

Java Spring Boot Workshop



Spring Boot

Xsis Mitra Utama



Oleh : Masyda Arrizaqu

Table of Contents

Pengenalan	1.1
Kebutuhan System	1.2
Buat Project	1.3
Hello World	1.4
Quiz Logika	1.5
Jawaban Quiz	1.6
Spring MVC	1.7
CRUD Data Form	1.8

Xsis Mitra Utama

LAB -Java SpringBoot

Berikut ini adalah Lab Praktek untuk pembuatan web aplikasi sederhana menggunakan Spring boot.

Panduan ini memberikan contoh bagaimana Spring Boot membantu dalam mempercepat dan memfasilitasi pengembangan aplikasi sehingga dapat membercepat dan memudahkan dalam mengembangkan aplikasi dengan bahasa pemrograman java spring framework.

Xsis Mitra Utama

Kebutuhan System

Dalam implementasinya, untuk workshop kali ini akan sangat dianjurkan untuk masing masing peserta memenuhi syarat untuk kemudian meng-Install di setiap komputer sebagai berikut :

IDE / Editor

Spring Tool Suit (STS) Latest Version

Database Server

MySQL (simple instal dengan : XAMPP)

Operating System

Windows, Linux, (Rekomendasi : Windows)

JDK

JDK 1.8 atau diatasnya.

Aplikasi Pendukung

XAMPP

Membuat Web Project

Spring Boot Starter

Untuk pertama kali adalah tentunya membuat terlebih web project terlebih dahulu sebagai berikut.

1. kunjungi <https://start.spring.io/>
2. Pilih Dependency standart diantaranya adalah :
 - i. JPA
 - ii. MySQL
 - iii. Thymeleaf

The screenshot shows the Spring Initializr interface. It's a form-based configuration tool for generating Spring Boot projects. The configuration fields include:

- Project**: Maven Project (selected), Gradle Project
- Language**: Java (selected), Kotlin, Groovy
- Spring Boot**: 2.2.0 M1, 2.2.0 (SNAPSHOT), 2.1.5 (SNAPSHOT), 2.1.4 (selected), 1.5.20
- Project Metadata**: Group (com.uinjakarta), Artifact (smartweb)
- Dependencies**: See all, Search dependencies to add (Web, Security, JPA, Actuator, Envtools...), Selected dependencies (JPA [SQL]): Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate
- Generate Project - alt + d** button

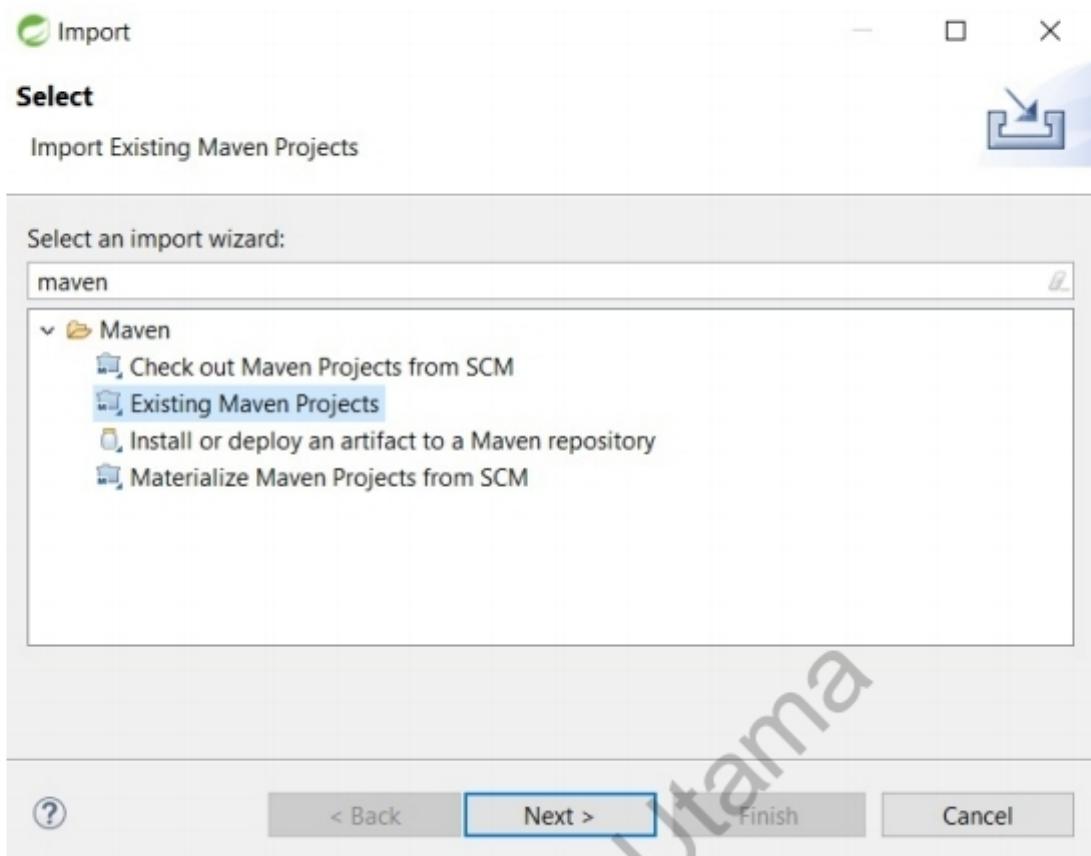
3. Setelah itu

seperti terlihat button dibawah form dan **enter**

Import File Project

Setelah melakukan generate Project dengan spring boot strater maka, selanjutnya adalah :

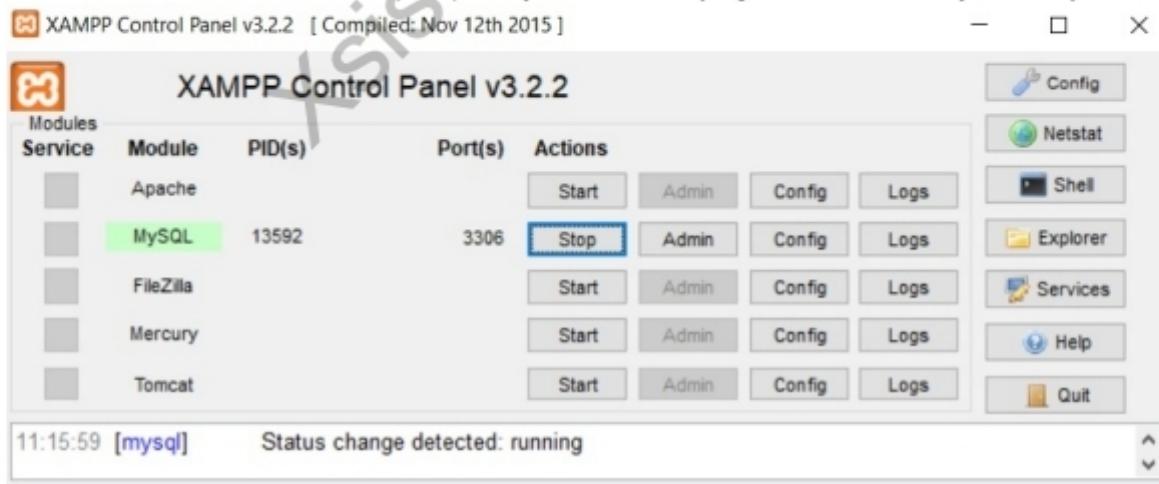
1. Extract file rar Generate Project ke dalam **project workspace** anda.
2. Buka dan import file project yang telah di extract menggunakan IDE Eclipse / STS, yaitu : **File -> Import** dan ketikan wizard "Maven" dan pilih Existing Maven Project, dan terakhir adalah cari file project yang telah di extract sebelumnya.



Koneksi Database

Dengan hadirnya dependency JPA, maka ketika project dijalankan maka biasanya akan error karena aplikasi harus memiliki koneksi dengan database, untuk kali ini menggunakan mysql, sebagai berikut:

1. Jalankan service database, untuk mempercepatkan jalankan XAMPP yang sudah terinstall, dan jalankan MySQL.



2. Create database dengan nama "sb-uin-jakarta"

```
C:\Users\arrizaqu>cd c:/xampp/mysql/bin
c:/xampp/mysql/bin>mysql -u root -p
Enter password:
mysql> create database sb_uin_jakarta;
```

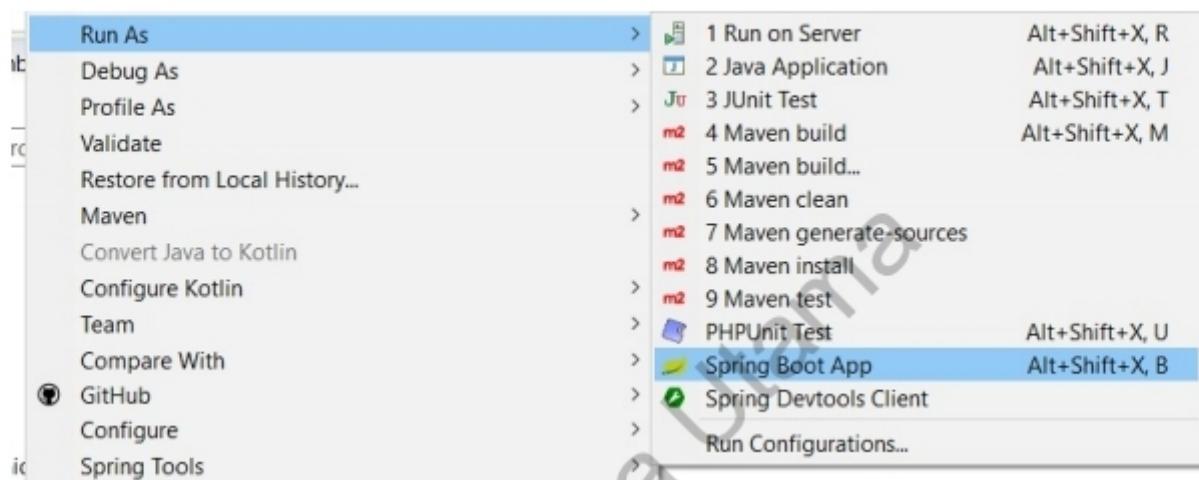
3. buka file **src -> main -> resources -> application.properties** dan tulis script sebagai berikut :

```
# koneksi Database MySQL
spring.jpa.hibernate.ddl-auto=create
spring.datasource.url=jdbc:mysql://localhost:3306/sb_uin_jakarta
spring.datasource.username=root
spring.datasource.password=
```

Menjalankan Program

Jalan program dengan banyak cara dalam hal ini secara cepat bisa dilakukan dengan menggunakan cara :

Klik kanan Project -> Pilih "RUN AS" -> pilih "Spring Boot App"

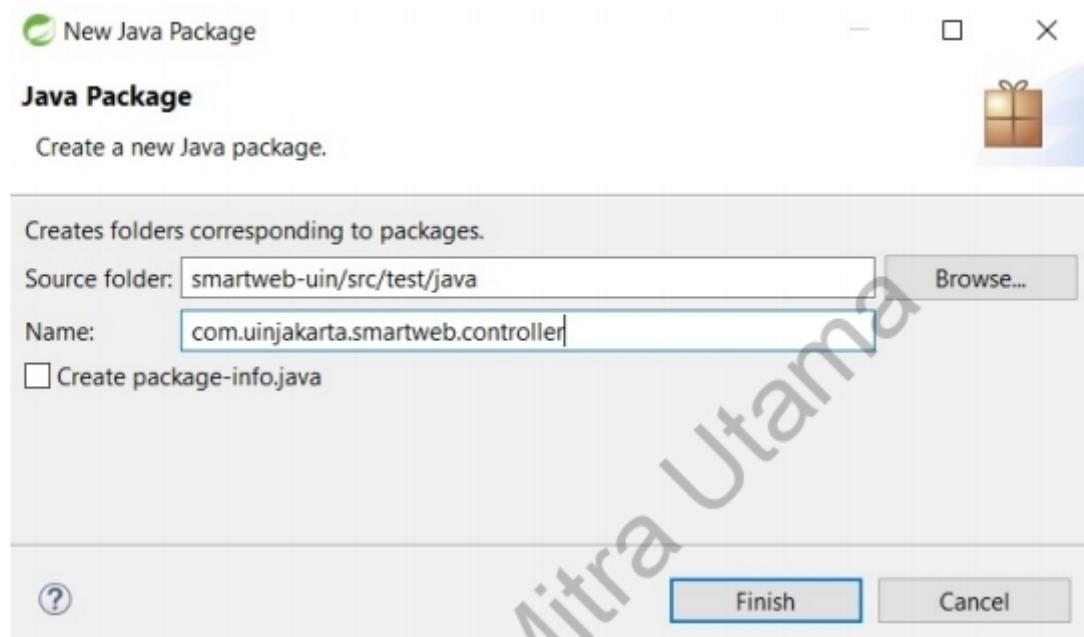


Hello World

Sudah menjadi hal yang umum ketika belajar suatu bahasa pemrograman adalah bagaimana membuat code yang sangat sederhana, salah satunya adalah "Hello World".

Controller

Membuat Package Controller



Membuat Controller Class

```
package com.uinjakarta.smartweb.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
@RequestMapping("/hello")
public class HelloWorld {

    @GetMapping
    @ResponseBody
    public String index() {
        return "hello world java springboot";
    }
}
```

Jalankan di Browser

- buka browser dan ketik url: <http://localhost:8080>
- output : "hello world java springboot"

+Sis Mitra Utama

Quiz Logika

Guidance Quiz

Menghasilkan bilangan Fibonacci dengan inputan N

- **Function Fibonacci**

```
public int[] getFibonacci(int n) {  
    int[] data = new int[n];  
    data[0] = 1;  
    data[1] = 1;  
    for(int i = 2; i < n; i++) {  
        data[i] = data[i - 1] + data[i - 2];  
    }  
  
    return data;  
}
```

- **Controller**

```
package com.uinjakarta.smartweb.controller;  
  
import java.io.ObjectInputStream.GetField;  
import java.util.Arrays;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.ResponseBody;  
  
@Controller  
@RequestMapping("/quiz")  
public class Quiz {  
  
    @GetMapping  
    @ResponseBody  
    public String index(@RequestParam("n") int n) {  
        int[] dataFib = this.getFibonacci(n);  
        String hasil = Arrays.toString(dataFib);  
        return hasil;  
    }  
  
    public int[] getFibonacci(int n) {  
        int[] data = new int[n];  
        data[0] = 1;  
        data[1] = 1;  
        for(int i = 2; i < n; i++) {  
            data[i] = data[i - 1] + data[i - 2];  
        }  
  
        return data;  
    }  
}
```

```
}
```

Quiz

Buatlah hasil output 2 digit menggunakan format Array dengan aturan sebagai berikut :

Digit 1 -> menuntukkan Panjang suatu Array yang dihasilkan dari Bilangan Fibonacci Quiz 1

Digit 2 -> menunjukkan Total Jumlah jika hasil Quiz 1 dijumlahkan.

```
Contoh : 1
* input : 3
* Fibonacci : [1, 1, 2]
* hasil : [3, 4]
```

```
Contoh : 2
* input : 5
* Fibonacci : [1, 1, 2, 3, 5]
* hasil : [5, 12]
```

Lengkapi penyelesain berikut ini :

```
package com.uinjakarta.smartweb.controller;

import java.io.ObjectInputStream.GetField;
import java.util.Arrays;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
@RequestMapping("/quiz")
public class Quiz {

    @GetMapping
    @ResponseBody
    public String index(@RequestParam("n") int n) {
        int[] dataFib = this.getFibonacci(n);
        int[] dataMax = this.getMaxValues(dataFib);
        String hasil = Arrays.toString(dataMax);

        return hasil;
    }

    public int[] getFibonacci(int n) {
        int[] data = new int[n];
        data[0] = 1;
        data[1] = 1;
        for(int i = 2; i < n; i++) {
            data[i] = data[i - 1] + data[i - 2];
        }

        return data;
    }
}
```

```
public int[] getMaxValues(int[] dataFib) {  
    //complete your code here !!  
}  
}
```

Xsis Mitra Utama

Spring MVC

Kerangka kerja Spring Web MVC menyediakan arsitektur Model-View-Controller (MVC) dan komponen-komponen siap yang dapat digunakan untuk mengembangkan aplikasi web yang fleksibel. Pola MVC menghasilkan pemisahan berbagai aspek aplikasi (logika input, logika bisnis, dan logika UI).

- Model merangkum data aplikasi dan secara umum akan terdiri dari POJO.
- View bertanggung jawab untuk memberikan data model dan secara umum menghasilkan output HTML yang dapat ditafsirkan oleh browser.
- Controller bertanggung jawab untuk memproses permintaan pengguna dan membangun model yang sesuai dan meneruskannya ke tampilan untuk rendering.

Simple CRUD (Create Read Update Delete)

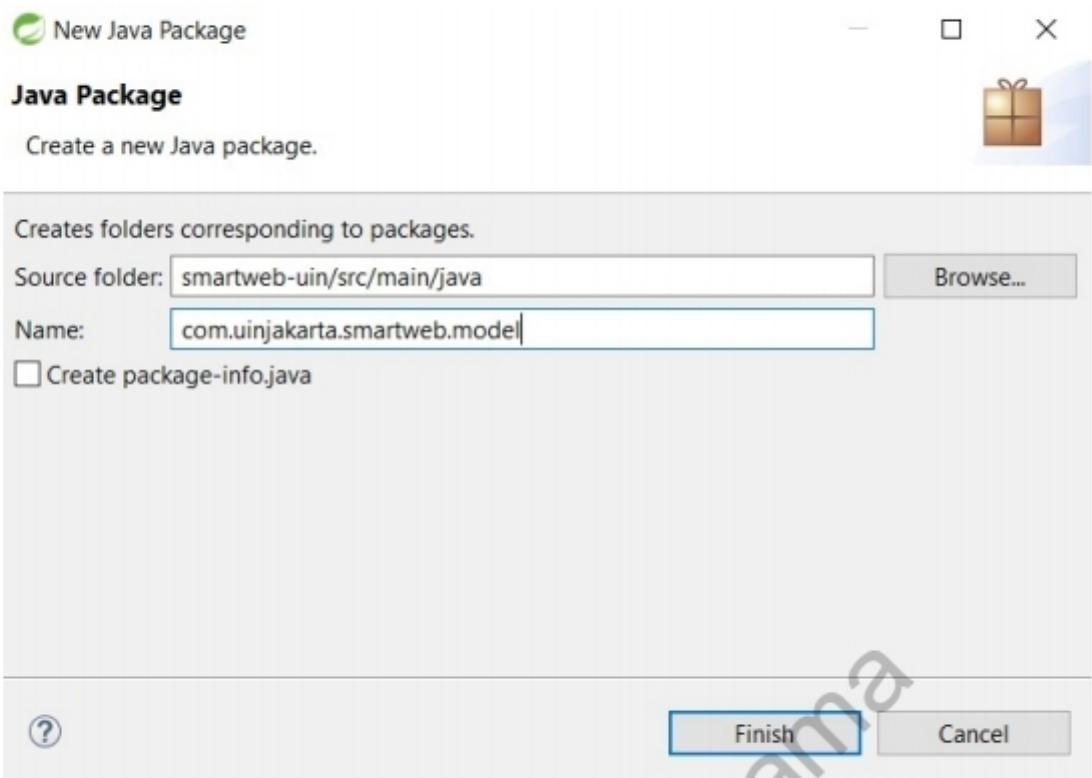
Table : Employee

Employee
• employeeld
• name
• address
• email

POJO sebagai Model

Berikut ini akan kita buatkan Model deskripsi table berbasis Object, dengan hadirnya JPA, hal ini sangat memungkin dimana sebelumnya biasa menggunakan SQL Native. dengan Model ini akan kita definisikan Object dan Properti sebagai bagian dari struktur data yang akan disimpan secara persistance di dalam database.

untuk itu perlu dilakukan pembuat package Model tersendiri sebagai berikut:



Membuat Class Employee

```
package com.uinjakarta.smartweb.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="EMPLOYEE")
public class Employee {

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name="employee_id")
    public int employeeId;
    public String name;
    @Column(name="address")
    public String address;
    public String email;
    public int getEmployeeId() {
        return employeeId;
    }
    public void setEmployeeId(int employeeId) {
        this.employeeId = employeeId;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

```

public String getAddress() {
    return address;
}
public void setAddress(String address) {
    this.address = address;
}
public String getEmail() {
    return email;
}
public void setEmail(String email) {
    this.email = email;
}

}

```

Sebelum dilanjutkan ke Controller, lebih baik untuk dicoba terlebih dahulu dengan menjalankan project. sehingga jika tidak terjadi masalah, maka tentunya POJO diatas akan membuatkan table EMPLOYEE pada MySQL. bisa di cek melalui commandline :

```

mysql> use sb_uin_jakarta
Database changed
mysql> show tables;
+-----+
| Tables_in_sb_uin_jakarta |
+-----+
| employee
| hibernate_sequence
+-----+
2 rows in set (0.01 sec)

mysql> desc employee;
+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| employee_id | int(11)   | NO   | PRI | NULL    |       |
| address     | varchar(255)| YES  |     | NULL    |       |
| email       | varchar(255)| YES  |     | NULL    |       |
| name        | varchar(255)| YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql>

```

Terlihat jelas pada command diatas, sudah berhasil membuat table Employee dengan model yang dibuat sebelumnya.

HTML sebagai View

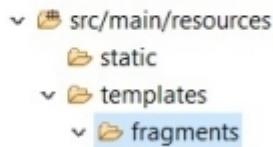
Seperti yang sudah di singgung sebelumnya View akan bertanggung jawab sebagai interaksi user dengan aplikasi dalam hal ini adalah tampilan yang ada di browser (Front End). namun demikian tentu web developer harus memiliki waktu yang cukup untuk mempelajari itu semua terutama adalah User Experiance (UX). dalam aplikasi kali ini akan menggunakan **Thymeleaf Engine Template**.

Untuk mempersingkat waktu sudah disiapkan untuk bisa di **Copy Paste**, yaitu UI Framework dengan simple Bootstrap dan Jquery bisa diakses sebagai layout di aplikasi yang sedang kita buat pada:

https://github.com/arrizaqu-matrial/workshop-uin-2019/blob/master/view_content.md.

Membuat Directory Fragment

Buatlah directory Fragments pada template, yaitu **src->main->resources->templates->fragments** seperti gambar berikut ini :



Membuat File HTML

1. template.html

Buatlah file tersebut di dalam directory **fragments**

```
<!DOCTYPE html>
<html>
<head th:fragment="header">
<title th:text="${title}">Template</title>
<meta charset="utf-8">
<meta name="viewport"
      content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="description" content="simple workshop UIN Jakarta">
<meta name="author" content="arrizaqu">
<title>Home</title>
<!-- Bootstrap core CSS -->
<link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
      integrity="sha384-ggOyR0iXcbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
      crossorigin="anonymous">
<!-- Bootstrap core JavaScript -->
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
      integrity="sha384-q8i/X+965Dz00rT7abK41StQIAqVgRVzbzo5smXKp4YfRvH+BabTE1Pi6jizo"
      crossorigin="anonymous"></script>
<script
      src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js"
      integrity="sha384-UQDzT0hWJ9WZ5IuJyFzq38VQWJUOo7t2fpxWVJmHlZBnJGZLXWqJZLWZPQW1"
      crossorigin="anonymous"></script>
<script
      src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
      integrity="sha384-J3JiH9eIqEiQ7hUdI5JNtqFkqIAiEhjWZJGZLXWqJZLWZPQW1"
      crossorigin="anonymous"></script>
</head>

<body>
<div th:fragment="app-nav">
    <!-- Navigation -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark static-top">
        <div class="container">
            <a class="navbar-brand" href="#">Xsis - Uin Jakarta</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse"
                   data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
                   aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <ul class="navbar-nav ml-auto">
                    <li class="nav-item active"><a class="nav-link" href="#">Home
                        <span class="sr-only" style="font-size: small;">(current)</span>
                    </a></li>
                </ul>
            </div>
        </div>
    </nav>
</div>
</body>
```

```
        <li class="nav-item"><a class="nav-link" href="#">Contact</a>
    </li>
</ul>
</div>
</div>
</nav>
</div>
</body>
</html>
```

2. view_employee.html

Buat file index.html pada directory templates.

```
<!DOCTYPE html>
<html lang="en">
<head th:replace="fragments/template :: header"></head>
<body>
    <!-- Page Content -->
    <th:block th:replace="fragments/template :: app-nav" ></th:block>
    <div class="container">
        <div class="row">
            <div class="col-lg-12 text-center">
                <h1 class="mt-5">Hello, Java Spring Boot</h1>
                <p class="lead">Table : Employee</p>
                Tulis konten disini ..
            </div>
        </div>
    </div>
</body>
</html>
```

Employee Controller

Pertama kali tentu membuat Controller dengan nama **EmployeeController** pada package controller.

```
@Controller
@RequestMapping("/employee")
public class EmployeeController {

    @GetMapping
    public String index() {
        return "view_employee";
    }
}
```

Hasil Program

Jalankan program, kemudian akses program pada <http://localhost:8080/employee>.



Hello, Java Spring Boot

Table : Employee

Tulis konten disini ..

Xsis Mitra Utama

Data Form

Data form biasanya digunakan untuk memasukkan data pada aplikasi, biasa dikenal INSERT atau SAVE Data, pada prosesnya java menginginkan adalah setiap form data yang ada harus berkesuaian dengan model yang sudah dibuatkan sebelumnya, pada sesi kali ini akan membahas tentang penerapan **CRUD** sebagai berikut :

HTML Form

```
<!-- Page Content -->
<th:block th:replace="fragments/template :: app-nav" ></th:block>
<div class="container">
  <div class="row">
    <div class="col-lg-12 text-center">
      <h1 class="mt-5">Hello, Java Spring Boot</h1>
      <p class="lead">Table : Employee</p>
      <form action="#" th:action="@{/employee/save}" th:object="${employee}" method="POST">
        Name : <input type="text" th:field="*{name}" /><br/>
        Email : <input type="text" th:field="*{email}" /><br/>
        <input type="submit" value="Save Data" />
      </form>
    </div>
  </div>
</div>
```

Membuat Repository Data

```
package com.uinjakarta.smartweb.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.uinjakarta.smartweb.model.Employee;

public interface EmployeeRepo extends JpaRepository<Employee, Integer> {
}
```

Membuat Service Layer sebagai Service

```
package com.uinjakarta.smartweb.service;
import javax.transaction.Transactional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.uinjakarta.smartweb.model.Employee;
import com.uinjakarta.smartweb.repository.EmployeeRepo;

@Service
@Transactional
public class EmployeeService {

  @Autowired
  private EmployeeRepo employeeRepo;

  public void save(Employee employee) {
    employeeRepo.save(employee);
  }
}
```

```
}
```

Controller

```
@Autowired  
private EmployeeService employeeService;  
  
@Controller  
@RequestMapping("/employee")  
public class EmployeeController {  
  
    @GetMapping  
    public String index(Model model) {  
        model.addAttribute("title", "Employee CRUD");  
        model.addAttribute("employee", new Employee());  
        return "view_employee";  
    }  
  
    @PostMapping("/save")  
    @ResponseStatus(HttpStatus.CREATED)  
    public void saveData(@ModelAttribute Employee employee) {  
        employeeService.save(employee);  
    }  
}
```

Jalankan Program

1. Compile ulang
2. Coba akses : <http://localhost:8080/employee>

Hello, Java Spring Boot

Table : Employee

Name :

Email :

Show Datatable

1. Tambahkan fungsi findAll pada service
2. memodifikasi Controller index

Menambahkan fungsi findALL pada service

```
public List<Employee> findAll(){  
    return employeeRepo.findAll();  
}
```

Memodifikasi Controller index

```
model.addAttribute("employees", employeeService.findAll());
```

Menambahkan HTML table

```
<table class="table">  
    <thead>  
        <tr>  
            <th scope="col">Name</th>  
            <th scope="col">Email</th>  
            <th scope="col">Address</th>  
            <th scope="col">Action</th>  
        </tr>  
    </thead>  
    <tbody>
```

```

<tr th:each="row : ${employees}">
    <td th:text="${row.name}"></td>
    <td th:text="${row.email}"></td>
    <td th:text="${row.address}"></td>
    <td>
        <a href="#" th:attr="data-id=${row.employeeId}" class="btn btn-sm btn-warning">Edit</a>
        <a href="#" th:attr="data-id=${row.employeeId}" class="btn btn-sm btn-danger">Hapus</a>
    </td>
</tr>
</tbody>
</table>

```

Hasil Browser

The screenshot shows a web application interface. At the top, there is a dark header bar with the text "Xsis - Uin Jakarta" on the left and a menu icon on the right. Below the header, the main content area has a large title "Hello, Java Spring Boot". Underneath the title is a subtitle "Table : Employee". A form is present with fields for "Name" and "Email", and a "Save Data" button. Below the form is a table with four columns: "Name", "Email", "Address", and "Action". Two rows of data are shown:

Name	Email	Action
masyda arrizaqu	arrizaqu@yahoo.com	<button>Edit</button> <button>Hapus</button>
xsis mitra utama	xmu@xisi.co.id	<button>Edit</button> <button>Hapus</button>

Delete Data

Javascript

```

<script type="text/javascript" th:inline="javascript">
$(document).ready(function(){
    var webRoot = /*[@{/}]]*/
    $('.btn-warning').on('click', function(){
        var id = $(this).attr('data-id');
        var conf=confirm("Are you sure delete this data ?");
        if(conf){
            window.location=webRoot + 'employee/delete/' + id;
        }
        return false;
    });
});
</script>

```

Controller

```
@GetMapping("/delete/{id}")
public String delete(@PathVariable("id") int id) {
    employeeService.delete(id);
    return "redirect:/employee";
}
```

Service

```
public void delete(int id) {
    // TODO Auto-generated method stub
    Employee employee = new Employee();
    employee.setEmployeeId(id);
    employeeRepo.delete(employee);
}
```

Update Data

Custome Data Modal

```
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
```

Controller

```
//for update record
// get record by id
@GetMapping("/get/{id}")
@ResponseBody
public Employee getEmployeeById(@PathVariable("id") int id) {
    Employee employee = employeeService.getEmployeeById(id);
    return employee;
}

// execute update
@PostMapping("/update")
public String update(@ModelAttribute("employee") Employee employee) {
    employeeService.save(employee);
    return "redirect:/employee";
}
```

Modal

```
<div class="modal" id="edit-modal" tabindex="-1" role="dialog">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title">Edit Employee</h5>
                <button type="button" class="close" data-dismiss="modal"
                    aria-label="Close">
                    <span aria-hidden="true">&times;</span>
                </button>
```

```

        </div>
        <form action="#" th:action="@{/employee/update}"
              th:object="${employee}" method="POST">
            <div class="modal-body">
              <input type="hidden" id="edit-employeeId" th:field="*{employeeId}" />
              Name : <input id="edit-name" type="text" th:field="*{name}" /><br />
              Email : <input id="edit-email" type="text" th:field="*{email}" /><br />

            </div>
            <div class="modal-footer">
              <input value="Update" type="submit" class="btn btn-primary" />
              <button type="button" class="btn btn-secondary"
                      data-dismiss="modal">Close</button>
            </div>
          </form>
        </div>
      </div>
    </div>

```

Javascript

```

$('.btn-warning').on('click', function(){
  var id = $(this).attr('data-id');
  $.ajax({
    url : webRoot + 'employee/get/'+ id,
    success: function(employee){
      console.log(employee);
      $('#edit-employeeId').val(employee.employeeId);
      $('#edit-name').val(employee.name);
      $('#edit-email').val(employee.email);
    }
  });
  $('#edit-modal').modal();
});

```

Form Validation

Modifikasi POJO

```

@NotNull
public String name;

@Email
public String email;

```

Modifikasi HTML form

```

<form action="#" th:action="@{/employee/save}" th:object="${employee}" method="POST">
  Name : <input type="text" th:field="*{name}" /><br />
  <p th:if="#{fields.hasErrors('name')}" class="label label-danger" th:errors="*{name}"/>
  Email : <input type="text" th:field="*{email}" /><br />
  <p th:if="#{fields.hasErrors('email')}" class="label label-danger" th:errors="*{email}"/>
  <input type="submit" value="Save Data" />
<!-- //show all error
<div>
```

```
<div style="color: red;" th:each="err : ${#fields.errors('*')}" th:text="${err}" />
</div>
-->
</form>
```

Modifikasi Controller

```
@PostMapping("/save")
public String saveData(Model model, @Valid @ModelAttribute("employee") Employee employee, BindingResult bindingResult) {
    if(bindingResult.hasErrors()) {
        model.addAttribute("title", "Employee CRUD");
        return "view_employee";
    }

    employeeService.save(employee);
    return "redirect:/employee";
}
```

Xsis Mitra Utama