Woodland Academy

1.7 PersonProject　中級

- AdminBlogController追加の設定
- Blog登録画面の作成
- Blog編集画面の作成

Shape Your Future

目次

AdminBlogControllerクラスの作成①

- 前回作成したAdminBlogControllerクラスに、下記のソースを追加してください。ソースの解説はコメントをみて理解を進めてください。

*AdminBlogController.java

```java
package blog.example.controller;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.util.List;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;

```

```java
@Controller
@RequestMapping("/admin/blog")
public class AdminBlogController {
    /**
     * accountテーブルを操作するための
     * Serviceクラス
     */
    @Autowired
    private UserService userService;
    /**
     * blogテーブルを操作するための
     * Serviceクラス
     */
    @Autowired
    BlogService blogService;
    /**
     * categoryテーブルを操作するための
     * Serviceクラス
     */
    @Autowired
    CategoryService categoryServie;

```

AdminBlogControllerクラスの作成②

- 前回作成したAdminBlogControllerクラスに、下記のソースを追加してください。ソースの解説はコメントをみて理解を進めてください。

```
94      /**
95       * ブログ記事の登録画面を表示させるための処理です。
96       * -ログインしている人のメールアドレスを使用して、ログインしている人のuserIdとuserNameを取得します。
97       * -カテゴリー覧を取得します
98       * -取得した情報をセットして画面から参照可能にします。
99       */
100
101     //ブログ記事の登録
102     @GetMapping("/register")
103     public String getBlogCreatePage(Model model) {
104         //      現在のリクエストに紐づく Authentication を取得するには SecurityContextHolder.getContext().getAuthentication() とする。
105         //      SecurityContextHolder.getContext() は、現在のリクエストに紐づく SecurityContext を返している。
106         //      Authentication.getAuthorities() で、現在のログインユーザーに付与されている権限(GrantedAuthority のコレクション)を取得できる。
107         Authentication auth = SecurityContextHolder.getContext().getAuthentication();
108         //ログインした人のメールアドレスを取得
109         String userEmail = auth.getName();
110         //accountテーブルの中から、ユーザーのEmailで検索をかけて該当するユーザーの情報を引っ張り出す。
111         UserEntity user = userService.selectById(userEmail);
112         //accountテーブルの中からログインしているユーザーのIDを取得
113         Long userId = user.getUserId();
114         //accountテーブルの中からログインしているユーザーの名前の取得
115         String userName = user.getUserName();
116         //カテゴリー一覧を取得
117         List<CategoryEntity>categoryList = categoryServie.findByAll();
118
119         //userId(管理者のId)、categoryList(カテゴリー覧)
120         //userName(管理者の名前)をmodelにセットし
121         //admin_blog_register.htmlから参照可能にする。
122         model.addAttribute("userId",userId);
123         model.addAttribute("categoryList",categoryList);
124         model.addAttribute("userName",userName);
125
126         return "admin_blog_register.html";
127     }
128
```

Woodland Academy

AdminBlogControllerクラスの作成③

- 前回作成したAdminBlogControllerクラスに、下記のソースを追加してください。ソースの解説はコメントをみて理解を進めてください。

```
129     /**
130      * ブログ記事を登録させるための処理です。
131      * ―画像名を取得し、blog-imageにアップロードする作業を行います。
132      * ―入力された情報によってblogテーブルに保存処理を行います。
133      * ―保存処理後は、ブログ一覧画面にリダイレクトさせます。
134      */
135     //登録内容を保存
136     @PostMapping("/register")
137     public String register(@RequestParam String blogTitle,
138             @RequestParam("blogImage") MultipartFile blogImage,
139             @RequestParam String categoryName,
140             @RequestParam String message,
141             @RequestParam Long userId) {
142 
143         //画像ファイル名を取得する
144         String fileName = blogImage.getOriginalFilename();
145 
146         //ファイルのアップロード処理
147         try {
148             //画像ファイルの保存先を指定する。
149             File blogFile = new File("./src/main/resources/static/blog-image/"+fileName);
150             //画像ファイルからバイナリデータを取得する
151             byte[] bytes = blogImage.getBytes();
152             //画像を保存(書き出し)するためのバッファを用意する
153             BufferedOutputStream out = new BufferedOutputStream(new FileOutputStream(blogFile));
154             //画像ファイルの書き出しする。
155             out.write(bytes);
156             //バッファを閉じることにより、書き出しを正常終了させる。
157             out.close();
158         }catch(Exception e) {
159             e.printStackTrace();
160         }
161         //ファイルのアップロード処理後に、サービスクラスのメソッドに値を渡して保存する
162         blogService.insert(blogTitle, fileName, categoryName, message, userId);
163 
164         return "redirect:/admin/blog/all";
165     }
```

AdminBlogControllerクラスの作成④

- 前回作成したAdminBlogControllerクラスに、下記のソースを追加してください。ソースの解説はコメントをみて理解を進めてください。

```
166
167     /**
168      * ブログ記事の編集画面を表示させるための処理です。
169      * -リンクからblogIdを取得する
170      * -blogIdに紐づくレコードを探す
171      * -取得した情報をセットして画面から参照可能にします。
172      */
173     //リンクタグで記載したblogIdを取得する
174     @GetMapping("/edit/{blogId}")
175     public String getBlogDetailPage(@PathVariable Long blogId, Model model) {
176         //      現在のリクエストに紐づく Authentication を取得するには SecurityContextHolder.getContext().getAuthentication() とする。
177         //      SecurityContextHolder.getContext() は、現在のリクエストに紐づく SecurityContext を返している。
178         //      Authentication.getAuthorities() で、現在のログインユーザーに付与されている権限（GrantedAuthority のコレクション）を取得できる。
179         Authentication auth = SecurityContextHolder.getContext().getAuthentication();
180         //ログインした人のメールアドレスを取得
181         String userEmail = auth.getName();
182         //accountテーブルの中から、ユーザーのEmailで検索をかけて該当するユーザーの情報を引っ張り出す。
183         UserEntity user = userService.selectById(userEmail);
184         //accountテーブルの中から、ユーザー名を取得
185         String userName = user.getUserName();
186         //accountテーブルの中から、ユーザーIDを取得
187         Long userId = user.getUserId();
188         //カテゴリーの一覧を取得
189         List<CategoryEntity>categoryList = categoryServie.findByAll();
190         //blogのテーブルの中から、blogIdで検索をかけて該当する該当するブログの情報を引っ張り出す。
191         BlogEntity blogs = blogService.selectByBlogId(blogId);
192
193         //userId(管理者のId)、categoryList(カテゴリー一覧)
194         //userName(管理者の名前)、blogs(idに紐づいたブログ記事)をmodelにセットし
195         //admin_blog_edit.htmlから参照可能にする。
196         model.addAttribute("userId",userId);
197         model.addAttribute("blogs",blogs);
198         model.addAttribute("categoryList",categoryList);
199         model.addAttribute("userName",userName);
200         return "admin_blog_edit.html";
201     }
```

AdminBlogControllerクラスの作成⑤

- 前回作成したAdminBlogControllerクラスに、下記のソースを追加してください。ソースの解説はコメントをみて理解を進めてください。

```
202     /**
203      * ブログ記事を更新させるための処理です。
204      * -画像名を取得し、blog-imageにアップロードする作業を行います。
205      * -入力された情報によってblogテーブルに更新処理を行います。
206      * -更新処理後は、ブログ一覧画面にリダイレクトさせます。
207      */
208     //登録内容を修正(更新)
209     @PostMapping("/update")
210     public String updateData(@RequestParam Long blogId,
211             @RequestParam String blogTitle,
212             @RequestParam("blogImage") MultipartFile blogImage,
213             @RequestParam String categoryName,
214             @RequestParam String message,
215             @RequestParam Long userId) {
216         //画像ファイル名を取得する
217         String fileName = blogImage.getOriginalFilename();
218         //ファイルのアップロード処理
219         try {
220             //画像ファイルの保存先を指定する。
221             File blogFile = new File("./src/main/resources/static/blog-image/"+fileName);
222             //画像ファイルからバイナリデータを取得する。
223             byte[] bytes = blogImage.getBytes();
224             //画像を保存(書き出し)するためのバッファを用意する。
225             BufferedOutputStream out = new BufferedOutputStream(new FileOutputStream(blogFile));
226             //画像ファイルの書き出しする。
227             out.write(bytes);
228             //バッファを閉じることにより、書き出しを正常終了させる。
229             out.close();
230         }catch(Exception e) {
231             e.printStackTrace();
232         }
233         //ファイルのアップロード処理後に、サービスクラスのメソッドに値を渡して更新を行う。
234         blogService.update(blogId, blogTitle, fileName, categoryName, message, userId);
235 
236         return "redirect:/admin/blog/all";
237     }
```

AdminBlogControllerクラスの作成⑥

- 前回作成したAdminBlogControllerクラスに、下記のソースを追加してください。ソースの解説はコメントをみて理解を進めてください。

```
238
239⊖    /**
240      * ブログ記事を削除させるための処理です。
241      * -blogIdに紐づくレコードを探す
242      * -紐づくレコードを削除する
243      */
244     //ブログの内容を削除
245⊖    @PostMapping("/delete")
246     public String blogDelete(@RequestParam Long blogId) {
247         //Serviceクラスに値をわたし、削除処理を行う。
248         blogService.deleteBlog(blogId);
249         return "redirect:/admin/blog/all";
250     }
251
252
253 }
```

Woodland Academy

目次

admin_blog_registerの作成①

- admin_blog_register.htmlを作成し、下記のソースを書き写してください。

admin_blog_register.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" th:href="@{/css/reset.css}">
    <link rel="stylesheet" th:href="@{/css/style.css}">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Kiwi+Maru:wght@300;400;500&display=swap" rel="stylesheet">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Gochi+Hand&family=Kiwi+Maru:wght@300;400;500&display=swap">
</head>

```

admin_blog_registerの作成②

- admin_blog_register.htmlを作成し、下記のソースを書き写してください。

```
18
19<body>
20    <header>
21        <!--スマートフォン-->
22        <nav class="menu">
23            <div class="logo">
24                <a href="#"><img th:src="@{/images/logo.png}" alt=""></a>
25            </div>
26            <div class="menu-contents">
27                <div class="menu-inner">
28                    <ul>
29
30                        <li class="menu__item"><a th:href="@{/admin/category/all}">カテゴリー一覧</a></li>
31                        <li class="menu__item"><a th:href="@{/admin/blog/all}">投稿記事一覧</a></li>
32                        <li class="menu__item">
33                            <form th:action="@{/logout}" method="post"><a href="#"
34                                    onclick="this.parentNode.submit()">ログアウト</a></form>
35                        </li>
36
37                    </ul>
38                </div>
39                <div class="menu-toggle_btn">
40                    <span></span>
41                    <span></span>
42                    <span></span>
43                </div>
44            </div>
45        </nav>
```

Woodland Academy

admin_blog_registerクラスの作成③

- admin_blog_register.htmlを作成し、下記のソースを書き写してください。

```
46          <!--pc-->
47          <nav class="pc">
48              <div class="pc-inner">
49                  <div class="pc-logo">
50                      <a href=""><img th:src="@{/images/logo.png}" alt=""></a>
51                  </div>
52                  <ul class="pc-list">
53                      <li class="pc-list__item" th:text=${userName}></li>
54                      <li class="pc-list__item"><a th:href="@{/admin/category/all}">カテゴリー一覧</a></li>
55                      <li class="pc-list__item"><a th:href="@{/admin/blog/all}">投稿記事一覧</a></li>
56                      <li class="pc-list__item">
57                          <form th:action="@{/logout}" method="post"><a href="#"
58                                  onclick="this.parentNode.submit()">ログアウト</a></form>
59                      </li>
60
61                  </ul>
62              </div>
63          </nav>
64      </header>
```

admin_blog_registerクラスの作成④

- admin_blog_register.htmlを作成し、下記のソースを書き写してください。

```
<main>
    <div class="main-inner">
        <section class="register-section">
            <h2>記事登録画面</h2>
            <form method="POST" th:action="@{/admin/blog/register}" enctype="multipart/form-data">
                <div class="register-section-details">
                    <div class="register-section-details_flex">
                        <div>タイトル</div>
                        <input type="text" name="blogTitle">
                    </div>
                    <div class="register-section-details_flex">
                        <div>ブログ画像</div>
                        <input type="file" name="blogImage">
                    </div>
                    <div class="register-section-details_flex">
                        <div>カテゴリー名</div>
                        <select name="categoryName" id="">
                            <option th:each="category : ${categoryList}" th:value="${category.categoryName}"
                                th:text="${category.categoryName}">
                            </option>
                        </select>
                    </div>
                    <div class="register-section-details_flex">
                        <div>Message</div>
                        <textarea name="message"></textarea>
                    </div>
                    <input type="hidden" name="userId" th:value="${userId}">
                    <div class="register-section-details_flex">
                        <button class="edit" id="register">登録</button>
                        <button class="back-btn" onclick="history.back();" type="button">戻る</button>'
                    </div>
                </div>
            </form>
        </section>
    </div>
</main>
```

Woodland Academy

admin_blog_registerの作成⑤

- admin_blog_register.htmlを作成し、下記のソースを書き写してください。

```
100     </main>
101     <script src="https://code.jquery.com/jquery-3.6.0.min.js"
102             integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>
103         <script th:src="@{/js/common.js}"></script>
104 </body>
105
106 </html>
```

目次

admin_blog_editの作成①

- admin_blog_edit.htmlを作成し、下記のソースを書き写してください。

admin_blog_edit.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" th:href="@{/css/reset.css}">
    <link rel="stylesheet" th:href="@{/css/style.css}">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Kiwi+Maru:wght@300;400;500&display=swap" rel="stylesheet">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Gochi+Hand&family=Kiwi+Maru:wght@300;400;500&display=swap">
</head>

```

admin_blog_editの作成②

- admin_blog_edit.htmlを作成し、下記のソースを書き写してください。

```
18
19<body>
20        <header>
21        <!--スマートフォン-->
22        <nav class="menu">
23            <div class="logo">
24                <a href="#"><img th:src="@{/images/logo.png}" alt=""></a>
25            </div>
26            <div class="menu-contents">
27                <div class="menu-inner">
28                    <ul>
29
30                        <li class="menu__item"><a th:href="@{/admin/category/all}">カテゴリー一覧</a></li>
31                        <li class="menu__item"><a th:href="@{/admin/blog/all}">投稿記事一覧</a></li>
32                        <li class="menu__item">
33                            <form th:action="@{/logout}" method="post"><a href="#"
34                                    onclick="this.parentNode.submit()">ログアウト</a></form>
35                        </li>
36
37                    </ul>
38                </div>
39                <div class="menu-toggle_btn">
40                    <span></span>
41                    <span></span>
42                    <span></span>
43                </div>
44            </div>
45        </nav>
```

admin_blog_editの作成③

- admin_blog_edit.htmlを作成し、下記のソースを書き写してください。

```
46        <!--pc-->
47        <nav class="pc">
48            <div class="pc-inner">
49                <div class="pc-logo">
50                    <a href=""><img th:src="@{/images/logo.png}" alt=""></a>
51                </div>
52                <ul class="pc-list">
53                    <li class="pc-list__item" th:text=${userName}></li>
54                    <li class="pc-list__item"><a th:href="@{/admin/category/all}">カテゴリー一覧</a></li>
55                    <li class="pc-list__item"><a th:href="@{/admin/blog/all}">投稿記事一覧</a></li>
56                    <li class="pc-list__item">
57                        <form th:action="@{/logout}" method="post"><a href="#"
58                                onclick="this.parentNode.submit()">ログアウト</a></form>
59                    </li>
60
61                </ul>
62            </div>
63        </nav>
64    </header>
```

admin_blog_editの作成④

- admin_blog_edit.htmlを作成し、下記のソースを書き写してください。

```
<main>
    <div class="main-inner">
        <section class="register-section">
            <h2>記事編集画面</h2>
            <form method="POST" th:action="@{/admin/blog/update}" enctype="multipart/form-data">
                <div class="register-section-details">
                    <div class="register-section-details_flex">
                        <div>タイトル</div>
                        <input type="text" name="blogTitle" th:value="${blogs.blogTitle}">
                    </div>
                    <div class="register-section-details_flex">
                        <div>登録済みブログ画像</div>
                        <img th:src="@{/blog-image/{items}(items=${blogs.blogImage})}">
                    </div>
                    <div class="register-section-details_flex">
                        <div>ブログ画像</div>
                        <input type="file" name="blogImage">
                    </div>
                    <div class="register-section-details_flex">
                        <div>カテゴリー名</div>
                        <select name="categoryName" id="">
                            <option th:each="category : ${categoryList}" th:value="${category.categoryName}"
                                th:text="${category.categoryName}"
                                th:selected="${category.categoryName == blogs.categoryName}">
                            </option>
                        </select>
                    </div>
                    <div class="register-section-details_flex">
                        <div>Message</div>
                        <textarea name="message" th:text="${blogs.message}"></textarea>
                    </div>
                    <input type="hidden" name="userId" th:value="${userId}">
                    <input type="hidden" name="blogId" th:value="${blogs.blogId}">
                    <div class="register-section-details_flex">
                        <button class="edit" id="register">更新</button>
                        <button class="back-btn" onclick="history.back();" type="button">戻る</button>
                    </div>
                </div>
            </form>
```

Woodland Academy

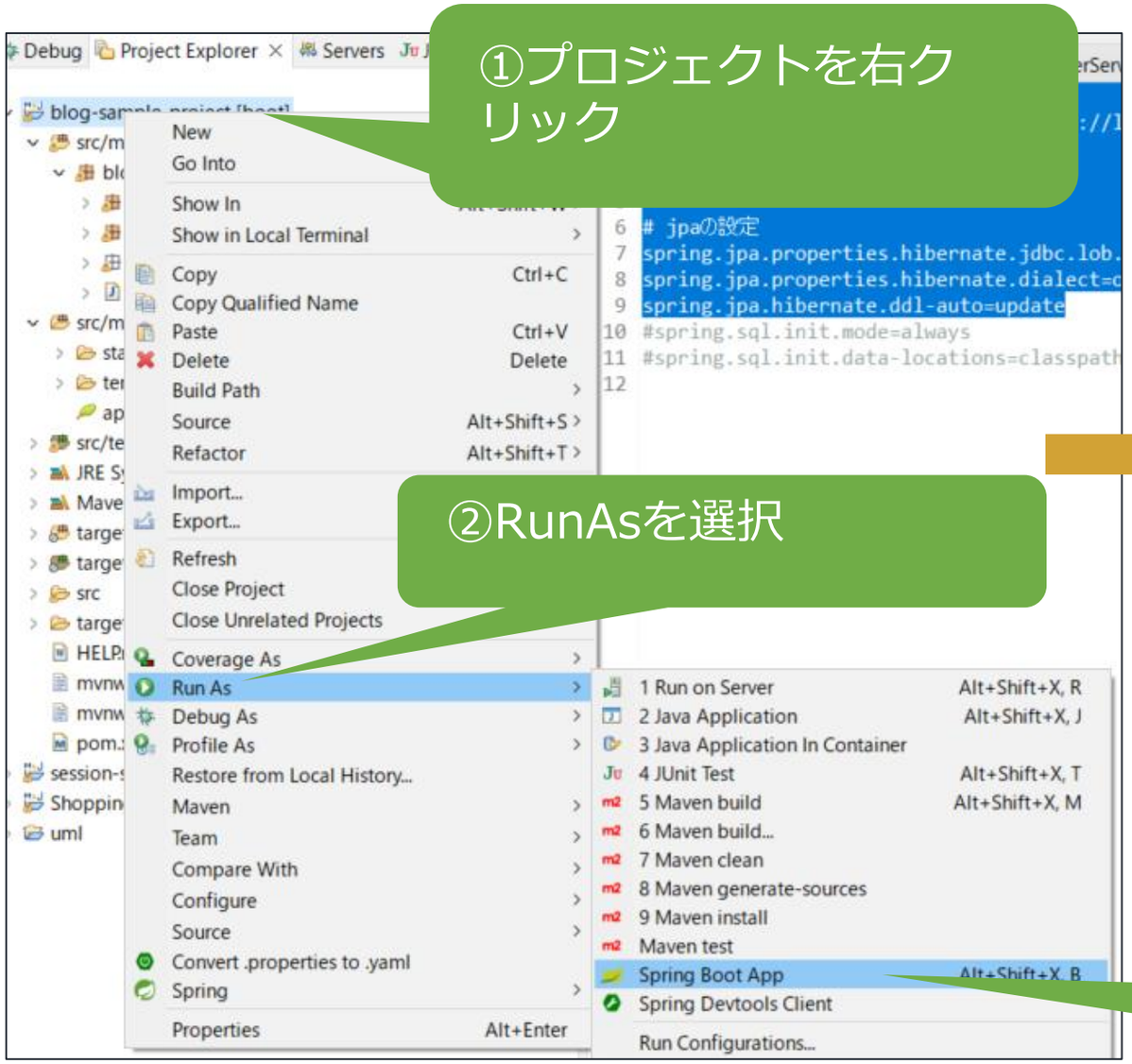
admin_blog_editの作成⑤

- admin_blog_edit.htmlを作成し、下記のソースを書き写してください。

```
104                     <div class="register-section-details_flex">
105                         <form method="post" th:action="@{/admin/blog/delete}">
106                             <input type="hidden" name="blogId" th:value="${blogs.blogId}">
107                             <button class="delete" id="delete">削除</button>
108                         </form>
109                     </div>
110                 </section>
111             </div>
112         </main>
113         <script src="https://code.jquery.com/jquery-3.6.0.min.js"
114             integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>
115         <script th:src="@{/js/common.js}"></script>
116 </body>
117
118 </html>
```

動作確認①

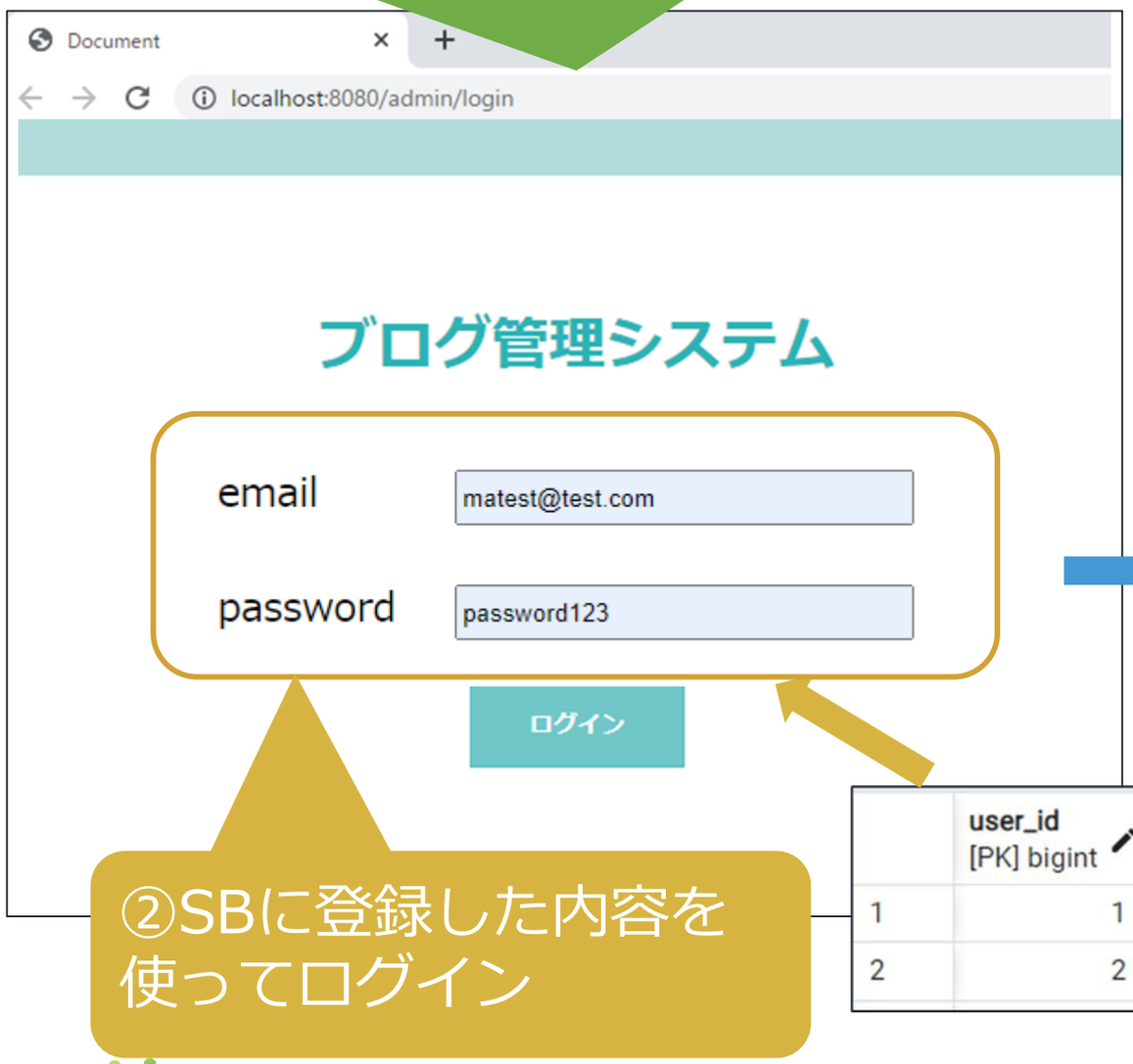
- Categoryの登録・編集・削除の確認。



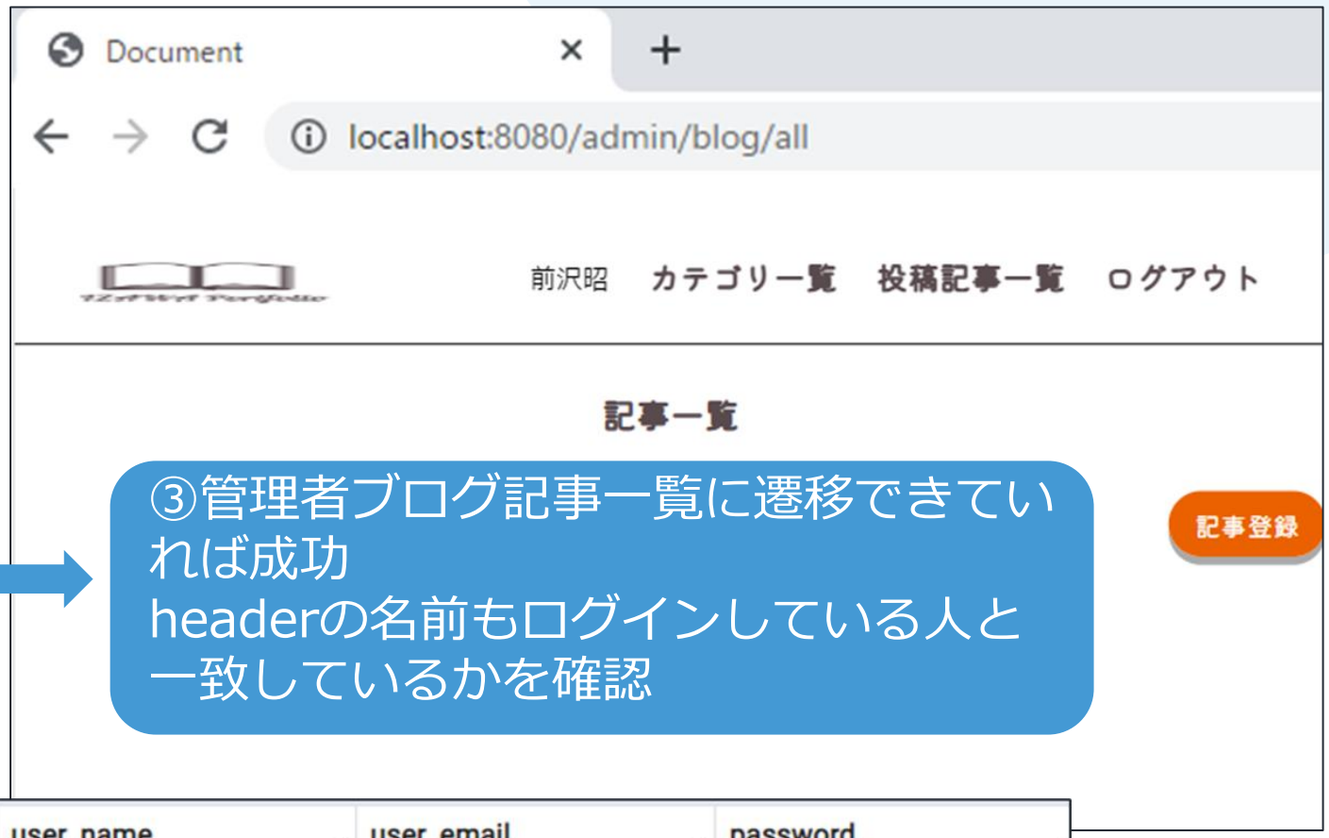
動作確認②

- ログイン画面してブログ一覧画面からCategory一覧が見れるかを確認

①ブラウザの検索欄に以下のURLを入力Enter
http://localhost:8080/admin/login



②SBに登録した内容を使ってログイン

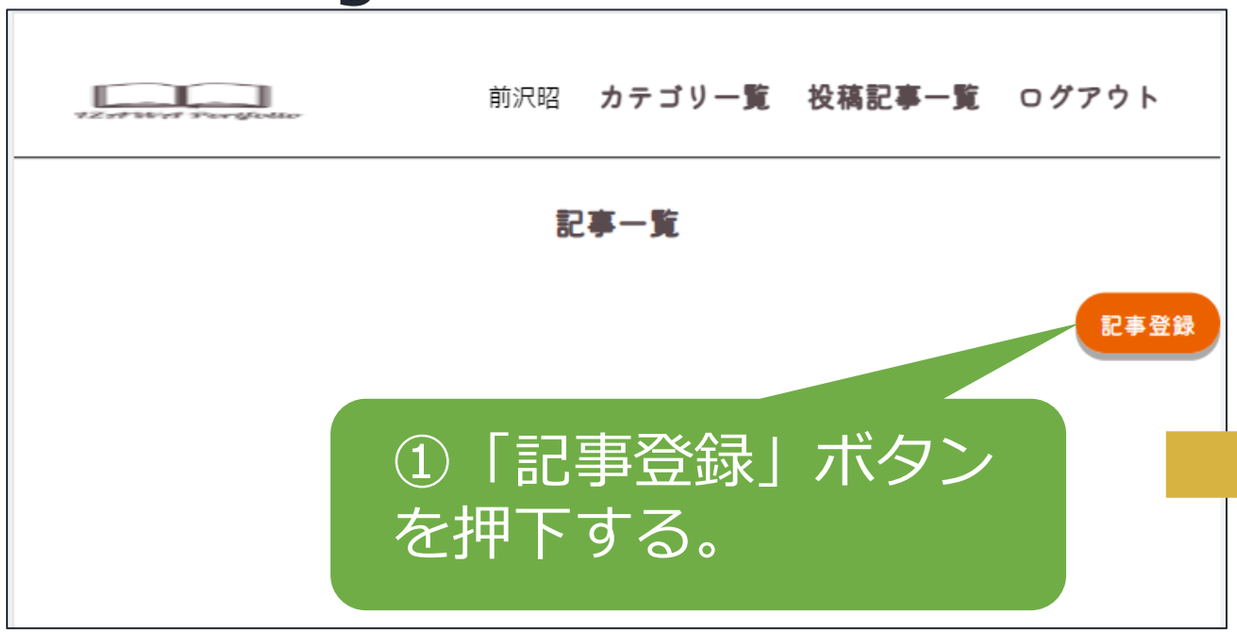


③管理者ブログ記事一覧に遷移できていれば成功
headerの名前もログインしている人と一致しているかを確認

	user_id [PK] bigint	user_name character varying (255)	user_email character varying (255)	password character varying (255)
1	1	前沢昭	matest@test.com	password123
2	2	大久保太郎	otest@test.com	password123

動作確認③

- Blog記事の登録の確認



動作確認④

- Blog記事の登録の確認

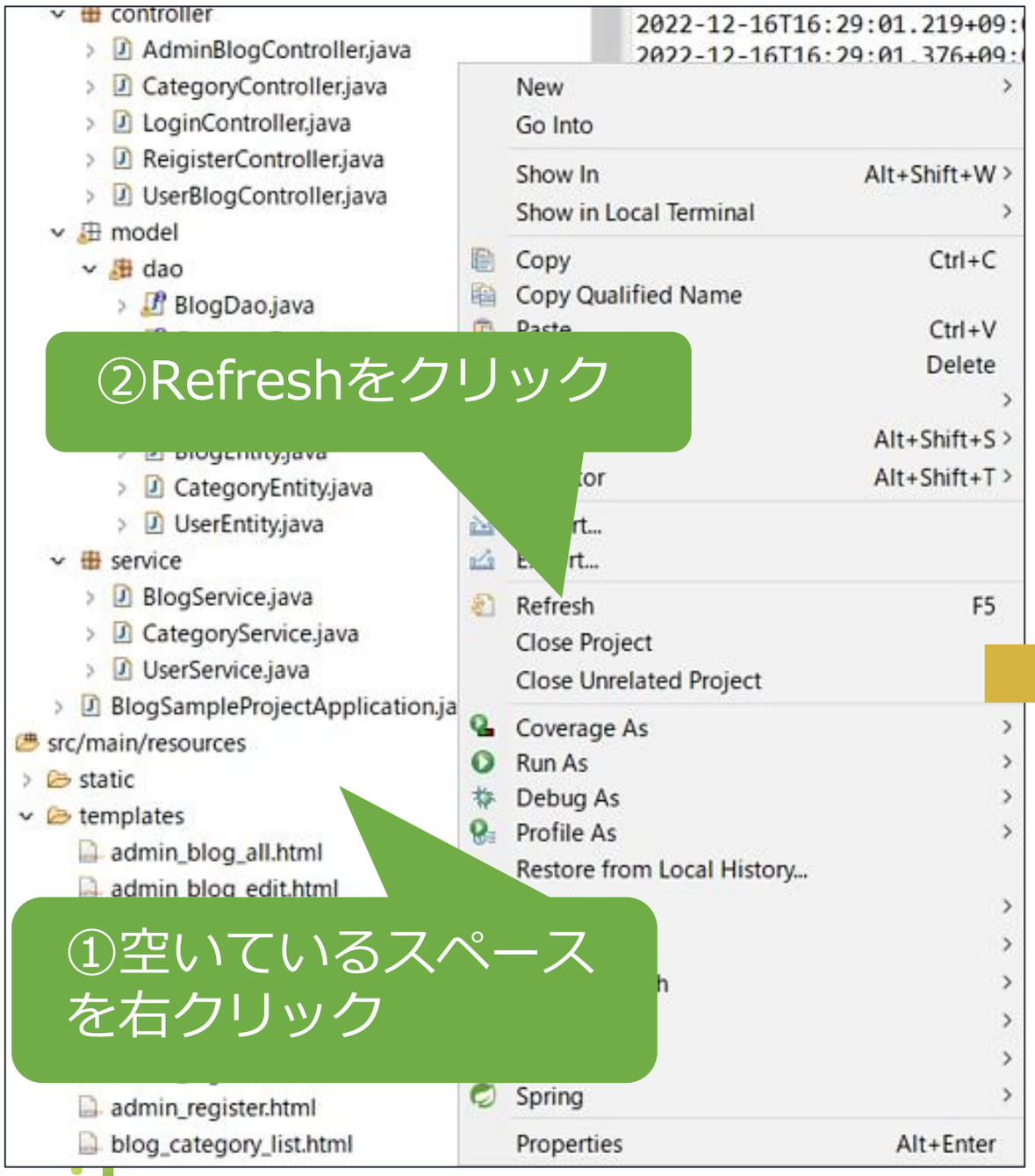
①「登録」ボタンを押下する。

DBに登録されているかを確認してください。

	blog_id [PK] bigint	blog_image character varying (255)	blog_title character varying (255)	category_name character varying (255)	message character varying (255)	user_id bigint
1	1	焼肉.jpg	焼肉食べました！	焼肉	焼肉はやっぱり最高！ …	1

動作確認⑤

- Blog記事の画像の表示確認



動作確認⑥

- Blog記事をもう一つ登録

DBに自分の入力した
内容が反映されている
あも必ず確認する。

	blog_id [PK] bigint	blog_image character varying (255)	blog_title character varying (255)	category_name character varying (255)	message character varying (255)	user_id bigint
1	1	焼肉.jpg	焼肉食べました！	焼肉	焼肉はやっぱり最高！ …	1
2	2	お寿司.jpg	お寿司食べました	お寿司	回転寿司は卒業！ 回ら…	1

動作確認⑥

- Blog記事編集動作の確認

記事をクリック

該当する内容が表示されているかを確認

	blog_id [PK] bigint	blog_image character varying (255)	blog_title character varying (255)	category_name character varying (255)	message character varying (255)	user_id bigint
1	1	焼肉.jpg	焼肉食べました！	焼肉	焼肉はやっぱり最高！ …	1
2	2	お寿司.jpg	お寿司食べました	お寿司	回転寿司は卒業！ 回ら…	1

Woodland Academy

動作確認⑦

- Blog記事編集動作の確認

いくつかの内容を変更する

画像は保持されないため、再度登録する必要がある。
相手の使いやすさを考えるとよくないが、処理が多くなるため、今回は、編集する場合は、再度、画像登録をするという仕様にする。

動作確認⑧

- Blog記事編集動作の確認

「更新」ボタンを押下

更新できているかを確認

	blog_id [PK] bigint	blog_image character varying (255)	blog_title character varying (255)	category_name character varying (255)	message character varying (255)	user_id bigint
1	1	焼肉.jpg	焼肉食べました！	焼肉	焼肉はやっぱり最高！ …	1
2	2	お寿司.jpg	贅沢ランチ	お寿司	回らないお寿司は、贅…	1

更新されているかテーブルも必ず確認

Woodland Academy

動作確認⑨

- Blog記事削除動作の確認

記事をクリック

記事の内容が一致しているかを確認

動作確認⑩

- Blog記事削除動作の確認

「削除」ボタンをクリック

記事が削除されていることを確認

	blog_id [PK] bigint	blog_image character varying (255)	blog_title character varying (255)	category_name character varying (255)	message character varying (255)	user_id bigint
1	1	焼肉.jpg	焼肉食べました！	焼肉	焼肉はやっぱり最高！ …	1

Woodland Academy

Thank you!

From Seeds to Woodland — Shape Your Future.

Shape Your Future