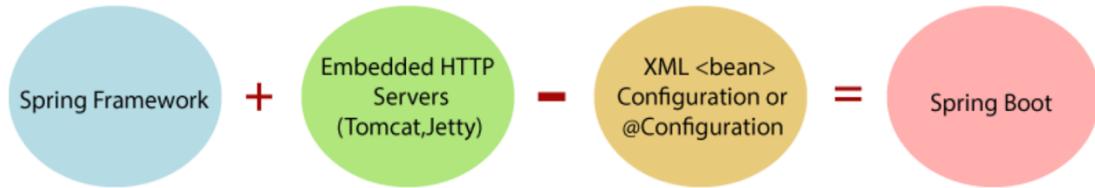


# Spring Boot

Spring Boot Framework គឺ Framework ដែលបង្កើតដោយ “The Spring Team”

ដើម្បីបង្កើតការ Development នៃ Spring Application ឬ ការក្រើង Spring Boot Application ត្រូវការ Spring configuration តិចជាបណ្តាលសំរាប់



ហេតុអិចាណាយេបិយ្យាស៊ែន Spring Boot?

ព្រោះវាបានរាយការនៃការ Development និងការបង្កើត Web Application ។

សារឈ្មោះនៅលើលេខ Spring Boot គឺសូវត្ថិភាព គុណភាព ហើយ Project ដែលបង្កើតដោយ

Spring Boot គឺមានការបង្កើតដោយប្រើប្រាស់និងមាន MVC មកជាមួយស្រាប់។

ជំបូទត្រូវត្រួតម៉ឺង៖

JDK version 8

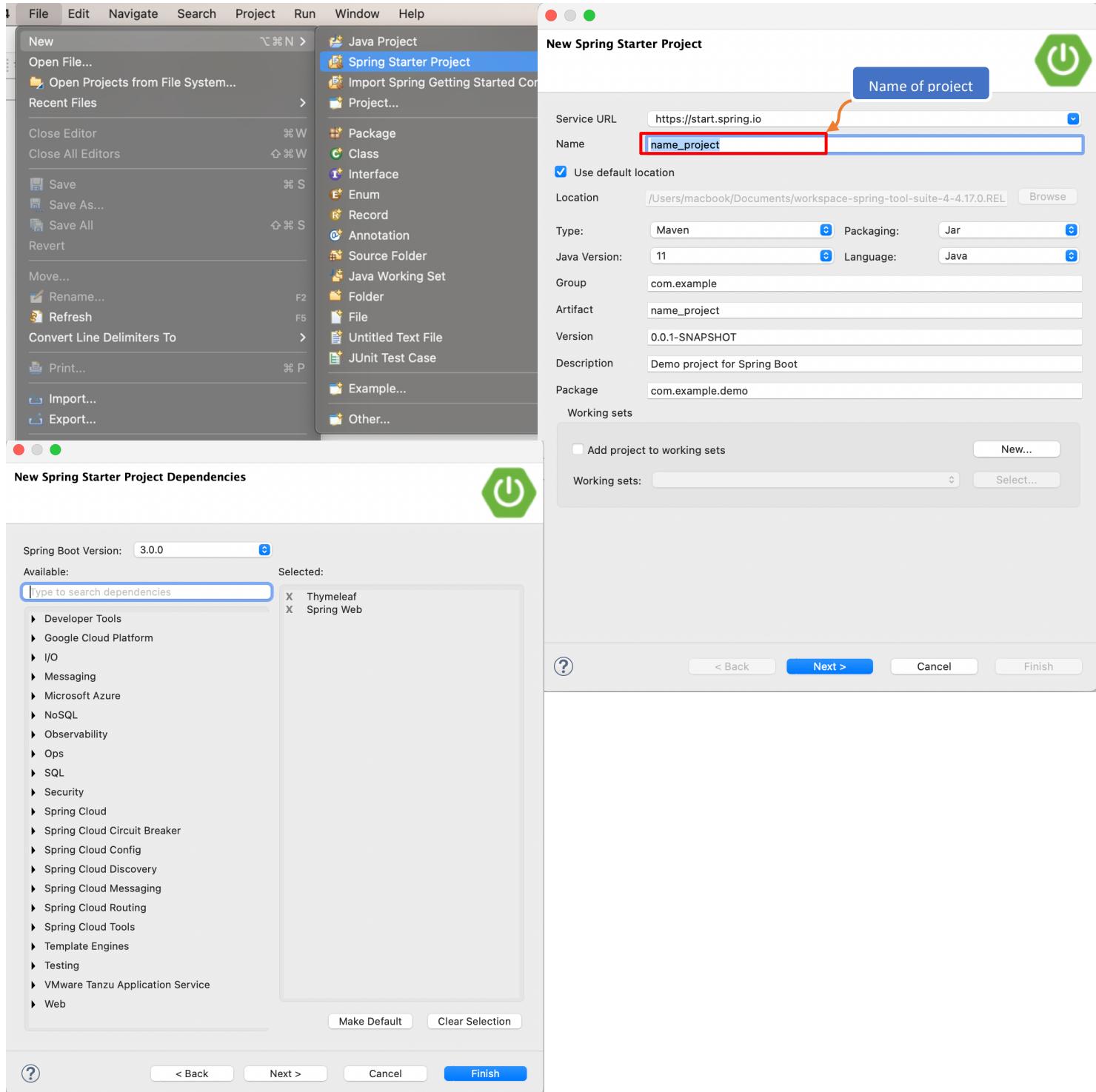
<https://www.oracle.com/java/technologies/downloads/>

IDE មានដូចជា



# កម្រិត Spring boot Project

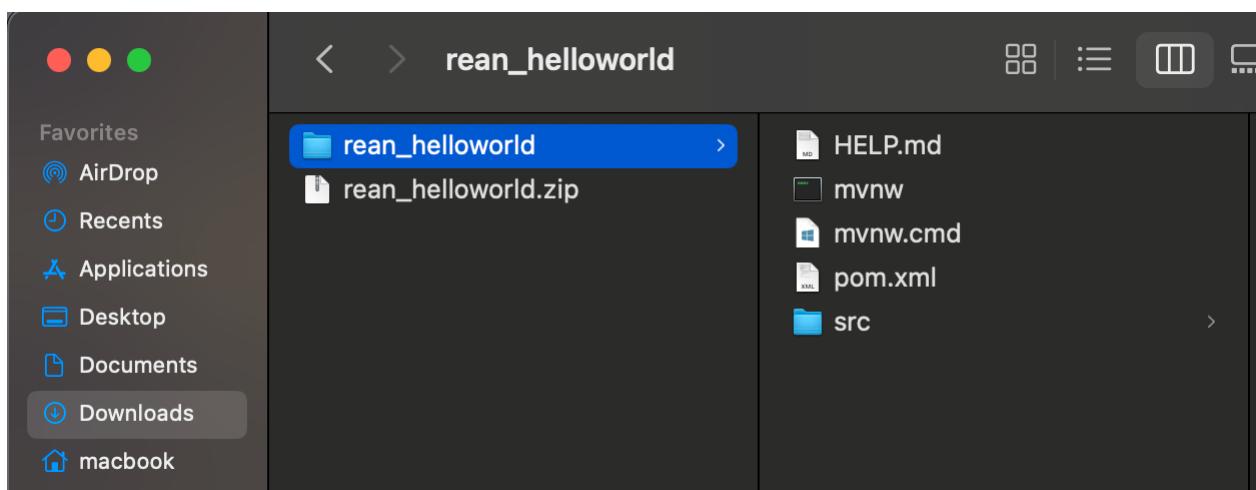
## របៀបទិន្នន័យបង្កើតនៅក្នុង Spring ToolSuite



របៀបទីនឹងរបស់នៅក្នុង Website (Link : <https://start.spring.io> )

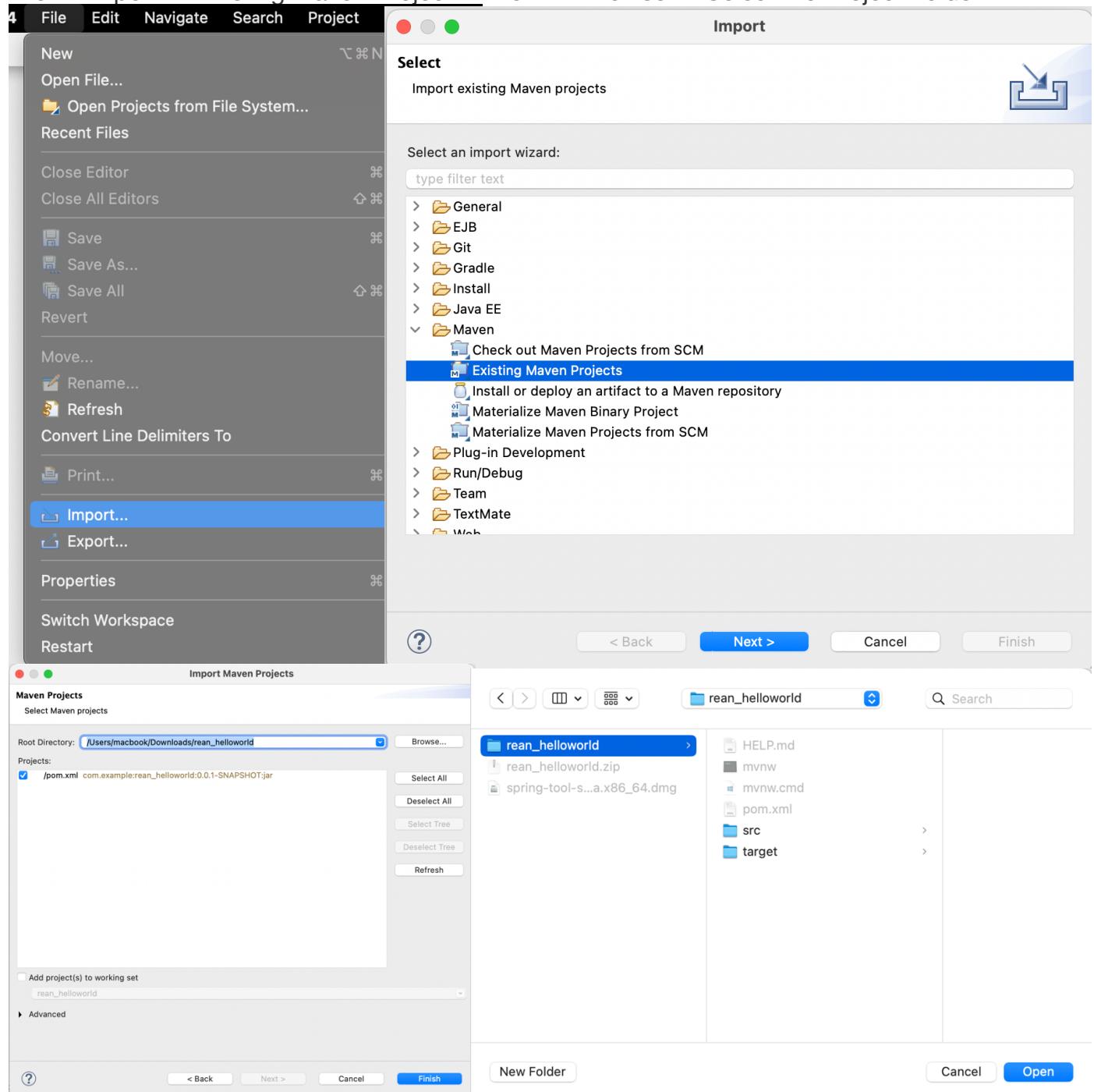
The screenshot shows the Spring Initializr website at <https://start.spring.io>. The configuration is set for a Spring Web application using Java 11, Maven, and Thymeleaf as the template engine. The project metadata includes a group of com.example, artifact of demo, and a name of demo\_helloworld. The description is "Demo project for Spring Boot". The package name is com.example.demo. The packaging is set to Jar. The dependencies section includes Spring Web (selected) and Thymeleaf (selected). Buttons at the bottom include GENERATE (⌘ + ↩), EXPLORE (CTRL + SPACE), and SHARE... .

បន្ទាប់ពីបូច Generate គឺខ្លួនបាន File Zip មួយនឹង Folder Project មួយ



## Import Project in Spring ToolSuite

File -> Import -> Existing Maven Project -> Next -> Browse -> Select the Project Folder

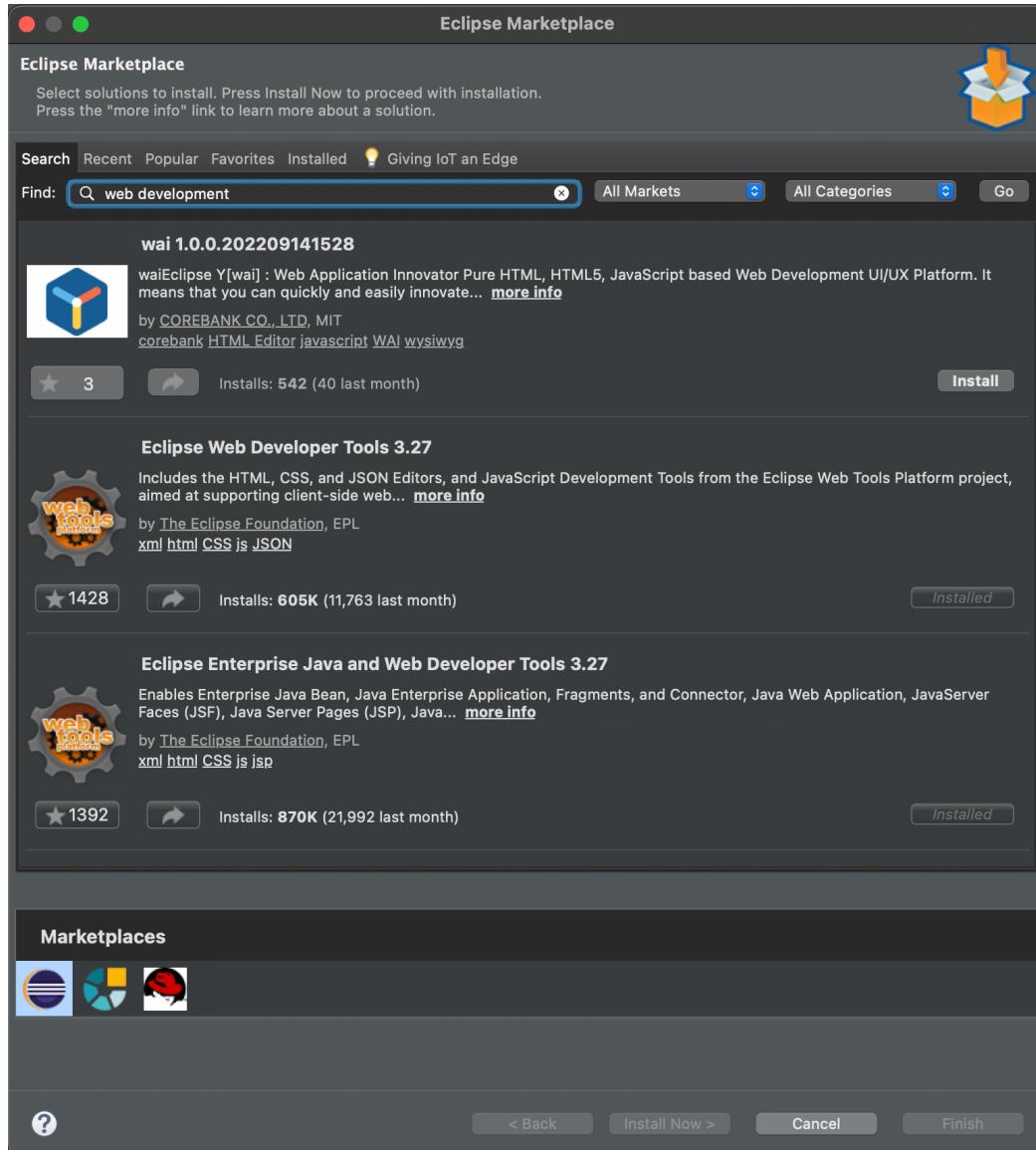


## Spring Boot កំណើត Hello world

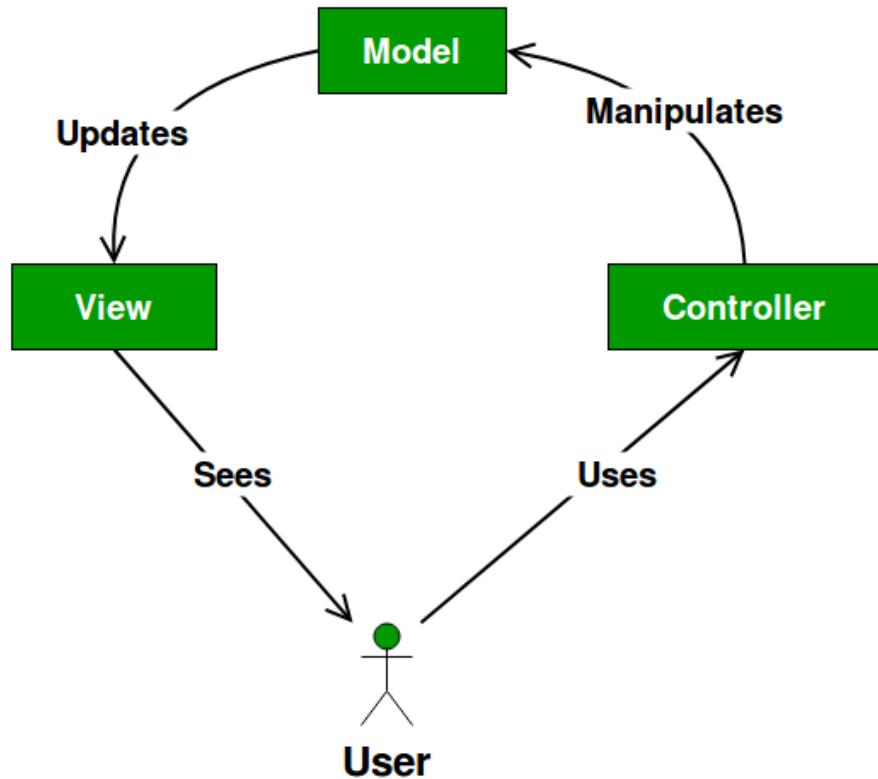
ដើម្បីសរសេរការណា Web ផ្តល់ជាមួយ HTML, CSS, Bootstrap, JavaScript...

បានយើងត្រូវចូលទៅក្នុង Help ->Eclipse Marketplace បន្ទាប់មកទៀត

Search ពាក្យ Web Development រួចរាល់ Install



# MVC = Model View Controller



សំគាល់

គ្នាន់ MVC

[docs.spring.io/spring-boot/docs/current/reference/html/index.html](https://docs.spring.io/spring-boot/docs/current/reference/html/index.html)

មាន់ MVC

[javatpoint.com/spring-boot-thymeleaf-view](https://javatpoint.com/spring-boot-thymeleaf-view)

## Thymeleaf Dependency

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

នៅក្នុងទីនេះ

DemoHelloworldApplication.java

```
1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 @Controller
7 public class DemoHelloworldApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(DemoHelloworldApplication.class, args);
11     }
12
13     @GetMapping("/")
14     public String home() {
15         return "index";
16     }
17 }
```

index.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Introduction</title>
6 </head>
7 <body>
8     <h3>Hello World</h3>
9     <h3>Welcome to Spring Boot</h3>
10 </body>
11 </html>
```

Terminal

```
2022-12-16T15:29:09.431+07:00 INFO 1162 --- [main] c.e.demo.DemoHelloworldApplication.main() : Starting DemoHelloworldApplication on DESKTOP-1QD9V9A with PID 1162 (C:\Users\DESKTOP-1QD9V9A\IdeaProjects\demo-helloworld\target\classes started by DESKTOP-1QD9V9A in C:\Users\DESKTOP-1QD9V9A\IdeaProjects\demo-helloworld)
2022-12-16T15:29:09.434+07:00 INFO 1162 --- [main] o.s.b.w.embedded.tomcat.TomcatEmbeddedServletContainer : Tomcat initialized with port(s): 8080 (http)
2022-12-16T15:29:10.138+07:00 INFO 1162 --- [main] o.s.b.w.embedded.tomcat.TomcatEmbeddedServletContainer : ContextLoaderListener: contextInitialized()
2022-12-16T15:29:10.138+07:00 INFO 1162 --- [main] o.s.b.w.embedded.tomcat.TomcatEmbeddedServletContainer : ContextLoaderListener: contextDestroyed()
2022-12-16T15:29:10.197+07:00 INFO 1162 --- [main] o.s.c.c.CatalinaListener : contextInitialized()
2022-12-16T15:29:10.197+07:00 INFO 1162 --- [main] o.s.c.c.CatalinaListener : contextDestroyed()
2022-12-16T15:29:10.352+07:00 INFO 1162 --- [main] o.s.b.a.w.s.WelcomeServlet : init()
2022-12-16T15:29:10.484+07:00 INFO 1162 --- [main] o.s.b.w.embedded.tomcat.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http) with context path ''
2022-12-16T15:29:18.146+07:00 INFO 1162 --- [nio-8080-exec-1] o.s.c.c.C.[Tomcat].StandardEngine : Server startup in 1 ms
2022-12-16T15:29:18.146+07:00 INFO 1162 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Hello World
2022-12-16T15:29:18.147+07:00 INFO 1162 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Welcome to Spring Boot
```

localhost:8080

Hello World

Welcome to Spring Boot

នៅក្នុង Controller កើតូចនៃរបៀប Spring Boot មាន MVC ក្រោម

Controller.java

```
1 package com.example.demo.controller;
2
3 import org.springframework.web.bind.annotation.GetMapping;
4
5 public class Controller {
6
7     @GetMapping
8     public String views() {
9         return "index";
10    }
11 }
```

index.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Introduction</title>
6 </head>
7 <body>
8     <h3>Hello World <br>
9     In String boot have MVC ready</h3>
10 </body>
11 </html>
```

localhost:8080

Hello World

In String boot have MVC ready

## ການເບີວິທີ່ແຈ້ງຍົດ Backend to Frontend

The screenshot shows a Java code editor and a browser window. The Java code in `MyController.java` defines a controller with a method `views` that adds attributes `a` and `b` to the model. The browser window displays the resulting HTML output with `Value A = 10` and `Value B = 20`.

```
DemoHelloworldApplication.java | MyController.java | index.html | Introduction | localhost:8080/value

1 package com.example.demo.controller;
2
3 import org.springframework.stereotype.Controller;
4 @Controller
5 public class MyController {
6
7     @GetMapping("/value")
8     public String views(Model model) {
9         int a=10,b=20;
10        model.addAttribute("a", a);
11        model.addAttribute("b", b);
12        return "index";
13    }
14
15 }
16
17

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Introduction</title>
6 </head>
7 <body>
8     <h3 th:text="' Value A = ${a}'></h3>
9     <h3 th:text="' Value B = ${b}'></h3>
10 </body>
11 </html>

Value A = 10
Value B = 20
```

## ການໃສ່ກໍລະກົດປົວເຕັມໃນ Frontend

The screenshot shows a Java code editor and a browser window. The Java code in `MyController.java` defines a controller with a method `views` that adds attributes `a` and `b` to the model. The browser window displays the resulting HTML output with `Value A = 100`, `Value B = 20`, `Max is :100`, and `Min is :20`. The Thymeleaf expressions `th:text` and `th:if`/`th:unless` are used to conditionally render parts of the page.

```
DemoHelloworldApplication.java | MyController.java | index.html | Introduction | localhost:8080/value

1 package com.example.demo.controller;
2
3 import org.springframework.stereotype.Controller;
4 @Controller
5 public class MyController {
6
7     @GetMapping("/value")
8     public String views(Model model) {
9         int a=100,b=20;
10        model.addAttribute("a", a);
11        model.addAttribute("b", b);
12        return "index";
13    }
14
15 }
16
17

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Introduction</title>
6 </head>
7 <body>
8     <h3 th:text="' Value A = ${a}'></h3>
9     <h3 th:text="' Value B = ${b}'></h3>
10    <div th:if="${a>b}">
11        <h3 th:text="'Max is :'+ ${a}'></h3>
12    </div>
13    <div th:unless="${a>b}">
14        <h3 th:text="'Min is :'+ ${b}'></h3>
15    </div>
16 </body>
17 </html>

Value A = 100
Value B = 20
Max is :100
Min is :20
```

## Class Model

```
1 package com.example.demo.model;
2
3 public class MyModel {
4
5     private String firstname;
6     private String lastname;
7     private String email;
8     private String address;
9     public MyModel() {
10         // TODO Auto-generated constructor stub
11     }
12     public MyModel(String firstname, String lastname, String email, String address) {
13         super();
14         this.firstname = firstname;
15         this.lastname = lastname;
16         this.email = email;
17         this.address = address;
18     }
19     public String getFirstname() {
20         return firstname;
21     }
22     public void setFirstname(String firstname) {
23         this.firstname = firstname;
24     }
25     public String getLastname() {
26         return lastname;
27     }
28     public void setLastname(String lastname) {
29         this.lastname = lastname;
30     }
31     public String getEmail() {
32         return email;
33     }
34     public void setEmail(String email) {
35         this.email = email;
36     }
37     public String getAddress() {
38         return address;
39     }
40     public void setAddress(String address) {
41         this.address = address;
42     }
43 }
44 }
```

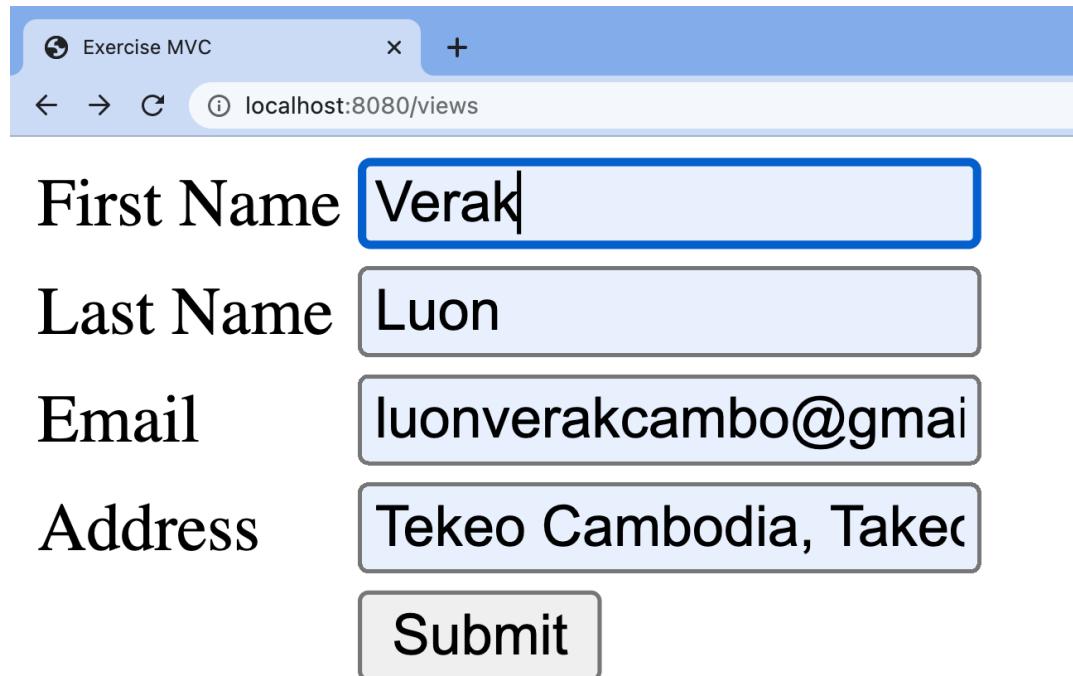
## Class Controller

```
1 package com.example.demo.controller;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.GetMapping;
5 import org.springframework.web.bind.annotation.ModelAttribute;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.bind.annotation.RequestMethod;
8 import org.springframework.web.servlet.ModelAndView;
9
10 import com.example.demo.model.MyModel;
11
12 @Controller
13 public class MyController {
14
15     @GetMapping("/views")
16     public String home() {
17         return "index";
18     }
19
20     @RequestMapping(value = "/save", method = RequestMethod.POST)
21     public ModelAndView save(@ModelAttribute MyModel model)
22     {
23         ModelAndView modelAndView = new ModelAndView();
24         modelAndView.setViewName("user-data");
25         modelAndView.addObject("user", model);
26         return modelAndView;
27     }
28 }
```

## Viesw

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Exercise MVC</title>
6 </head>
7 <body>
8 <form action="save" method="post"> | 
9   <table>
10    <tr>
11      <td><label for="firstname">First Name</label></td>
12      <td><input type="text" name="firstname"></input></td>
13    </tr>
14    <tr>
15      <td><label for="lastname">Last Name</label></td>
16      <td><input type="text" name="lastname"></input></td>
17    </tr>
18    <tr>
19      <td><label for="email">Email</label></td>
20      <td><input type="text" name="email"></input></td>
21    </tr>
22    <tr>
23      <td><label for="address">Address</label></td>
24      <td><input type="text" name="address"></input></td>
25    </tr>
26    <tr>
27      <td></td>
28      <td><input type="submit" value="Submit"></input></td>
29    </tr>
30   </table>
31 </form>
32 </body>
33 </html>
```

## Result

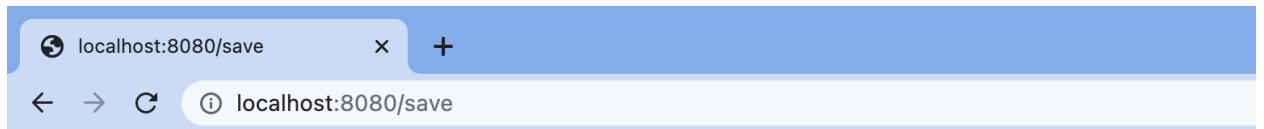


The screenshot shows a web browser window with the title "Exercise MVC". The address bar indicates the URL is "localhost:8080/views". The page displays a form with four input fields and a submit button. The first field is labeled "First Name" and contains the value "Verak". The second field is labeled "Last Name" and contains the value "Luon". The third field is labeled "Email" and contains the value "luonverakcambo@gmail.com". The fourth field is labeled "Address" and contains the value "Tekeo Cambodia, Takeo". A blue rectangular box highlights the "First Name" field.

First Name	Verak
Last Name	Luon
Email	luonverakcambo@gmail.com
Address	Tekeo Cambodia, Takeo
Submit	

```
1<html xmlns:th="https://thymeleaf.org">
2  <table>
3    <tr>
4      <td><h4>Full Name: </h4></td>
5      <td><h4 th:text="${user.firstname} +' '+ ${user.lastname}"></h4></td>
6    </tr>
7    <tr>
8      <td><h4>Email ID: </h4></td>
9      <td><h4 th:text="${user.email}"></h4></td>
10   </tr>
11   <tr>
12     <td><h4>Address: </h4></td>
13     <td><h4 th:text="${user.address}"></h4></td>
14   </tr>
15 </table>
16 </html>
```

## Result



**Full Name: Verak Luon**

**E-mail : luonverakcambo@gmail.com**

**Address : Tekeo Cambodia**

# Crud Operation with MVC

## Project One

### Backend

```
1 package com.example.demo;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.boot.CommandLineRunner;
5 import org.springframework.boot.SpringApplication;
6 import org.springframework.boot.autoconfigure.SpringBootApplication;
7 import com.example.demo.repository.StudentRepository;
8
9 @SpringBootApplication
10 public class StudentApplication implements CommandLineRunner {
11
12     public static void main(String[] args) {
13
14         SpringApplication.run(StudentApplication.class, args);
15     }
16
17     @Autowired
18     private StudentRepository studentRepository;
19
20     @Override
21     public void run(String... args) throws Exception {
22
23         // Student stu1=new Student("Luon", "Verak", "luonverak@gmail.com");
24         // studentRepository.save(stu1);
25         //
26         // Student stu2=new Student("Sok", "Cheata", "sokcheata@gmail.com");
27         // studentRepository.save(stu2);
28         //
29         // Student stu3=new Student("Seng", "Lyna", "senglyna@gmail.com");
30         // studentRepository.save(stu3);
31         //
32     }
33
34 }
```

# Class Controller

```
2@import org.springframework.stereotype.Controller;
10 @Controller
11 public class StudentController {
12
13     private StudentService studentService;
14
15     public StudentController(StudentService studentService) {
16         super();
17         this.studentService = studentService;
18     }
19     @GetMapping("/students")
20     public String listStudents(Model mod) {
21         mod.addAttribute("students",studentService.getAllStudents());
22         return "students";
23     }
24     @GetMapping("/students/new")
25     public String createStudentForm(Model mod) {
26
27         Student student=new Student();
28         mod.addAttribute("student", student);
29         return "create_student";
30     }
31     @PostMapping("/students")
32     public String saveStudent(@ModelAttribute ("student") Student student) {
33         studentService.saveStudent(student);
34         return "redirect:/students";
35     }
36     @GetMapping("/students/edit/{id}")
37     public String editStudentForm(@PathVariable Long id,Model mod) {
38         mod.addAttribute("student", studentService.getStudentById(id));
39         return "edit_student";
40     }
41     @PostMapping("/students/{id}")
42     public String updateStudent(@PathVariable Long id,@ModelAttribute("student") Student student,Model mod) {
43         Student existingStudent = studentService.getStudentById(id);
44         existingStudent.setId(id);
45         existingStudent.setFirstname(student.getFirstname());
46         existingStudent.setLastname(student.getLastname());
47         existingStudent.setEmail(student.getEmail());
48         existingStudent.setScore(student.getScore());
49         studentService.updateStudent(existingStudent);
50         return "redirect:/students";
51     }
52     @GetMapping("/students/{id}")
53     public String deleteStudent(@PathVariable Long id) {
54         studentService.deleteStudentById(id);
55         return "redirect:/students";
56     }
57 }
```

```
1 package com.example.demo.entity;
2+import jakarta.persistence.Column;[]
12 @NoArgsConstructor
13 @AllArgsConstructor
14 @Getter
15 @Setter
16 @Entity
17 @Table(name="students")
18 public class Student {
19@   @Id
20     @GeneratedValue(strategy = GenerationType.IDENTITY)
21   private long id;
22@   @Column(name = "first_name", nullable = false)
23   private String firstname;
24@   @Column(name = "last_name", nullable = false)
25   private String lastname;
26@   @Column(name = "email", nullable = false)
27   private String email;
28@   @Column(name="score", nullable = false)
29   private float score;
30 }
```

```
1 package com.example.demo.repository;
2
3+import org.springframework.data.jpa.repository.JpaRepository;
4
5 import com.example.demo.entity.Student;
6
7 public interface StudentRepository extends JpaRepository<Student, Long>{
8
9 }
```

```
StudentService.java X
1 package com.example.demo.service;
2
3+import java.util.List;
4
5 import com.example.demo.entity.Student;
6
7 public interface StudentService {
8     List<Student> getAllStudents();
9     Student saveStudent(Student student);
10    Student getStudentById(Long id);
11    Student updateStudent(Student student);
12    void deleteStudentById(Long id);
13 }
```

```
1 package com.example.demo.service.impl;
2
3 import java.util.List;
4
5 import org.springframework.stereotype.Service;
6
7 import com.example.demo.entity.Student;
8 import com.example.demo.repository.StudentRepository;
9 import com.example.demo.service.StudentService;
10 @Service
11 public class StudentImp implements StudentService{
12
13     private StudentRepository studentRepository;
14
15     public StudentImp(StudentRepository studentRepository) {
16         super();
17         this.studentRepository = studentRepository;
18     }
19
20     @Override
21     public List<Student> getAllStudents() {
22
23         return studentRepository.findAll();
24     }
25
26     @Override
27     public Student saveStudent(Student student) {
28
29         return studentRepository.save(student);
30     }
31
32     @Override
33     public Student getStudentById(Long id) {
34         // TODO Auto-generated method stub
35         return studentRepository.findById(id).get();
36     }
37
38     @Override
39     public Student updateStudent(Student student) {
40         // TODO Auto-generated method stub
41         return studentRepository.save(student);
42     }
43
44     @Override
45     public void deleteStudentById(Long id) {
46         studentRepository.deleteById(id);
47
48     }
49
50
51 }
52 }
```

# Student GUI

<https://getbootstrap.com/docs/4.0/getting-started/introduction/>

[https://www.w3schools.com/bootstrap/bootstrap\\_navbar.asp](https://www.w3schools.com/bootstrap/bootstrap_navbar.asp)

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4   <meta charset="UTF-8">
5   <title>Student Management</title>
6   <link rel="stylesheet"
7     href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
8     integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
9     crossorigin="anonymous">
10 </head>
11 <body>
12
13   <nav class="navbar navbar-expand-md bg-dark navbar-dark">
14     <!-- Brand -->
15     <a class="navbar-brand" th:href="@{/students}">Students Management</a>
16
17     <!-- Toggler/collapsible Button -->
18     <button class="navbar-toggler" type="button" data-toggle="collapse"
19       data-target="#collapsibleNavbar">
20       <span class="navbar-toggler-icon"></span>
21     </button>
22
23     <!-- Navbar links -->
24     <div class="collapse navbar-collapse" id="collapsibleNavbar">
25       <ul class="navbar-nav">
26         <li class="nav-item"><a class="nav-link" th:href="@{/students}">Students
27           Management</a></li>
28       </ul>
29     </div>
30   </nav>
31
32   <div class="container">
33     <div class="row">
34       <h1>List of Students</h1>
35     </div>
36
37     <div class="row">
38       <div class="col-lg-3">
39         <a th:href="@{/students/new}" class="btn btn-primary btn-sm mb-3">
40           Add Student</a>
41       </div>
42     </div>
43
44     <table class="table table-striped table-bordered">
45       <thead class="table-dark">
46         <tr>
47           ...
48       </tr>
49     </thead>
50     <tbody>
51       ...
52     </tbody>
53   </div>
```

Content-Type:html

Writable

```
26          <li class="nav-item"><a class="nav-link" th:href="@{/students}">Students  
27              Management</a></li>  
28      </ul>  
29  </div>  
30 </nav>  
31  
32 <div class="container">  
33     <div class="row">  
34         <h1>List of Students</h1>  
35     </div>  
36  
37     <div class="row">  
38         <div class="col-lg-3">  
39             <a th:href="@{/students/new}" class="btn btn-primary btn-sm mb-3">  
40                 Add Student</a>  
41         </div>  
42     </div>  
43  
44     <table class="table table-striped table-bordered">  
45         <thead class="table-dark">  
46             <tr>  
47                 <th>First Name</th>  
48                 <th>Last Name</th>  
49                 <th>Email</th>  
50                 <th>Score</th>  
51                 <th>Actions</th>  
52             </tr>  
53         </thead>  
54         <tbody>  
55             <tr th:each="student: ${students}">  
56                 <td th:text="${student.firstname}"></td>  
57                 <td th:text="${student.lastname}"></td>  
58                 <td th:text="${student.email}"></td>  
59                 <td th:text="${student.score}"></td>  
60  
61                 <td><a th:href="@{/students/edit/{id}(id=${student.id})}"  
62                     class="btn btn-primary">Update</a> <a  
63                     th:href="@{/students/{id}(id=${student.id})}"  
64                     class="btn btn-danger">Delete</a></td>  
65             </tr>  
66         </tbody>  
67     </table>  
68 </div>  
69  
70 </body>  
71 </html>
```

## Create New Student

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4   <meta charset="UTF-8">
5   <title>Student Management</title>
6   <link rel="stylesheet"
7     href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
8     integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
9     crossorigin="anonymous">
10 </head>
11 <body>
12
13   <nav class="navbar navbar-expand-md bg-dark navbar-dark">
14     <!-- Brand -->
15     <a class="navbar-brand" th:href="@{/students}">Students Management</a>
16
17     <!-- Toggler/collapse Button -->
18     <button class="navbar-toggler" type="button" data-toggle="collapse"
19       data-target="#collapsibleNavbar">
20       <span class="navbar-toggler-icon"></span>
21     </button>
22
23     <!-- Navbar links -->
24     <div class="collapse navbar-collapse" id="collapsibleNavbar">
25       <ul class="navbar-nav">
26         <li class="nav-item"><a class="nav-link" th:href="@{/students}">Students
27           Management</a></li>
28       </ul>
29     </div>
30   </nav>
31
32   <br>
33   <br>
34   <div class="container">
35     <div class="row">
36       <div
37         class="col-lg-6 col-md col-sm-6 container justify-content-center card">
38         <div class="text-center">
39           <h1>Create New Students</h1>
40           <div class="card-body">
41             <form th:action="@{/students}" th:object="${student}"
42               method="POST">
43               <div class="form-group">
44                 <label>Student First Name</label> <input type="text"
45                   name="firstname" th:field="*{firstname}" class="form-control"
46                   placeholder="Enter Student First Name" />
47               </div>
48               <div class="form-group">
49                 <label>Student Last Name</label> <input type="text"
50                   name="lastname" th:field="*{lastname}" class="form-control"
51                   placeholder="Enter Student Last Name" />
52               </div>
53               <div class="form-group">
54                 <label>Student Email</label> <input type="text" name="email"
55                   th:field="*{email}" class="form-control"
56                   placeholder="Enter Student Email" />
57               </div>
58               <div class="form-group">
59                 <label>Student Score</label> <input type="text" name="score"
60                   th:field="*{score}" class="form-control"
61                   placeholder="Enter Student Score" />
62               </div>
63               <div class="box-footer">
64                 <button type="submit" class="btn btn-primary">Submit</button>
65               </div>
66             </form>
67           </div>
68         </div>
69       </div>
70     </div>
71   </body>
72 </html>
```

## Update Data Student

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4   <meta charset="UTF-8">
5   <title>Student Management</title>
6   <link rel="stylesheet"
7     href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
8     integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
9     crossorigin="anonymous">
10 </head>
11 <body>
12
13   <nav class="navbar navbar-expand-md bg-dark navbar-dark">
14     <!-- Brand -->
15     <a class="navbar-brand" th:href="@{/students}">Students Management</a>
16
17     <!-- Toggler/collapsible Button -->
18     <button class="navbar-toggler" type="button" data-toggle="collapse"
19       data-target="#collapsibleNavbar">
20       <span class="navbar-toggler-icon"></span>
21     </button>
22
23     <!-- Navbar links -->
24     <div class="collapse navbar-collapse" id="collapsibleNavbar">
25       <ul class="navbar-nav">
26         <li class="nav-item"><a class="nav-link" th:href="@{/students}">Students
27           Management</a></li>
28       </ul>
29     </div>
30   </nav>
31   <br>
32   <br>
33   <div class="container">
34     <div class="row">
35       <div
36         class="col-lg-6 col-md col-sm-6 container justify-content-center card">
37         <div class="text-center">
38           <h1>Update Students</h1>
39           <div class="card-body">
40             <form th:action="@{/students (id=${student.id})}"
41               th:object="${student}" method="POST">
42               <div class="form-group">
43                 <label>Student First Name</label> <input type="text"
44                   name="firstname" th:field="*{firstname}" class="form-control"
45                   placeholder="Enter Student First Name" />
46               </div>
47               <div class="form-group">
48                 <label>Student Last Name</label> <input type="text"
49                   name="lastname" th:field="*{lastname}" class="form-control"
50                   placeholder="Enter Student Last Name" />
51             </div>
52             <div class="form-group">
53               <label>Student Email</label> <input type="text" name="email"
54                 th:field="*{email}" class="form-control"
55                 placeholder="Enter Student Email" />
56             </div>
57             <div class="form-group">
58               <label>Student Score</label> <input type="text" name="score"
59                 th:field="*{score}" class="form-control"
60                 placeholder="Enter Student Score" />
61             </div>
62             <div class="box-footer">
63               <button type="submit" class="btn btn-primary">Submit</button>
64             </div>
65           </form>
66         </div>
67       </div>
68     </div>
69   </div>
70 </body>
71 </html>
```

# Databases with MySQL

## Create and Connection

```
1 spring.datasource.url=jdbc:mysql://localhost:3306/dbstudents?createDatabaseIfNotExist=true&allowPublicKeyRetrieval=true&useSSL=false
2 spring.datasource.username=root
3 spring.datasource.password=
4
5 spring.datasource.hikari.connection-timeout=20000
6 spring.datasource.hikari.minimum-idle=5
7 spring.datasource.hikari.maximum-pool-size=100
8 spring.datasource.hikari.idle-timeout=300000
9 spring.datasource.hikari.max-lifetime=1200000
10 spring.datasource.hikari.auto-commit=true
11
12 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
13 spring.jpa.properties.hibernate.id.new_generator_mappings=false
14 spring.jpa.properties.hibernate.format_sql=true
15
16 spring.jpa.hibernate.ddl-auto=update
17 spring.jpa.show-sql=true
18
19 spring.jpa.open-in-view=true
20
21 spring.servlet.multipart.max-file-size=15MB
22 spring.servlet.multipart.max-request-size=15MB
23
```

## Table

<input type="checkbox"/>				2	luonverakcambo@gmail.com	Verak	Luon	90
<input type="checkbox"/>				3	sokcheata@gmail.com	Sok	Cheata	98
<input type="checkbox"/>				4	nhebrany@gmail.com	Nheb	Rany	97

Check all    With selected:

## Result

First Name	Last Name	Email	Score	Actions
Verak	Luon	luonverakcambo@gmail.com	90.0	<button>Update</button> <button>Delete</button>
Sok	Cheata	sokcheata@gmail.com	98.0	<button>Update</button> <button>Delete</button>
Nheb	Rany	nhebrany@gmail.com	97.0	<button>Update</button> <button>Delete</button>

When Click Button Add or Update

Create New Students

Student First Name  
Verak

Student Last Name  
Luon

Student Email  
luonverakcambo@gmail.com

Student Score  
98

Submit

# API = Application Programming Interface

## JSON = JavaScript Object Notation

The screenshot shows an IDE interface with four code editor panes:

- People.java**: A JPA entity class with annotations like @Table, @Id, and @GeneratedValue.
- PeopleService.java**: A service layer class with methods for finding people by ID, address, saving, updating, and deleting.
- MyController.java**: A REST controller with mappings for listing all people, finding a person by ID, and finding people by address.
- MyController2.java**: Another REST controller with similar functionality, using @PostMapping and @PutMapping annotations.

```
1 package com.example.demo.model;
2
3 import jakarta.persistence.Entity;
4
5 @Getter
6 @Setter
7 @NoArgsConstructor
8 @AllArgsConstructor
9 @Entity
10 @Table(name="tblpeople")
11 public class People {
12
13     @Id
14     @GeneratedValue(strategy = GenerationType.AUTO)
15     private long id;
16     private String name;
17     private String address;
18 }
19
20
21
22
23
24
25
26
27 }
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45 }
```

```
1 package com.example.demo.service;
2
3 import java.util.List;
4
5 @Service
6 public class PeopleService {
7
8     @Autowired
9     private PeopleRepo peopleRepo;
10
11     public List<People> findAllpeople() {
12         return peopleRepo.findAll();
13     }
14
15     public People findById(long id) {
16
17         return peopleRepo.findById(id).orElse(new People());
18     }
19
20     public List<People> findByAddress(String address) {
21
22         return peopleRepo.findByAddress(address);
23     }
24
25     public void save(People people) {
26
27         peopleRepo.save(people);
28     }
29
30     public void update(People people) {
31
32         People pl=peopleRepo.findById(people.getId()).orElse(new People());
33         if(pl.getId()==0)
34             pl.setName("No People");
35         else
36             peopleRepo.save(people);
37     }
38
39     public void delete(People people) {
40
41         peopleRepo.deleteById(people.getId());
42     }
43 }
```

```
1 package com.example.demo.controller;
2
3 import java.util.List;
4
5 @RestController
6 @RequestMapping("/api/people")
7 public class MyController {
8
9     @Autowired
10     private PeopleService people;
11
12     @GetMapping("")
13     public List<People> listAllpeople() {
14
15         return people.findAllpeople();
16     }
17
18     @GetMapping("/{id}")
19     public People findByID(@PathVariable long id) {
20
21         return people.findById(id);
22     }
23
24     @GetMapping("/address/{address}")
25     public List<People> findByaddress(@PathVariable String address) {
26
27         return people.findByAddress(address);
28     }
29
30 }
```

```
1 package com.example.demo.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 @RestController
6 @RequestMapping("/api/people")
7 public class MyController2 {
8
9     @Autowired
10     private PeopleService peopleService;
11
12     @PostMapping("")
13     public People add(@RequestBody People people) {
14
15         peopleService.save(people);
16         return people;
17     }
18
19     @PutMapping("")
20     public People update(@RequestBody People people) {
21
22         peopleService.update(people);
23         return people;
24     }
25
26     @DeleteMapping("")
27     public People delete(@RequestBody People people) {
28
29         peopleService.delete(people);
30         return people;
31     }
32 }
```

```

1 package com.example.demo.repo;
2
3 import java.util.List;
4
5 public interface PeopleRepo extends JpaRepository<People, Long>{
6     List<People> findByAddress(String address);
7 }
8
9
10
11
12
13 }
14
15
16
17
18
19

```

application.properties

```

1 spring.datasource.url=jdbc:mysql://localhost:3306/springapi?createDatabaseIfNotExist=true&allowPublicKeyRetrieval=true&useSSL=false
2 spring.datasource.username=root
3 spring.datasource.password=
4
5 spring.datasource.hikari.connection-timeout=20000
6 spring.datasource.hikari.minimum-idle=5
7 spring.datasource.hikari.maximum-pool-size=100
8 spring.datasource.hikari.idle-timeout=300000
9 spring.datasource.hikari.max-lifetime=1200000
10 spring.datasource.hikari.auto-commit=true
11
12 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
13 spring.jpa.properties.hibernate.id.new_generator_mappings=false
14 spring.jpa.properties.hibernate.format_sql=true
15
16 spring.jpa.hibernate.ddl-auto=update
17 spring.jpa.show-sql=true
18
19 spring.jpa.open-in-view=true

```

		<b>id</b>	<b>address</b>	<b>name</b>
<input type="checkbox"/>	 Edit	 Copy	 Delete	1 Phnom Penh Luon Verak
<input type="checkbox"/>	 Edit	 Copy	 Delete	2 Phnom Penh Heng Visal
<input type="checkbox"/>	 Edit	 Copy	 Delete	3 Phnom Penh Jonh Son
<input type="checkbox"/>	 Edit	 Copy	 Delete	4 Angtasom Seng Chan Lyna
<input type="checkbox"/>	 Edit	 Copy	 Delete	5 Phnom Penh Chomrern Sopheak
<input type="checkbox"/>	 Edit	 Copy	 Delete	7 Takeo Chhorn Chenda
<input type="checkbox"/>	 Edit	 Copy	 Delete	102 Phnom Penh Leng Dalin
<input type="checkbox"/>	 Edit	 Copy	 Delete	152 Phnom Penh So Da

We need Install Postman for testing API

<https://www.postman.com/downloads/>



The image shows a screenshot of a web browser window displaying a JSON response from a local API endpoint. The URL in the address bar is `localhost:8080/api/people`. The JSON data lists several people with their IDs, names, and addresses:

```
[{"id": 1, "name": "Luon Verak", "address": "Phnom Penh"}, {"id": 2, "name": "Heng Visal", "address": "Phnom Penh"}, {"id": 3, "name": "Jonh Son", "address": "Phnom Penh"}, {"id": 4, "name": "Seng Chan Lyna", "address": "Angtasom"}, {"id": 5, "name": "Chomrern Sopheak", "address": "Phnom Penh"}, {"id": 7, "name": "Chhorn Chenda", "address": "Takeo"}, {"id": 102, "name": "Leng Dalin", "address": "Phnom Penh"}, {"id": 152, "name": "So Da", "address": "Phnom Penh"}]
```

## Get API in Postman សំដែរលើការទាញពិន្ទុយតាមរយៈEndpoint people

The screenshot shows the Postman interface with a successful GET request to `http://localhost:8080/api/people`. The response body is a JSON array of four objects, each representing a person with fields `id`, `name`, and `address`.

```
[{"id": 1, "name": "Luon Verak", "address": "Phnom Penh"}, {"id": 2, "name": "Heng Visal", "address": "Phnom Penh"}, {"id": 3, "name": "Johh Son", "address": "Phnom Penh"}, {"id": 4, "name": "Seng Chan Lyna", "address": "Angtasom"}]
```

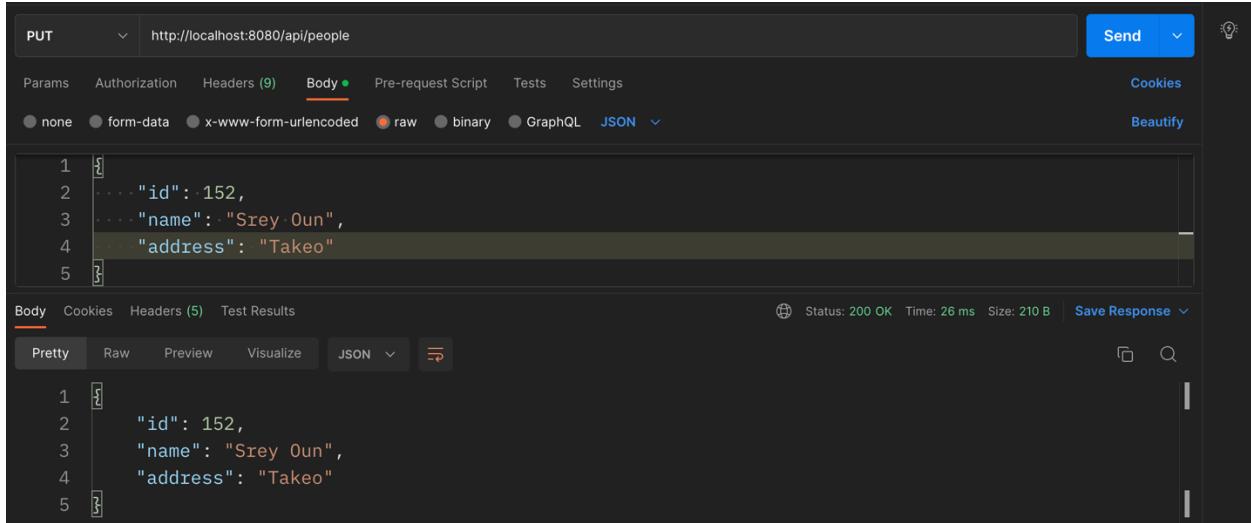
## Post API in Postman សំដែរលើការបញ្ចប់ពិន្ទុយតាមរយៈEndpoint people

The screenshot shows the Postman interface with a successful POST request to `http://localhost:8080/api/people`. The request body contains a JSON object with `name` and `address` fields. The response body shows the newly created object with an `id` of 202.

```
{"name": "New Member", "address": "In City"}
```

```
{"id": 202, "name": "New Member", "address": "In City"}
```

## Put API in Postman សំដែរលើការកែត្របន្ទីយតាមរយៈ Endpoint people

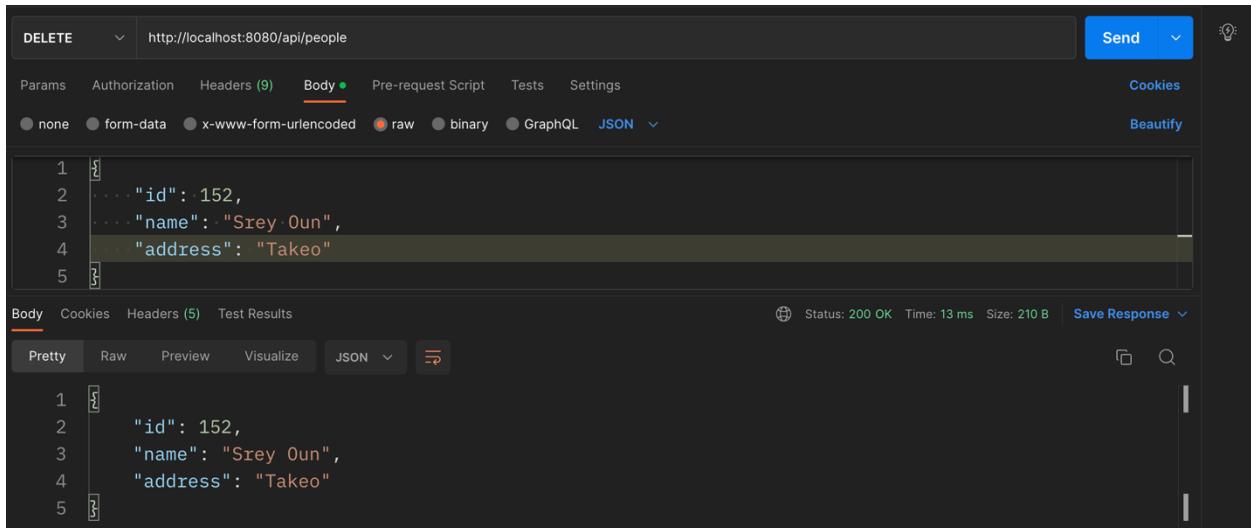


The screenshot shows the Postman interface with a PUT request to `http://localhost:8080/api/people`. The Body tab is selected, displaying a JSON object with fields `id`, `name`, and `address`. The response status is 200 OK.

```
1 {  
2   "id": 152,  
3   "name": "Srey Oun",  
4   "address": "Takeo"  
5 }
```

Status: 200 OK Time: 26 ms Size: 210 B Save Response

## Delete API in Postman សំដែរលើការលើបន្ទីយតាមរយៈ Endpoint people



The screenshot shows the Postman interface with a DELETE request to `http://localhost:8080/api/people`. The Body tab is selected, displaying a JSON object with fields `id`, `name`, and `address`. The response status is 200 OK.

```
1 {  
2   "id": 152,  
3   "name": "Srey Oun",  
4   "address": "Takeo"  
5 }
```

Status: 200 OK Time: 13 ms Size: 210 B Save Response

# Restful Spring boot + React Js

## Backend

```
1 package com.example.demo;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 @SpringBootApplication
6 public class Project1Application implements CommandLineRunner{
7
8     public static void main(String[] args) {
9         SpringApplication.run(Project1Application.class, args);
10    }
11
12    @Autowired
13    private EmployeeRepository employeeRepository;
14    @Override
15    public void run(String... args) throws Exception {
16        //        Employee employee=new Employee();
17        //        employee.setFirstName("Verak");
18        //        employee.setLastName("Luon");
19        //        employee.setEmail("luonverak@gmail.com");
20        //
21        //        employeeRepository.save(employee);
22    }
23
24}
25
26
27
28
29
30 }
```

```
1 package com.example.demo.controller;
2
3 import java.util.List;
4
5 @CrossOrigin("http://localhost:3000/")
6 @RestController
7 @RequestMapping("/api/employee")
8 public class EmployeeController {
9
10     @Autowired
11     private EmployeeRepository employeeRepository;
12
13     @GetMapping
14     public List<Employee> getAllEmployee(){
15         return employeeRepository.findAll();
16     }
17
18     @PostMapping("")
19     public Employee createEmployee(@RequestBody Employee employee) {
20         return employeeRepository.save(employee);
21     }
22
23     //find by id
24     @GetMapping("{id}")
25     public ResponseEntity<Employee> getEmployeeByID(@PathVariable long id){
26         Employee employee=employeeRepository.findById(id)
27             .orElseThrow( () -> new ResourceNotFoundException("Employee not exist with id :" + id));
28         return ResponseEntity.ok(employee);
29     }
30
31     @PutMapping("{id}")
32     public ResponseEntity<Employee> updateEmployeeByID(@PathVariable long id,@RequestBody Employee employeeDetail){
33         Employee updaetEmployee=employeeRepository.findById(id)
34             .orElseThrow( () -> new ResourceNotFoundException("Employee not exist with id :" + id));
35
36         updaetEmployee.setFirstName(employeeDetail.getFirstName());
37         updaetEmployee.setLastName(employeeDetail.getLastName());
38         updaetEmployee.setEmail(employeeDetail.getEmail());
39
40         employeeRepository.save(updaetEmployee);
41         return ResponseEntity.ok(updaetEmployee);
42     }
43
44     @DeleteMapping("{id}")
45     public ResponseEntity<HttpStatus> deleteEmployeeByID(@PathVariable long id){
46         Employee employee=employeeRepository.findById(id)
47             .orElseThrow( () -> new ResourceNotFoundException("Employee not exist with id :" + id));
48         employeeRepository.delete(employee);
49         return new ResponseEntity<>(HttpStatus.NO_CONTENT);
50     }
51 }
52 }
```

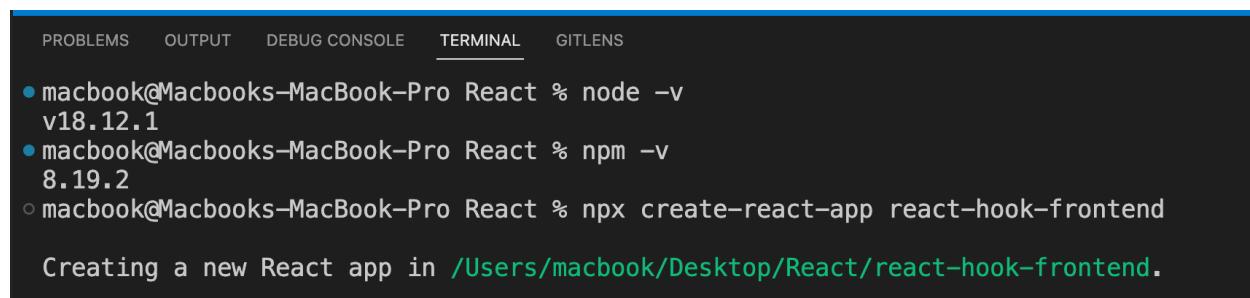
```
1 package com.example.demo.exception;
2
3 import org.springframework.http.HttpStatus;
4
5
6 @ResponseStatus(value = HttpStatus.NOT_FOUND)
7 public class ResourceNotFoundException extends RuntimeException{
8
9     public ResourceNotFoundException(String message) {
10         super(message);
11     }
12 }
```

```
EmployeeRepository.java x
1 package com.example.demo.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5
6 public interface EmployeeRepository extends JpaRepository<Employee, Long>{
7
8
9 }
10
```

```
1 package com.example.demo.model;
2
3 import jakarta.persistence.Column;...
4
5
6
7
8
9
10
11
12
13
14 @Getter
15 @Setter
16 @NoArgsConstructor
17 @AllArgsConstructor
18 @Entity
19 @Table(name = "tbemployee")
20 public class Employee {
21
22     @Id
23     @GeneratedValue(strategy = GenerationType.IDENTITY)
24     private long id;
25     @Column(name = "first_name")
26     private String firstName;
27     @Column(name = "last_name")
28     private String lastName;
29     @Column(name = "email")
30     private String email;
31 }
```

## Frontend React Js

### Command for install project



The screenshot shows a terminal window with the following command history:

- macbook@Macbooks-MacBook-Pro React % node -v  
v18.12.1
- macbook@Macbooks-MacBook-Pro React % npm -v  
8.19.2
- macbook@Macbooks-MacBook-Pro React % npx create-react-app react-hook-frontend

At the bottom of the terminal, it says: Creating a new React app in /Users/macbook/Desktop/React/react-hook-frontend.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS
● macbook@Macbooks-MacBook-Pro React % npm install bootstrap --save
  added 2 packages, and audited 3 packages in 1s
  2 packages are looking for funding
    run `npm fund` for details
  found 0 vulnerabilities
○ macbook@Macbooks-MacBook-Pro React % █

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL GITLENS
● macbook@Macbooks-MacBook-Pro React % npm install axios --save
  added 9 packages, and audited 12 packages in 3s
  3 packages are looking for funding
    run `npm fund` for details
  found 0 vulnerabilities
○ macbook@Macbooks-MacBook-Pro React % □

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS
● macbook@Macbooks-MacBook-Pro React % npm install react-router-dom --save
  added 8 packages, and audited 20 packages in 5s
  3 packages are looking for funding
    run `npm fund` for details
  found 0 vulnerabilities
○ macbook@Macbooks-MacBook-Pro React % █
```

The screenshot shows a dark-themed code editor with two open files: `App.js` and `App.css`.

**App.js**

```
You, 2 seconds ago | 1 author (You)
1 import './App.css';
2 import { BrowserRouter , Route,Routes} from 'react-router-dom';
3 import Footer from './components/Footer';
4 import HeaderComponents from './components/HeaderComponents';
5 import ListEmployeeComponents from './components/ListEmployeeComponents';
6 import AddEmployee from './components/AddEmployee';
7
8 function App() {
9   return [
10   <div>
11     <HeaderComponents/>
12     <BrowserRouter>
13       <Routes>
14         <Route path="/" element={<ListEmployeeComponents/>}></Route>
15         <Route path="/employee" element={<ListEmployeeComponents/>}></Route>
16         <Route path="/add-employee" element={<AddEmployee/>}></Route>
17         <Route path="/edit-employee/:id" element={<AddEmployee/>}></Route>
18       </Routes>
19     </BrowserRouter>
20     <Footer/>
21   ] You, 2 seconds ago • Uncommitted changes
22
23
24   </div>
25 }
26
27
28 export default App;
29
```

**App.css**

```
You, 1 second ago | 1 author (You)
1 .footer{
2   position: absolute;
3   bottom: 0;
4   width: 100%;
5   height: 50px; You, 1 second ago • Uncommitted changes
6   background-color: red;
7   text-align: center;
8   color: white;
9 }
```

EmployeeService.js

```
1 import axios from 'axios';
2
3 const EMPLOYEE_BASE_REST_API_URL ="http://localhost:8080/api/employee";
4
5 class EmployeeService{
6
7     getAllEmployee(){
8         return axios.get(EMPLOYEE_BASE_REST_API_URL)
9     }
10    createEmployee(employee){
11        return axios.post(EMPLOYEE_BASE_REST_API_URL,employee)
12    }
13    getEmployeeByID(employeeId){
14        return axios.get(EMPLOYEE_BASE_REST_API_URL + '/' + employeeId)
15    }
16    updateEmployee(employeeId,employee){
17        return axios.put(EMPLOYEE_BASE_REST_API_URL + '/' + employeeId,employee)
18    }
19    deleteEmployee(employeeId){
20        return axios.delete(EMPLOYEE_BASE_REST_API_URL + '/' + employeeId)
21    }
22 }
23 export default new EmployeeService();
```

AddEmployee.js

```
1 import React, { useEffect, useState } from 'react'
2 import EmployeeService from '../service/EmployeeService'
3 import {Link, useNavigate, useParams} from 'react-router-dom';
4 const AddEmployee = () => {
5
6     const [firstName, setFirstName]=useState('')
7     const [lastName, setLastName]=useState('')
8     const [email, setEmail]=useState('')
9
10    const navigate = useNavigate();
11    const {id}=useParams();
12
13    const saveOrUpdateEmployee=(e) =>{
14        e.preventDefault();
15        const employee={firstName,lastName,email}
16
17        if(id){
18            EmployeeService.updateEmployee(id,employee).then((response)=>{
19                navigate('/employee')
20            }).catch(error => {
21                console.log(error);
22            })
23        }else{
24            EmployeeService.createEmployee(employee).then((response)=>{
25
26                console.log(response.data);
27                navigate('/employee');
28            }).catch(error => {
29                console.log(error);
30            })
31        }
32
33    }
34
35    useEffect(()=>{
36        EmployeeService.getEmployeeByID(id).then((response)=>{
37            setFirstName(response.data.firstName)
```

```
  AddEmployee.js 1, U ×
react-hook-frontend > src > components > AddEmployee.js > AddEmployee
  ↵4
  35  useEffect(()=>{
  36      EmployeeService.getEmployeeByID(id).then((response)=>{
  37          setFirstName(response.data.firstName)
  38          setLastName(response.data.lastName)
  39          setEmail(response.data.email)
  40      }).catch(error=>{
  41          console.log(error);
  42      })
  43  },[ ])
  44
  45  const title = ()=>{
  46      if(id){
  47          return <h2 className='text-center'>Update Information Employee</h2>
  48      }else{
  49          return <h2 className='text-center'>Add Information Employee</h2>
  50      }
  51  }
  52
  ↵2
  53  return (
  54      <div>
  55          <div className='container'>
  56              <div className='row'>
  57
  58                  <div className='card col-md-6 offset-md-3'>
  59                      <br></br>
  60                      <br></br>
  61                      {
  62                          title()
  63                      }
  64                  <div className='card-body'>
  65                      <form>
  66                          <div className='form-group mb-2'>
  67                              <label className='form-label'> First Name </label>
  68                              <input
  69                                  type="text"
  70                                  placeholder='Enter First Name'
  71                                  className='form-control'
  72                                  value={firstName}
  73                                  onChange={(e) => setFirstName(e.target.value)}>
  74
  75
  76                          </input>
  77                      </div>
  78                      <div className='form-group mb-2'>
  79                          <label className='form-label'> Last Name </label>
  80                          <input
  81                              type="text"
  82                              placeholder='Enter Last Name'
  83                              className='form-control'
  84                              value={lastName}
  85                              onChange={(e) => setLastName(e.target.value)}>
  86
  87                          </input>
  88                  </div>

```

Ln 107, Col 11 Spaces: 4 UTF-8 LF {} JavaScript ⚡ Go Live ⌂ ⌂

The screenshot shows a code editor with two tabs open:

- AddEmployee.js**: A React component for adding an employee. It contains form fields for First Name, Last Name, and Email, and a button for saving or updating.
- Footer.js**: A simple footer component that includes a copyright notice.

```
react-hook-frontend > src > components > AddEmployee.js > AddEmployee
1 import React from 'react'
2
3 function AddEmployee() {
4   return (
5     <div>
6       <form>
7         <div>
8           <label>First Name</label>
9           <input type="text" placeholder="Enter First Name" className="form-control" value={firstName} onChange={(e) => setFirstName(e.target.value)}>
10          </input>
11        </div>
12        <div>
13          <label>Last Name</label>
14          <input type="text" placeholder="Enter Last Name" className="form-control" value={lastName} onChange={(e) => setLastName(e.target.value)}>
15          </input>
16        </div>
17        <div>
18          <label>Email</label>
19          <input type="text" placeholder="Enter E-mail" className="form-control" value={email} onChange={(e) => setEmail(e.target.value)}>
20          </input>
21        </div>
22        <button className="btn btn-success" onClick={(e)=>saveOrUpdateEmployee(e)}>Save</button>
23        <Link to="/employee" className="btn btn-danger" style={{marginLeft:"10px"}}>Cancel</Link>
24      </form>
25    </div>
26  )
27}
28
29 export default AddEmployee

react-hook-frontend > src > components > Footer.js > Footer
1 import React from 'react'
2
3 function Footer() {
4   return (
5     <div>
6       <footer>
7         <nav className='footer'>
8           <span>All Right Spring Boot Develop on 23 December 2022 by Luon Verak</span>
9         </nav>
10      </footer>
11    </div>
12  )
13}
14
15 export default Footer
```

The image shows two code editor windows side-by-side.

**HeaderComponents.js**

```
HeaderComponents.js U ×  
react-hook-frontend > src > components > HeaderComponents.js > HeaderComponents  
1 import React from 'react'  
2  
3 function HeaderComponents() {  
4     return (  
5         <div>  
6             <header>  
7                 <nav className='navbar navbar-expand-md navbar-dark bg-dark'>  
8                     <div>  
9                         <a href='https://www.facebook.com/?_rdc=2&_rdr' className='navbar-brand'>  
10                            Employee Management System  
11                        </a>  
12                    </div>  
13                </nav>  
14            </header>  
15        </div>  
16    )  
17}  
18  
19 export default HeaderComponents
```

**ListEmployeeComponents.js**

```
ListEmployeeComponents.js U ×  
react-hook-frontend > src > components > ListEmployeeComponents.js > ListEmployeeComponents  
1 import React, { useEffect, useState } from 'react'  
2 import { Link } from 'react-router-dom';  
3 import EmployeeService from '../service/EmployeeService';  
4  
5 function ListEmployeeComponents() {  
6     const [employee, setEmployee] = useState([]);  
7     useEffect(() => {  
8         getAllEmployee();  
9     }, [])  
10  
11     const getAllEmployee=()=>{  
12         EmployeeService.getAllEmployee().then((response) => {  
13             setEmployee(response.data)  
14             console.log(response.data)  
15         }).catch(  
16             error => {  
17                 console.log(error);  
18             }  
19         )  
20     }  
21     const deleteEmployee = (employeeId) => {  
22         EmployeeService.deleteEmployee(employeeId).then((response)=>{  
23             getAllEmployee();  
24         }).catch(error =>{  
25             console.log(error);  
26         })  
27     }  
28     return (  
29         <div className='container'>  
30             <h2 className='text-center'>Information of Employee</h2>  
31             <Link to="/add-employee" className="btn btn-primary mb-2"> Add Employee</Link>  
32             <br><br>  
33             <table className='table table-bordered table-striped'>  
34                 <thead>  
35                     <th>Employee-ID</th>  
36                     <th>FirstName</th>  
37                     <th>LastName</th>
```

```

ListEmployeeComponents.js U ×
react-hook-frontend > src > components > ListEmployeeComponents.js > ListEmployeeComponents
28     return [
29       <div className='container'>
30         <h2 className='text-center'>Information of Employee</h2>
31         <Link to="/add-employee" className="btn btn-primary mb-2"> Add Employee</Link>
32         <br><br>
33         <table className='table table-bordered table-striped'>
34           <thead>
35             <th>Employee-ID</th>
36             <th>First Name</th>
37             <th>Last Name</th>
38             <th>E-mail</th>
39             <th>Actions</th>
40           </thead>
41           <tbody>
42             {
43               employee.map(
44                 employees =>
45                   <tr key={employees.id}>
46                     <td>{employees.id}</td>
47                     <td>{employees.firstName}</td>
48                     <td>{employees.lastName}</td>
49                     <td>{employees.email}</td>
50                     <td>
51                       <Link className='btn btn-info' to={`/edit-employee/${employees.id}`}>Update</Link>
52                       <button className='btn btn-danger' onClick={() => deleteEmployee(employees.id)} style={{marginLeft:"10px"}}>Delete</button>
53                     </td>
54                   </tr>
55                 )
56               )
57             </tbody>
58           </table>
59         </div>
60       ]
61     ]
62   ]
63 export default ListEmployeeComponents

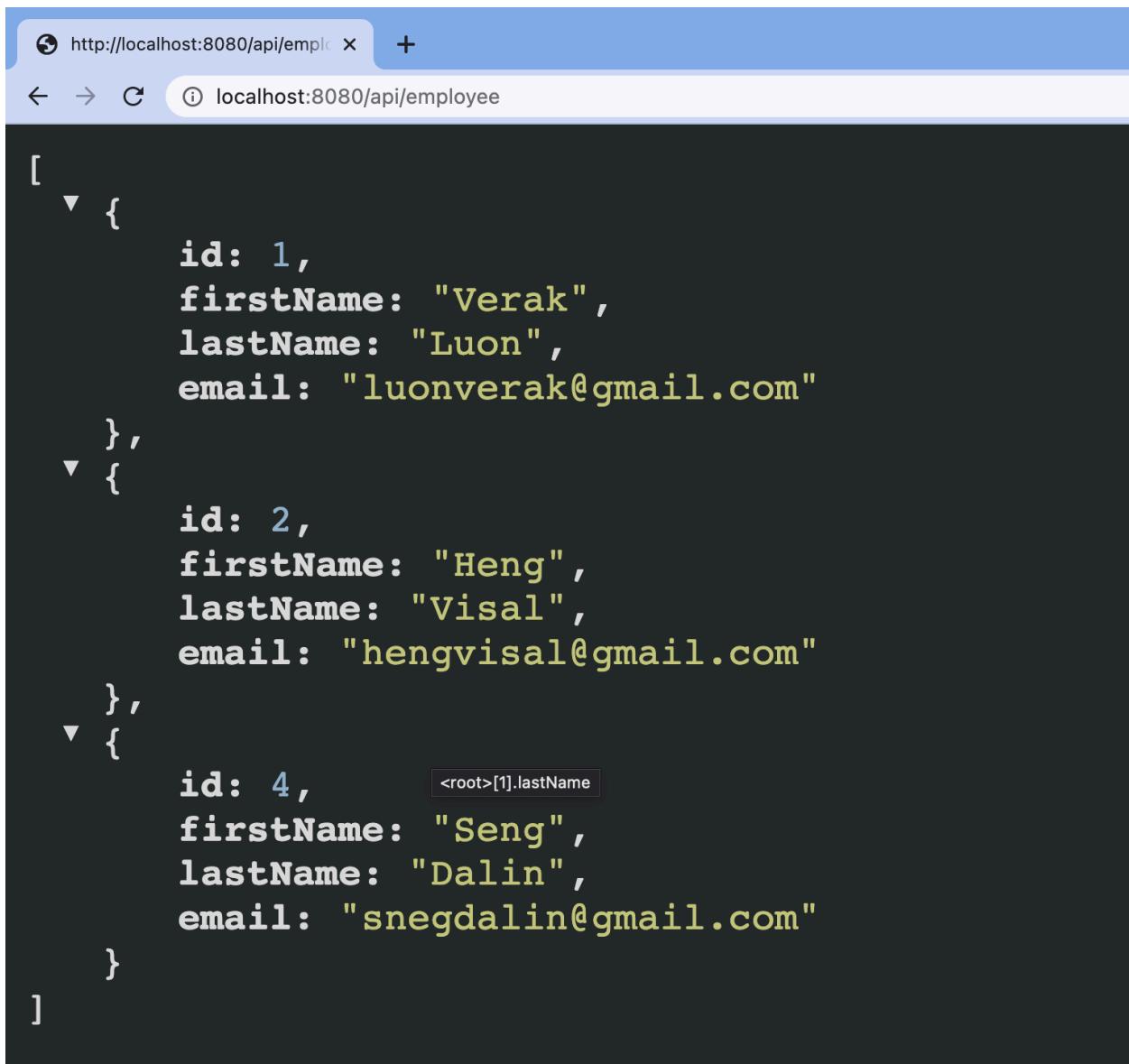
```

## Table Databases

				<b>id</b>	<b>email</b>	<b>first_name</b>	<b>last_name</b>
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	luonverak@gmail.com	Verak	Luon
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	hengvisal@gmail.com	Heng	Visal
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	snegdalin@gmail.com	Seng	Dalin

 Check all    With selected:  Edit     Copy     Delete     Export

## Result API



The screenshot shows a browser window with the URL `http://localhost:8080/api/employee`. The page displays a JSON array of employee objects. Each object contains the following fields: id, firstName, lastName, and email. The JSON is formatted with indentation and line breaks for readability.

```
[  
  {  
    "id": 1,  
    "firstName": "Verak",  
    "lastName": "Luon",  
    "email": "luonverak@gmail.com"  
  },  
  {  
    "id": 2,  
    "firstName": "Heng",  
    "lastName": "Visal",  
    "email": "hengvisal@gmail.com"  
  },  
  {  
    "id": 4,  
    "firstName": "Seng",  
    "lastName": "Dalin",  
    "email": "snegdalin@gmail.com"  
  }]
```

Information of Employee				
Add Employee				
Employee-ID	FirstName	LastName	E-mail	Actions
1	Verak	Luon	luonverak@gmail.com	<button>Update</button> <button>Delete</button>
2	Heng	Visal	hengvisal@gmail.com	<button>Update</button> <button>Delete</button>
4	Seng	Dalin	snegdalin@gmail.com	<button>Update</button> <button>Delete</button>

All Right Spring Boot Develop on 23 December 2022 by Luon Verak

## Button Add New Employee or Update

Add Information Employee				
First Name	Enter First Name			
Last Name	Enter Last Name			
Email	Enter E-mail			
	<button>Save Employee</button> <button>Cencel</button>			

All Right Spring Boot Develop on 23 December 2022 by Luon Verak