SpringBoot　コード解説

2.Hello Spring Boot! と表示させよう

2-1 メインクラスを確認します（パッケージとアノテーション）

src/main/java/com/example/demo/DemoApplication.java

```
1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class DemoApplication {
8
9    public static void main(String[] args) {
10       SpringApplication.run(DemoApplication.class, args);
11    }
12
13 }
```

概要説明

このクラス（基礎 P2） には、業務処理や画面制御のコードは記述されていません。
主な役割は、Spring Boot の仕組みを起動し、アプリケーションを実行可能な状態にすることです。

@SpringBootApplication アノテーション（基礎 P22）が付与されていることで、Spring Boot は以下の処理を自動的に行います。

- アプリケーションの各種設定を自動的に適用する
- Controller や Service、Mapper などのクラスを自動的に検出する
- Web アプリケーションの場合、組み込みサーバーを起動する

処理の流れ

main メソッド（基礎 P4）は、Java プログラムが起動した際に最初に呼び出される特別なメソッドです。
このメソッド内で呼び出されている
SpringApplication.run(...) により、Spring Boot の初期化処理が開始されます。

この 1 行の実行によって、

- Spring の実行環境が準備され
- 設定ファイルが読み込まれ
- アプリケーション全体が起動状態になります

```
1 package com.example.demo;
 // このクラスが属するパッケージ（基礎 P7）を宣言しています。
 // Spring Boot では、このパッケージを起点としてクラスの自動探索が行われます。

3 import org.springframework.boot.SpringApplication;（基礎 P9）
 // Spring Boot アプリケーションを起動するためのクラスを読み込んでいます。
 // アプリ全体の起動処理を担当する中心的なクラスです。

4 import org.springframework.boot.autoconfigure.SpringBootApplication;
 // Spring Boot の自動設定機能を有効にするためのアノテーションを読み込んでいます。

6 @SpringBootApplication（基礎 P23）
 // このクラスが Spring Boot アプリケーションの起動クラスであることを示します。
 // 設定クラスとしての役割、自動設定の有効化、
 // そして指定パッケージ配下のクラス探索をまとめて行います。
7 public class DemoApplication {

9    public static void main(String[] args) {
    // Java アプリケーションのエントリーポイントとなる main メソッドです。
    // プログラムは必ずこのメソッドから実行が開始されます。

10       SpringApplication.run(DemoApplication.class, args);
        // Spring Boot アプリケーションを起動する処理です。
        // Spring の管理機構を起動し、必要なクラスの初期化を行います。
        // あわせて、組み込みの Web サーバーも起動されます。
```

2-2Controller を作る

ソースコードの概要

1. クラスの役割

HelloController は、Web からのリクエストを受け取り、レスポンスを返す コントローラーです。
特にこのコードでは @RestController を使用しているため、HTML 画面（テンプレート）を返すのではなく、
文字列をそのまま HTTP レスポンスとして返す構成になります。

2. リクエストの受け取りと返却内容

@GetMapping("/") により、次の URL への GET リクエストを受け取ります。
/（ルートパス）
リクエストを受け取ると、hello() メソッドが呼び出され、戻り値である
"Hello Spring Boot!"
がそのまま画面（レスポンス本文）として返されます。

まとめ

このコードは、Spring Boot における基本動作である

- コントローラーの作成
- URL とメソッドの紐付け（ルーティング）
- 文字列のレスポンス返却
- を最小構成で確認できる例です。

コード解説

以下のディレクトリ
ディレクトリ　src/main/java/jp/edix/demo/

新規ファイルを作成
ファイル名　HelloController.java

```
1 package com.example.demo;
2
3 import org.springframework.web.bind.annotation.GetMapping;
4 import org.springframework.web.bind.annotation.RestController;
5
6 @RestController
7 public class HelloController {
8
9    @GetMapping("/")
10    public String hello() {
11        return "Hello Spring Boot!";
12    }
13 }
```

```
1 package com.example.demo;
 // このクラスが属するパッケージを宣言しています。
 // Spring Boot では、このパッケージ配下が自動的に探索対象となります。

3 import org.springframework.web.bind.annotation.GetMapping;
 // HTTP の GET リクエストを処理するためのアノテーションを読み込んでいます。

4 import org.springframework.web.bind.annotation.RestController;
 // このクラスが REST コントローラであることを示すアノテーションを読み込んでいます。

6 @RestController
 // このクラスが Web からのリクエストを受け取るコントローラであることを示します。
 // @Controller と @ResponseBody を組み合わせた役割を持ち、
 // 戻り値は画面ではなく、そのままレスポンス本文として返されます。
7 public class HelloController {
```

```
9    @GetMapping("/")
     //「/」という URL に対する HTTP GET リクエストを処理することを示します。
     // ブラウザでルートパスにアクセスすると、このメソッドが呼び出されます。

10   public String hello() {
       // リクエストを受け取った際に実行される処理です。
       // このメソッドは文字列を返すだけのシンプルな処理を行います。
11       return "Hello Spring Boot!";
       // 返却された文字列は、HTTP レスポンスの本文として
       // そのままブラウザに表示されます。
```

CRUD システム構築

各ファイルコーディング

index.html

```
1 <!DOCTYPE html>
2 <html lang="ja">
3 <head>
4    <meta charset="UTF-8">
5    <title>CRUD App - 一覧 </title>
6 </head>
7 <body>
8 <h1>CRUD App: 一覧 </h1>
9 <p> ここに一覧を表示します。</p>
10 <nav>
11    <ul>
12       <li><a href="/crudapp/input"> 新規登録 </a></li>
13       <li><a href="/crudapp/confirm"> 登録確認 </a></li>
14       <li><a href="/crudapp/complete"> 登録完了 </a></li>
15    </ul>
16    <a href="/crudapp"> 一覧へ </a>
17 </nav>
18 </body>
19 </html>
```

input.html

```
1 <!DOCTYPE html>
2 <html lang="ja">
3 <head>
4    <meta charset="UTF-8">
5    <title>CRUD App - 登録 </title>
```

```
</head>
<body>
<h1>CRUD App: 登録 </h1>
<p> ここに入力フォームを配置します。</p>
<nav>
    <a href="/crudapp"> 一覧へ戻る </a>
</nav>
</body>
</html>
```

confirm.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <title>CRUD App - 確認 </title>
</head>
<body>
<h1>CRUD App: 登録確認 </h1>
<p> ここに確認内容を表示します。</p>
<nav>
    <a href="/crudapp/input"> 入力へ戻る </a>
    <a href="/crudapp/complete"> 登録完了へ </a>
    <a href="/crudapp"> 一覧へ戻る </a>
</nav>
</body>
</html>
```

complete.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <title>CRUD App - 完了 </title>
</head>
<body>
<h1>CRUD App: 登録完了 </h1>
<p> 登録が完了しました。</p>
<nav>
    <a href="/crudapp"> 一覧へ戻る </a>
</nav>
</body>
</html>
```

CrudappController.java

```
1 package com.example.demo;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.GetMapping;
5 import org.springframework.web.bind.annotation.RequestMapping;
6
7 @Controller
8 @RequestMapping("/crudapp")
9 public class CrudappController {
10
11    @GetMapping("/index")
12    public String index() {
13        return "crudapp/index";
14    }
15
16    @GetMapping("/input")
17    public String input() {
18        return "crudapp/input";
19    }
20
21    @GetMapping("/confirm")
22    public String confirm() {
23        return "crudapp/confirm";
24    }
25
26    @GetMapping("/complete")
27    public String complete() {
28        return "crudapp/complete";
29    }
30 }
```

```
1 package com.example.demo;
 // このクラスが属するパッケージを宣言しています。
 // Spring Boot では、このパッケージ配下がコンポーネント探索の対象となります。

3 import org.springframework.stereotype.Controller;
 // このクラスが画面（HTML）を返すコントローラであることを示すための
 // @Controller アノテーションを使用するために読み込んでいます。

4 import org.springframework.web.bind.annotation.GetMapping;
 // HTTP の GET リクエストを処理するための @GetMapping アノテーションを
 // 使用するために読み込んでいます。

5 import org.springframework.web.bind.annotation.RequestMapping;
 // クラスやメソッドに共通の URL パスを割り当てるための
```

```
// @RequestMapping アノテーションを使用するために読み込んでいます。

7 @Controller（基礎 P24）
// このクラスが Spring MVC（基礎 P31）のコントローラであることを示します。
// 戻り値として返す文字列は、表示するテンプレート名として解釈されます。

8 @RequestMapping("/crudapp")
// このコントローラに対する共通の URL パスを指定しています。
//「/crudapp」およびルートパスへのアクセスを、このコントローラで処理します。

9 public class CrudappController {（基礎 P10）
// CRUD アプリケーションにおける画面表示と画面遷移を担当するコントローラクラスです。

11   @GetMapping("/index")（基礎 P26）
    // クラスに指定された共通パスに対する GET リクエストを処理します。
    // 追加のパス指定がないため、一覧画面の表示を担当します。

12   public String index() {
        // 一覧画面を表示するためのメソッドです。

13       return "crudapp/index";  （基礎 P12）
        //「crudapp/index」というテンプレート名を返しています。
        // 実際には templates/crudapp/index が表示されます。

14   }
15

16   @GetMapping("/input")
    //「/crudapp/input」への GET リクエストを処理します。
    // 新規登録などの入力画面を表示するための設定です。

17   public String input() {
        // 入力画面を表示するためのメソッドです。

18       return "crudapp/input";
        // templates/crudapp/input.html が表示されます。

19   }
20

21   @GetMapping("/confirm")
    //「/crudapp/confirm」への GET リクエストを処理します。
    // 入力内容を確認する画面を表示するための設定です。

22   public String confirm() {
        // 確認画面を表示するためのメソッドです。

23       return "crudapp/confirm";
```

```
        // templates/crudapp/confirm.html が表示されます。

    }

    @GetMapping("/complete")
    //「/crudapp/complete」への GET リクエストを処理します。
    // 処理完了後に表示される完了画面を担当します。

    public String complete() {
        // 完了画面を表示するためのメソッドです。

        return "crudapp/complete";
        // templates/crudapp/complete.html が表示されます。

    }
}
```

入力フォームの内容を画面間で受け渡す

input.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <title>CRUD App - 登録 </title>
</head>
<body>
<h1>CRUD App: 登録 </h1>
<p> ここに入力フォームを配置します。</p>
<form th:action="@{/crudapp/confirm}" method="post">
    <label for="content"> 内容 </label><br>
    <textarea id="content" name="content" rows="6" cols="50" required></textarea><br>
    <button type="submit"> 送信 </button>
</form>
<nav>
    <a href="/crudapp"> 一覧へ戻る </a>
</nav>
</body>
</html>
```

confirm.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <title>CRUD App - 確認 </title>
</head>
<body>
<h1>CRUD App: 登録確認 </h1>
<p> ここに確認内容を表示します。</p>
<section>
  <h2> 入力内容 </h2>
  <p th:text="${content}"> 入力内容がここに表示されます。</p>
</section>
<nav>
  <a href="/crudapp/input"> 入力へ戻る </a>
  <a href="/crudapp/complete"> 登録完了へ </a>
  <form th:action="@{/crudapp/input}" method="get" style="display:inline;">
    <button type="submit"> 戻る </button>
  </form>
  <form th:action="@{/crudapp/complete}" method="post" style="display:inline;">
    <input type="hidden" name="content" th:value="${content}">
    <button type="submit"> 確定する </button>
  </form>
  <a href="/crudapp"> 一覧へ戻る </a>
</nav>
</body>
</html>
```

CrudappController.java

```
package com.example.demo;

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

@Controller
@RequestMapping("/crudapp")
public class CrudappController {

  @GetMapping
```

```
15    public String index() {
16        return "crudapp/index";
17    }
18
19    @GetMapping("/input")
20    public String input() {
21        return "crudapp/input";
22    }
23
24    @GetMapping("/confirm")
25    public String confirm() {
26        return "crudapp/confirm";
27    }
28
29    @PostMapping("/confirm")
30    public String confirm(@RequestParam("content") String content, Model model) {
31        model.addAttribute("content", content);
32        return "crudapp/confirm";
33    }
34
35    @GetMapping("/complete")
36    public String complete() {
37        return "crudapp/complete";
38    }
39
40    @PostMapping("/complete")
41    public String completePost() {
42        return "crudapp/complete";
43    }
44 }
```

```
1 package com.example.demo;
 // このクラスが属するパッケージを宣言しています。
 // Spring Boot では、このパッケージ配下がコンポーネントスキャンの対象となります。

2
3 import org.springframework.stereotype.Controller;
 // このクラスが画面（HTML）を返すコントローラであることを示す
 // @Controller アノテーションを使用するために読み込んでいます。

4 import org.springframework.web.bind.annotation.GetMapping;
 // HTTP の GET リクエストを処理するための @GetMapping アノテーションを
 // 使用するために読み込んでいます。

5 import org.springframework.web.bind.annotation.PostMapping;
 // HTTP の POST リクエストを処理するための @PostMapping アノテーションを
```

```
 // 使用するために読み込んでいます。

6 import org.springframework.web.bind.annotation.RequestMapping;
 // クラスやメソッドに共通の URL パスを割り当てるための
 // @RequestMapping アノテーションを使用するために読み込んでいます。

7 import org.springframework.web.bind.annotation.RequestParam;
 // フォームなどから送信されたリクエストパラメータを
 // メソッド引数として受け取るためのアノテーションを使用するために読み込んでいます。

8 import org.springframework.ui.Model;
 // コントローラからビュー（HTML）へデータを渡すための Model インターフェースを
 // 使用するために読み込んでいます。

9

10 @Controller
 // このクラスが Spring MVC のコントローラであることを示します。
 // メソッドの戻り値として返される文字列は、表示するテンプレート名として扱われます。

11 @RequestMapping("/crudapp")
 // このコントローラに対する共通の URL パスを指定しています。
 //「/crudapp」およびルートパスへのアクセスが、このコントローラに割り当てられます。

12 public class CrudappController {
 // CRUD アプリケーションにおける画面表示および入力処理を担当するコントローラクラスです。

13
14    @GetMapping
    // 共通パスに対する GET リクエストを処理します。
    // 主に一覧画面の表示を担当します。

15    public String index() {
       // 一覧画面を表示するためのメソッドです。

16       return "crudapp/index";
       // templates/crudapp/index.html が表示されます。
17    }

18
19    @GetMapping("/input")
    //「/crudapp/input」への GET リクエストを処理します。
    // ユーザーが入力を行うための入力画面を表示します。

20    public String input() {
       // 入力画面を表示するためのメソッドです。
21       return "crudapp/input";
```

```
        // templates/crudapp/input.html が表示されます。
22    }
23
24    @GetMapping("/confirm")
      //「/crudapp/confirm」への GET リクエストを処理します。
      // 入力前または直接アクセス時に表示される確認画面です。
25    public String confirm() {
        // 確認画面を表示するためのメソッドです。

26        return "crudapp/confirm";
        // templates/crudapp/confirm.html が表示されます。
27    }
28
29    @PostMapping("/confirm")（基礎 P28）
      //「/crudapp/confirm」への POST リクエストを処理します。
      // フォームから送信された入力内容を受け取り、確認画面に渡します。

30    public String confirm(@RequestParam("content") String content, Model model) {
        // リクエストパラメータ「content」を受け取ります。
        // フォームの input 要素の name 属性と対応しています。

31        model.addAttribute("content", content);
        // 受け取った値を Model に追加し、画面側で参照できるようにします。

32        return "crudapp/confirm";
        // 入力内容を反映した確認画面を再表示します。
33    }


34
35    @GetMapping("/complete")
      //「/crudapp/complete」への GET リクエストを処理します。
      // 処理完了後に表示される完了画面を表示します。
36    public String complete() {
        // 完了画面を表示するためのメソッドです。
37        return "crudapp/complete";
        // templates/crudapp/complete.html が表示されます。
38    }
39
40    @PostMapping("/complete")
      //「/crudapp/complete」への POST リクエストを処理します。
      // 確認画面からの確定操作を受け取るためのエンドポイントです。

41    public String completePost() {
        // 現時点では処理を行わず、完了画面を表示します。

42        return "crudapp/complete";
        // templates/crudapp/complete.html が表示されます。
```

```
43     }
44 }
```

データベース構築

docker-compose.yml NEW

```
1
2 services:
3   mysql:
4     image: mysql:8.0
5     container_name: crudapp-mysql
6     restart: unless-stopped
7     environment:
8       MYSQL_DATABASE: crudapp_db
9       MYSQL_USER: crudapp_user
10      MYSQL_PASSWORD: crudapp_pass
11      MYSQL_ROOT_PASSWORD: rootpass
12    ports:
13      - "3306:3306"
14    volumes:
15      - db_data:/var/lib/mysql
16    command: >
17      --character-set-server=utf8mb4
18      --collation-server=utf8mb4_unicode_ci
19      --default-authentication-plugin=caching_sha2_password
20 volumes:
21   db_data:
```

```
1 version: "3.9"
  # docker-compose ファイルのバージョンを指定しています。
  # Docker Compose がどの仕様でこのファイルを解釈するかを決定します。

2 services:
  # この docker-compose で起動するサービス（コンテナ群）を定義します。（基礎 P21）

3   mysql:
    # サービス名です。
    # この名前は、他のコンテナから接続する際のホスト名としても使用されます。

4     image: mysql:8.0
      # 使用する Docker イメージを指定しています。
```

```
    # ここでは MySQL の公式イメージのバージョン 8.0 を使用します。

5   container_name: crudapp-mysql
    # 起動されるコンテナの名前を明示的に指定しています。
    # docker ps などで識別しやすくなります。

6   restart: unless-stopped
    # コンテナの再起動ポリシーを指定しています。
    # Docker が停止しない限り、コンテナが落ちた場合は自動的に再起動されます。

7   environment:
    # コンテナ起動時に渡す環境変数を定義しています。

8     MYSQL_DATABASE: crudapp_db
      # 起動時に自動作成されるデータベース名を指定しています。

9     MYSQL_USER: crudapp_user
      # データベース接続用の一般ユーザー名を指定しています。

10    MYSQL_PASSWORD: crudapp_pass
      # 上記ユーザーのパスワードを指定しています。

11    MYSQL_ROOT_PASSWORD: rootpass
      # MySQL の root ユーザー用パスワードを指定しています。
      # 必須項目であり、未指定の場合はコンテナが起動しません。

12  ports:
    # ホストマシンとコンテナのポートを紐付ける設定です。

13    - "3306:3306"
      # ホスト側の 3306 番ポートを、コンテナ内の 3306 番ポートに接続します。
      # ローカル環境から MySQL クライアントで直接接続できるようになります。

14  volumes:
    # データを永続化するためのボリューム設定です。

15    - db_data:/var/lib/mysql
      # MySQL のデータ保存先を Docker ボリュームにマウントしています。
      # コンテナを削除してもデータが保持されます。

16  command: >
    # MySQL 起動時に追加で渡すオプションを指定しています。
    #「>」は複数行を 1 つの文字列として扱う YAML の書き方です。

17    --character-set-server=utf8mb4
      # デフォルトの文字コードを utf8mb4 に設定しています。
      # 絵文字や多言語を安全に扱うための推奨設定です。
```

```
18      --collation-server=utf8mb4_unicode_ci
        # 文字の比較ルール（照合順序）を指定しています。
        # 大文字・小文字や言語差異を考慮した比較が行われます。

19      --default-authentication-plugin=caching_sha2_password
        # MySQL 8.0 の標準認証方式を明示的に指定しています。
        # Spring Boot などのクライアントとの互換性を保つための設定です。

20 volumes:
  # docker-compose 全体で使用するボリュームを定義します。

21   db_data:
     # MySQL のデータ保存用に使用する名前付きボリュームです。
     # 実体は Docker が管理し、明示的に削除しない限り保持されます。
```

pom.xml

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
 3    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
 4    <modelVersion>4.0.0</modelVersion>
 5    <parent>
 6       <groupId>org.springframework.boot</groupId>
 7       <artifactId>spring-boot-starter-parent</artifactId>
 8       <version>3.4.1</version>
 9       <relativePath/> <!-- lookup parent from repository -->
10    </parent>
11    <groupId>com.example</groupId>
12    <artifactId>demo</artifactId>
13    <version>0.0.1-SNAPSHOT</version>
14    <name>demo</name>
15    <description>demodiscripttion</description>
16    <url/>
17    <licenses>
18       <license/>
19    </licenses>
20    <developers>
21       <developer/>
22    </developers>
23    <scm>
24       <connection/>
25       <developerConnection/>
26       <tag/>
27       <url/>
```

```
28  </scm>
29  <properties>
30    <java.version>17</java.version>
31  </properties>
32  <dependencies>
33    <dependency>
34      <groupId>org.springframework.boot</groupId>
35      <artifactId>spring-boot-starter-jdbc</artifactId>
36    </dependency>
37    <dependency>
38      <groupId>org.springframework.boot</groupId>
39      <artifactId>spring-boot-starter-thymeleaf</artifactId>（基礎 P35）
40    </dependency>
41    <dependency>
42      <groupId>org.springframework.boot</groupId>
43      <artifactId>spring-boot-starter-web</artifactId>
44    </dependency>
45    <dependency>
46      <groupId>org.flywaydb</groupId>
47      <artifactId>flyway-mysql</artifactId>
48    </dependency>
49    <dependency>
50      <groupId>org.flywaydb</groupId>
51      <artifactId>flyway-core</artifactId>
52    </dependency>
53    <dependency>
54      <groupId>com.mysql</groupId>
55      <artifactId>mysql-connector-j</artifactId>
56    </dependency>
57    <dependency>
58      <groupId>org.springframework.boot</groupId>
59      <artifactId>spring-boot-starter-test</artifactId>
60      <scope>test</scope>
61    </dependency>
62  </dependencies>
63  <build>
64    <plugins>
65      <plugin>
66        <groupId>org.springframework.boot</groupId>
67        <artifactId>spring-boot-maven-plugin</artifactId>
68      </plugin>
69    </plugins>
70  </build>
71 </project>
```

```
45      <dependency>
<!-- Flyway（基礎 P50）の MySQL 向け拡張モジュール。MySQL 固有仕様に対応するための追加ライブラリ -->
46          <groupId>org.flywaydb</groupId>
47          <artifactId>flyway-mysql</artifactId>
48      </dependency>
49      <dependency>
            <!-- Flyway の本体。起動時に SQL マイグレーション（基礎 P52）を実行し DB 構造を管理する -->
50          <groupId>org.flywaydb</groupId>
51          <artifactId>flyway-core</artifactId>
52      </dependency>
53      <dependency>
            <!-- MySQL に接続するための JDBC ドライバ。DB と通信するために必須 -->
54          <groupId>com.mysql</groupId>
55          <artifactId>mysql-connector-j</artifactId>
56      </dependency>
```

src/main/resources/application.properties

```
1 spring.application.name=demo
3 spring.datasource.url=jdbc:mysql://localhost:3306/crudapp_db?useSSL=false&allowPublicKeyRetrieval=true&se
rverTimezone=UTC
4 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5 spring.datasource.username=crudapp_user
6 spring.datasource.password=crudapp_pass
7
8 spring.flyway.enabled=true
9
10 spring.jpa.show-sql=true
```

```
3 spring.datasource.url=jdbc:mysql://localhost:3306/crudapp_db（基礎 P41）
?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC
        # 接続先の MySQL 情報。DB 名・SSL 無効・タイムゾーン指定まで含めた JDBC（基礎 P39）
        接続 URL
4 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
        # 使用する MySQL 用 JDBC ドライバクラスを指定

5 spring.datasource.username=crudapp_user
        # MySQL に接続するユーザー名

6 spring.datasource.password=crudapp_pass
        # MySQL に接続するユーザーのパスワード
7
8 spring.flyway.enabled=true
        # アプリ起動時に Flyway による DB マイグレーションを有効化
9
10 spring.jpa.show-sql=true
```

```
# 実行される SQL をコンソールに出力（学習・デバッグ用）
```

src/main/resources/db/migration/V1__create_note.sql NEW

```
1 CREATE TABLE notes (
2   id BIGINT NOT NULL AUTO_INCREMENT,
3   text VARCHAR(1000) NOT NULL,
4   created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
5   PRIMARY KEY (id)
6 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
1 CREATE TABLE notes (
        -- notes という名前のテーブルを新しく作成する
2   id BIGINT NOT NULL AUTO_INCREMENT,
        -- 主キー用の ID。BIGINT で大量データにも耐え、AUTO_INCREMENT で自動採番される
3   text VARCHAR(1000) NOT NULL,
        -- メモ本文を保存するカラム。最大 1000 文字で、必ず値が入る
4   created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
         -- レコード作成日時。INSERT 時に自動で現在時刻が入る
5   PRIMARY KEY (id)
        -- id カラムを主キーとして設定し、一意性と検索性能を確保する
6 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
        -- トランザクション対応の InnoDB を使用し、絵文字も扱える utf8mb4 を指定
```

SpringBoot からデータベースに接続する

MyBatis のを利用できるように設定

pom.xml（基礎 P54）

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5   <parent>
6     <groupId>org.springframework.boot</groupId>
7     <artifactId>spring-boot-starter-parent</artifactId>
8     <version>3.4.1</version>
9     <relativePath/> <!-- lookup parent from repository -->
```

```
</parent>
<groupId>com.example</groupId>
<artifactId>demo</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>demo</name>
<description>demodiscriptttion</description>
<url/>
<licenses>
  <license/>
</licenses>
<developers>
  <developer/>
</developers>
<scm>
  <connection/>
  <developerConnection/>
  <tag/>
  <url/>
</scm>
<properties>
  <java.version>17</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jdbc</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.flywaydb</groupId>
    <artifactId>flyway-mysql</artifactId>
  </dependency>
  <dependency>
    <groupId>org.flywaydb</groupId>
    <artifactId>flyway-core</artifactId>
  </dependency>
  <dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
  </dependency>
```

```
    <dependency>
      <groupId>org.mybatis.spring.boot</groupId>
      <artifactId>mybatis-spring-boot-starter</artifactId>
      <version>3.0.4</version>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>（基礎 P58）

      </plugin>
    </plugins>
  </build>
</project>
```

```
    <dependency>（基礎 P56）
      <!-- Spring Boot で MyBatis（基礎 P42）を使うためのスターター依存関係。
                          MyBatis と Spring の連携設定を自動化する -->
      <groupId>org.mybatis.spring.boot</groupId>
              <!-- MyBatis 公式が提供する Spring Boot 向けグループ -->
      <artifactId>mybatis-spring-boot-starter</artifactId>
              <!-- MyBatis を Spring Boot に組み込むためのスターター -->
      <version>3.0.4</version>
               <!-- Spring Boot 3 系に対応した MyBatis スターターのバージョン -->
    </dependency>
```

application.properties（基礎 P60）

```
spring.application.name=demo

spring.datasource.url=jdbc:mysql://localhost:3306/crudapp_db?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.username=crudapp_user
spring.datasource.password=crudapp_pass

spring.flyway.enabled=true

# ローカル環境向けの SQL ログ（必要に応じて OFF に）
```

```
11 spring.jpa.show-sql=true
12
13 # MyBatis configuration
14 mybatis.mapper-locations=classpath*:mapper/**/*.xml（基礎 P44）（基礎 P62）
15 mybatis.type-aliases-package=com.example.demo
16 mybatis.configuration.map-underscore-to-camel-case=true
```

```
13 # MyBatis configuration
14 mybatis.mapper-locations=classpath*:mapper/**/*.xml

# MyBatis の Mapper XML の配置場所。resources/mapper 配下を再帰的に読み込む

15 mybatis.type-aliases-package=com.example.demo
# エンティティ（DTO/Domain）を型エイリアスとして登録するパッケージ

16 mybatis.configuration.map-underscore-to-camel-case=true
# DB の snake_case と Java の camelCase を自動変換する設定
```

MyBatis 動作確認

- Mapper XML を読み込み（基礎 P48）
- SQL を実行
- Spring Boot 起動時のエラー確認

③ Mapper XML を 1 つ作る（疎通用）

```
src/main/resources/mapper/NoteMapper.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.example.demo.mapper.NoteMapper">

    <!-- 疎通確認用：件数を取るだけ -->
    <select id="count" resultType="int">
        SELECT COUNT(*) FROM notes
    </select>

</mapper>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
        <!-- XML ファイルであることと、文字コードが UTF-8 であることを宣言 -->
```

```
2 <!DOCTYPE mapper
        <!-- MyBatis 用 Mapper XML であることを宣言 -->
3   PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        <!-- MyBatis 公式が定義する Mapper 用 DTD -->
4   "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
        <!-- Mapper XML の構造ルールを示す DTD の URL -->
5
6 <mapper namespace="com.example.demo.mapper.NoteMapper">
        <!-- 対応する Mapper インターフェースの完全修飾クラス名 -->
7
8       <!-- 疎通確認用：件数を取るだけ -->
        <!-- DB 接続・MyBatis 設定が正しいか確認するためのテスト用 SQL -->

9    <select id="count" resultType="int">
        <!-- NoteMapper の count メソッドに対応し、結果を int で受け取る -->

10      SELECT COUNT(*) FROM notes
        <!-- notes テーブルの全レコード件数を取得 -->
11   </select>
12
13 </mapper>
<!-- Mapper 定義の終了 -->
```

④ Mapper インターフェースを作る

src/main/java/com/example/demo/mapper/NoteMapper.java

```
package com.example.demo.mapper;

import org.apache.ibatis.annotations.Mapper;

@Mapper
public interface NoteMapper {

   int count();

}
```

```
1 package com.example.demo.mapper;
        // Mapper インターフェースを配置するパッケージ。XML の namespace と一致させる必要がある
2
3 import org.apache.ibatis.annotations.Mapper;
        // MyBatis にこのインターフェースが Mapper であることを認識させるためのアノテーション
4
5 @Mapper
        // Spring Boot 起動時に MyBatis の Mapper として自動登録され、DI 可能になる（基礎 P16）
```

```
6 public interface NoteMapper {
        // SQL 定義（XML）と Java をつなぐためのインターフェース
7
8   int count();
        // Mapper XML の <select id="count"> に対応するメソッド。件数を int として受け取る
9
10 }
```

⑤ Spring Boot 起動（疎通チェック）

```
./mvnw spring-boot:run
```

> 起動エラーが出ない
> コンソールに MyBatis / HikariCP / Flyway の通常ログが出る
> Mapper XML 読み込みエラーが出ない

通常起動すれば成功です。

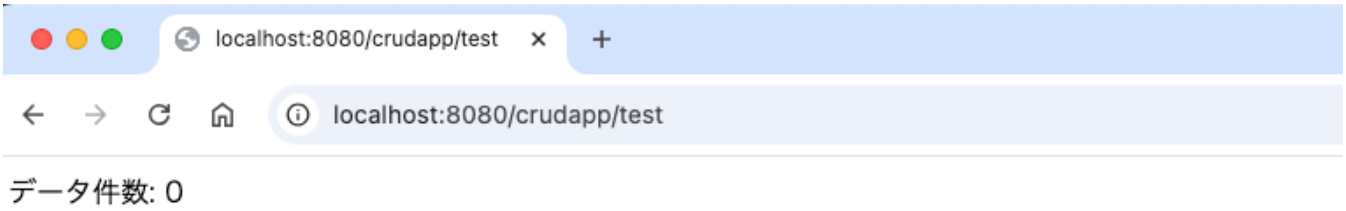
⑥ さらに一歩確認）
Controller から呼んでみます。

```
1 package com.example.demo;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.GetMapping;
5 import org.springframework.web.bind.annotation.PostMapping;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.bind.annotation.RequestParam;
8 import org.springframework.web.bind.annotation.ResponseBody;
9
10 import com.example.demo.mapper.NoteMapper;
11
12 import org.springframework.ui.Model;
13
14 @Controller
15 @RequestMapping("/crudapp")
16 public class CrudappController {
17
18   private final NoteMapper noteMapper;（基礎 P10）
19
20   public CrudappController(NoteMapper noteMapper) {
21     this.noteMapper = noteMapper;
22   }
```

```
23
24  @GetMapping
25  public String index() {
26      return "crudapp/index";
27  }
28
29  @GetMapping("/input")
30  public String input() {
31      return "crudapp/input";
32  }
33
34  @GetMapping("/confirm")
35  public String confirm() {
36      return "crudapp/confirm";
37  }
38
39  @PostMapping("/confirm")
40  public String confirm(@RequestParam("content") String content, Model model) {
41      model.addAttribute("content", content);
42      return "crudapp/confirm";
43  }
44
45  @GetMapping("/complete")
46  public String complete() {
47      return "crudapp/complete";
48  }
49
50  @PostMapping("/complete")
51  public String completePost() {
52      return "crudapp/complete";
53  }
54
55  @GetMapping("/test")
56  @ResponseBody
57  public String test() {
58      int count = noteMapper.count();
59      return " データ件数 : " + count;
60  }
61 }
```

⑦ Spring Boot 起動（疎通チェック）

```
./mvnw spring-boot:run
```



データ取得し一覧画面に表示

demo/CrudappController.java

```
 1 package com.example.demo;
 2
 3 import org.springframework.stereotype.Controller;
 4 import org.springframework.web.bind.annotation.GetMapping;
 5 import org.springframework.web.bind.annotation.PostMapping;
 6 import org.springframework.web.bind.annotation.RequestMapping;
 7 import org.springframework.web.bind.annotation.RequestParam;
 8 import org.springframework.web.bind.annotation.ResponseBody;
 9
10 import com.example.demo.mapper.NoteMapper;
11 import com.example.demo.model.Note;
12
13 import org.springframework.ui.Model;
14 import java.util.List;
15
16 @Controller
17 @RequestMapping("/crudapp")
18 public class CrudappController {
19
20     private final NoteMapper noteMapper;
21
22     public CrudappController(NoteMapper noteMapper) {
23         this.noteMapper = noteMapper;
24     }
25
26     @GetMapping("/index")
27     public String index() {
27     public String index(Model model) {
28         List<Note> notes = noteMapper.findAll();
29         model.addAttribute("notes", notes);
30         return "crudapp/index";
31     }
32
```

```
33    @GetMapping("/input")
34    public String input() {
35        return "crudapp/input";
36    }
37
38    @GetMapping("/confirm")
39    public String confirm() {
40        return "crudapp/confirm";
41    }
42
43    @PostMapping("/confirm")
44    public String confirm(@RequestParam("content") String content, Model model) {
45        model.addAttribute("content", content);
46        return "crudapp/confirm";
47    }
48
49    @GetMapping("/complete")
50    public String complete() {
51        return "crudapp/complete";
52    }
53
54    @PostMapping("/complete")
55    public String completePost() {
56        return "crudapp/complete";
57    }
58
59    @GetMapping("/test")
60    @ResponseBody
61    public String test() {
62        int count = noteMapper.count();
63        return " データ件数 : " + count;
64    }
65 }
```

追加分のみ表示

```
11 import com.example.demo.model.Note;
            // notes テーブルに対応するドメイン（モデル）クラス

14 import java.util.List;
            // 複数件データを扱うための List

27    public String index(Model model) {
            // 一覧画面に表示するデータを Model に詰めるための正しい定義

28        List<Note> notes = noteMapper.findAll();
            // notes テーブルの全件データを取得
```

```
29      model.addAttribute("notes", notes);（基礎 P36）
          // 取得した一覧を View に渡す
```

demo/mapper/NoteMapper.java

```
1 package com.example.demo.mapper;
2
3 import java.util.List;
4
5 import org.apache.ibatis.annotations.Mapper;
6
7 import com.example.demo.model.Note;
8
9 @Mapper
10 public interface NoteMapper {
11
12    int count();
13
14    List<Note> findAll();
15 }
```

```
1 package com.example.demo.mapper;      // MyBatis の Mapper インターフェースを配置するパッケージ
2
3 import java.util.List;                // 複数件のデータを扱うためのコレクション型
4
5 import org.apache.ibatis.annotations.Mapper;
          // このインターフェースを MyBatis の Mapper として認識させるためのアノテーション
6
7 import com.example.demo.model.Note;  // notes テーブル 1 件分を表すモデル（ドメイン）クラス
8
9 @Mapper                               // Spring Boot 起動時に Mapper として自動登録され、DI 可能になる
10 public interface NoteMapper {        // SQL（XML）と Java を結びつけるためのインターフェース
11
12    int count();                      // notes テーブルの件数を取得する SQL に対応するメソッド
13
14    List<Note> findAll();             // notes テーブルの全件データを取得し、Note の一覧として受け取る
15 }
```

demo/model/Note.java　NEW

```
1 package com.example.demo.model;
2
```

```
3 import java.time.LocalDateTime;
4
5 public class Note {
6    private Long id;
7    private String text;
8    private LocalDateTime createdAt;
9
10    public Long getId() {
11       return id;
12    }
13
14    public void setId(Long id) {
15       this.id = id;
16    }
17
18    public String getText() {
19       return text;
20    }
21
22    public void setText(String text) {
23       this.text = text;
24    }
25
26    public LocalDateTime getCreatedAt() {
27       return createdAt;
28    }
29
30    public void setCreatedAt(LocalDateTime createdAt) {
31       this.createdAt = createdAt;
32    }
33 }
```

```
1 package com.example.demo.model;      // モデル（ドメイン）クラスを配置するパッケージ
2
3 import java.time.LocalDateTime;      // 作成日時などの日付・時刻を扱うためのクラス
4
5 public class Note {                  // notes テーブル 1 件分のデータを表すモデルクラス
6    private Long id;                  // 主キーに対応する ID。DB の AUTO_INCREMENT と対応
7    private String text;               // メモ本文に対応するカラム
8    private LocalDateTime createdAt;  // 作成日時。DB の created_at カラムと対応
9
10    public Long getId() {            // id フィールドの値を取得する getter
11       return id;                    // 現在の id 値を返す
12    }
13
14    public void setId(Long id) {     // id フィールドに値を設定する setter
```

```
15      this.id = id;                           // 引数（基礎 P14）で受け取った値をフィールドに代入
16    }
17
18    public String getText() {                 // text フィールドの値を取得する getter

19      return text;                            // メモ本文を返す
20    }
21
22    public void setText(String text) {        // text フィールドに値を設定する setter
23      this.text = text;                       // 引数で受け取った本文をフィールドに代入
24    }
25
26    public LocalDateTime getCreatedAt() {                 // createdAt フィールドの値を取得する getter
27      return createdAt;                                   // 作成日時を返す
28    }
29
30    public void setCreatedAt(LocalDateTime createdAt) {   // createdAt フィールドに値を設定する setter
31      this.createdAt = createdAt;                         // 引数で受け取った日時をフィールドに代入
32    }
33 }
```

resources/mapper/NoteMapper.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE mapper
3   PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4   "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5
6 <mapper namespace="com.example.demo.mapper.NoteMapper">
7
8    <!-- 疎通確認用：件数を取るだけ -->
9    <select id="count" resultType="int">
10       SELECT COUNT(*) FROM notes
11    </select>
12
13    <select id="findAll" resultType="com.example.demo.model.Note">
14       SELECT
15          id,
16          text,
17          created_at
18       FROM notes
19       ORDER BY id DESC
20    </select>
21 </mapper>
```

13 <select id="findAll" resultType="com.example.demo.model.Note">

<!-- NoteMapper の findAll() メソッドに対応し、結果を Note クラスとして受け取る -->

resources/templetes/crudapp/index.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <title>CRUD App - 一覧 </title>
</head>
<body>
<h1>CRUD App: 一覧 </h1>
<p> ここに一覧を表示します。</p>
<nav>
  <ul>
    <li><a href="/crudapp/input"> 新規登録 </a></li>
    <li><a href="/crudapp/confirm"> 登録確認 </a></li>
    <li><a href="/crudapp/complete"> 登録完了 </a></li>
  </ul>
  <a href="/crudapp"> 一覧へ </a>
</nav>
<div>
  <a href="/crudapp/input"> 新規登録 </a>
</div>

<div th:if="${#lists.isEmpty(notes)}">
  <p> データがありません </p>
</div>

<table border="1" cellspacing="0" cellpadding="8" th:if="${!#lists.isEmpty(notes)}">
  <thead>
  <tr>
    <th>ID</th>
    <th>Text</th>
    <th>Created At</th>
  </tr>
  </thead>
  <tbody>
  <tr th:each="note : ${notes}">
    <td th:text="${note.id}">1</td>
    <td th:text="${note.text}"> サンプル </td>
    <td th:text="${note.createdAt}">2024-01-01T00:00:00</td>
  </tr>
  </tbody>
</table>
</body>
</html>
```

フォームデータを登録

2) NoteMapper に insert を追加

demo/mapper/NoteMapper.java

```
 1 package com.example.demo.mapper;
 2
 3 import java.util.List;
 4
 5 import org.apache.ibatis.annotations.Mapper;
 6
 7 import com.example.demo.model.Note;
 8
 9 @Mapper
10 public interface NoteMapper {
11
12     int count();
13
14     List<Note> findAll();
15
16     int insert(String text);
17 }
```

```
16     int insert(String text);              // メモ本文を新規登録する SQL に対応し、登録された行数を返す
```

3) NoteMapper.xml に INSERT を追加
resources/mapper/NoteMapper.xml

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <!DOCTYPE mapper
 3   PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
 4   "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
 5
 6 <mapper namespace="com.example.demo.mapper.NoteMapper">
 7
 8   <!-- 疎通確認用：件数を取るだけ -->
 9   <select id="count" resultType="int">
10       SELECT COUNT(*) FROM notes
11   </select>
12
13   <select id="findAll" resultType="com.example.demo.model.Note">
14       SELECT
15           id,
16           text,
17           created_at
18       FROM notes
19       ORDER BY id DESC
20   </select>
21
22   <insert id="insert" parameterType="string">
23     INSERT INTO notes (text)
24     VALUES (#{text})
25   </insert>
26 </mapper>
```

4) Controller の completePost で登録する
demo/CrudappController.java

```
 1 package com.example.demo;
 2
 3 import org.springframework.stereotype.Controller;
 4 import org.springframework.web.bind.annotation.GetMapping;
 5 import org.springframework.web.bind.annotation.PostMapping;
 6 import org.springframework.web.bind.annotation.RequestMapping;
 7 import org.springframework.web.bind.annotation.RequestParam;
 8 import org.springframework.web.bind.annotation.ResponseBody;
 9
10 import com.example.demo.mapper.NoteMapper;
11 import com.example.demo.model.Note;
12
13 import org.springframework.ui.Model;
```

```
14 import java.util.List;
15
16 @Controller
17 @RequestMapping("/crudapp")
18 public class CrudappController {
19
20   private final NoteMapper noteMapper;
21
22   public CrudappController(NoteMapper noteMapper) {
23     this.noteMapper = noteMapper;
24   }
25
26   @GetMapping("/index")
27   public String index(Model model) {
28     List<Note> notes = noteMapper.findAll();
29     model.addAttribute("notes", notes);
30     return "crudapp/index";
31   }
32
33   @GetMapping("/input")
34   public String input() {
35     return "crudapp/input";
36   }
37
38   @GetMapping("/confirm")
39   public String confirm() {
40     return "crudapp/confirm";
41   }
42
43   @PostMapping("/confirm")
44   public String confirm(@RequestParam("content") String content, Model model) {
45     model.addAttribute("content", content);
46     return "crudapp/confirm";
47   }
48
49   @GetMapping("/complete")
50   public String complete() {
51     return "crudapp/complete";
52   }
53
54   @PostMapping("/complete")
55   public String completePost() {
55   public String completePost(@RequestParam("content") String content) {
56     noteMapper.insert(content);
57     return "redirect:/crudapp";
58   }
59
```

```
60    @GetMapping("/test")
61    @ResponseBody
62    public String test() {
63        int count = noteMapper.count();
64        return " データ件数 : " + count;
65    }
66 }
```

```
54    @PostMapping("/complete")
          // フォーム送信（POST）された /complete リクエストを処理する
55    public String completePost(@RequestParam("content") String content) {
          // 画面から送信された content パラメータを受け取る
56        noteMapper.insert(content);
          // MyBatis の Mapper を使って、入力内容を DB（notes テーブル）に登録する
57        return "redirect:/crudapp";
          // 登録処理後に一覧画面へリダイレクトし、二重送信を防止する
58    }
```

登録されたデータを編集

demo/CrudappController.java

```
 1 package com.example.demo;
 2
 3 import org.springframework.stereotype.Controller;
 4 import org.springframework.web.bind.annotation.GetMapping;
 5 import org.springframework.web.bind.annotation.PostMapping;
 6 import org.springframework.web.bind.annotation.RequestMapping;
 7 import org.springframework.web.bind.annotation.RequestParam;
 8 import org.springframework.web.bind.annotation.ResponseBody;
 9
10 import com.example.demo.mapper.NoteMapper;
11 import com.example.demo.model.Note;
12
13 import org.springframework.ui.Model;
14 import java.util.List;
15
16 @Controller
17 @RequestMapping("/crudapp")
18 public class CrudappController {
19
20    private final NoteMapper noteMapper;
21
22    public CrudappController(NoteMapper noteMapper) {
23        this.noteMapper = noteMapper;
```

```
24  }
25
26  @GetMapping("/index")
27  public String index(Model model) {
28      List<Note> notes = noteMapper.findAll();
29      model.addAttribute("notes", notes);
30      return "crudapp/index";
31  }
32
33  @GetMapping("/input")
34  public String input() {
35      return "crudapp/input";
36  }
37
38  @GetMapping("/confirm")
39  public String confirm() {
40      return "crudapp/confirm";
41  }
42
43  @PostMapping("/confirm")
44  public String confirm(@RequestParam("content") String content, Model model) {
45      model.addAttribute("content", content);
46      return "crudapp/confirm";
47  }
48
49  @GetMapping("/complete")
50  public String complete() {
51      return "crudapp/complete";
52  }
53
54  @PostMapping("/complete")
55  public String completePost(@RequestParam("content") String content) {
56       noteMapper.insert(content);
57       return "redirect:/crudapp";
58  }
59
60  @GetMapping("/edit")
61  public String edit(@RequestParam("id") Long id, Model model) {
62      Note note = noteMapper.findById(id);
63      model.addAttribute("note", note);
64      return "crudapp/edit";
65  }
66
67  @PostMapping("/edit")
68  public String update(@RequestParam("id") Long id, @RequestParam("content") String content) {
69      Note note = new Note();
70      note.setId(id);
```

```
71      note.setText(content);
72      noteMapper.update(note);
73      return "redirect:/crudapp";
74  }
75
76  @GetMapping("/test")
77  @ResponseBody
78  public String test() {
79      int count = noteMapper.count();
80      return " データ件数 : " + count;
81  }
82 }
```

```
60  @GetMapping("/edit")
        // 編集画面を表示するための GET リクエストを処理
61  public String edit(@RequestParam("id") Long id, Model model) {
        // 編集対象の ID を受け取り、画面表示用の Model を利用
62      Note note = noteMapper.findById(id);
        // 指定された ID のデータを DB から取得
63      model.addAttribute("note", note);
        // 取得したデータを View に渡す
64      return "crudapp/edit";
        // 編集画面のテンプレートを表示
65  }
66
67  @PostMapping("/edit")
        // 編集内容を保存するための POST リクエストを処理
68  public String update(@RequestParam("id") Long id, @RequestParam("content") String content) {
        // 更新対象 ID と入力内容を受け取る
69      Note note = new Note();
        // 更新用の Note オブジェクトを生成
70      note.setId(id);
         // 更新対象を特定するために ID を設定
71      note.setText(content);
         // 新しい本文を設定
72      noteMapper.update(note);
        // Mapper を通して DB の更新処理を実行
73      return "redirect:/crudapp";
        // 更新後は一覧画面へリダイレクト
74  }
```

demo/mapper/NoteMapper.java

```
1 package com.example.demo.mapper;
2
3 import java.util.List;
```

```
4
5 import org.apache.ibatis.annotations.Mapper;
6
7 import com.example.demo.model.Note;
8
9 @Mapper
10 public interface NoteMapper {
11
12    int count();
13
14    List<Note> findAll();
15
16    int insert(String text);
17
18    Note findById(Long id);
19
20    int update(Note note);
21 }
```

resources/mapper/NoteMapper.html

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE mapper
3   PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4   "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5
6 <mapper namespace="com.example.demo.mapper.NoteMapper">
7
8    <!-- 疎通確認用：件数を取るだけ -->
9    <select id="count" resultType="int">
10       SELECT COUNT(*) FROM notes
11    </select>
12
13    <select id="findAll" resultType="com.example.demo.model.Note">
14       SELECT
15          id,
16          text,
17          created_at
18       FROM notes
19       ORDER BY id DESC
20    </select>
21
22    <insert id="insert" parameterType="string">
23      INSERT INTO notes (text)
24      VALUES (#{text})
25    </insert>
26
```

```
27    <select id="findById" parameterType="long" resultType="com.example.demo.model.Note">
28      SELECT
29        id,
30        text,
31        created_at
32      FROM notes
33      WHERE id = #{id}
34    </select>
35
36    <update id="update" parameterType="com.example.demo.model.Note">
37      UPDATE notes
38      SET text = #{text}
39      WHERE id = #{id}
40    </update>
41
42 </mapper>
```

resources/templates/crudapp/index.html

```
 1 <!DOCTYPE html>
 2 <html lang="ja">
 3 <head>
 4    <meta charset="UTF-8">
 5    <title>CRUD App - 一覧 </title>
 6 </head>
 7 <body>
 8 <h1>CRUD App: 一覧 </h1>
 9 <div>
10    <a href="/crudapp/input"> 新規登録 </a>
11 </div>
12
13 <div th:if="${#lists.isEmpty(notes)}">
14    <p> データがありません </p>
15 </div>
16
17 <table border="1" cellspacing="0" cellpadding="8" th:if="${!#lists.isEmpty(notes)}">
18    <thead>
19    <tr>
20      <th>ID</th>
21      <th>Text</th>
22      <th>Created At</th>
23    </tr>
24    </thead>
25    <tbody>
26    <tr th:each="note : ${notes}">
27      <td th:text="${note.id}">1</td>
```

```
28       <td th:text="${note.text}"> サンプル </td>
29       <td th:text="${note.createdAt}">2024-01-01T00:00:00</td>
30       <td>
31          <a th:href="@{/crudapp/edit(id=${note.id})}"> 編集 </a>
32       </td>
33    </tr>
34    </tbody>
35 </table>
36 </body>
37 </html>
```

resources/templates/crudapp/index.html NEW

```
<!DOCTYPE html>
<html lang="ja">
<head>
   <meta charset="UTF-8">
   <title>CRUD App - 編集 </title>
</head>
<body>
<h1>CRUD App: 編集 </h1>
<form th:action="@{/crudapp/edit}" method="post">
   <input type="hidden" name="id" th:value="${note.id}">
   <label for="content"> 内容 </label><br>
   <textarea id="content" name="content" rows="6" cols="50" required
         th:text="${note.text}"> 既存の内容 </textarea><br>
   <button type="submit"> 更新する </button>
</form>
<nav>
   <a href="/crudapp"> 一覧へ戻る </a>
</nav>
</body>
</html>
```

登録されたデータを削除

demo/CrudappController.java

```
1 package com.example.demo;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.GetMapping;
5 import org.springframework.web.bind.annotation.PostMapping;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.bind.annotation.RequestParam;
8 import org.springframework.web.bind.annotation.ResponseBody;
```

```
 9
10 import com.example.demo.mapper.NoteMapper;
11 import com.example.demo.model.Note;
12
13 import org.springframework.ui.Model;
14 import java.util.List;
15
16 @Controller
17 @RequestMapping("/crudapp")
18 public class CrudappController {
19
20     private final NoteMapper noteMapper;
21
22     public CrudappController(NoteMapper noteMapper) {
23         this.noteMapper = noteMapper;
24     }
25
26     @GetMapping("/index")
27     public String index(Model model) {
28         List<Note> notes = noteMapper.findAll();
29         model.addAttribute("notes", notes);
30         return "crudapp/index";
31     }
32
33     @GetMapping("/input")
34     public String input() {
35         return "crudapp/input";
36     }
37
38     @GetMapping("/confirm")
39     public String confirm() {
40         return "crudapp/confirm";
41     }
42
43     @PostMapping("/confirm")
44     public String confirm(@RequestParam("content") String content, Model model) {
45         model.addAttribute("content", content);
46         return "crudapp/confirm";
47     }
48
49     @GetMapping("/complete")
50     public String complete() {
51         return "crudapp/complete";
52     }
53
54     @PostMapping("/complete")
55     public String completePost(@RequestParam("content") String content) {
```

```
56        noteMapper.insert(content);
57        return "redirect:/crudapp";
58    }
59
60    @GetMapping("/edit")
61    public String edit(@RequestParam("id") Long id, Model model) {
62        Note note = noteMapper.findById(id);
63        model.addAttribute("note", note);
64        return "crudapp/edit";
65    }
66
67    @PostMapping("/edit")
68    public String update(@RequestParam("id") Long id, @RequestParam("content") String content) {
69        Note note = new Note();
70        note.setId(id);
71        note.setText(content);
72        noteMapper.update(note);
73        return "redirect:/crudapp";
74    }
75
76    @PostMapping("/delete")
77    public String delete(@RequestParam("id") Long id) {
78        noteMapper.softDelete(id);
79        return "redirect:/crudapp";
80    }
81
82    @PostMapping("/toggle")
83    public String toggleVisibility(@RequestParam("id") Long id, @RequestParam("isVisible") Integer isVisible) {
84        int newVisible = (isVisible != null && isVisible == 1) ? 0 : 1;
85        noteMapper.toggleVisibility(id, newVisible);
86        return "redirect:/crudapp";
87    }
88
89    @GetMapping("/test")
90    @ResponseBody
91    public String test() {
92        int count = noteMapper.count();
93        return " データ件数 : " + count;
94    }
95 }
```

```
60    @GetMapping("/edit")
          // 編集画面を表示するための GET リクエストを処理する
61    public String edit(@RequestParam("id") Long id, Model model) {
          // 編集対象の ID を受け取り、画面に渡すための Model を使用する
62        Note note = noteMapper.findById(id);
          // 指定された ID に対応するデータをデータベースから取得する
```

```
63      model.addAttribute("note", note);
          // 取得したデータを編集画面で使えるように Model へ格納する
64      return "crudapp/edit";
          // 編集用テンプレートを表示する
65  }
66
67  @PostMapping("/edit")
          // 編集内容を送信した後の POST リクエストを処理する
68  public String update(@RequestParam("id") Long id, @RequestParam("content") String content) {
          // 更新対象の ID と新しい本文を受け取る
69      Note note = new Note();
          // 更新処理に使用する Note オブジェクトを新しく作成する
70      note.setId(id);
          // どのデータを更新するかを特定するために ID を設定する
71      note.setText(content);
          // フォームで入力された新しい内容を設定する
72      noteMapper.update(note);
          // Mapper を通してデータベースの更新処理を実行する
73      return "redirect:/crudapp";
          // 更新後は一覧画面へリダイレクトし、二重送信を防止する
74  }
```

demo/mapper/NoteMapper.java

```
1 package com.example.demo.mapper;
2
3 import java.util.List;
4
5 import org.apache.ibatis.annotations.Mapper;
6 import org.apache.ibatis.annotations.Param;
7
8 import com.example.demo.model.Note;
9
10 @Mapper
11 public interface NoteMapper {
12
13    int count();
14
15    List<Note> findAll();
16
17    int insert(String text);
18
19    Note findById(Long id);
20
21    int update(Note note);
22
23    int softDelete(Long id);
```

```
24
25    int toggleVisibility(@Param("id") Long id, @Param("isVisible") Integer isVisible);
26 }
```

```
6 import org.apache.ibatis.annotations.Param;
        // 複数パラメータを SQL に渡す際に名前付きで指定するためのアノテーション
23    int softDelete(Long id);
        // データを物理削除せず、削除フラグを立てる論理削除用 SQL に対応
25    int toggleVisibility(@Param("id") Long id, @Param("isVisible") Integer isVisible);
        // 表示／非表示を切り替える更新処理で、複数引数を名前付きで SQL に渡す
```

demo/model/Note.java

```
 1 package com.example.demo.model;
 2
 3 import java.time.LocalDateTime;
 4
 5 public class Note {
 6    private Long id;
 7    private String text;
 8    private LocalDateTime createdAt;
 9    private Integer isDeleted;
10    private Integer isVisible;
11
12    public Long getId() {
13        return id;
14    }
15
16    public void setId(Long id) {
17        this.id = id;
18    }
19
20    public String getText() {
21        return text;
22    }
23
24    public void setText(String text) {
25        this.text = text;
26    }
27
28    public LocalDateTime getCreatedAt() {
29        return createdAt;
30    }
31
```

```
32    public void setCreatedAt(LocalDateTime createdAt) {
33        this.createdAt = createdAt;
34    }
35
36    public Integer getIsDeleted() {
37        return isDeleted;
38    }
39
40    public void setIsDeleted(Integer isDeleted) {
41        this.isDeleted = isDeleted;
42    }
43
44    public Integer getIsVisible() {
45        return isVisible;
46    }
47
48    public void setIsVisible(Integer isVisible) {
49        this.isVisible = isVisible;
50    }
51 }
```

```
32    public void setCreatedAt(LocalDateTime createdAt) {
          // 作成日時を表す createdAt フィールドに値を設定するための setter
33        this.createdAt = createdAt;
          // 引数で受け取った日時をフィールドに代入する
34    }
35
36    public Integer getIsDeleted() {
          // 論理削除状態を表す isDeleted フィールドの値を取得する getter
37        return isDeleted;
          // 現在の削除フラグの値を返す
38    }
39
40    public void setIsDeleted(Integer isDeleted) {
          // 論理削除フラグを設定するための setter
41        this.isDeleted = isDeleted;
          // 引数で受け取った削除フラグをフィールドに代入する
42    }
43
44    public Integer getIsVisible() {
          // 表示／非表示の状態を表す isVisible フィールドの値を取得する getter
45        return isVisible;
          // 現在の表示状態の値を返す
46    }
47
48    public void setIsVisible(Integer isVisible) {
```

```
        // 表示／非表示フラグを設定するための setter
49      this.isVisible = isVisible;
        // 引数で受け取った表示フラグをフィールドに代入する
50  }
```

resources/db/migration/V2__add_flags_to_notes.sql　NEW

```
1 ALTER TABLE notes
2   ADD COLUMN is_deleted TINYINT(1) NOT NULL DEFAULT 0 AFTER created_at,
3   ADD COLUMN is_visible TINYINT(1) NOT NULL DEFAULT 1 AFTER is_deleted;
```

resources/mapper/NoteMapper.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE mapper
3   PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4   "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5
6 <mapper namespace="com.example.demo.mapper.NoteMapper">
7
8   <!-- 疎通確認用：件数を取るだけ -->
9   <select id="count" resultType="int">
10      SELECT COUNT(*) FROM notes
11    </select>
12
13    <select id="findAll" resultType="com.example.demo.model.Note">
14      SELECT
15        id,
16        text,
17        created_at
17        created_at,
18        is_deleted,
19        is_visible
20      FROM notes
21      WHERE is_deleted = 0
22        AND is_visible = 1
23      ORDER BY id DESC
24    </select>
25
26    <insert id="insert" parameterType="string">
27      INSERT INTO notes (text)
28      VALUES (#{text})
29    </insert>
```

```
30
31   <select id="findById" parameterType="long" resultType="com.example.demo.model.Note">
32     SELECT
33       id,
34       text,
35       created_at
35       created_at,
36       is_deleted,
37       is_visible
38     FROM notes
39     WHERE id = #{id}
40   </select>
41
42   <update id="update" parameterType="com.example.demo.model.Note">
43     UPDATE notes
44     SET text = #{text}
45     WHERE id = #{id}
46   </update>
47
48   <update id="softDelete" parameterType="long">
49     UPDATE notes
50     SET is_deleted = 1
51     WHERE id = #{id}
52   </update>
53
54   <update id="toggleVisibility">
55     UPDATE notes
56     SET is_visible = #{isVisible}
57     WHERE id = #{id}
58   </update>
59 </mapper>
```

resources/templates/crudapp/index.html

```
1 <!DOCTYPE html>
2 <html lang="ja">
3 <head>
4   <meta charset="UTF-8">
5   <title>CRUD App - 一覧 </title>
6 </head>
7 <body>
8 <h1>CRUD App: 一覧 </h1>
9 <div>
10   <a href="/crudapp/input"> 新規登録 </a>
11 </div>
12
13 <div th:if="${#lists.isEmpty(notes)}">
```

```
14    <p> データがありません </p>
15 </div>
16
17 <table border="1" cellspacing="0" cellpadding="8" th:if="${!#lists.isEmpty(notes)}">
18    <thead>
19    <tr>
20      <th>ID</th>
21      <th>Text</th>
22      <th>Created At</th>
23      <th> 編集 </th>
24      <th> 削除 </th>
25      <th> 表示切替 </th>
26    </tr>
27    </thead>
28    <tbody>
29    <tr th:each="note : ${notes}">
30      <td th:text="${note.id}">1</td>
31      <td th:text="${note.text}"> サンプル </td>
32      <td th:text="${note.createdAt}">2024-01-01T00:00:00</td>
33      <td><a th:href="@{/crudapp/edit(id=${note.id})}"> 編集 </a></td>
34      <td>
35        <form th:action="@{/crudapp/delete}" method="post">
36          <input type="hidden" name="id" th:value="${note.id}">
37          <button type="submit"> 削除 </button>
38        </form>
39      </td>
40      <td>
31        <a th:href="@{/crudapp/edit(id=${note.id})}"> 編集 </a>
41        <form th:action="@{/crudapp/toggle}" method="post">
42          <input type="hidden" name="id" th:value="${note.id}">
43          <input type="hidden" name="isVisible" th:value="${note.isVisible}">
44              <button type="submit" th:text="${note.isVisible == 1} ? ' 非表示にする ' : ' 表示する '"> 表示切替 </
button>
45        </form>
46      </td>
47    </tr>
48    </tbody>
49 </table>
50 </body>
51 </html>
```