MODUL 1 SPRING FRAMEWORK

1.1 Deskripsi Singkat

Spring adalah kerangka kerja (*framework*) yang luas dan populer dalam pengembangan aplikasi Java. Dikembangkan dengan tujuan mempermudah proses pembuatan dan pengelolaan aplikasi, Spring menyediakan solusi yang kuat dan fleksibel untuk berbagai kebutuhan pengembangan perangkat lunak.

Spring menawarkan pendekatan pemrograman yang terfokus pada objek *(Object-Oriented Programming)* dengan menggunakan *Inversion of Control (IoC)*. Ini memungkinkan pengembang untuk lebih fokus pada logika bisnis aplikasi dan mengurangi ketergantungan kuat antara komponen-komponen aplikasi, sehingga memudahkan pengujian dan perubahan.

Selain itu, Spring menyediakan modul-modul yang dapat diintegrasikan dengan mudah, termasuk Spring Boot untuk memulai proyek dengan cepat, Spring Data untuk interaksi dengan database secara efisien, dan Spring Security untuk mengamankan aplikasi dengan autentikasi dan otorisasi. Spring juga mendukung berbagai protokol komunikasi, termasuk RESTful API dan SOAP, serta menyediakan dukungan untuk aspek-aspek seperti keamanan, manajemen transaksi, dan keamanan.

Dalam pengembangan web, Spring MVC menyediakan alat yang kuat untuk membangun aplikasi web berbasis *model-view-controller (MVC)*, memisahkan logika bisnis, tampilan, dan kontrol dalam komponen terpisah. Selain itu, Spring menyediakan fitur-fitur seperti *Dependency Injection, Aspek-Oriented Programming*, dan integrasi dengan berbagai teknologi seperti *Hibernate, JPA, dan Thymeleaf*.

Secara keseluruhan, Spring adalah ekosistem lengkap yang dapat mempercepat proses pengembangan aplikasi Java, membantu pengembang dalam menghasilkan kode yang lebih bersih, mudah diuji, dan dapat diandalkan. Dengan berbagai modul dan dukungan yang

disediakan, Spring terus menjadi pilihan populer dalam dunia pengembangan perangkat lunak.

1.2 Tujuan Praktikum

Tujuan praktikum ini adalah mahasiswa membuat aplikasi web sederhana menggunakan Spring MVC dan Thymeleaf.

Setelah praktikum ini kompetensi yang akan dicapai adalah mahasiswa mampu memahami penggunaan *Spring Framework* dalam pengembangan aplikasi Java, memahami alur pengembangan aplikasi web dengan Spring, memahami teknologi atau *library-library* yang berkaitan dengan Spring.

1.3 Alokasi Waktu

Alokasi waktu pada praktikum 1 adalah sebagai berikut :

1. Tatap muka di kelas : 50 menit
2. Kerja mandiri (mengerjakan penugasan) : 120 menit

1.4 Material Praktikum

Praktikum ini memerlukan beberapa material sebagai berikut:

1) Java Development Kit (JDK) versi 17
2) Java IDE: Netbeans, IntelliJ IDEA, atau Spring Tool Suite (STS)
3) Maven 3.5+

1.5 Kegiatan Praktikum

Kegiatan praktikum pada modul 1 dimulai dari menyiapkan lingkungan pengembangan, membuat proyek Spring Boot, membuka proyek Spring, menjalankan proyek Spring, menulis kode program, menjalankan program, dan membuat operasi CRUD.

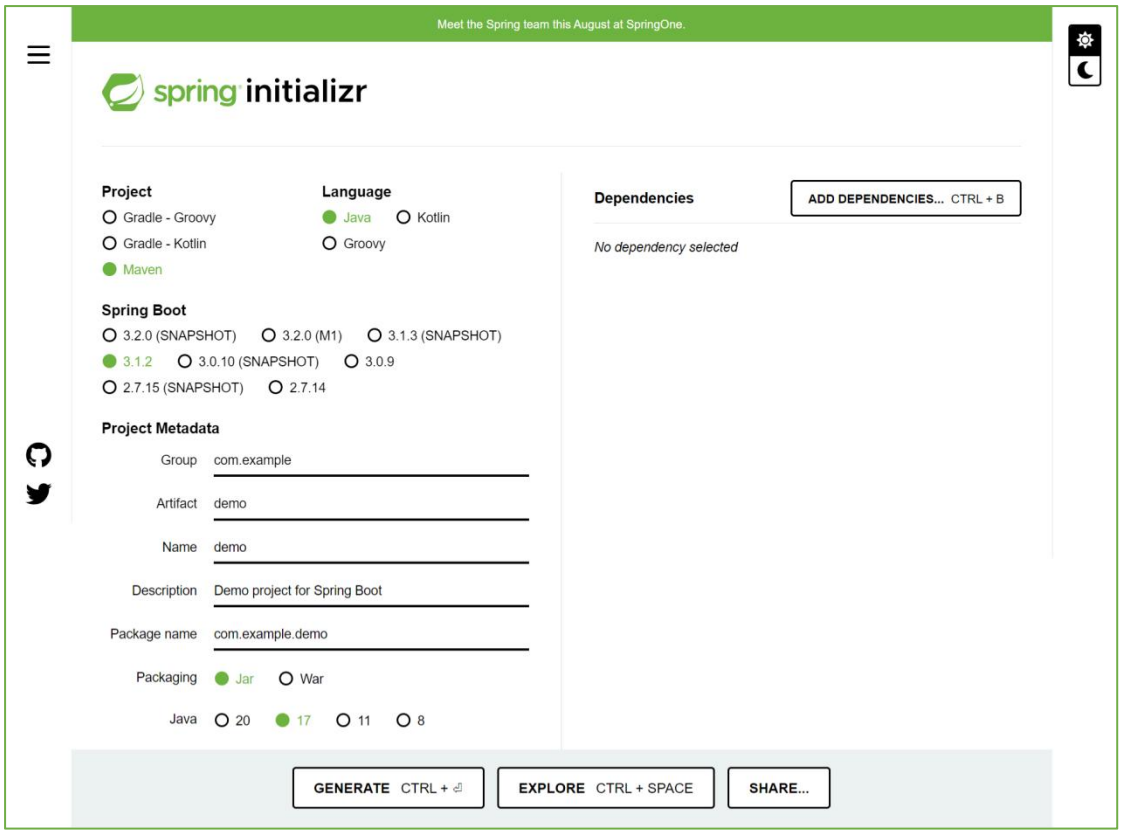
A. Menyiapkan Lingkungan Pengembangan

Sebelum memulai membuat proyek Spring, siapkan lingkungan pengembangan sebagai berikut.

1. Pastikan *Java Development Kit (JDK)* sudah terpasang di komputer.
2. Pastikan *Integrated Developer Environment (IDE)* sudah terpasang di komputer dan dapat digunakan dengan baik. Pada praktikum ini digunakan NetBeans IDE.

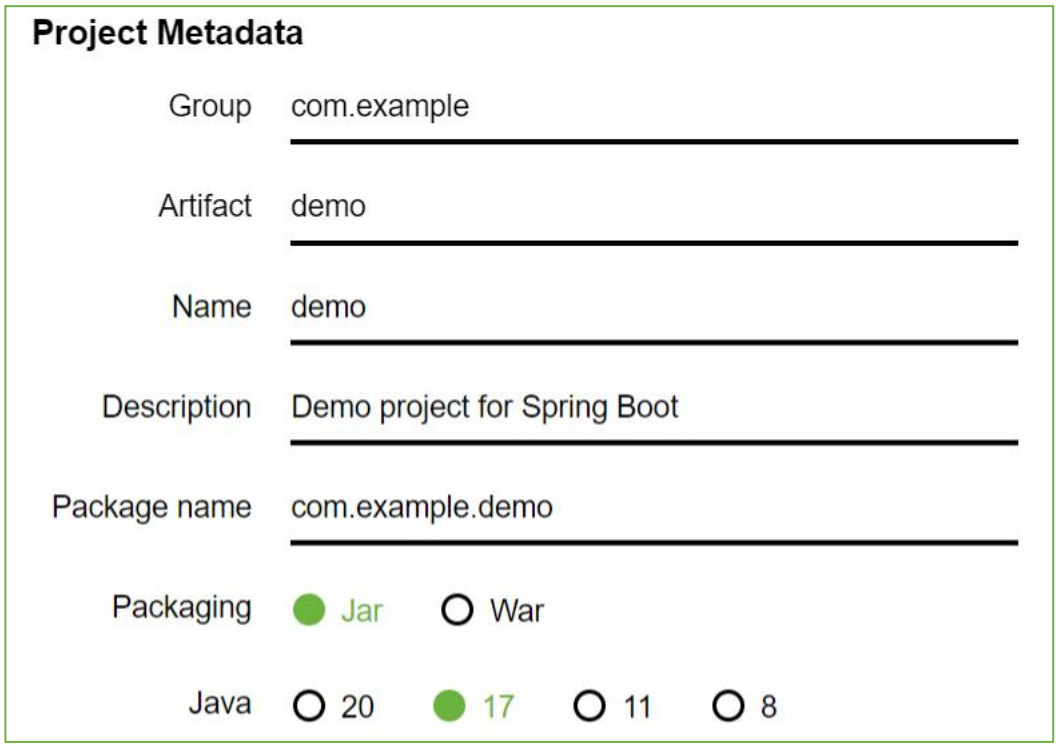
B. Membuat Proyek Spring Boot

Untuk memudahkan membuat proyek Spring, gunakan *Spring Initializr* yang dapat diakses pada https://start.spring.io/.

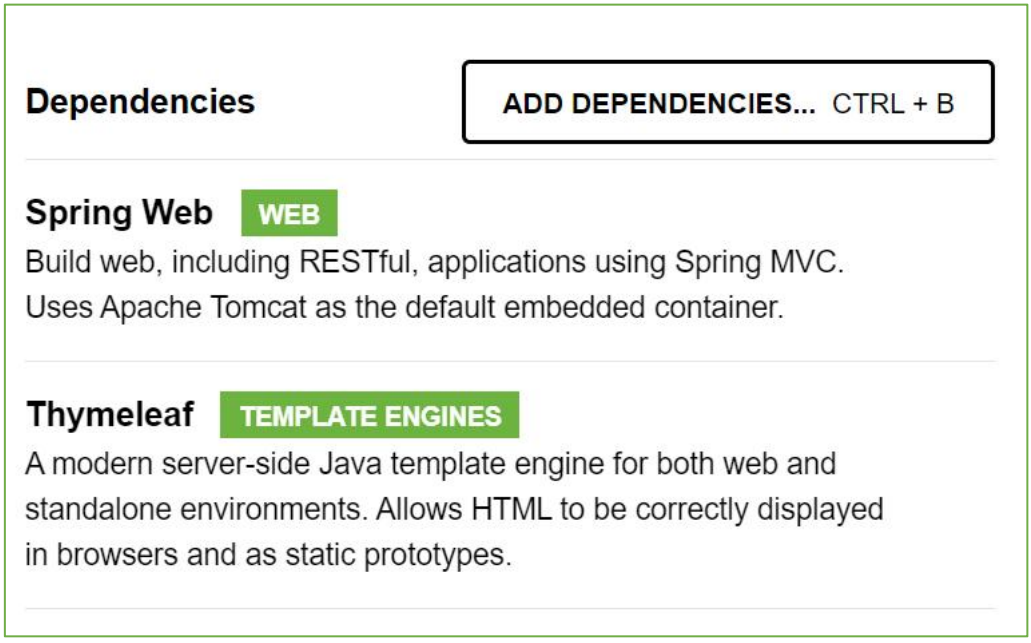


Spring Initializr akan membantu dalam menyiapkan konfigurasi awal proyek Spring. Ikuti langkah-langkah berikut ini.

1. Pilih Project **Maven** dan Language **Java**.
2. Pilih versi Spring Boot **3.1.2**.
3. Isikan Project Metadata.



4. Tambahkan Dependencies yang diperlukan, antara lain: Spring Web dan Thymeleaf. Klik tombol ADD DEPENDENCIE. Selanjutnya cari dan pilih Depedencies tersebut.

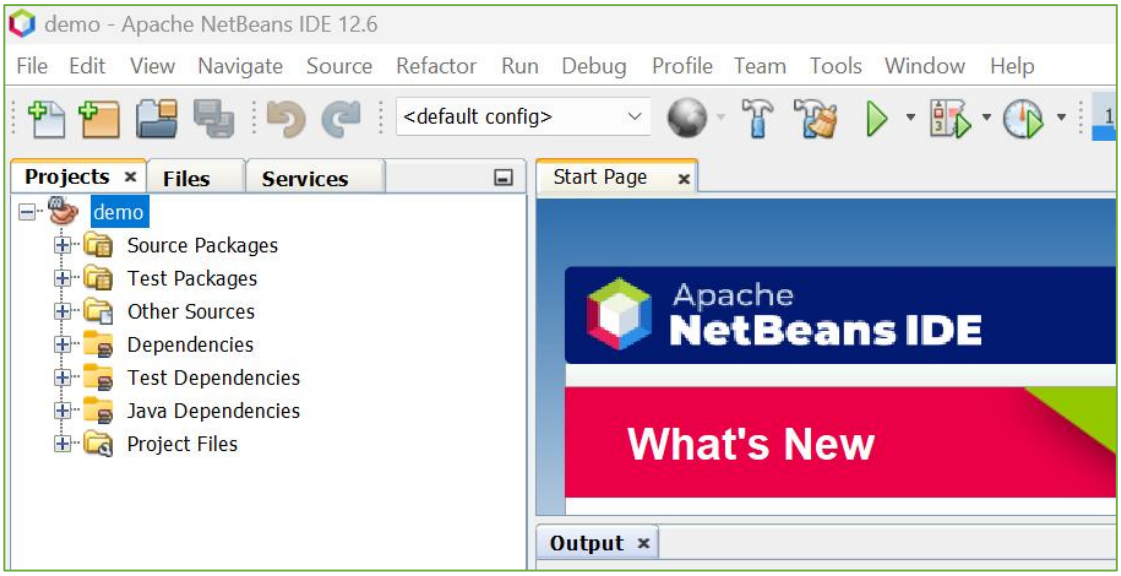


5. Terakhir, klik tombol GENERATE untuk mengunduh proyek Spring.

C. Membuka Proyek Spring

Proyek Spring yang telah di-*generate* dari *Spring Initializr* dapat dibuka melalui NetBeans IDE. Ikuti langkah berikut ini.

1. Ekstrak file proyek Spring yang telah diunduh.
2. Buka NetBeans IDEA. Setelah window tampil, klik tombol *Open Project* atau menu *File>Open Project*.
3. Telusuri folder proyek Spring yang telah diekstrak. Pilih folder proyek dan klik *Open Project*.
4. Proyek akan ditampilkan pada window NetBeans IDE.



5. Selanjutnya klik kanan project demo. Pilih menu *Build with Dependencies*. Proyek akan mengunduh dependencies Spring MVC dan Thymeleaf yang telah didefinisikan sebelumnya serta dependencies terkait lainnya.

D. Menulis Kode Program

Melalui Intellij IDEA, buka file DemoApplication.java di dalam folder proyek /src/main/java/com/example/demo. Ubah kode program dengan menambahkan satu method seperti berikut.

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class DemoApplication {
   public static void main(String[] args) {
    SpringApplication.run(DemoApplication.class, args);
      }

   @GetMapping("/hello")
   public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
    return String.format("Hello %s!", name);
   }
}
```

Kode di atas adalah contoh sederhana dari aplikasi Spring Boot dengan menggunakan Spring MVC.

a) *package com.example.demo;:* Baris ini menunjukkan bahwa kelas-kelas dalam file ini berada dalam paket (package) bernama com.example.demo.

b) *import statements:* Baris-baris import digunakan untuk mengimpor kelas-kelas yang dibutuhkan dari pustaka Spring Framework.

c) *@SpringBootApplication:* Anotasi ini ditempatkan di kelas utama aplikasi dan menggabungkan beberapa anotasi seperti @Configuration, @EnableAutoConfiguration, dan @ComponentScan. Ini adalah konfigurasi utama untuk aplikasi Spring Boot.

d) *@RestController:* Anotasi ini digunakan untuk memberi tahu Spring bahwa kelas ini adalah komponen RESTful dan akan meng-handle permintaan HTTP.

e) *public static void main(String[] args) { ... }:* Metode main ini adalah titik masuk utama aplikasi. Ini akan memulai aplikasi Spring Boot.

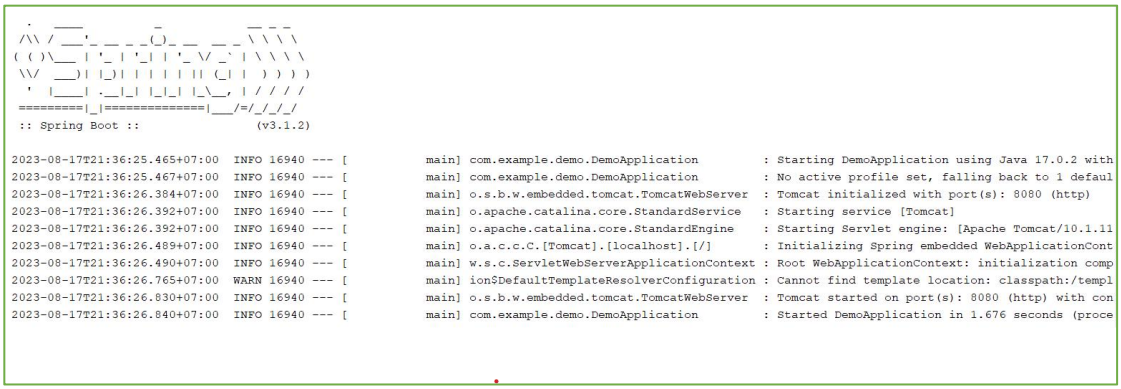
f) *@GetMapping("/hello"):* Anotasi ini mendefinisikan bahwa metode hello akan merespons permintaan HTTP dengan metode GET pada jalur (path) "/hello".

g) *public String hello(@RequestParam(value = "name", defaultValue = "World") String name) { ... }:* Ini adalah metode yang akan menangani permintaan HTTP GET /hello. Parameternya adalah name yang dapat diberikan sebagai parameter query dalam URL. Nilai default untuk name adalah "World". Metode ini mengembalikan pesan "Hello [nama]!".

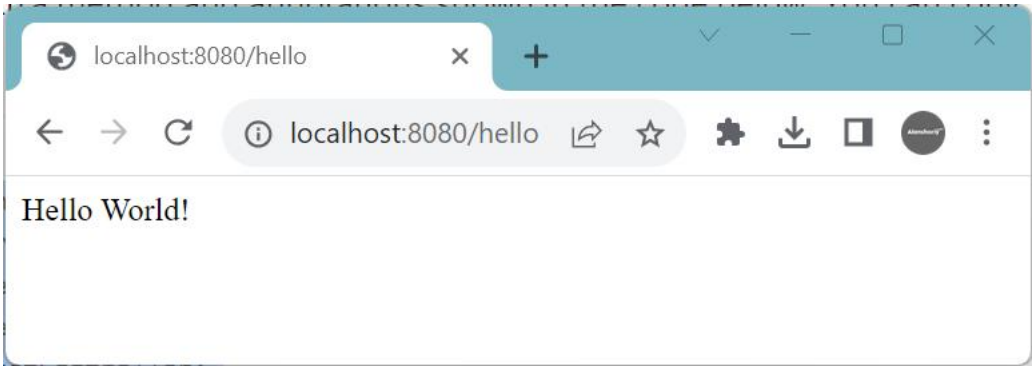
Dengan kode di atas, Anda memiliki aplikasi Spring Boot yang dapat diakses melalui URL seperti http://localhost:8080/hello dan mengembalikan pesan "Hello World!" atau "Hello [nama]!" tergantung pada nilai name yang diberikan dalam parameter query.

E. Mejalankan Proyek Spring

Untuk menjalankan proyek demo, klik kanan pada file DemoApplication.java. Pilih menu *Run File*. Tunggu sampai proses persiapan eksekusi program selesai.



Setelah eksekusi program selesai, buka *browser* untuk mengakses halaman web dengan alamat http://localhost:8080/hello. Halaman *browser* akan menampilkan seperti gambar berikut.



F. Membuat Operasi CRUD

Selanjutnya modifikasi file DemoApplication.java untuk menambahkan operasi *CRUD (Create, Read, Update, Delete)* sederhana. Operasi CRUD ini hanya memodifikasi nilai variabel items. Modifikasi file DemoApplication.java sebagai berikut.

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.*;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;

import java.util.ArrayList;
import java.util.List;

@SpringBootApplication
@Controller
public class DemoApplication {

   private List<String> items = new ArrayList<>();

   public static void main(String[] args) {
      SpringApplication.run(DemoApplication.class, args);
   }

   @GetMapping("/")
   public String index(Model model) {
      model.addAttribute("items", items);
      return "index";
   }

   @PostMapping("/add")
   public String addItem(@RequestParam String item) {
      items.add(item);
      return "redirect:/";
   }

   @GetMapping("/edit/{index}")
   public String editItem(@PathVariable int index, Model model) {
      String item = items.get(index);
      model.addAttribute("index", index);
      model.addAttribute("item", item);
      return "edit";
   }

   @PostMapping("/update/{index}")
```

```
    public String updateItem(@PathVariable int index, @RequestParam String item) {
        items.set(index, item);
        return "redirect:/";
    }

    @GetMapping("/delete/{index}")
    public String deleteItem(@PathVariable int index) {
        items.remove(index);
        return "redirect:/";
    }
}
```

Selain kode di atas, Anda juga perlu menambahkan file HTML di direktori resources/templates untuk mendukung tampilan aplikasi. Berikut adalah contoh file index.html dan edit.html:

src/main/resources/templates/index.html:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>CRUD Application</title>
</head>
<body>
    <h1>CRUD Application</h1>

    <form action="/add" method="post">
        <input type="text" name="item" placeholder="Enter item" required>
        <button type="submit">Add</button>
    </form>

    <ul>
        <li th:each="item, index : ${items}">
            <span th:text="${item}"></span>
            <a th:href="@{/edit/{index}(index=${index.index})}">Edit</a>
            <a th:href="@{/delete/{index}(index=${index.index})}">Delete</a>
        </li>
    </ul>
</body>
</html>
```

src/main/resources/templates/edit.html:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
   <title>Edit Item</title>
</head>
<body>
   <h1>Edit Item</h1>

   <form th:action="@{/update/{index}(index=${index})}" method="post">
      <input type="text" name="item" th:value="${item}" required>
      <button type="submit">Update</button>
   </form>

   <a th:href="@{/}">Back to List</a>
</body>
</html>
```

Berikut penjelasan kode program DemoApplication.java.

a) *Package dan Imports:* Baris pertama hingga baris ke-6 adalah deklarasi package dan import statements yang diperlukan untuk mengimpor kelas-kelas yang diperlukan dari pustaka Spring Framework.

b) Anotasi *@SpringBootApplication:* Ini adalah anotasi dari Spring Boot yang menggabungkan beberapa anotasi lain menjadi satu, yaitu @Configuration, @EnableAutoConfiguration, dan @ComponentScan. Anotasi ini digunakan di kelas utama aplikasi untuk mengkonfigurasi aplikasi Spring Boot.

c) Anotasi *@Controller:* Ini adalah anotasi dari Spring Framework yang menandakan bahwa kelas ini adalah komponen yang mengendalikan permintaan HTTP. Dalam konteks ini, kelas DemoApplication bertindak sebagai controller.

d) Deklarasi List Items: Baris ke-12 mendeklarasikan sebuah List bernama items, yang akan digunakan untuk menyimpan item dalam aplikasi CRUD.

e) Metode main: Ini adalah metode utama yang menjalankan aplikasi Spring Boot. Ketika aplikasi dijalankan, metode ini akan dipanggil, dan SpringApplication.run(DemoApplication.class, args) akan memulai aplikasi Spring Boot.

f) Metode index: Ini adalah metode yang akan menangani permintaan HTTP GET /. Anotasi @GetMapping("/") mengaitkan metode ini dengan URL root. Metode ini menerima objek Model yang digunakan untuk memberikan data ke tampilan. Dalam contoh ini, daftar items disimpan dalam model dan akan digunakan dalam tampilan index.html.

g) Metode addItem: Ini adalah metode yang akan menangani permintaan HTTP POST /add. Anotasi @PostMapping("/add") mengaitkan metode ini dengan URL /add yang menerima permintaan POST. Metode ini menerima parameter item dari form input HTML. Item ditambahkan ke dalam daftar items dan kemudian pengguna akan diarahkan kembali ke halaman utama menggunakan redirect:/.

h) Metode editItem: Ini adalah metode yang akan menangani permintaan HTTP GET /edit/{index}. Anotasi @GetMapping("/edit/{index}") mengaitkan metode ini dengan URL /edit/{index} yang menerima permintaan GET dengan variabel index sebagai path parameter. Metode ini mengambil item dari items berdasarkan indeks yang diberikan dan menyiapkan data untuk tampilan edit.html.

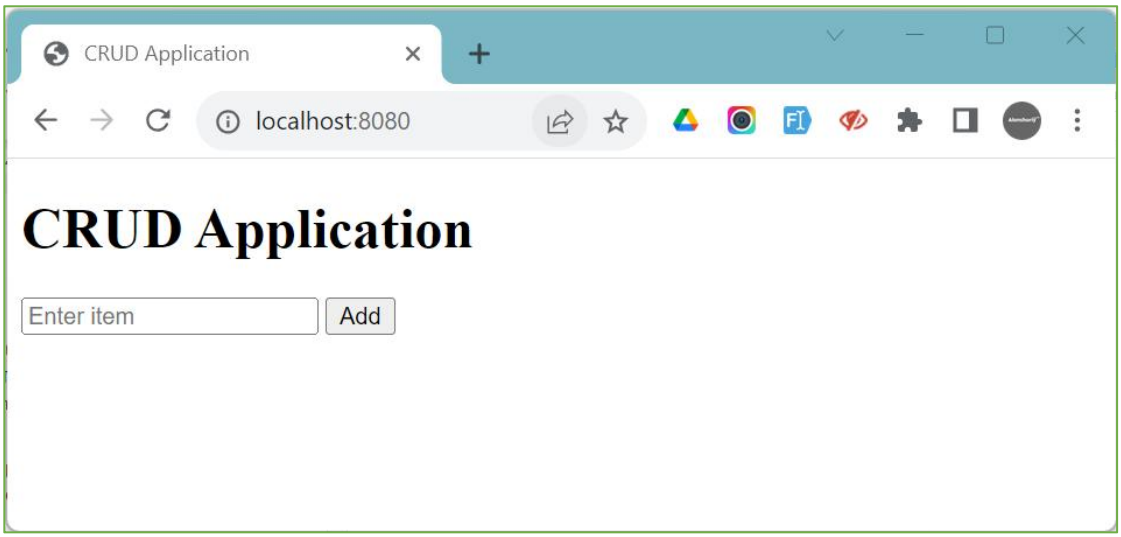
i) Metode updateItem: Ini adalah metode yang akan menangani permintaan HTTP POST /update/{index}. Anotasi @PostMapping("/update/{index}") mengaitkan metode ini dengan URL /update/{index} yang menerima permintaan POST dengan variabel index sebagai path parameter. Metode ini menerima parameter item dari form input HTML dan mengganti item yang ada pada indeks yang diberikan dalam daftar items.

j) Metode deleteItem: Ini adalah metode yang akan menangani permintaan HTTP GET /delete/{index}. Anotasi @GetMapping("/delete/{index}") mengaitkan metode ini dengan URL /delete/{index} yang menerima permintaan GET dengan variabel index

sebagai path parameter. Metode ini menghapus item dari daftar items berdasarkan indeks yang diberikan.

Secara keseluruhan, kode program DemoApplication.java ini merupakan inti dari aplikasi CRUD sederhana menggunakan Spring MVC yang mengatur pengolahan permintaan HTTP untuk menambahkan, mengedit, dan menghapus item dari daftar items. Tampilan HTML dikelola menggunakan *template engine Thymeleaf*.

Jalankan proyek aplikasi CRUD. Setelah aplikasi berjalan, Anda dapat mengaksesnya menggunakan browser web. Buka browser dan kunjungi URL berikut: http://localhost:8080/.



Anda akan diarahkan ke halaman utama aplikasi CRUD. Pada halaman ini, Anda akan melihat daftar item yang telah Anda tambahkan, dan Anda dapat melakukan operasi CRUD seperti berikut:

Tambahkan Item:

- Masukkan teks item yang ingin Anda tambahkan dalam kotak input.
- Klik tombol "Add".
- Item baru akan ditambahkan ke daftar dan ditampilkan di halaman.

Edit Item:

- Klik tautan "Edit" di samping item yang ingin Anda edit.

- Anda akan diarahkan ke halaman edit, di mana Anda dapat mengedit teks item.
- Setelah selesai, klik tombol "Update" untuk menyimpan perubahan.

Hapus Item:

- Klik tautan "Delete" di samping item yang ingin Anda hapus.
- Item akan dihapus dari daftar.

1.6 Penugasan

Buat aplikasi CRUD menggunakan Spring MVC, Thymeleaf, MySQL, Spring JPA, dan Hibernate dengan petunjuk sebagai berikut.

1. Create (Tambah Data)
- Buatlah entitas Mahasiswa dengan atribut NIM, Nama, Jurusan, dan Tanggal Lahir.
- Implementasikan sebuah form HTML untuk memasukkan data mahasiswa baru.
- Sesuaikan kode controller untuk menyimpan data mahasiswa baru ke dalam database.

2. Read (Tampilan Data)
- Tampilkan daftar semua mahasiswa yang ada dalam tabel.
- Buat halaman HTML yang menampilkan data mahasiswa dalam tabel.
- Sesuaikan kode controller untuk mengambil data mahasiswa dari database dan meneruskannya ke tampilan.

3. Update (Edit Data)
- Implementasikan fitur untuk mengedit data mahasiswa berdasarkan NIM.
- Buatlah form HTML untuk mengedit atribut Nama, Jurusan, dan Tanggal Lahir dari mahasiswa.
- Sesuaikan kode controller untuk mengambil data mahasiswa berdasarkan NIM, menampilkan dalam form edit, dan menyimpan perubahan ke database.

4. Delete (Hapus Data)

- Tambahkan fungsi untuk menghapus data mahasiswa berdasarkan NIM.
- Sesuaikan kode controller untuk menghapus data mahasiswa dari database.

5. Pencarian Data
- Tambahkan fitur pencarian mahasiswa berdasarkan NIM atau Nama.
- Sesuaikan kode controller untuk mengambil data mahasiswa yang sesuai dengan kriteria pencarian.

6. Validasi Data
- Terapkan validasi pada form input untuk memastikan data yang dimasukkan sesuai format yang diharapkan.
- Tampilkan pesan error yang relevan jika validasi gagal.

7. Exception Handling
- Sesuaikan kode program untuk menangani exception yang mungkin terjadi saat berinteraksi dengan database atau melakukan operasi CRUD.
- Tampilkan pesan error yang informatif kepada pengguna.

8. Tampilan Detail Mahasiswa
- Buat halaman detail untuk menampilkan informasi lengkap tentang seorang mahasiswa berdasarkan NIM.
- Sesuaikan kode controller untuk mengambil data mahasiswa berdasarkan NIM dan mengirimnya ke tampilan detail.

9. Menghubungkan dengan Halaman Utama
- Tambahkan tautan di halaman utama untuk mengakses semua fitur CRUD yang telah diimplementasikan.

Buat laporan pekerjaan Anda dengan penjelasan selengkap mungkin dalam format pdf.