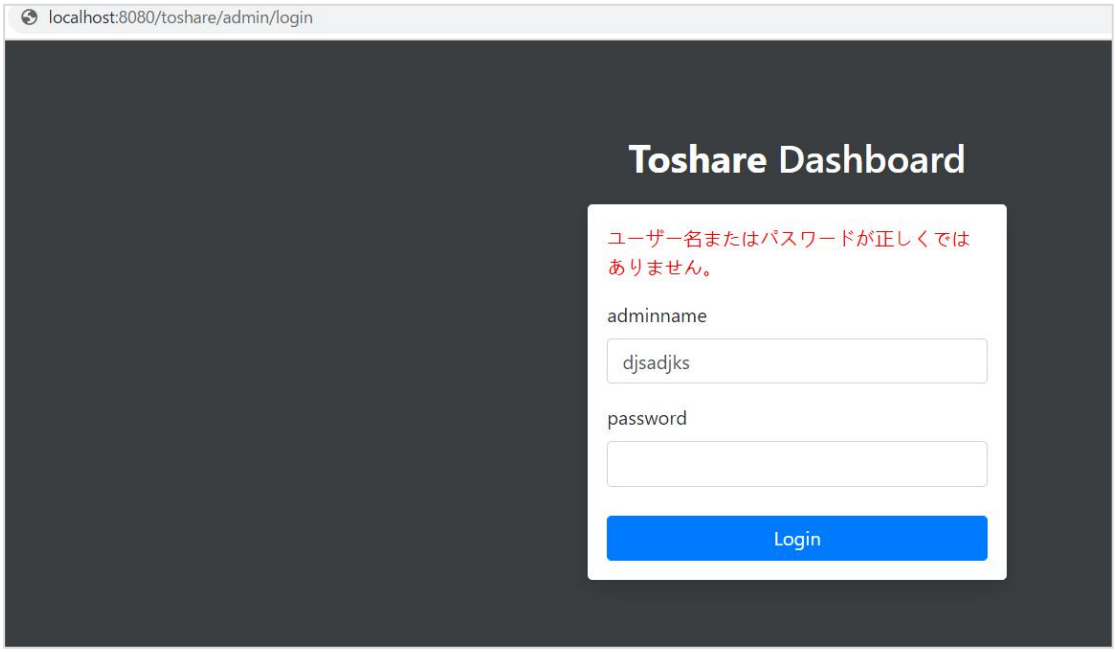

図 19　ログイン

②　記事のタイプの管理：増加，編集，削除

ログインした後，「dashboard」画面に移動します(図 20)．管理者はタイプについて supertype(例：Language)と subtype(例：Language 下の C，C++，Java など)の増加，編集，削除をすることができます.

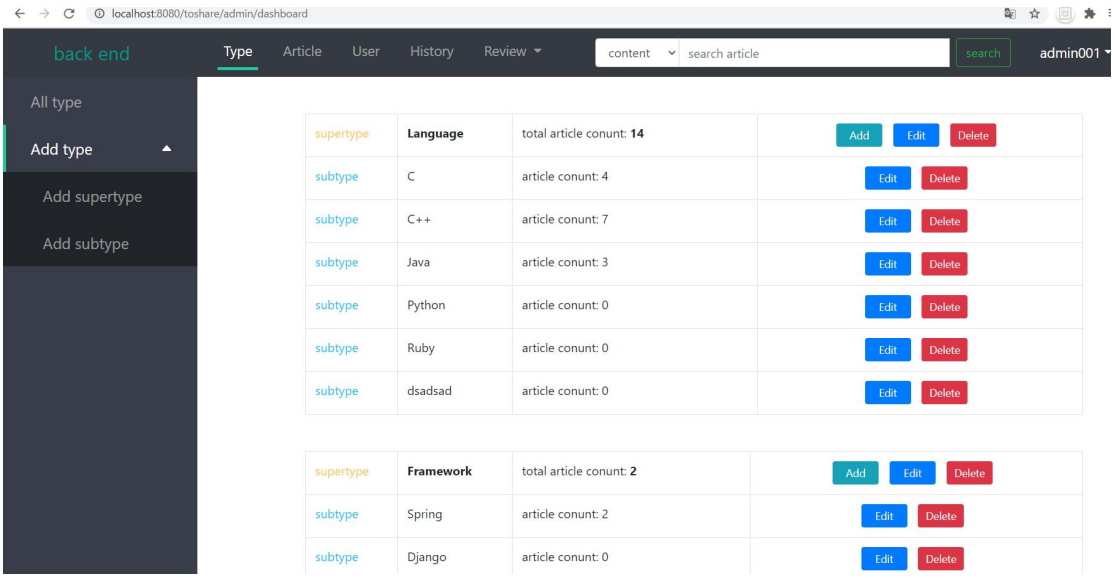

図 20　dashboard

- 増加

Package：src/main/java/com/wjc/admin/controller/LoginController.java

Request：@RequestMapping("/admin")@PostMapping("/addSupertype")

@PostMapping("/addSubtype")
Method：public String addSupertype()　public String addSubtype()

図 20 のようにテーブルの「Add」ボタンをクリックするか，サイドバーの「Add supertype」ボタンをクリックすると supertype を増加する画面に移動します．サイドバーの「Add subtype」ボタンをクリックすると subtype を増加する画面に移動します，subtype を増加する前に所属する supertype を選択することが必要です(図 21)．

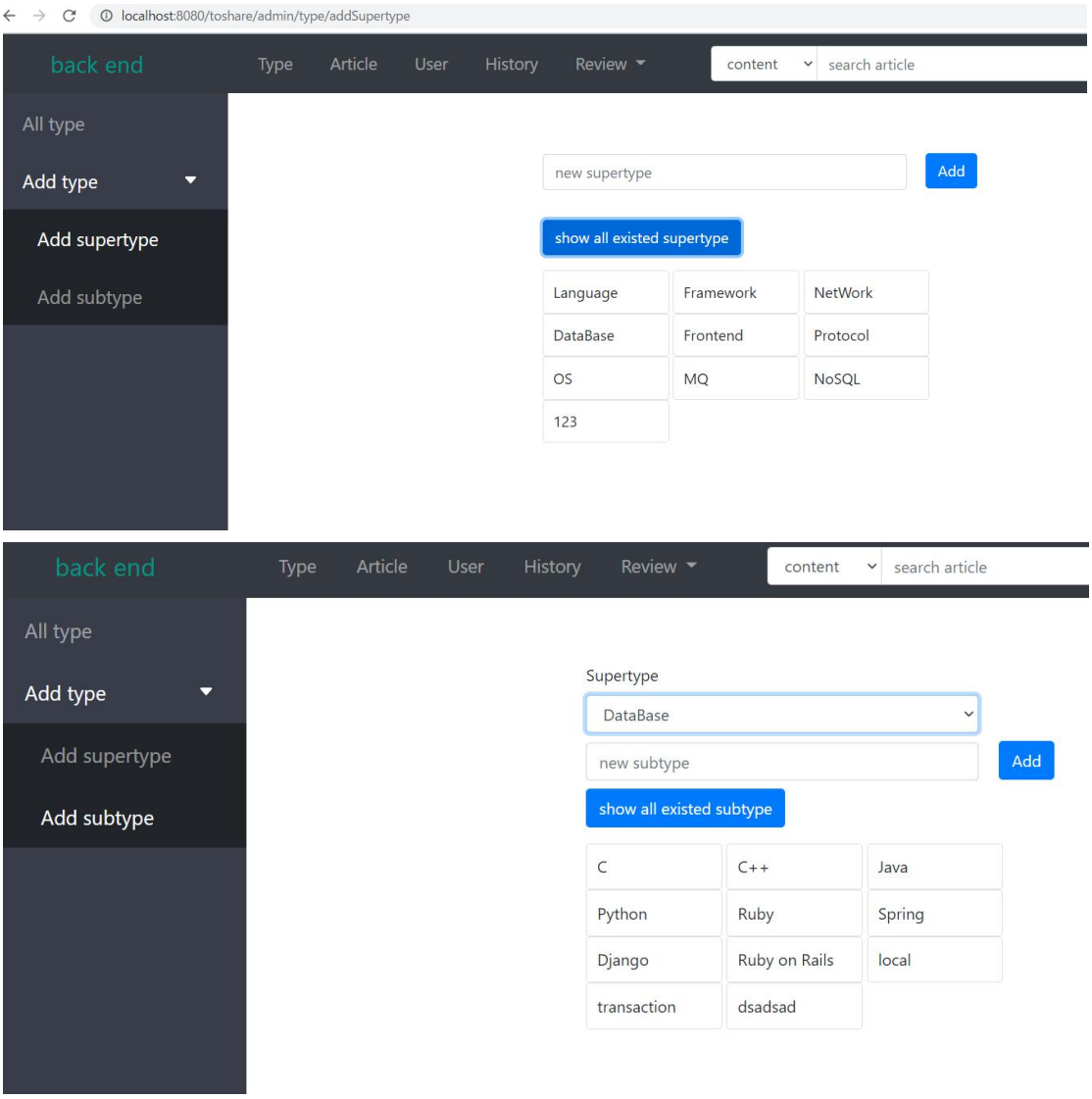

図 21　タイプ増加

- 編集

Package：src/main/java/com/wjc/admin/controller/LoginController.java
Request：@RequestMapping("/admin")@PostMapping("/editType")
Method：public String editType()

「dashboard」画面にテーブルの各タイプの後，「Edit」ボタンをクリックすると，このタイプを編集するモーダルウィンドウが表示されます(図 22).

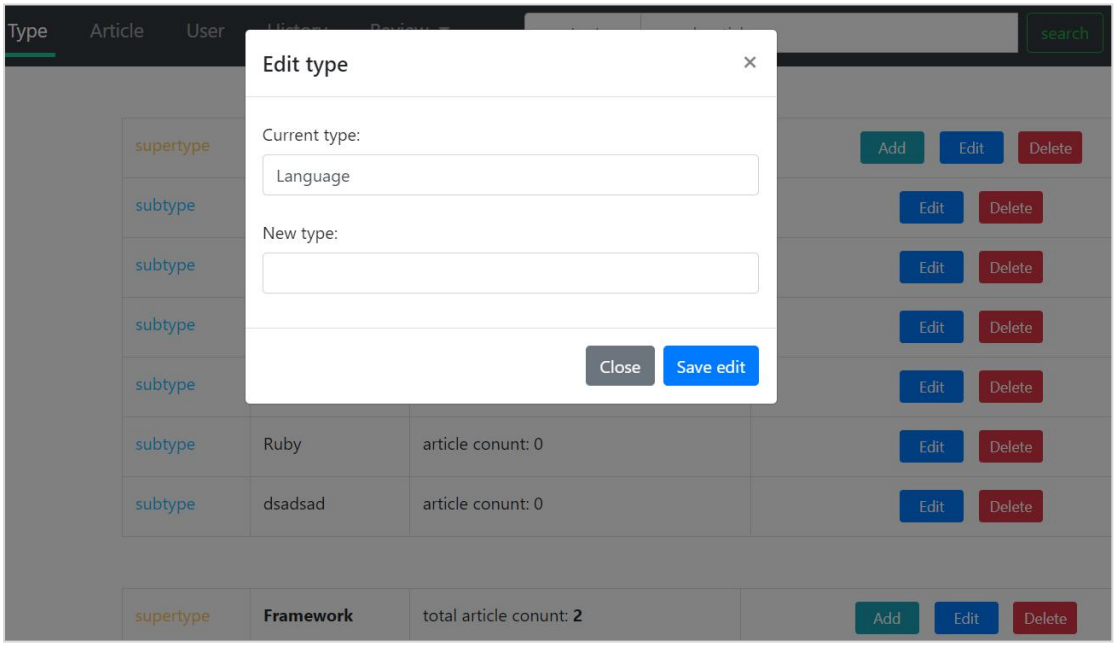

図 21　タイプ編集

- 削除

Package：src/main/java/com/wjc/admin/controller/LoginController.java
Request：@RequestMapping("/admin")@PostMapping("/deleteType")
Method：public String deleteType()

「Delete」ボタンをクリックすると，タイプを削除することができます．ただし supertype を削除する前に，このタイプに属するすべての subtype を削除することが必要です．subtype を削除する前に，このタイプに属するすべての記事をほかの subtype に変更することが必要です．

③　発表した記事一覧

Package：src/main/java/com/wjc/admin/controller/ArticleManage.java
Request：@RequestMapping("/admin")@GetMapping("/article")
Method：public String findAllArticle()

アドレスバーで「toshare/admin/article」を入力するか，ナビゲーションバーで「Article」ボタンをクリックすると，発表した記事一覧画面に移動します(図 22)．

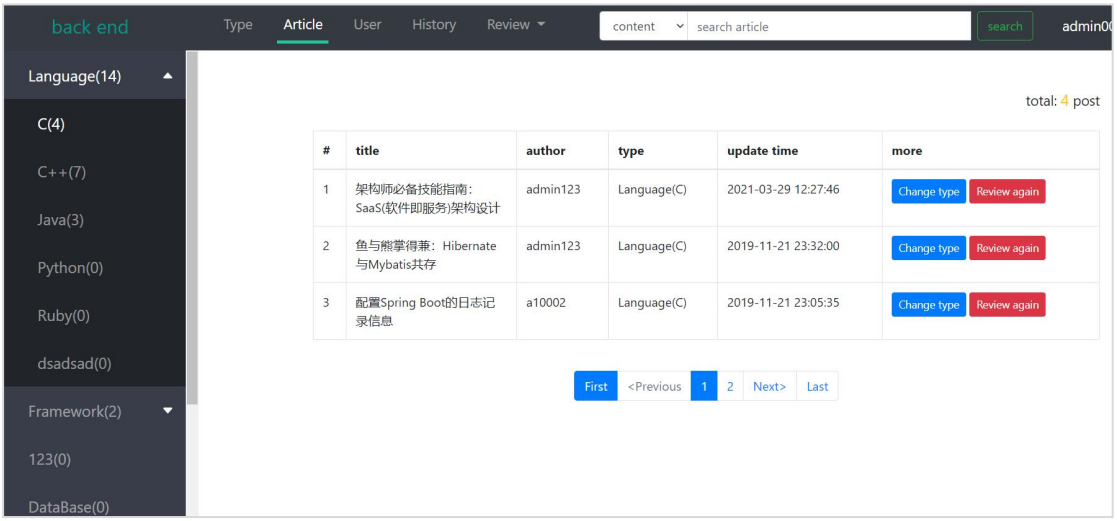

図２２　記事一覧

④　ユーザーの一覧と詳細表示

Package：src/main/java/com/wjc/admin/controller/UserList.java

Request：@RequestMapping("/admin")@GetMapping("/article")

Method：public String findAllArticle()

アドレスバーで「toshare/admin/user」を入力するか，ナビゲーションバーで「User」ボタンをクリックすると，すべてのユーザーの一覧画面に移動します(図２３)．「Detail」をクリックすると，このユーザーの詳細画面に移動します(図 24　ユーザー「admin123」の詳細画面)．詳細画面に左側の一番下にユーザーの発表した記事，審査中の記事，気に入りの記事を表示することができます．

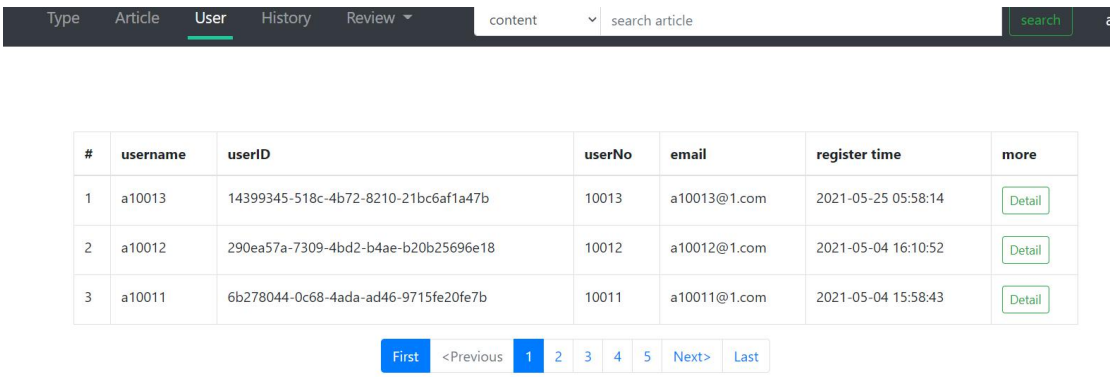

図２３　ユーザー一覧

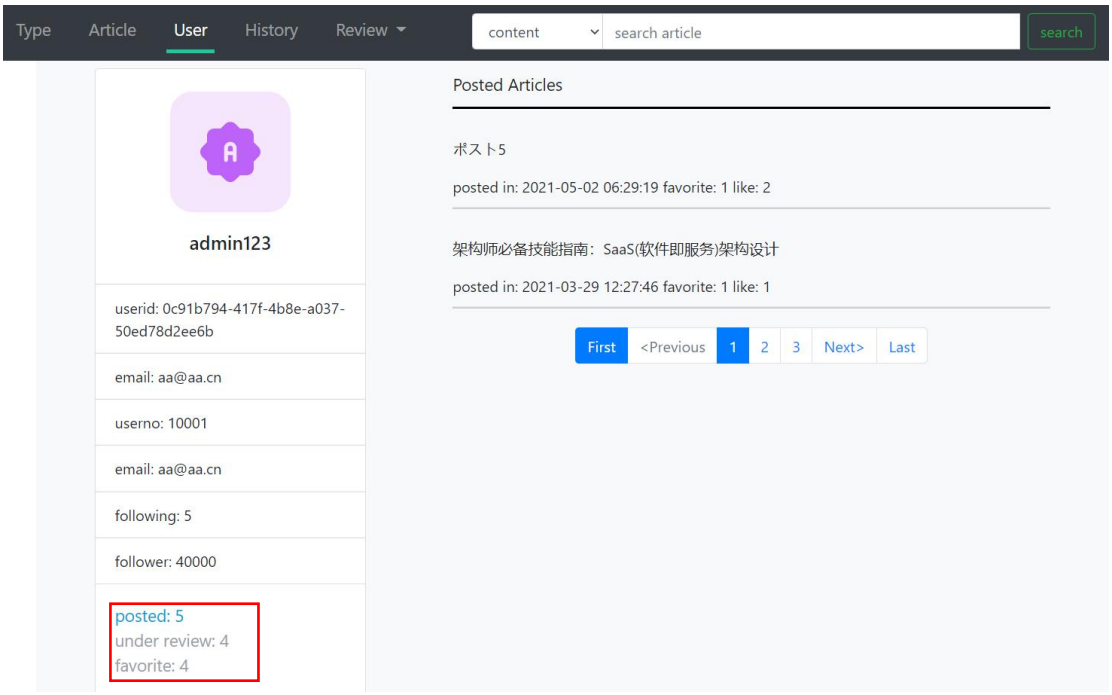

図２４　ユーザー「admin123」の詳細画面

⑤　各管理者の操作の履歴チェック

アドレスバーで「toshare/admin/history」を入力するか，ナビゲーションバーで「History」ボタンをクリックすると，各管理者の操作の履歴に移動します．履歴は二つがあります．

Article history：管理者は記事を審査する履歴(図 25).

Type history：管理者はタイプに対する操作の履歴.

- Article history

Package：src/main/java/com/wjc/admin/controller/History.java

Request：@RequestMapping("/admin")@GetMapping("/history")

Method：public String history_article()

Database：t_review_message

- Type history

Package：src/main/java/com/wjc/admin/controller/History.java

Request：@RequestMapping("/admin")@GetMapping("/history_type")

Method：public String history_type()

Database：t_history_type

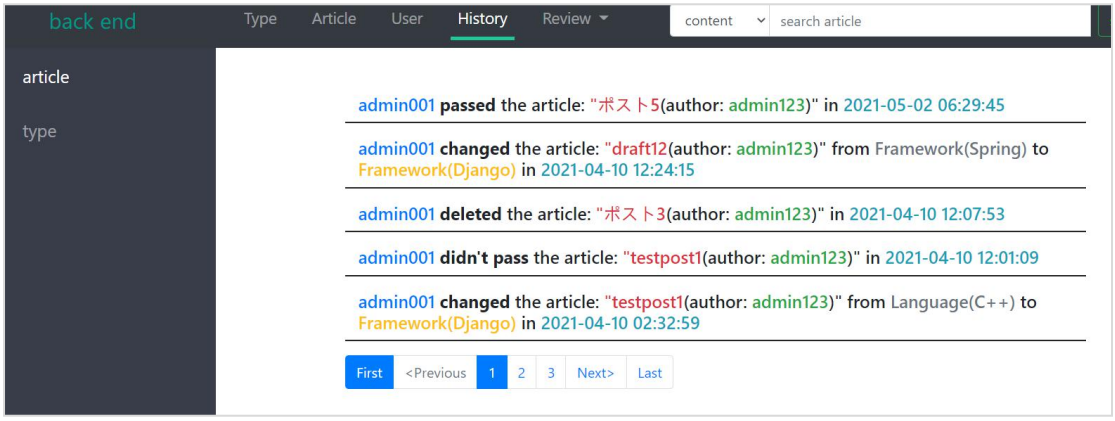

図２５　article history

⑥　記事審査：通過，不通過

Package：src/main/java/com/wjc/admin/controller/ReviewManage.java

Request：@RequestMapping("/admin")@GetMapping("/review")

Method：public String showUnderReviewPosts()

アドレスバーで「toshare/admin/review」を入力するか，ナビゲーションバーで「Review」ボタンをクリックすると，審査中の記事一覧画面に移動します(図 26)．テーブルの黄色「Review」ボタンをクリックすると，この記事の審査画面に移動します(図 27)．

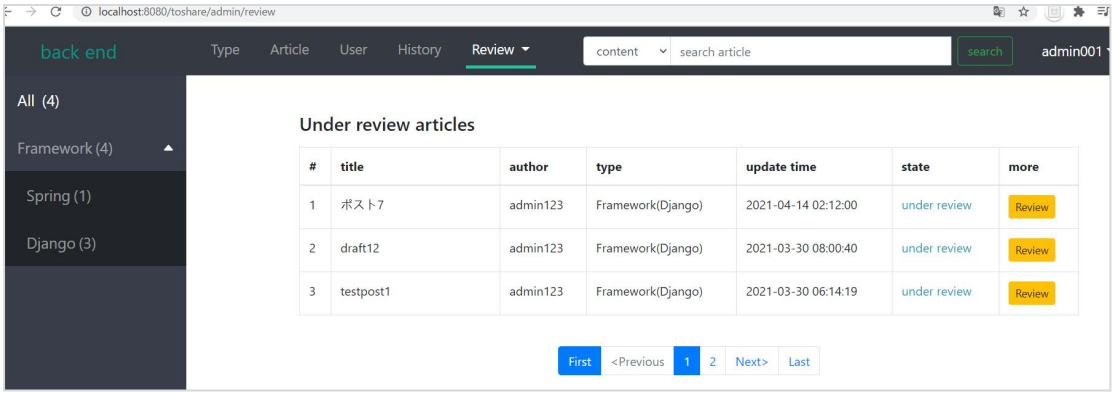

図２６　審査一覧画面

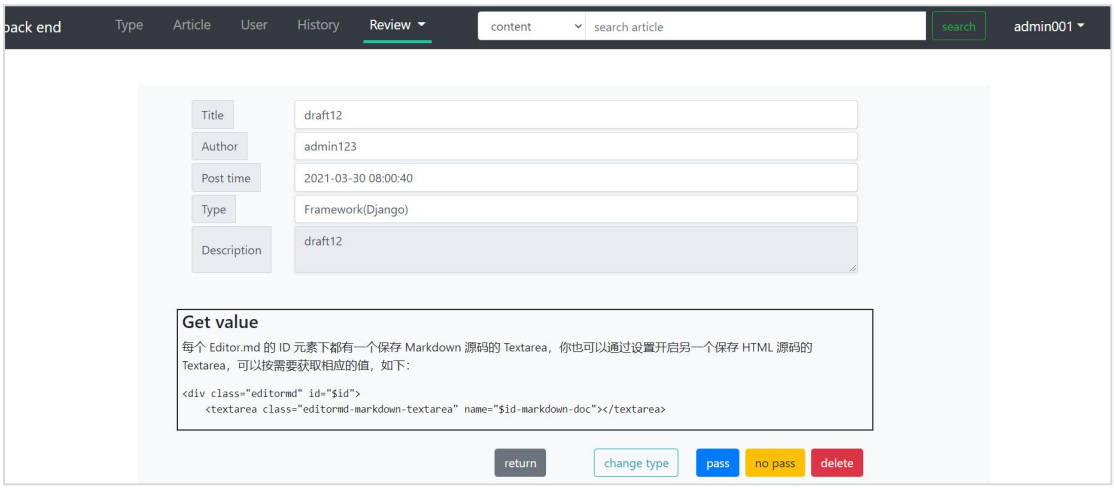

図２７　審査画面

図 27 のように記事の審査に対する「pass」「no pass」「delete」三つの選択があります．各選択はデータベースに記事の状態を変更します(database の t_posts の state)．

t_posts の state について，「２」「１」「0」三つの数字で表します．特に「0」になった記事は「draft」と「deleted」二つ状態があります．

２：「pass」の記事の「state」2 になり，記事一覧画面に表示されます．
１：審査中の記事を表します．
0：deleted：ユーザーは記事管理画面に一時的に削除した記事，「Trash」に入りました．
draft：ユーザーは記事管理画面に編集中の記事，「Drafts」に入りました．

state	draft	deleted
0	0	1
0	1	0
0	1	0
0	0	1
0	1	0
0	1	0
0	0	1
1	0	0
1	0	0
1	0	0
2	0	0

四、権限管理

この web サービスは Role-Based Access Control というストラテジーを使用しています．shiro に基づいてカスタマイズ filter を設定し，ユーザーは「user」(テーブルの t_user の role アイテム)，管理者は「admin」(テーブルの t_admin の role アイテム)というロールをそれぞれにロールの権限を付与します(Guest さんはロールがない)．

1. Filter

Package：src/main/java/com/wjc/config/ShiroConfig.java

src/main/java/com/wjc/shiro/filters/MyRolesAuthorizationFilter.java

```
@Bean
    public ShiroFilterFactoryBean
    getShiroFilterFactoryBean(@Qualifier("getDefaultSM
    ") DefaultWebSecurityManager
    defaultWebSecurityManager){
        ShiroFilterFactoryBean shiroFilterFactoryBean = new
        ShiroFilterFactoryBean();



        Map<String,String> map = new LinkedHashMap<>();

        map.put("/user/login","anon");
        map.put("/user/registration","anon");
        map.put("/main","anon");
        map.put("/*","anon");
        map.put("/search","anon");

        map.put("/*/myPost","roles[user]");
        map.put("/*/message","roles[user]");
        map.put("/*/post/*","roles[user]");
        map.put("/*/edit/*","roles[user]");
        map.put("/*/edit/","roles[user]");
        map.put("/*/delete/*","roles[user]");
        map.put("/*/delete/","roles[user]");
        map.put("/user/password","roles[user]");
        map.put("/user/*","anon");

        map.put("/admin/login","anon");

        map.put("/admin/*","roles[admin]");

        Map<String, Filter> filters = shiroFilterFactoryBean.getFilters();
        filters.put("roles",new MyRolesAuthorizationFilter());

        shiroFilterFactoryBean.setFilterChainDefinitionMap(map);
        shiroFilterFactoryBean.setFilters(filters);

        return shiroFilterFactoryBean;
    }
```

図２８　shiro filter

図 28 のようにログインしていない方は，RolesAuthorizationFilter のメソッドをオーバーライドしたことで，「user」ロールが必要な画面にアクセスしようとすると，自動的に「toshare/user/login」に移動します．同様に「admin」ロールが必要な画面にアクセスしようとすると，自動的に「toshare/admin/login」に移動します．

2．認証

Package：src/main/java/com/wjc/shiro/realms/CustomerRealm.java

doGetAuthenticationInfo をオーバライドして，ログインしている方の身分はユーザーか，管理者かを判断します．

また，データベースの t_user テーブルに，一部のアカウントのパスワードは平文で表示されています．ほかのアカウントのパスワードは MD5+salt で暗号化されている．暗号化のアカウントでログインしたいならば，

src/main/java/com/wjc/config/ShiroConfig.java に

```
    @Bean
    public Realm getRealm(){
        CustomerRealm customerRealm = new CustomerRealm();
//        HashedCredentialsMatcher hashedCredentialsMatcher = new
HashedCredentialsMatcher();
//        hashedCredentialsMatcher.setHashAlgorithmName("MD5");
//        hashedCredentialsMatcher.setHashIterations(1024);
//        customerRealm.setCredentialsMatcher(hashedCredentialsMatcher());
        return customerRealm;
    }
```

コメントアウトした部分を戻して，ログイン認証が進めます．

3．許可

Http リクエストにカスタマイズ filter で権限を管理する以外に，特定の Ajax リクエストに shiro のアノテーション(例：@RequiresRoles("user"))で権限を管理します．例えば guest さんはユーザーをフォローしする権限がないため(「like」「favorite」を利用する権限もない)，フォロー Ajax リクエストを拒否されます(図 29)．

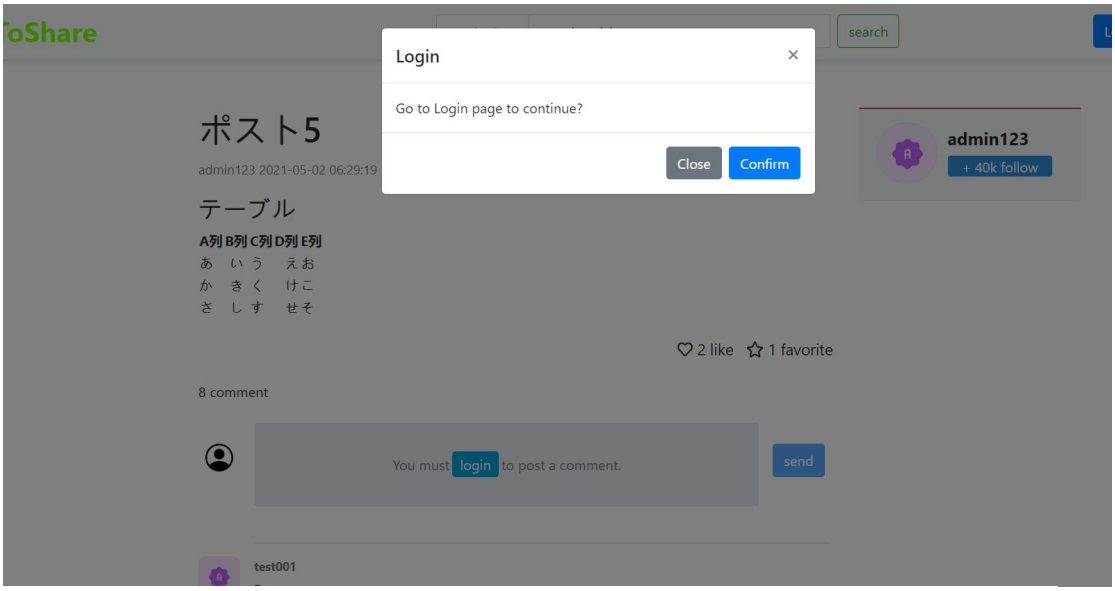

図２９　Ajax リクエスト拒否

五、MySQL 最適化

1. 複数テーブル結合(WHERE と JOIN)

複数テーブルの結合について，最初 WHERE で記述しました．JOIN を勉強した後，JOIN で記述しました．ネット上で資料を調べると，両方が正しいで，効率も同じという．この原因で，両方を利用しました．

（1）二つのテーブル結合

Package：src/main/resources/com/wjc/user/mapper/UserMapper.xml

Servlet：src/main/java/com/wjc/user/controller/myPage.java の showFollow メソッド.

説明：「mypage」に「following」ボタンをクリックすると，このユーザー(follower_uid)がフォローしているユーザーを表示します.

```
<select id="findFollowingUser" resultType="User">
        select username,userno,followernum,followunum
        from t_user u ,t_follow f
        where u.uid=f.follow_uid and follower_uid=#{follower_uid}
        order by update_time
        DESC limit #{pageCode},#{pageSize}
</select>
```

（2）三つのテーブル結合

Package：src/main/resources/com/wjc/post/mapper/PostMapper.xml

Servlet：src/main/java/com/wjc/user/controller/myPage.java の showDynamic メソッド.

説明：「mypage」に「Dynamic」ボタンをクリックすると，このユーザーがフォローしているユーザーが，発表した記事を表示します.

先にこのユーザーの uid でフォローしているユーザーを検索し，そしてフォローしているユーザーの userno で発表した記事を検索します.

```
<select id="findDynamicPost" resultType="Post">
        select *
        from (select u.userno
                from t_follow f right join t_user u
                on f.follow_uid=u.uid
                where f.follower_uid=#{uid}) as u
            left join t_posts
            on u.userno=t_posts.userno
            order by t_posts.update_time DESC limit #{pageCode},#{pageSize}
</select>
```

2. Spring boot の transaction 管理

Service 層で同時に二つの操作を実行する場合，例えば，管理者は記事のタイプを増加

する同時に，この操作を履歴として保存する場合に，この二つの操作を transaction として，spring boot のアノテーションで管理します．一つの操作が失敗したら，ロールバックを行います．

@Transactional(rollbackFor=Exception.class,propagation = Propagation.*REQUIRED*)

Package：src/main/java/com/wjc/admin/service/AdminServiceImpl.java

3．データのグループ化

Package：src/main/resources/com/wjc/admin/mapper/AdminMapper.xml

Servlet：src/main/java/com/wjc/admin/controller/ArticleManage.java の findAllArticle メソッド．

Service：src/main/java/com/wjc/admin/service/AdminServiceImpl.java の findAllTypeAndCount メソッド．

目的：すべての super type と各 super type に属する記事の数を検索します．

```
<select id="supertypeAndPostCount" resultMap="superCountResultMap">
        select f.type_first_name, f.type_first_id, ifnull(p.count,0) count
            from t_type_first f
            left join(
                select count(*) count, p.type, f.type_first_name
                from t_posts p, t_type_first f
                where p.state=2 and p.type=f.type_first_id
                group by p.type,f.type_first_id order by p.type) as p
            on f.type_first_id=p.type
</select>
<resultMap id="superCountResultMap" type="com.wjc.type.pojo.Type_first">
  <result column="type_first_name" property="type_first_name" javaType="java.lang.String"/>
  <result column="type_first_id" property="type_first_id" javaType="java.lang.Integer"/>
  <result column="count" property="count" javaType="java.lang.Integer"/>
</resultMap>
```

4．再帰 SQL の代わり

一つの記事の下に，コメントとそのコメントの返事を表示するため，mybatis の collection でテーブルを再帰します．しかし，データベースに効率が悪いです．その代わりにすべてのコメントを検索し，service 層で再帰を行います．

cid	username	content	userno	create_time	post_id	parent_id	author_comment	pname	pno
5	lisi1234	A	10004	2021-04-22 02:16:58	a5144653154516613456	NULL	0	NULL	NULL
6	admin123	B	10001	2021-04-22 02:18:25	a5144653154516613456	NULL	1	NULL	NULL
7	test001	B-b	10003	2021-04-22 02:19:03	a5144653154516613456	6	0	NULL	NULL
8	a10001	A-a	10006	2021-04-22 02:19:38	a5144653154516613456	5	0	NULL	NULL
9	a10002	A-a-1	10007	2021-04-22 02:20:06	a5144653154516613456	8	0	a10001	10006
10	a10003	A-b	10008	2021-04-22 02:20:29	a5144653154516613456	5	0	NULL	NULL
11	a10003	C	10008	2021-04-22 02:50:05	a5144653154516613456	NULL	0	NULL	NULL
13	admin123	D	10001	2021-04-26 01:59:59	a5144653154516613456	NULL	1	NULL	NULL
14	admin123	A	10001	2021-04-26 02:04:28	a6922654633470260995	NULL	1	NULL	NULL
15	admin123	B	10001	2021-04-26 02:05:55	a6922654633470260995	NULL	1	NULL	NULL
16	admin123	A-a-1-...	10001	2021-04-26 02:06:55	a5144653154516613456	9	1	a10002	10007
18	test001	A-a-1-...	10003	2021-04-26 02:19:55	a5144653154516613456	9	0	a10002	10007
29	admin123	A-c	10001	2021-04-26 03:49:33	a5144653154516613456	5	1	NULL	NULL
30	a10003	B-b-1	10008	2021-04-26 04:00:39	a5144653154516613456	7	0	test001	10003

```xml
<select id="findComByPID" parameterType="java.lang.String"
resultType="Comment">
        select *
        from t_comment
        where post_id=#{postID}
        order by create_time DESC
</select>
```

```java
@Override
    public List<Comment> listCommentByBlogId(String postID) {
        List<Comment> comByPID = postMapper.findComByPID(postID);
        return combineChildren(comByPID);
    }

    private List<Comment> combineChildren(List<Comment> comments) {
        List<Comment> commentsParents = new ArrayList<>();
        List<Comment> commentsChildren = new ArrayList<>();
        for (Comment comment : comments) {
            if (comment.getParent_id()==null){
                commentsParents.add(comment);
            }else {
                commentsChildren.add(comment);
            }
        }
        for (Comment commentParent:commentsParents){
            if (commentsChildren.size()>0){
                childRecursion(commentsChildren,commentParent.getCid());
            }

 tempReplys.sort(Comparator.comparing(Comment::getCreate_time).reversed());
            commentParent.setReplyComments(tempReplys);
            tempReplys = new ArrayList<>();
        }
        return commentsParents;
    }

    private void childRecursion(List<Comment> commentsChildren,Integer cid)
{
        List<Comment> nextParents=new ArrayList<>();
        for (int i=0;i<commentsChildren.size();i++){
            if (commentsChildren.get(i).getParent_id().equals(cid)) {
                tempReplys.add(commentsChildren.get(i));
                nextParents.add(commentsChildren.get(i));
                commentsChildren.remove(commentsChildren.get(i));
                i--;
            }
        }
        if (nextParents.size()>0){
            for (Comment comment:nextParents)
            childRecursion(commentsChildren,comment.getCid());
        }
    }
    private List<Comment> tempReplys = new ArrayList<>();
```