v2158269
ラスエドジャメル

2023 年度 ソフトウェア開発演習 2
最終レポート

(1) 作成したアプリケーションについて

本課題では、Java のウェブアプリ開発フレームワーク「Spring Boot」を利用して、「食品マップ」という単純なデータの可視化ツールの開発に取り組んだ。

このアプリは、バックエンド側で Java の MVC（モデル、ビュー、コントローラ）アーキテクチャで構築されており、クライアント側は html でユーザインターフェースを表示する。

本課題では、e-stat.go が提供している「都市階級・地方・都道府県庁所在市別（支出金額及び購入数量のみ）－二人以上の世帯」のデータセット、具体的に食料のデータセット（消費物には食料以外のデータセットも含まれている）を利用して、日本の地図に可視化した。ユーザが、このデータセットより提供されている食品（飲み物を含む）のカテゴリーから選択、または検索ができ、選択されたカテゴリー、例えば「ねぎ」（次ページの画面キャプチャを参照）を e-stat.go の api からデータを要求して、応答が到着したら必要なデータを抽出し、ユーザに地図の形で表示する。

地図の作成には Google 社が提供している Google Chart Api を利用して地図の作成・表示を行う（クライアント側の JavaScript で）。Java 言語では地図を作成するライブラリーがあまり無かったため Google の API の利用が選ばれた。この API を利用するために、2 列のテーブルという構造でデータを Google API に送信する。1 列目は都道府県名、2 列目は可視化した食品の 1 世帯あたりの年間消費量（単位付き）。可視化されたデータを見ると、どの件がどの食品を一番消費しているか、あるいは他のけんに比べて全く消費量が少ないなどのような情報が簡単にわかるやすく表示できる（次ページの画像を参照）。

また、このウェブアプリでは、複数のユーザが同時に利用することが可能となっており、ユーザ認証はセッション ID で行う。これは Spring Boot が提供しているクライアント側のデータの保存方法に基づいて実装ができた。または、そのユーザのアカウントが存在しなかった場合に、アカウント作成も可能である。本課題では認証は非常に単純な実装になっており、サーバ側では、ただ csv ファイルに保存する。

それに加えて、ユーザが登録したユーザ名とパスワードをその時に利用してログインし、ホーム画面から食品のカテゴリを選択、または検索する。E-stat.go から情報を要求する前に、サーバ側で選択されたカテゴリーの有無について調査する。これは、ユーザが最初にウェブアプリに至った時に、このデータセットのメタ情報を取得しサーバで保存する。それによって API の利用を最小限にすることができる。また、Spring Boot のキャッシュ機能を利用して、ユーザへの応答を高速化することができた。

(2) 実装計画と最終的な成果

- 最初に、ログインボタンのあるホーム画面が表示される。

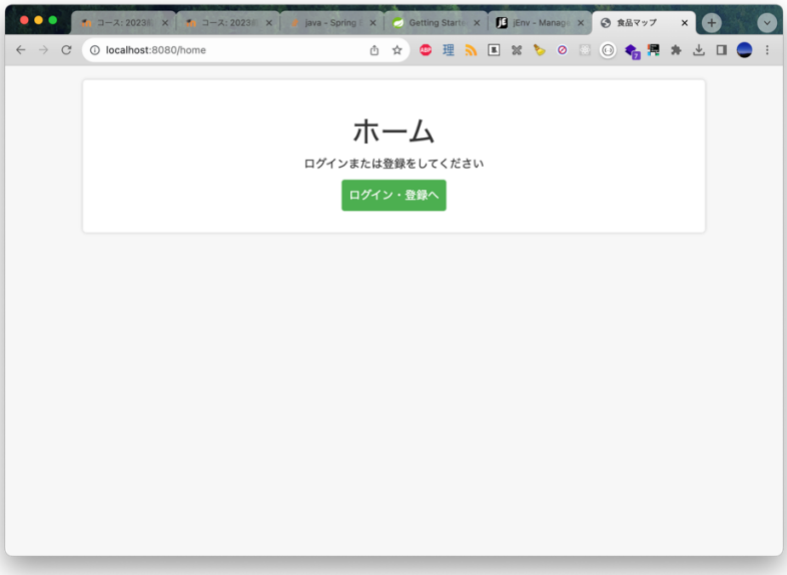

図 1.ホーム画面（認証なし）

- その「ログイン・登録へ」のボタンにクリックすると、ユーザがユーザ名とパスワードが入力できる画面に移動される。

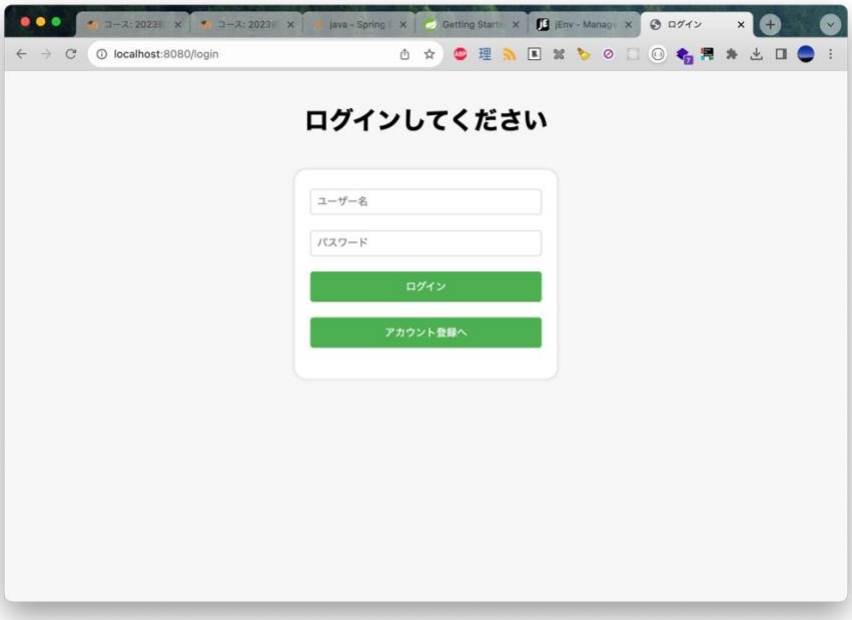

図 2.　ログイン画面

- 本課題の開発にあたり提供されたユーザの模擬データを利用して、認証機能が実装できた。または、ユーザのアカウント登録も可能である。

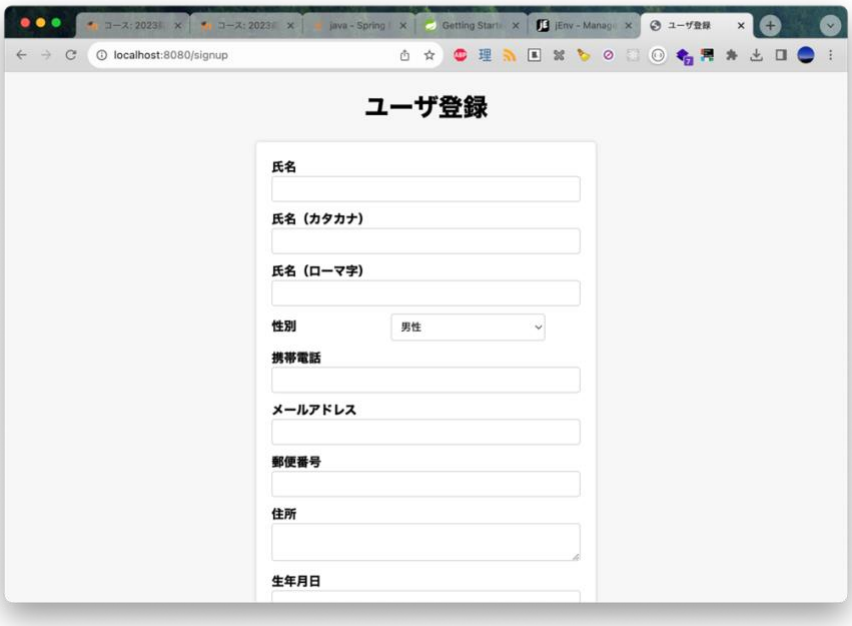

図 3.1. ユーザ情報登録画面(入力前に)

- ユーザ情報を入力し、サーバ側で保存されている csv ファイルにそのデータを書き込みすることができる。そのあと、認証するときに再びそのファイルから情報を読み込むことが可能である

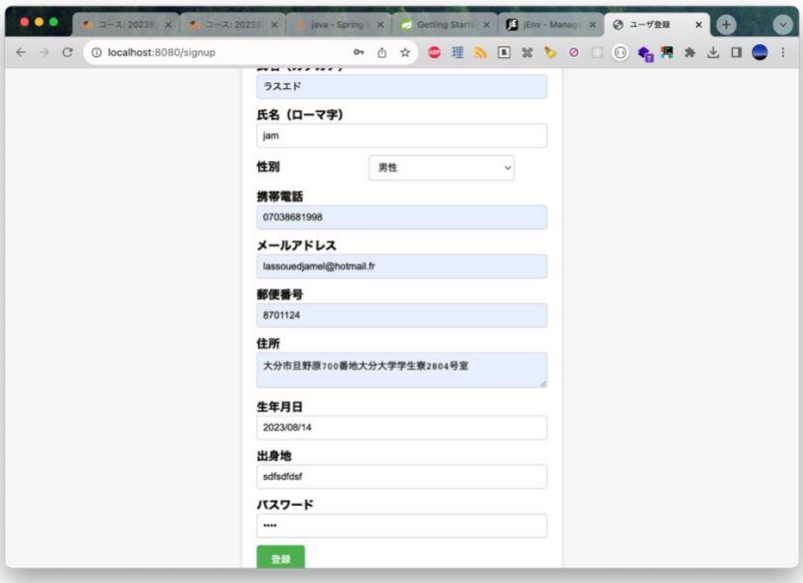

図 3.2. ユーザ情報登録画面(入力後)

- ユーザが入力した情報を確認し、ログイン画面へ戻るボタンが表示される。

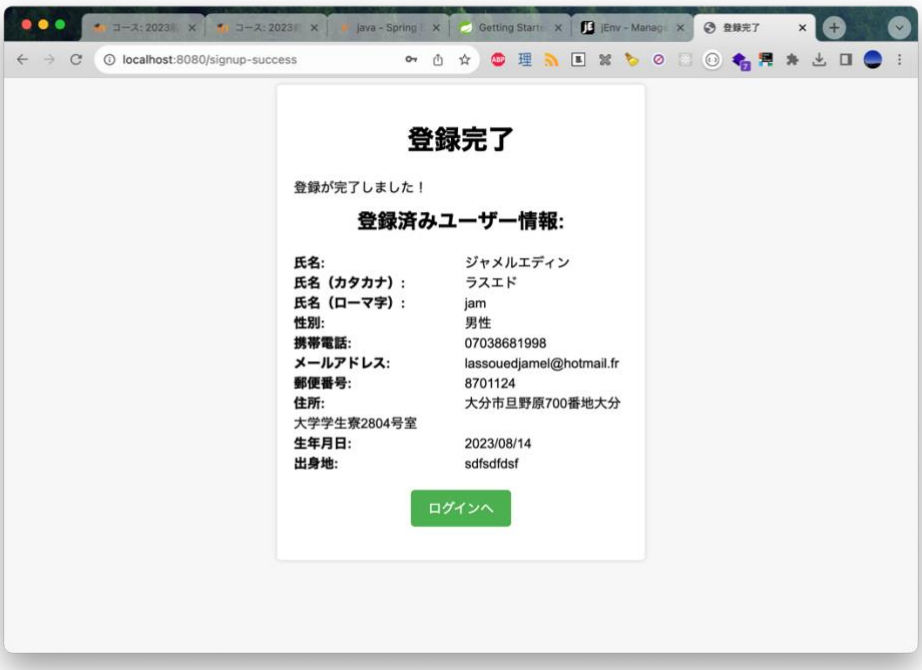

図 4.ユーザ情報の確認画面

- 直前に登録したユーザ名とパスワードを入力し、本機能にアクセスすることができる

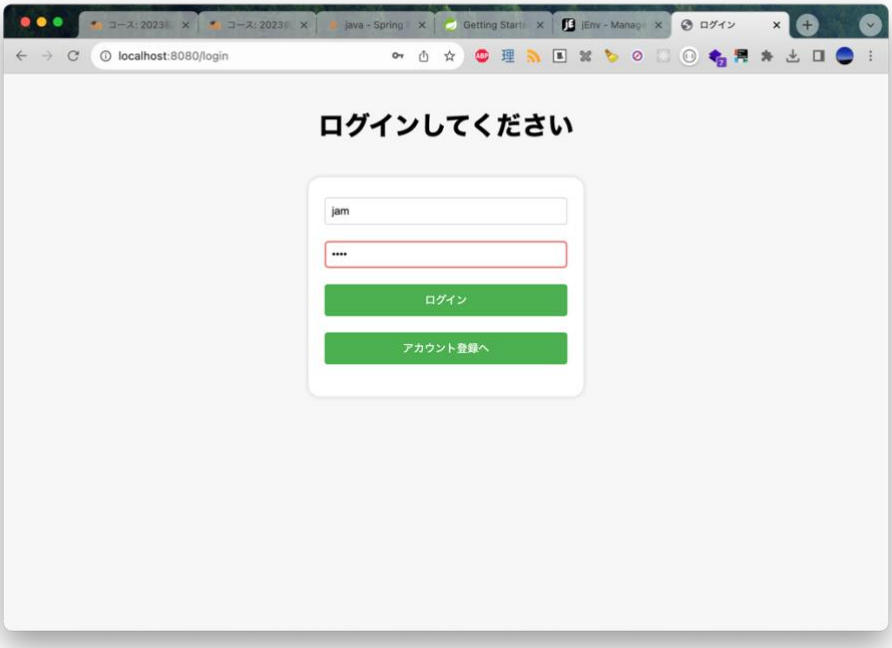

図 5.ログイン画面（登録後）

- 無事に認証ができたら（パスワードあるいはユーザ名が誤っていない場合）、フタたぶホーム画面に移り、実際のアプリの機能が表示される。

図 6.認証後のホーム画面：検索機能

- 食品のカテゴリーのボタンに押し、もしくは食品名を入力し検索ボタンを押したら可視化されたデータの画面に映る。

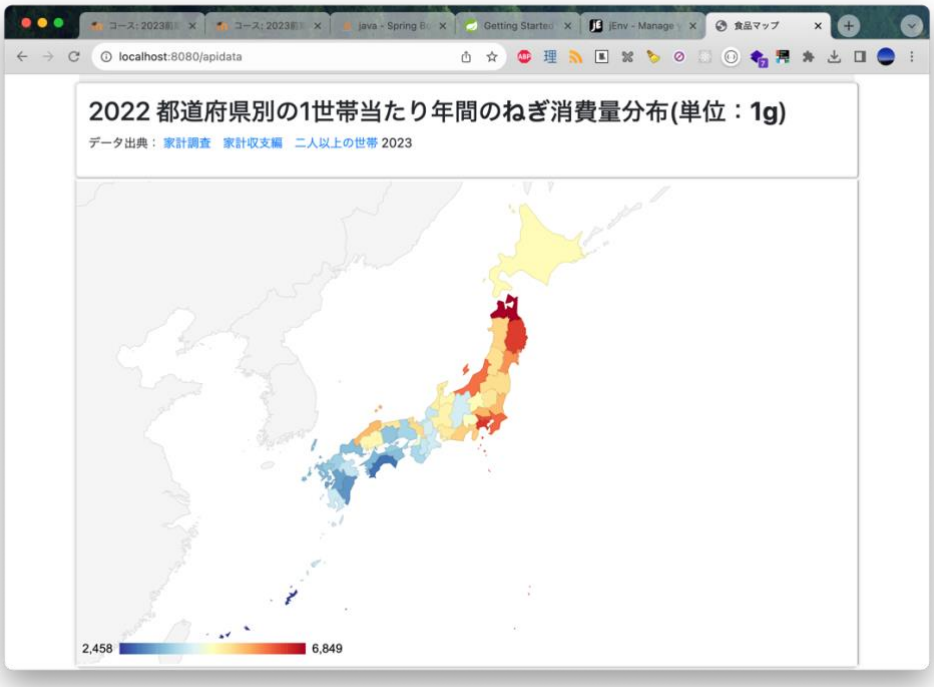

図 7.1.データ表示画面

- この地図にマウスを持っていくと、選択された食品の県別の食料が確認できる。または、ヒートマップのような色付けで消費量が多い県と少ない県が簡単に見分けることが可能である。

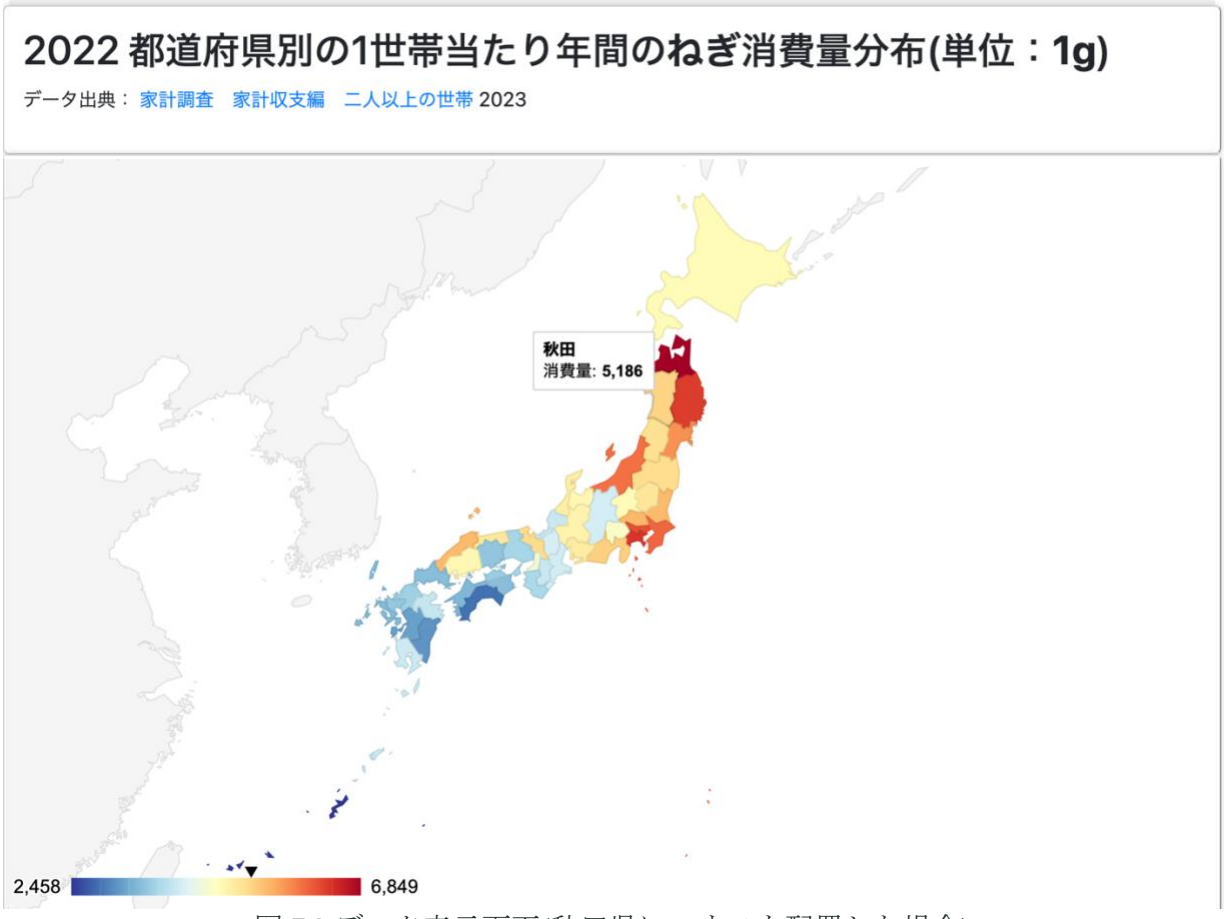

図 7.2.データ表示画面(秋田県にマウスを配置した場合)

このように、日本全国のデータを比べることができる。または、その食品の単位もデータと共に取得・表示しているため、その総量を 365 日で割ったら 1 日分の消費量も計算できる。

- 次は、食品の入力が誤った例を表示する。ここでは前述の通り、すでに取得した対象の統計データのメタ情報からカテゴリー名を抽出し、入力の正当性を判断する。

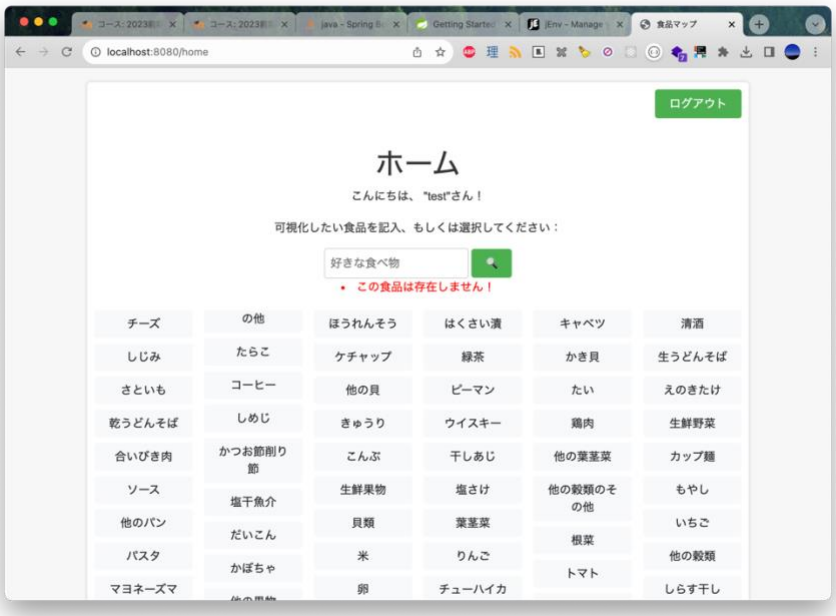

図 8. 入力が誤って例

ユーザがどのような食品を可視化できるかどうかを見やすくするために、それぞれのカテゴリーをボタンとして表示する。

図 9. ホーム画面（詳細のカテゴリーの表示）

(3)UML の作成と利用について

本課題では、MVC パターンを利用して簡単 Web アプリの作成ができた。その Web ページの機能の繋がりやデータの取得・処理を行う部分の計画を立てるために付属の UML 図を利用した。UML のクラス図によってクラスの分け方やクラス間の関係を計画的に考えることができた。例えば、本アプリではユーザデータの書き込み、読み込みを行うために、DTO・DAO パターンを用いる。これは、データ転送オブジェクト・データアクセスオブジェクトをクラス化した、認証コントローラやアカウント登録コントローラで利用する（実際に、ユーザマネジャークラスを介して利用している）。また、API とのやりとりを行うためは、3 つのクラスを利用する：

- ApiManager:
  API を呼び出すためのクラスになっており、実際の API の URL やパラメータを利用するクラスである。ここでは、パラメータの指定（特定のカテゴリーの情報を要求するため）、URL の作成などを行うメソッドを鄭義している。このクラスのよって、メタ情報を返す API クラス、または実際の統計データを返す API クラスをインスタンスとして定義することができる。
- ApiController
  これはデータの表示を制御するクラスである。このクラスは ApiManager のインスタンスを利用して、データを JSON 型として取得し、ApiDataProcessor にデータの抽出やクラスかを託す。ApiDataProcessor から変換されたデータをウェブアプリの“/apidata”エンドポイントへ送信する。
- ApiDataProcessor
  Api から取得したデータを処理するクラスである。実際に e-stat.go の API は都道府県のデータではなく、市ごとのデータを提供している。したがって、各市の該当データを対照する市に変換しなければならない。しかも、同じ県の複数の市の場合もあるため、それにも対応しなければならない。したがって、このデータを変換するメソッドを導入した。

追加情報：

- このプロジェクトは Java 20 を利用する。

```
~#@〉java --version
openjdk 20.0.2 2023-07-18
OpenJDK Runtime Environment Zulu20.32+11-CA (build 20.0.2+9)
OpenJDK 64-Bit Server VM Zulu20.32+11-CA (build 20.0.2+9, mixed mode, sharing)
```

- または、ビルドツールとして Gradle を利用している。このプロジェクトを実行するために、プロジェクトルートから以下のような Gradle コマンドを利用することができる：./gradlew bootRun

```
~#@〉./gradlew bootRun
Starting a Gradle Daemon, 2 incompatible and 1 stopped Daemons could not be reused, use --status for details

> Task :bootRun
Standard Commons Logging discovery in action with spring-jcl: please remove commons-logging.jar from classpath in order to avoid potential conflicts
Standard Commons Logging discovery in action with spring-jcl: please remove commons-logging.jar from classpath in order to avoid potential conflicts

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.1.2)

2023-08-08T15:49:16.965+09:00  INFO 92588 --- [  restartedMain] com.example.demo.DemoApplication         : Starting DemoApplication using Java 20.0.2
with PID 92588 (/Users/jam/Documents/大学/ソフトウェア開発演習２/demo/build/classes/java/main started by jam in /Users/jam/Documents/大学/ソフトウェア
開発演習２/demo)
2023-08-08T15:49:16.966+09:00  INFO 92588 --- [  restartedMain] com.example.demo.DemoApplication         : No active profile set, falling back to 1 de
fault profile: "default"
2023-08-08T15:49:16.998+09:00  INFO 92588 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spr
ing.devtools.add-properties' to 'false' to disable
2023-08-08T15:49:16.998+09:00  INFO 92588 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider
 setting the 'logging.level.web' property to 'DEBUG'
2023-08-08T15:49:17.595+09:00  INFO 92588 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http
)
2023-08-08T15:49:17.602+09:00  INFO 92588 --- [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2023-08-08T15:49:17.602+09:00  INFO 92588 --- [  restartedMain] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.
1.11]
```

- 上の画面のように、localhost:8080（ポート 8080)にウェブアプリが提供される。