

# 1.6 PersonProject 中級

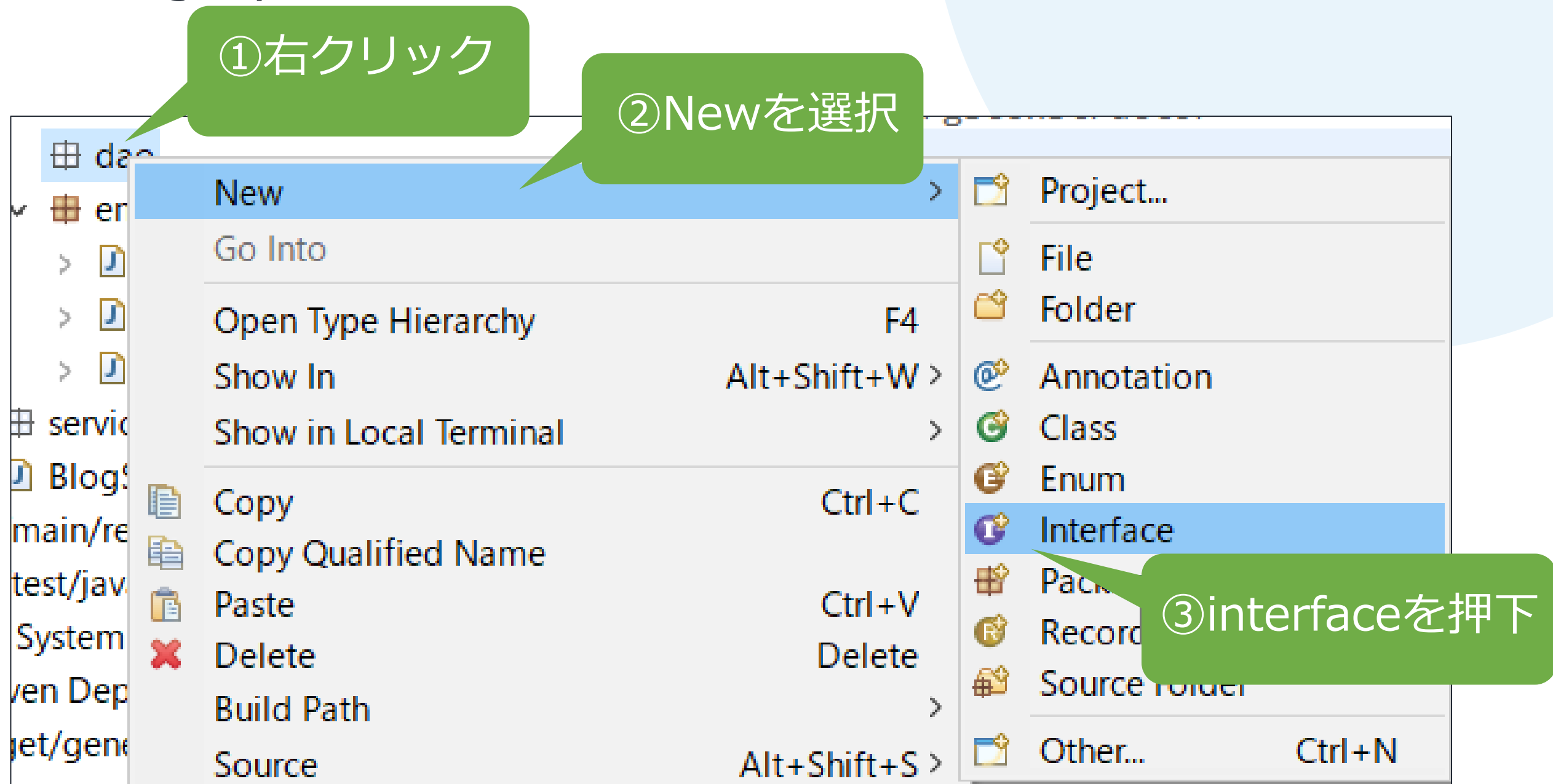
- CategoryDao作成
- CategoryServiceクラスの設定
- CategoryControllerクラスの設定
- 一覧画面の作成
- Category登録・編集画面の作成

# 目次

- 1 **CategoryDaoの作成**
- 2 **CategoryService  
クラスの設定**
- 3 **CategoryController  
クラスの設定**
- 4 **一覧画面の作成**
- 5 **Category登録・編  
集画面の作成**

# CategoryDaoインターフェースの作成①

- CategoryDaoクラスの作成方法を説明します。





# CategoryDaoインターフェースの作成②

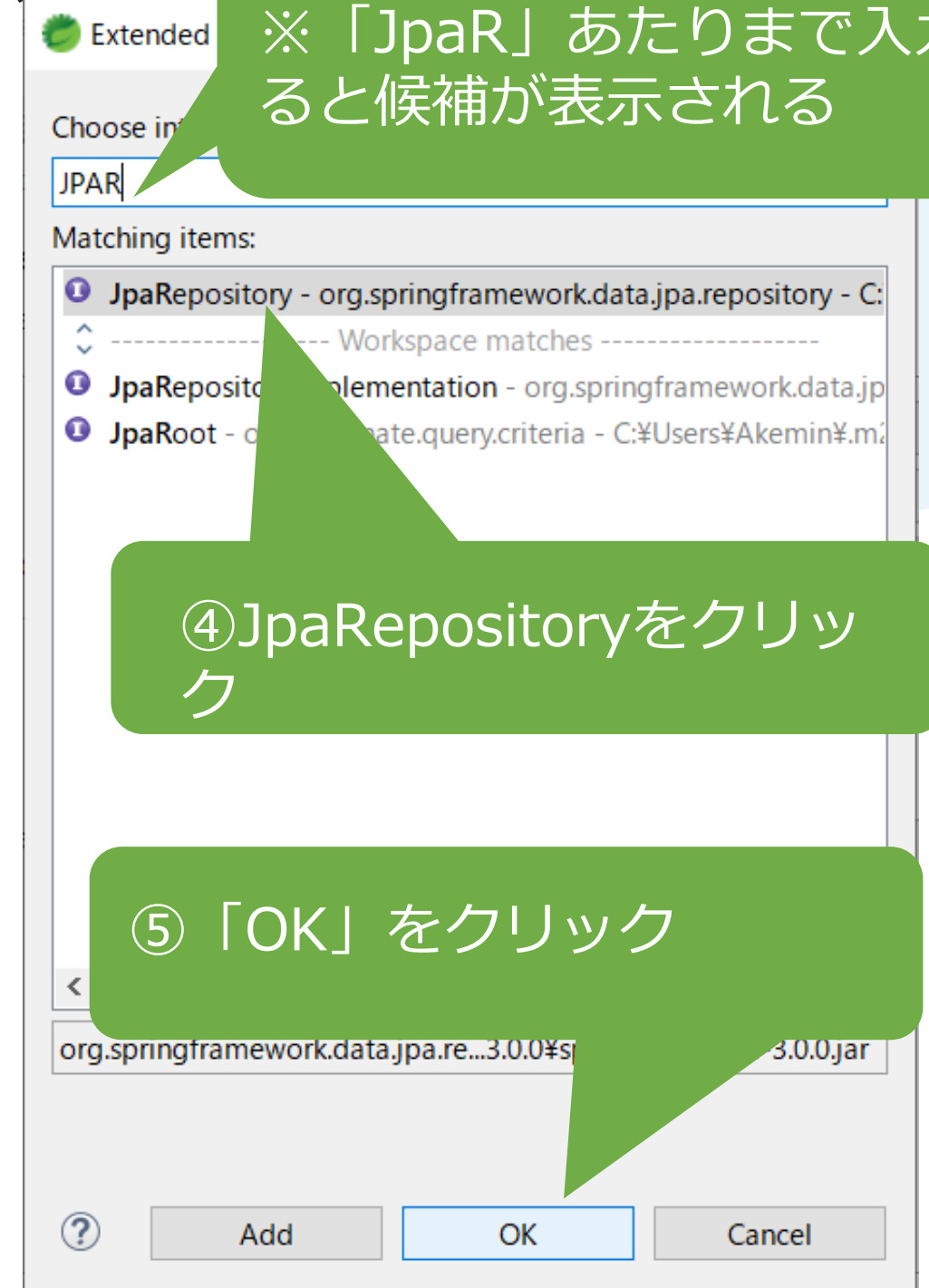
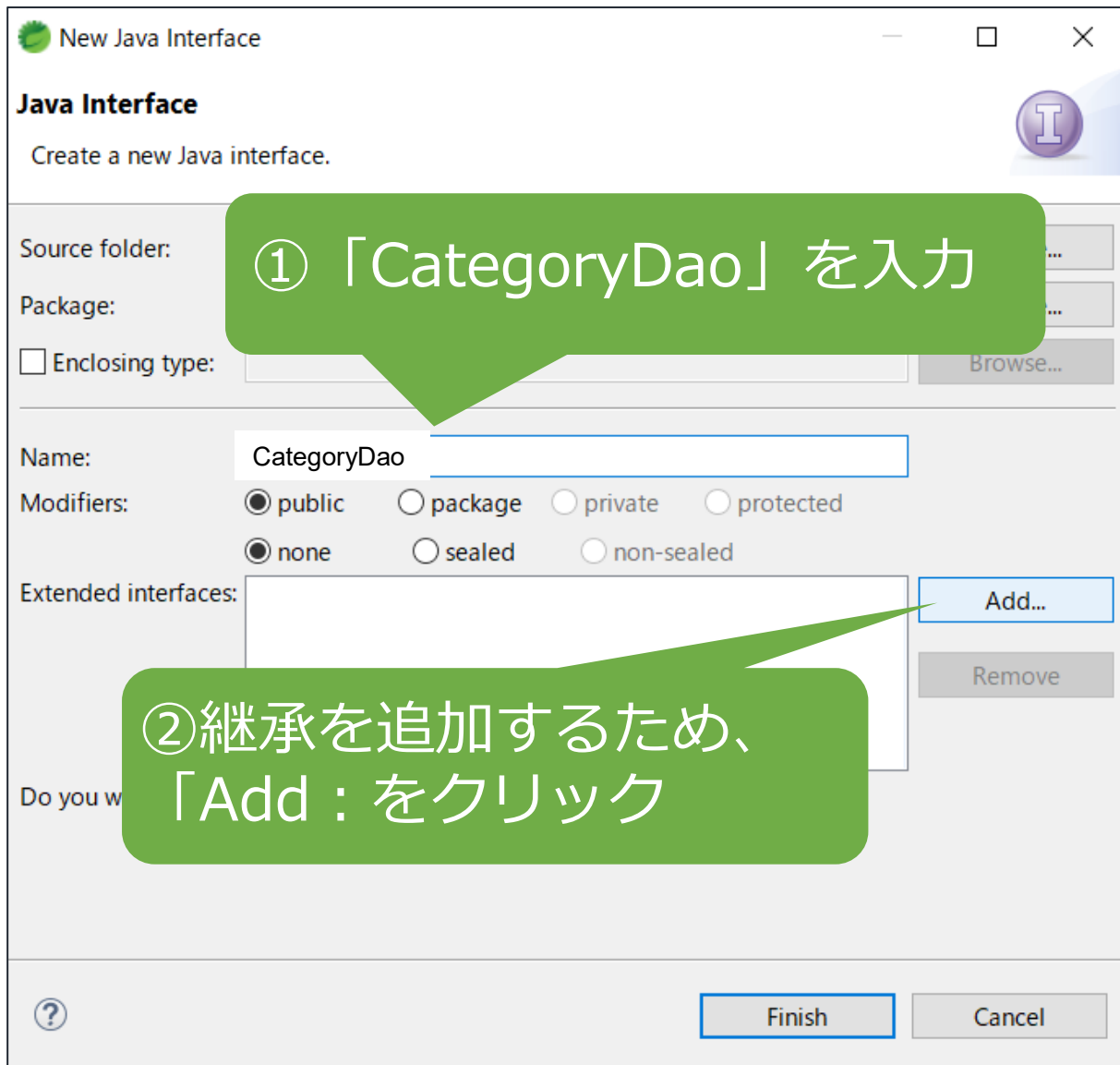
- CategoryDaoクラスの作成方法を説明し

①「CategoryDao」を入力

③「JpaRepository」を入力  
※「JpaR」あたりまで入力すると候補が表示される

④JpaRepositoryをクリック

⑤「OK」をクリック



# CategoryDaoインターフェースの作成③

- CategoryDaoクラスの作成方法を説明します。

**New Java Interface**

**Java Interface**  
Create a new Java interface.

☐ Enclosing type:

Name: UserDao

Modifiers: ☒ public ☐ package ☐ private ☒ none ☐ sealed ☐ non-sealed

Extended interfaces: ☒ org.springframework.data.jpa.repository.JpaRepository...

Do you want to add comments? (Configure templates and default value [here](#))  
☐ Generate comments

① 「userDao」が入力されているかを確認

② JpaRepositoryが追加されていることを確認

③ 「Finish」をクリック

# CategoryDaoインターフェースの作成④

- 以下の内容を書き写してください。

```

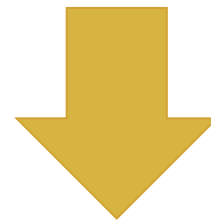
CategoryDao.java ×
1 package blog.example.model.dao;
2
3 import java.util.List;
4
5 import org.springframework.data.jpa.repository.JpaRepository;
6
7 import blog.example.model.entity.CategoryEntity;
8 import jakarta.transaction.Transactional;
9
10
11
12 public interface CategoryDao extends JpaRepository<CategoryEntity, Long> {
13     //カテゴリー内容の保存
14     CategoryEntity save(CategoryEntity categoryEntity);
15     //categoryIdを使用してDBに検索をかける
16     CategoryEntity findById(Long categoryId);
17     //削除
18     @Transactional
19     List<CategoryEntity> deleteById(Long categoryId);
20     //カテゴリー一覧を取得
21     List<CategoryEntity> findAll();
22 }

```

# CategoryDaoインターフェースの作成⑤

## ● ソースの説明

```
//カテゴリー内容の保存
CategoryEntity save(CategoryEntity categoryEntity);
```



saveメソッドは要注意  
insertとupdateの両方の役割を持っている  
1：更新対象のキーでセレクト  
2：同じキーのレコードがあるかをチェック  
    1：あったらupdate  
    2：なかったらinsert  
という順で処理をしている。

# CategoryDaoインターフェースの作成⑥

- ソースの説明。

```
//categoryIdを使用してDBに検索をかける  
CategoryEntity findById(Long categoryId);
```

categoryの編集の際に使用

```
select category_id, category_name from category where category_id=?1
```

```
//カテゴリー一覧を取得  
List<CategoryEntity> findAll();
```

category一覧画面で一覧を出す際に使用

```
select category_id,category_name from category
```

```
//削除  
@Transactional  
List<CategoryEntity> deleteById(Long categoryId);
```

categoryの内容を削除する際に使用

```
delete from category where blog_id=?1
```



# 目次

- 1 CategoryDaoの作成
- 2 CategoryService  
クラスの設定
- 3 CategoryController  
クラスの設定
- 4 一覧画面の作成
- 5 Category登録・編集  
画面の作成

# CategoryServiceクラスの作成①

- CategoryServiceクラスを作成し、下記のソースを書き写してください。

```
CategoryService.java ×
1 package blog.example.service;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import blog.example.model.dao.CategoryDao;
9 import blog.example.model.entity.CategoryEntity;
10
11
12 @Service
13 public class CategoryService {
14     @Autowired
15     CategoryDao categoryDao;
16
17     //一覧表示
18     public List<CategoryEntity> findByAll() {
19         return categoryDao.findAll();
20     }
21
22     //内容を保存
23     public void insert(String categoryName) {
24         categoryDao.save(new CategoryEntity(categoryName));
25     }
26 }
```

# CategoryServiceクラスの作成②

- CategoryServiceクラスを作成し、下記のソースを書き写してください。

```

26
27 //編集
28 public void update(Long categoryId,String categoryName) {
29     categoryDao.save(new CategoryEntity(categoryId,categoryName));
30 }
31
32 //CategoryIdを基にDBで検索をかける
33 public CategoryEntity selectCategoryId(Long categoryId) {
34     return categoryDao.findById(categoryId);
35 }
36
37 //削除
38 public void delete(Long categoryId) {
39     categoryDao.deleteById(categoryId);
40 }
41 }

```

# CategoryServiceクラスの作成③

## ● ソースの解説

```
@Autowired
CategoryDao categoryDao;

//一覧表示
public List<CategoryEntity> findByAll() {
    return categoryDao.findAll();
}

//内容を保存
public void insert(String categoryName) {
    categoryDao.save(new CategoryEntity(categoryName));
}
```

CategoryDaoのメソッドを使用できるように設定

Category一覧を取得の際に使用

CategoryControllerから値を受け取る

保存処理をするメソッドをに引数をセットし、BlogDaoに渡して保存処理を行う。

# CategoryServiceクラスの作成④

- ソースの解説。

ブログ編集内容を更新する際に使用

```

27 //編集
28 public void update(Long categoryId,String categoryName) {
29     categoryDao.save(new CategoryEntity(categoryId,categoryName));
30 }
31
32 //CategoryIdを基にDBで検索をかける
33 public CategoryEntity selectCategoryId(Long categoryId) {
34     return categoryDao.findById(categoryId);
35 }
36
37 //削除
38 public void delete(Long categoryId) {
39     categoryDao.deleteById(categoryId);
40 }
41 }
    
```

Category編集画面を表示する際に必要

categoryの削除に使用



# 目次

- 1 CategoryDaoの作成
- 2 CategoryService  
クラスの設定
- 3 CategoryController  
クラスの設定
- 4 一覧画面の作成
- 5 Category登録・編集  
画面の作成

# CategoryControllerクラスの作成①

- CategoryControllerクラスを作成し、下記のソースを書き写してください。ソースの解説はコメントをみて理解を進めてください。

```

25 @Controller
26 @RequestMapping("/admin/category")
27 public class CategoryController {
28
29     /**
30      * categoryテーブルを操作するための
31      * Serviceクラス
32      */
33     @Autowired
34     private CategoryService categoryService;
35
36     /**
37      * accountテーブルを操作するための
38      * Serviceクラス
39      */
40     @Autowired
41     private UserService userService;
42
43     /**
44      * category一覧画面を出すための処理です
45      * -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。
46      * -category一覧を取得します。
47      * -取得した情報をセットして画面から参照可能にします。
48      */

```

# CategoryControllerクラスの作成②

- CategoryControllerクラスを作成し、下記のソースを書き写してください。ソースの解説はコメントをみて理解を進めてください。

```

49 //カテゴリー一覧を表示
50 @GetMapping("/all")
51 public String getCategoryAll(Model model) {
52     // 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。
53     // SecurityContextHolder.getContext() は、現在のリクエストに紐づく SecurityContext を返している。
54     // 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。
55     Authentication auth = SecurityContextHolder.getContext().getAuthentication();
56
57     //ログインした人のメールアドレスを取得
58     String userEmail = auth.getName();
59     //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ID情報を引っ張り出す。
60     UserEntity user = userService.selectById(userEmail);
61
62     //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
63     String userName = user.getUserName();
64     //categoryテーブルに入っている内容を全て取得し、categorylistに格納する。
65     List<CategoryEntity>categorylist = categoryService.findAll();
66
67     //categorylist(category一覧情報)とuserName(管理者の名前)をmodelにセットし
68     //admin_category_all.htmlから参照可能にする。
69     model.addAttribute("categorylist",categorylist);
70     model.addAttribute("userName",userName);
71
72     return "admin_category_all.html";
73 }

```

# CategoryControllerクラスの作成③

- CategoryControllerクラスを作成し、下記のソースを書き写してください。ソースの解説はコメントをみて理解を進めてください。

```

74  /**
75   * categoryの内容を保存する画面を表示する処理です。
76   * -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。
77   * -取得した情報をセットして画面から参照可能にします。
78   */
79   //カテゴリー登録画面を表示
80   @GetMapping("/register")
81   public String getCategoryListRegister(Model model) {
82       // 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。
83       // SecurityContextHolder.getContext() は、現在のリクエストに紐づく SecurityContext を返している。
84       // 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。
85       Authentication auth = SecurityContextHolder.getContext().getAuthentication();
86
87       //ログインした人のメールアドレスを取得
88       String userEmail = auth.getName();
89
90       //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ID情報を引っ張り出す。
91       UserEntity user = userService.selectById(userEmail);
92
93       //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
94       String userName = user.getUserName();
95
96       //userName(管理者の名前)をmodelにセットし
97       //admin_category_register.htmlから参照可能にする。
98       model.addAttribute("userName", userName);
99       return "admin_category_register.html";
100  }

```



# CategoryControllerクラスの作成④

- CategoryControllerクラスを作成し、下記のソースを書き写してください。ソースの解説はコメントをみて理解を進めてください。

```

101
102  /**
103   * categoryの内容を保存する処理です。
104   * -入力された情報によってcategoryテーブルに保存処理を行います。
105   * -取得したuserIdを使用してログインしている人のブログ記事を取得します。
106   * -保存処理後は、Category一覧画面にリダイレクトさせます。
107   */
108   //カテゴリー登録内容を保存
109   @PostMapping("/register")
110   public String register(@RequestParam String categoryName) {
111       //サービスクラスのメソッドに値を渡して保存する
112       categoryService.insert(categoryName);
113       return "redirect:/admin/category/all";
114   }

```



# CategoryControllerクラスの作成⑤

- CategoryControllerクラスを作成し、下記のソースを書き写してください。ソースの解説はコメントをみて理解を進めてください。

```

115
116  /**
117   * categoryの編集画面を表示させるための処理です。
118   * -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。
119   * -リンクからcategoryIdを取得する
120   * -categoryIdに紐づくレコードを探す
121   * -取得した情報をセットして画面から参照可能にします。
122   */
123  //カテゴリー編集画面の表示
124  @GetMapping("/edit/{categoryId}")
125  public String getCategoryEditPage(@PathVariable Long categoryId, Model model) {
126      // 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。
127      // SecurityContextHolder.getContext() は、現在のリクエストに紐づく SecurityContext を返している。
128      // 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。
129      Authentication auth = SecurityContextHolder.getContext().getAuthentication();
130
131      //ログインした人のメールアドレスを取得
132      String userEmail = auth.getName();
133
134      //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ID情報を引っ張り出す。
135      UserEntity user = userService.selectById(userEmail);
136
137      //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
138      String userName = user.getUserName();
139
140      //categoryテーブルの中から、category_idで検索をかけて該当するcategory情報を引っ張り出す。
141      CategoryEntity categoryEntity = categoryService.selectCategoryId(categoryId);
142
143      //categoryEntity(category情報)とuserName(管理者の名前)をmodelにセットし
144      //admin_category_edit.htmlから参照可能にする。
145      model.addAttribute("userName", userName);
146      model.addAttribute("category", categoryEntity);
147      return "admin_category_edit.html";
148  }

```

# CategoryControllerクラスの作成⑥

- CategoryControllerクラスを作成し、下記のソースを書き写してください。ソースの解説はコメントをみて理解を進めてください。

```

149
150  /**
151   * categoryの編集内容を保存する処理です。
152   * -画面から情報を受け取りcategoryテーブルに更新処理を行います。
153   * -リンクからcategoryIdを取得する
154   * -更新処理後は、category一覧画面にリダイレクトさせます。
155   */
156  @PostMapping("/update")
157  public String update(@RequestParam String categoryName,@RequestParam Long categoryId) {
158      //CategoryServiceのメソッドに値を渡して更新を行う。
159      categoryService.update(categoryId, categoryName);
160      return "redirect:/admin/category/all";
161  }
162
163  /**
164   * categoryを削除させるための処理です。
165   * categoryIdに紐づくレコードを探す
166   * -紐づくレコードを削除する
167   */
168  //カテゴリ登録内容を削除
169  @PostMapping("/delete")
170  public String deleteCategory(@RequestParam Long categoryId) {
171      //Serviceクラスに値を渡し、削除処理を行う。
172      categoryService.delete(categoryId);
173      return "redirect:/admin/category/all";
174  }
175
176
177 }

```

# 目次

- 1 CategoryDaoの作成
- 2 CategoryService  
クラスの設定
- 3 CategoryController  
クラスの設定
- 4 一覧画面の作成
- 5 Category登録・編  
集画面の作成

# category一覧画面の作成①

- admin\_category\_all.htmlを作成し、下記の内容を書き写してください。

```
admin_category_all.html ×
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>Document</title>
9   <link rel="stylesheet" th:href="@{/css/reset.css}">
10  <link rel="stylesheet" th:href="@{/css/style.css}">
11  <link rel="preconnect" href="https://fonts.googleapis.com">
12  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
13  <link href="https://fonts.googleapis.com/css2?family=Kiwi+Maru:wght@300;400;500&display=swap" rel="stylesheet">
14  <link rel="preconnect" href="https://fonts.googleapis.com">
15  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
16  <link href="https://fonts.googleapis.com/css2?family=Gochi+Hand&family=Kiwi+Maru:wght@300;400;500&display=swap">
17 </head>
18 |
```



# category一覧画面の作成②

- admin\_category\_all.htmlを作成し、下記の内容を書き写してください。

```

19<body>
20  <header>
21    <!--スマートフォン-->
22    <nav class="menu">
23      <div class="logo">
24        <a href="#"></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>

```



# category一覧画面の作成③

- admin\_category\_all.htmlを作成し、下記の内容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=""></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              </ul>
61          </div>
62      </nav>
63  </header>
64

```

# category一覧画面の作成④

- admin\_category\_all.htmlを作成し、下記の内容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=""></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              </ul>
61          </div>
62      </nav>
63  </header>
64

```

# category一覧画面の作成⑤

- admin\_category\_all.htmlを作成し、下記の内容を書き写してください。

```

65= <main>
66=   <div class="main-inner">
67=     <div class="genre">
68=
69=       <div class="genre-register__button">
70=         <a th:href="@{/admin/category/register}" class="btn btn--orange btn--radius">カテゴリー新規登録</a>
71=       </div>
72=       <table>
73=         <tr>
74=           <th>カテゴリー名</th>
75=           <th>編集</th>
76=           <th>削除</th>
77=         </tr>
78=
79=         <tr th:each="category:${categorylist}">
80=           <td th:text="${category.categoryName}"></td>
81=           <td><a th:href="'/admin/category/edit/' + ${category.categoryId}" class="edit">編集</a></td>
82=           <td>
83=             <form th:action="@{/admin/category/delete}" method="post">
84=               <input type="hidden" name="categoryId" th:value="${category.categoryId}">
85=               <button class="delete">削除</button>
86=             </form>
87=           </td>
88=         </tr>
89=
90=       </table>
91=     </div>
92=   </div>
93= </main>

```

# category一覧画面の作成⑥

- admin\_category\_all.htmlを作成し、下記の内容を書き写してください。

```
94 <script src="https://code.jquery.com/jquery-3.6.0.min.js"  
95     integrity="sha256-/xUj+30JU5yExlq6GSYGS7k7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>  
96 <script th:src="@{/js/common.js}"></script>  
97 </body>  
98  
99 </html>
```

# Category一覧画面の作成⑦

## ● ソースの説明

th:each="変数:\${コレクション}"  
変数.フィールド（今回はCategory Entityのフィールド）で値を表示できる

```
<tr th:each="category:${categorylist}">
  <td th:text="${category.categoryName}"></td>
  <td><a th:href="'/admin/category/edit/' + ${category.categoryId}" class="edit">編集</a></td>
  <td>
    <form th:action="@{/admin/category/delete}" method="post">
      <input type="hidden" name="categoryId" th:value="${category.categoryId}">
      <button class="delete">削除</button>
    </form>
  </td>
</tr>
```

### CategoryEntity

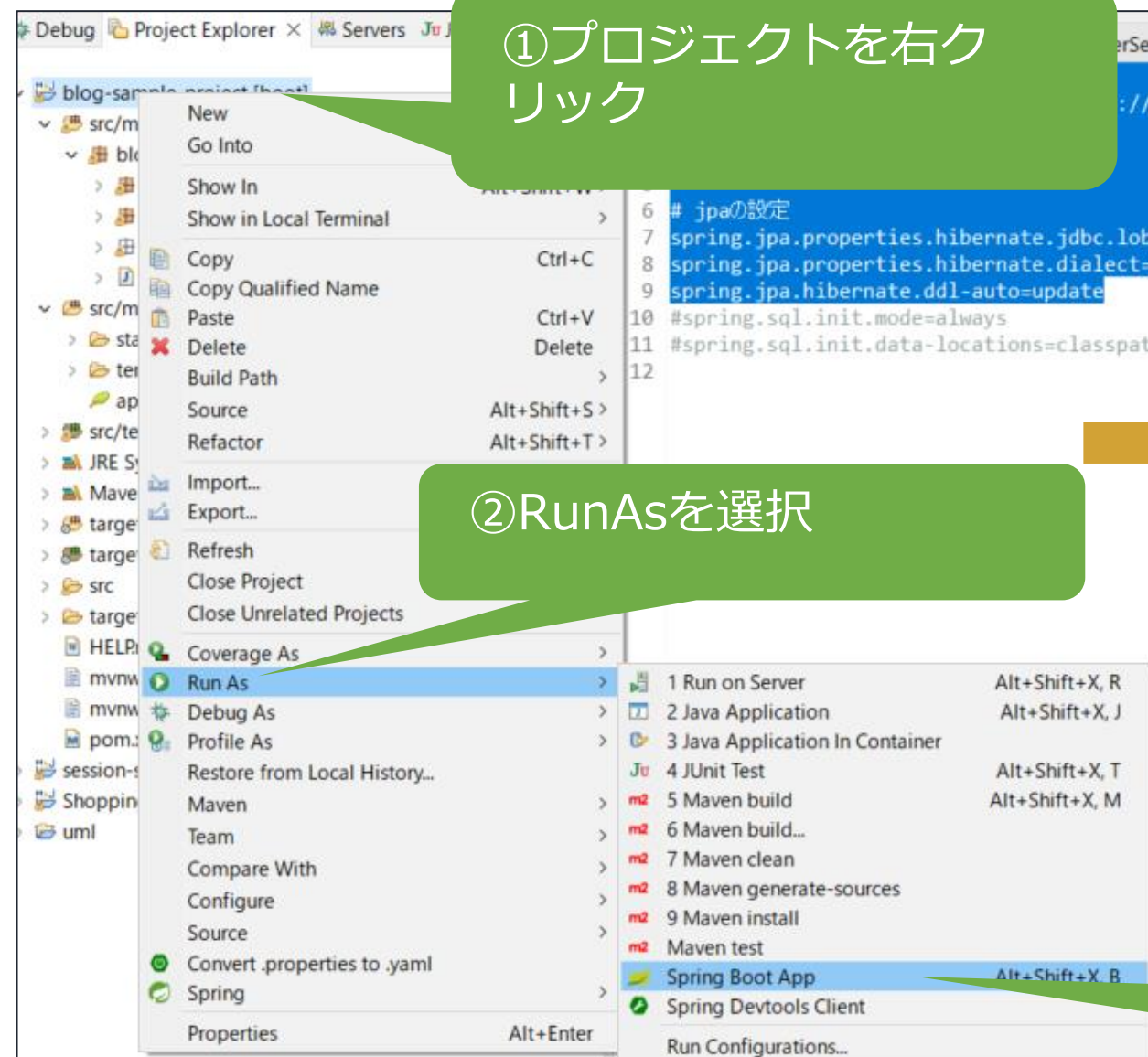
```
private Long categoryId;
private String categoryName;
```

<input>タグのtype属性でtype="hidden"を指定すると、ブラウザ上に表示されない非表示データを送信することができます。隠しデータとも呼ばれますが、完全に隠しきれているわけではなく、HTMLソースを表示すれば見ることができるので注意してください。



# 動作確認①

- Category一覧画面が見れるかを確認していきます。



# 動作確認②

- ログイン画面してブロッグ一覧画面からCategory一覧が見れ

①ブラウザの検索欄に以下のURLを入力Enter

http://localhost:8080/admin/login

Document x +

localhost:8080/admin/login

## ブログ管理システム

email matest@test.com

password password123

ログイン

②SBに登録した内容を使ってログイン

Document x +

localhost:8080/admin/blog/all

前沢昭 カテゴリー一覧 投稿記事一覧 ログアウト

## 記事一覧

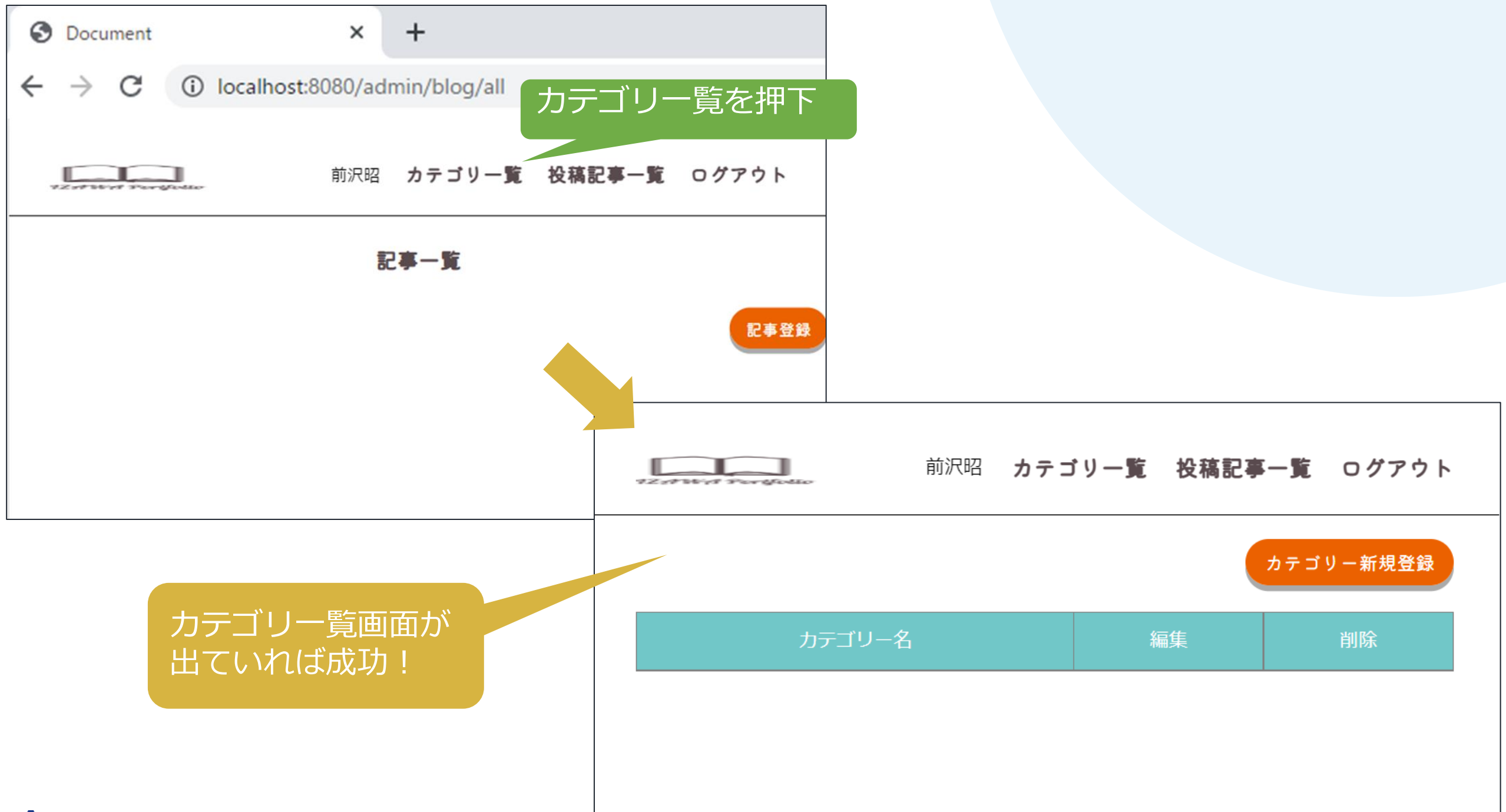
記事登録

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

- ログイン画面してブロッグ一覧画面からCategory一覧が見れるかを確認



The screenshot shows a web browser window with the address bar at `localhost:8080/admin/blog/all`. The page has a header with a logo, the name '前沢昭', and navigation links: 'カテゴリー一覧', '投稿記事一覧', and 'ログアウト'. The main content area is titled '記事一覧' (Article List) and contains a button '記事登録' (Register Article). A green callout bubble points to the 'カテゴリー一覧' link with the text 'カテゴリー一覧を押下' (Click Category List). Below the main content area, a yellow arrow points to a second screenshot of the 'カテゴリー一覧' (Category List) page. This page also has the same header. It features a button 'カテゴリー新規登録' (Register New Category) and a table with three columns: 'カテゴリー名' (Category Name), '編集' (Edit), and '削除' (Delete). A yellow callout bubble points to the category list page with the text 'カテゴリー一覧画面が出ていれば成功！' (Success if the category list screen appears!).

# 目次

- 1 CategoryDaoの作成
- 2 CategoryService  
クラスの設定
- 3 CategoryController  
クラスの設定
- 4 一覧画面の作成
- 5 Category登録・編集  
画面の作成



# Category登録画面の作成①

- admin\_category\_regitser.htmlを作成し、下記の内容を書き写してください。

```
admin_category_register.html ×
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>Document</title>
9   <link rel="stylesheet" th:href="@{/css/reset.css}">
10  <link rel="stylesheet" th:href="@{/css/style.css}">
11  <link rel="preconnect" href="https://fonts.googleapis.com">
12  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
13  <link href="https://fonts.googleapis.com/css2?family=Kiwi+Maru:wght@300;400;500&display=swap" rel="stylesheet">
14  <link rel="preconnect" href="https://fonts.googleapis.com">
15  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
16  <link href="https://fonts.googleapis.com/css2?family=Gochi+Hand&family=Kiwi+Maru:wght@300;400;500&display=swap">
17 </head>
18
```



# Category登録画面の作成②

- admin\_category\_regitser.htmlを作成し、下記の内容を書き写してください。

```

19<body>
20  <header>
21    <!--スマートフォン-->
22    <nav class="menu">
23      <div class="logo">
24        <a href="#"></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>

```

# Category登録画面の作成③

- admin\_category\_regitser.htmlを作成し、下記の内容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=""></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>

```

# Category登録画面の作成④

- admin\_category\_regitser.htmlを作成し、下記の内容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=""></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>

```

# Category登録画面の作成⑤

- admin\_category\_regitser.htmlを作成し、下記の内容を書き写してください。

```

65<main>
66  <div class="main-inner">
67    <section class="register-section">
68      <h2>カテゴリー登録画面</h2>
69      <form method="POST" th:action="@{/admin/category/register}">
70        <div class="register-section-details">
71          <div class="register-section-details_flex">
72            <div>Category</div>
73            <input type="text" name="categoryName" value="">
74          </div>
75          <div class="register-section-details_flex">
76            <button id="register" class="edit_disable">登録</button>
77            <button class="back-btn" onclick="history.back();" type="button">戻る</button>
78          </div>
79        </div>
80      </form>
81    </section>
82  </div>
83</main>
84<script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-/xUj+30JU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>
85<script src="https://code.jquery.com/jquery-3.6.0.min.js"
86  integrity="sha256-/xUj+30JU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>
87<script th:src="@{/js/common.js}"></script>
88</body>
89
90</html>

```



# Category編集画面の作成①

- admin\_category\_edit.htmlを作成し、下記の内容を書き写してください。

```
admin_category_edit.html ×
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta http-equiv="X-UA-Compatible" content="IE=edge">
7     <meta name="viewport" content="width=device-width, initial-scale=1.0">
8     <title>Document</title>
9     <link rel="stylesheet" th:href="@{/css/reset.css}">
10    <link rel="stylesheet" th:href="@{/css/style.css}">
11    <link rel="preconnect" href="https://fonts.googleapis.com">
12    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
13    <link href="https://fonts.googleapis.com/css2?family=Kiwi+Maru:wght@300;400;500&display=swap" rel="stylesheet">
14    <link rel="preconnect" href="https://fonts.googleapis.com">
15    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
16    <link href="https://fonts.googleapis.com/css2?family=Gochi+Hand&family=Kiwi+Maru:wght@300;400;500&display=swap">
17 </head>
18
```



# Category編集画面の作成②

- admin\_category\_edit.htmlを作成し、下記の内容を模写してください。

```

18
19<body>
20  <header>
21    <!--スマートフォン-->
22    <nav class="menu">
23      <div class="logo">
24        <a href="#"></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>

```

# Category編集画面の作成③

- admin\_category\_edit.htmlを作成し、下記の内容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=""></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              </ul>
61          </div>
62      </nav>
63  </header>
64

```

# Category編集画面の作成④

- admin\_category\_edit.htmlを作成し、下記の内容を書き写してください。

```
65<main>
66  <div class="main-inner">
67    <section class="register-section">
68      <h2>カテゴリ編集画面</h2>
69      <form method="POST" th:action="@{/admin/category/update}">
70        <div class="register-section-details">
71          <div class="register-section-details_flex">
72            <div>Category</div>
73            <input type="text" name="categoryName" th:value="${category.categoryName}">
74            <input type="hidden" name="categoryId" th:value="${category.categoryId}">
75          </div>
76          <div class="register-section-details_flex">
77            <button id="register" class="edit_disable">登録</button>
78            <button class="back-btn" onclick="history.back();" type="button">戻る</button>
79          </div>
80        </div>
81      </form>
82    </section>
83  </div>
84</main>
```

# Category編集画面の作成⑤

- admin\_category\_edit.htmlを作成し、下記の内容を書き写してください。

```
85 <script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-/xUj+30JU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>
86 <script src="https://code.jquery.com/jquery-3.6.0.min.js"
87     integrity="sha256-/xUj+30JU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>
88 <script th:src="@{/js/common.js}"></script>
89 </body>
90
91 </html>
```



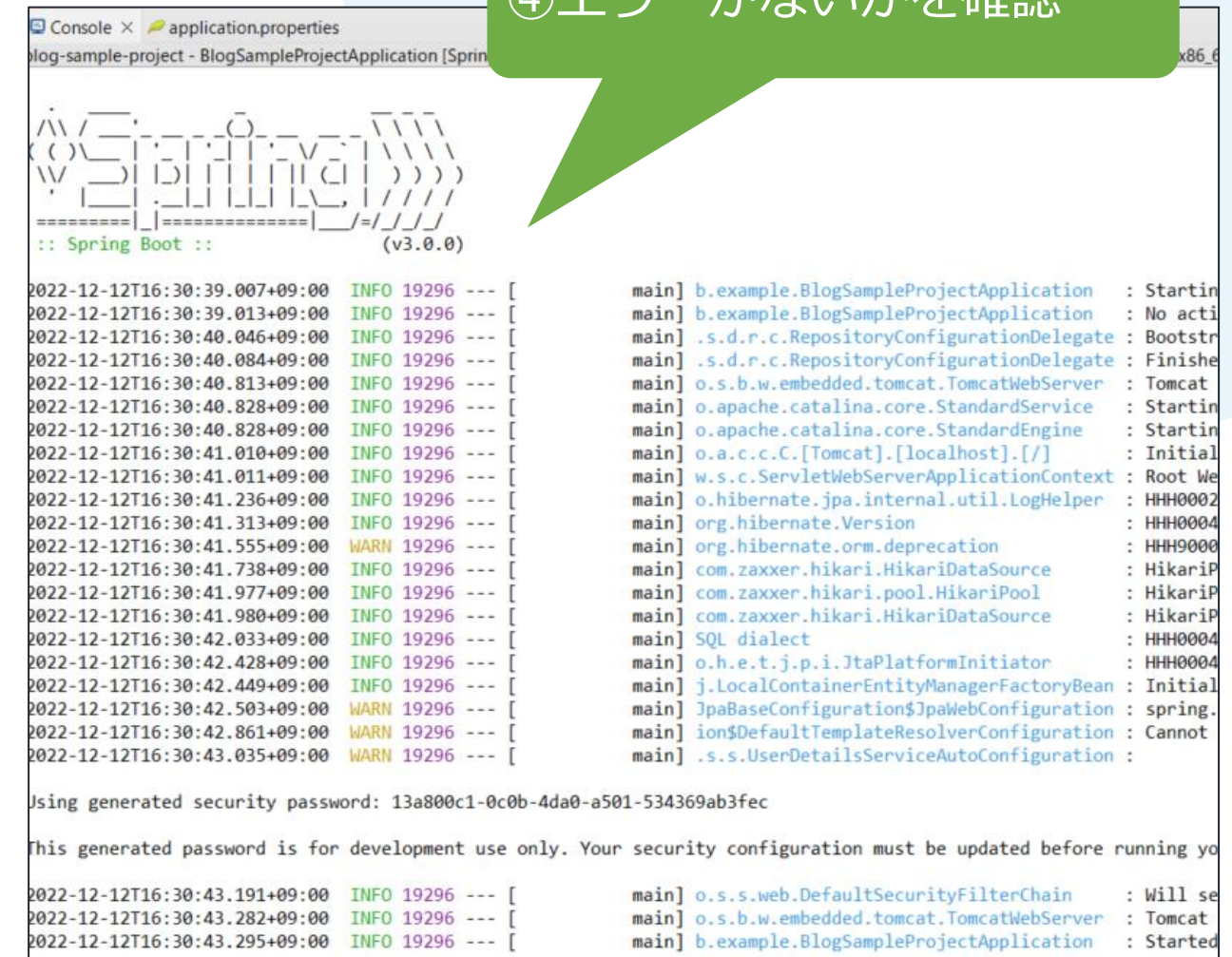
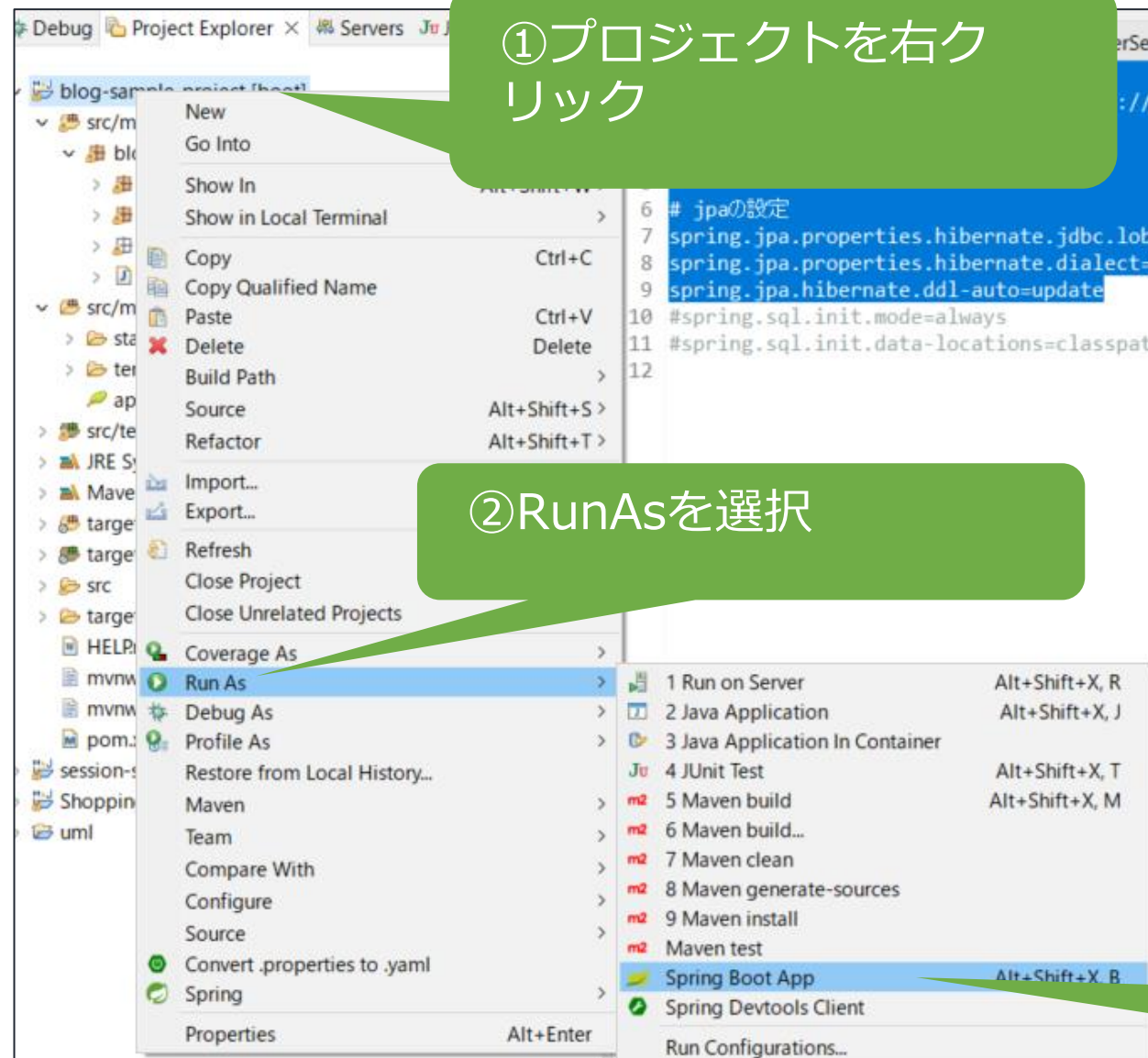
# 動作確認①

- Categoryの登録・編集・削除の確認。

①プロジェクトを右クリック

②RunAsを選択

④エラーがないかを確認



③SpringBootApplicationをクリック



## 動作確認②

- ログイン画面してブロッグ一覧画面からCategory一覧が見れるかを確認

	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

- ログイン画面してブロッグ一覧画面からCategory一覧が見れるかを確認



The screenshot shows a web browser window with the address bar displaying `localhost:8080/admin/blog/all`. The page title is "Document". The main content area is titled "記事一覧" (Article List). A green callout box with the text "カテゴリー一覧を押下" (Click Category List) points to the "カテゴリー一覧" (Category List) link in the top navigation bar. A yellow arrow points from the "記事一覧" title to the "カテゴリー一覧" link in the bottom navigation bar. The bottom navigation bar also includes "前沢昭" (Maizawa Shiro), "投稿記事一覧" (Posted Article List), and "ログアウト" (Logout). The main content area on the right shows a table with columns for "カテゴリー名" (Category Name), "編集" (Edit), and "削除" (Delete). A button labeled "記事登録" (Article Registration) is visible in the top right corner. A button labeled "カテゴリー新規登録" (Category New Registration) is visible in the bottom right corner.

# 動作確認④

## ● Categoryの登録の確認

前沢昭 カテゴリー一覧 投稿記事一覧 ログアウト

カテゴリー新規登録

カテゴリー名	編集	削除
--------	----	----

「カテゴリー新規登録」を押下



前沢昭 カテゴリー一覧 投稿記事一覧 ログアウト

カテゴリー登録画面

Category

登録 戻る

カテゴリー新規登録が表示  
されていれば成功！

# 動作確認⑤

## ● Categoryの登録の確認

前沢昭 カテゴリー一覧 投稿記事一覧 ログアウト

カテゴリー登録画面

Category

カテゴリーを入力後、「登録」を押下

前沢昭 カテゴリー一覧 投稿記事一覧 ログアウト

登録されている内容が表示されれば成功

カテゴリー新規登録

カテゴリー名	編集	削除
お寿司	<input type="button" value="編集"/>	<input type="button" value="削除"/>

	category_id [PK] bigint	category_name character varying (255)
1	1	お寿司

DBも必ず確認を行うこと！

# 動作確認⑥

## ● Categoryの登録の確認

 <span>前沢昭</span> <span>カテゴリー一覧</span> <span>投稿記事一覧</span> <span>ログアウト</span>		
<div>カテゴリー新規登録</div>		
カテゴリー名	編集	削除
お寿司	編集	削除
焼肉	編集	削除
すき焼き	編集	削除

カテゴリーをさらに2つ登録

- ・ 焼肉
- ・ すき焼き

	category_id [PK] bigint	category_name character varying (255)
1	1	お寿司
2	2	焼肉
3	3	すき焼き



# 動作確認⑦

## ● Categoryの編集の確認

前沢昭 カテゴリー一覧 投稿記事一覧 ログアウト

カテゴリー新規登録

カテゴリー名	編集	削除
お寿司	編集	削除
焼肉	編集	削除
すき焼き	編集	削除



前沢昭 カテゴリー一覧 投稿記事一覧 ログアウト

カテゴリー編集画面

Category すき焼き

登録 戻る

「すき焼き」が表示されるかを確認

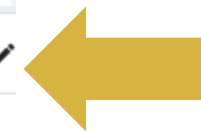


「すき焼き」の「編集」ボタンを押下

前沢昭 カテゴリー一覧 投稿記事一覧 ログアウト

カテゴリー新規登録

カテゴリー名	category_id [PK] bigint	category_name character varying (255)	編集	削除
お寿司	1	お寿司		
焼肉	2	焼肉		
お節料理	3	お節料理	編集	削除



前沢昭 カテゴリー一覧 投稿記事一覧 ログアウト

カテゴリー編集画面

Category お節料理

登録 戻る

「お節料理」と入力して「登録」を押下

「お節料理」になっていれば成功！

# 動作確認⑧

## ● Categoryの削除の確認

前沢昭 カテゴリー一覧 投稿記事一覧 ログアウト		
カテゴリー新規登録		
カテゴリー名	編集	削除
お寿司	編集	削除
焼肉	編集	削除
お節料理	編集	削除

「お節料理」の「削除」ボタンを押下

「お節料理」がDBから削除されているかを確認

	category_id [PK] bigint	category_name character varying (255)
1	1	お寿司
2	2	焼肉

「お節料理」が削除されているかを確認

前沢昭 カテゴリー一覧 投稿記事一覧 ログアウト		
カテゴリー新規登録		
カテゴリー名	編集	削除
お寿司	編集	削除
焼肉	編集	削除



*Light in Your Career.*

**THANK YOU!**