## CURD project for student database using spring boot and react

**Spring initializer Setup**

🡪Spring web

🡪Spring data JPA

🡪Mysql driver

🡪Lomboak

**Step 1:**

🡪Creation of database

🡪Creation of table

**Step 2:**

🡪Connecting database to spring boot using application properties

🡪Code:

spring.datasource.url = jdbc:mysql://localhost:3306/ems?useSSL=false  
spring.datasource.username = root  
spring.datasource.password = Vignesh@123  
spring.jpa.database-platform = org.hibernate.dialect.MySQLDialect  
spring.jpa.hibernate.ddl-auto = update

**Step 3:**

🡪creation of entity

🡪Code:

package com.javrect.curd.model;  
  
import jakarta.persistence.\*;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@NoArgsConstructor  
@AllArgsConstructor  
@Entity //This means that the class represents a table in a database, and each instance of the class corresponds to a row in that table.  
@Table(name = "employees") //This defines the table name  
public class Employee {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*) //automatically generate the values  
 private long id;  
  
 @Column(name = "first\_name") //represent the column name in table  
 private String firstName;  
  
 @Column(name = "last\_name")  
 private String lastName;  
  
 @Column(name="email\_id")  
 private String emailId;  
}

🡪This will create attributes in table name

**Step 4:**

🡪creation of JPA repository

🡪That is an interface which extends JPA repository

🡪Code:

package com.javrect.curd.repository;  
  
import com.javrect.curd.model.Employee;  
import org.springframework.data.jpa.repository.JpaRepository;  
// a JPA Repository is an interface that provides a set of methods for performing CRUD (Create, Read, Update, Delete) operations on entities  
  
public interface EmployeeRepository extends JpaRepository<Employee,Long> {  
  
}

## Jumping into UI

Step 6(Read operation):

🡪Create the list of employee UI

🡪initially we use, use State that is

 const [employees,setEmployees] = useState([])

🡪**employees** we are using for frontend where as **setEmployees** we are using for backend

🡪Basic creation of UI(frontend code)

<div className="container">

        <h2 className="text-center">

            List Employess

        </h2>

        <table className="table table-bordered table-striped">

            <thead>

                <th>Employee id</th>

                <th>Employee first name</th>

                <th>Employee last name</th>

                <th>Employee Email id</th>

                <th>Actions</th>

            </thead>

            <tbody>

                {  //Iterates over the employees array and returns a table row for each employee.

                   employees.map(

                    employee =>//.id,.firstName....etc taken from spring boot to print the value

                        <tr>

                            <td>{employee.id}</td>

                            <td>{employee.firstName}</td>

                            <td>{employee.lastName}</td>

                            <td>{employee.emailId}</td>

                        </tr>

                   )

                }

            </tbody>

        </table>

    </div>

🡪Backend code(TO Call API)

useEffect(() => {

        EmployeeService.getAllEmployees().then((response) =>{

            setEmployees(response.data) //print the values in console

        }).catch(error =>{

            console.log(error); //this will perform when url is wrong

        })

     }, [])

🡪API(need to create a normal class)

const EMPLOYEE\_BASE\_REST\_API\_URL = 'http://localhost:8080/api/v1/employees';

getAllEmployees(){

        return axios.get(EMPLOYEE\_BASE\_REST\_API\_URL)

    }

🡪Controller(getAllValues This need to be created 1st in backend)

@GetMapping //main page  
public List<Employee> getAllEmployee(){  
  
 return employeeRepository.findAll();  
}

Step 7:

🡪In order to test the code we need to set some values

🡪Code

package com.javrect.curd;  
  
import com.javrect.curd.model.Employee;  
import com.javrect.curd.repository.EmployeeRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class CurdApplication implements CommandLineRunner {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(CurdApplication.class, args);  
 }  
 @Autowired  
 private EmployeeRepository employeeRepository;  
 @Override  
 public void run(String... args) throws Exception {  
 Employee employee = new Employee();  
 employee.setFirstName("vignesh");  
 employee.setLastName("venkatachalam");  
 employee.setEmailId("vigneshv1867@gmail.com");  
 employeeRepository.save(employee);  
  
 Employee employee1 = new Employee();  
 employee1.setFirstName("karthik");  
 employee1.setLastName("venkatachalam");  
 employee1.setEmailId("karthikv1867@gmail.com");  
 employeeRepository.save(employee1);  
 }  
}

import 'bootstrap/dist/css/bootstrap.min.css'

keep in note index.js

import axios from 'axios'

Keep in note for api

How to create react app

🡪 npx create-react-app student (1)

Important imports

🡪 npm install axios –save

🡪npm install bootstrap –save (2)

🡪npm install react-router-dom;

Note:

🡪For instant imports without manual insertion we can use a json that is

🡪Code

{

    "exclude"= [node\_module]

}

## Jumping into UI

Step 8:Creation of header and footer

Code:

<div>

        <header>

            <nav className="navbar navbar-expand-md navbar-dark bg-dark">

                <div>

                    <a href="https://javaguides.net" className="navbar-brand">

                    Employee Management Application

                    </a>

                </div>

            </nav>

        </header>

    </div>

Code:

<div>

        <footer className="footer">

            <span className="text-muted">

                All Rights Reserved 2021 @JavaGuides

            </span>

        </footer>

    </div>

Code(css):

.footer{

  position: relative;

  bottom: 0;

  width: 100%;

  height: 50px;

  background-color:white;

  text-align: center;

  color:black;

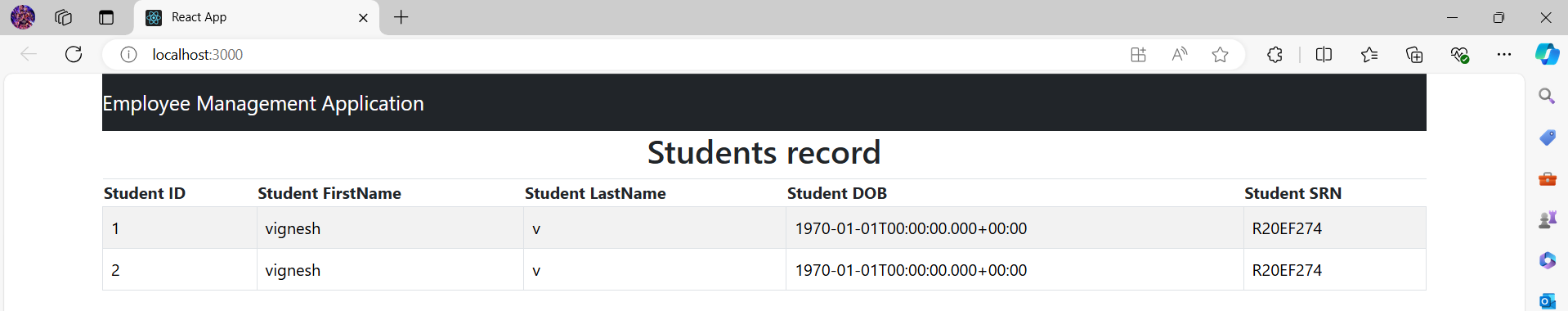
}

Difference between using a container and not using

🡪Using container

  <div className="container">

</div>

Output:

🡪With out using a container

  <div>

</div>

Output:

A screenshot of a computer

Description automatically generated

Step 9:(Create operation)

Controller

Code:

@PostMapping  
public Student creatStudent(@RequestBody Student student){  
 return studentRepository.save(student);  
}

Api:

createStudent(stud){

            return axios.post(url,stud)

        }

## Jumping into UI

Add student details:

A screenshot of a computer

Description automatically generated

A blue rectangle with white text

Description automatically generated <Link to = "/add-student" className="btn btn-primary mb-2">Add Employee </Link>

Read.js file

A screenshot of a computer

Description automatically generated

<Route path = "/add-student" Component= {Add}></Route>

App.js file

Add student Details UI

Code: for Add.js

useStates

 const[firstName,setFirstName] = useState("")

        const[lastName,setLatName] = useState("")

        const[dob,setDob] = useState("")

        const[srn,setSrnName] = useState("")

Here firstName,lastName…. Should be same as entity because we are doing post operation where we need to give the data. So that json formate must be correct. This fields will be the reference of API of backend. Which we will use in save function

Ex:

{

    "srn": "jsbh",

    "firstName": "sjsmdnk",

    "lastName": "jfkwn",

    "dob": "2001-12-03"

}

UI design

<div>

      <br></br>

        <div className="container">

          <div className="row">

              <div className="card col-md-6 offset-md-3 offset-md-3">

                  <h1 className='text-center'>Add Student Details</h1>

                  <div className="card-body">

                    <form>

                      <div className="form-group mb-2">

                        <label className="form-label">First Name:</label>

                        <input type='text' placeholder='Enter your first name' className="form-control" onChange={(e) => setFirstName(e.target.value)}></input>

                      </div>

                      <div className="form-group mb-2">

                        <label className="form-label">Last Name:</label>

                        <input type='text' placeholder='Enter your last name' className="form-control" onChange={(e) => setLatName(e.target.value)}></input>

                      </div>

                      <div className="form-group mb-2">

                        <label className="form-label">Date Of Birth:</label>

                        <input type='text' placeholder='YYYY-MM-DD' className="form-control" onChange={(e) => setDob(e.target.value)}></input>

                      </div>

                      <div className="form-group mb-2">

                        <label className="form-label">SRN:</label>

                        <input type='text' placeholder='Enter your SRN' className="form-control" onChange={(e) => setSrnName(e.target.value)}></input>

                      </div>

                      <button className="btn btn-success me-2" onClick={(e) => saveStudent(e)}>Save</button>

                      <Link to = "/employees" className="btn btn-danger">Cancel</Link>

                    </form>

                  </div>

              </div>

          </div>

        </div>

    </div>

This is a function to save the data where we have given in the input field, after we click the save button we will be directed to initial landing page without any refresh

const history = useNavigate(); //used to navigate to initial page with all prev data exist

        const saveStudent = (e) =>{

            e.preventDefault(); //stop the refresh

            const student = {firstName,lastName,dob,srn}

            Student.createStudent(student).then((response)=>{

                console.log(response.data)

                history("/")

            }).catch((error)=>{

                console.log(error)

            })

        }

Save function

Search bar UI:

<form>

            <div className="input-container">

                <input className="form-control mr-sm-2" type="search" placeholder="Search" onChange={(e) => setSearch(e.target.value)} aria-label="Search"/>

            </div>

        </form>

Logic code:

const[search,setSearch] = useState("")

<tbody>

                {

                    first.filter((sear)=>{

                        return search === ''? sear : sear.srn.includes(search)

                    }).map(

                        stud =>

                        <tr>

                            <td>{stud.firstName}</td>

                            <td>{stud.lastName}</td>

                            <td>{stud.dob}</td>

                            <td>{stud.srn}</td>

                        </tr>

                    )

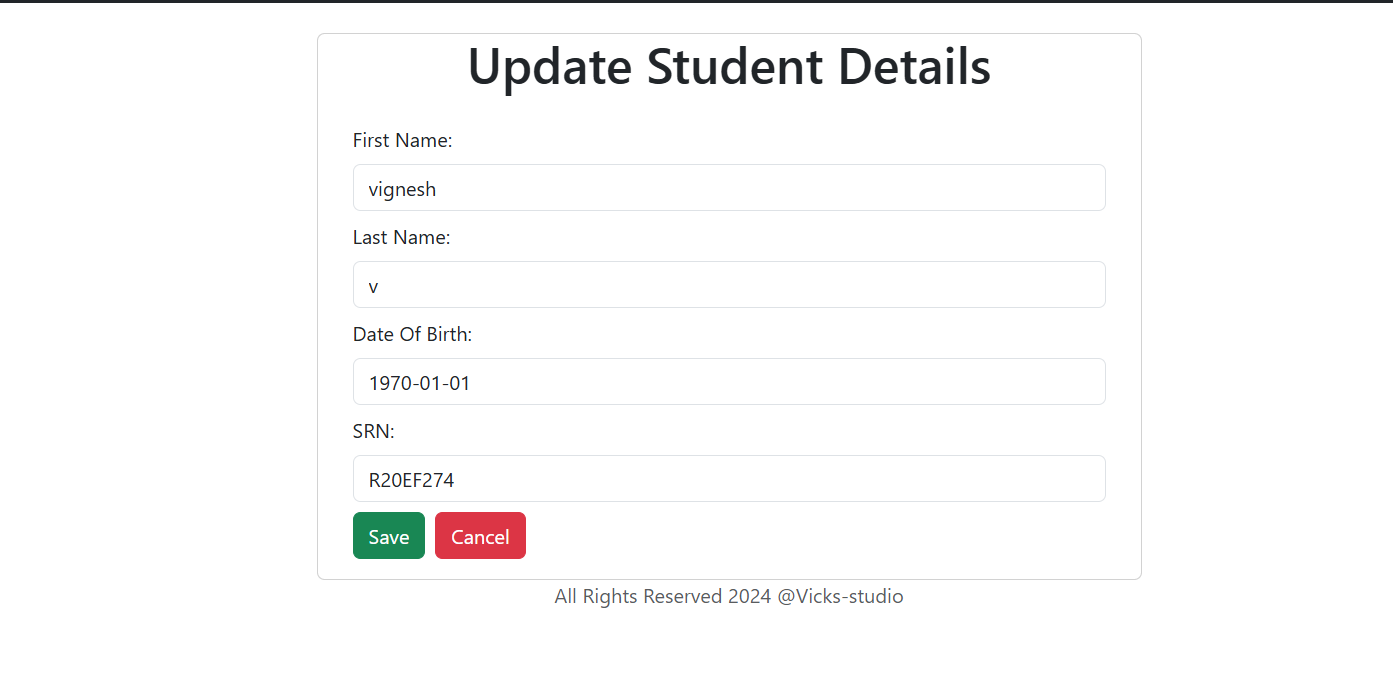
                }

            </tbody>

Step 10:

🡪This step is very crucial because we have 2 webservice for one update button

🡪One just displays the data in update student details after clicking update button



🡪Backend code

@GetMapping("{id}")  
public ResponseEntity<Student> getStudentById(@PathVariable long id){  
 Student student = studentRepository.findById(id).orElseThrow(()->new ResourceNotFoundException("Not found"));  
 return ResponseEntity.*ok*(student);  
  
}

Front end api:

 getStudentById(studid){

            return axios.get(url+"/"+studid)

        }

useEffect code to call api:

const {id} = useParams();

        useEffect(() => {

          Student.getStudentById(id).then((response)=>{

            setFirstName(response.data.firstName)

            setLatName(response.data.lastName)

            setDob(response.data.dob)

            setSrnName(response.data.srn)

          }).catch((error)=>{

            console.log(error)

          })

Front end code:

 const title = () =>{

          if(id){

            return <h1 className='text-center'>Update Student Details</h1>

          }

          else{

            return <h1 className='text-center'>Add Student Details</h1>

          }

        }

  return (

    <div>

      <br></br>

        <div className="container">

          <div className="row">

              <div className="card col-md-6 offset-md-3 offset-md-3">

                  {

                    title()

                  }

                  <div className="card-body">

                    <form>

                      <div className="form-group mb-2">

                        <label className="form-label">First Name:</label>

                        <input type='text' placeholder='Enter your first name' value={firstName} className="form-control" onChange={(e) => setFirstName(e.target.value)}></input>

                      </div>

                      <div className="form-group mb-2">

                        <label className="form-label">Last Name:</label>

                        <input type='text' placeholder='Enter your last name' value={lastName} className="form-control" onChange={(e) => setLatName(e.target.value)}></input>

                      </div>

                      <div className="form-group mb-2">

                        <label className="form-label">Date Of Birth:</label>

                        <input type='text' placeholder='YYYY-MM-DD' className="form-control" value={dob} onChange={(e) => setDob(e.target.value)}></input>

                      </div>

                      <div className="form-group mb-2">

                        <label className="form-label">SRN:</label>

                        <input type='text' placeholder='Enter your SRN' className="form-control" value={srn} onChange={(e) => setSrnName(e.target.value)}></input>

                      </div>

                      <button className="btn btn-success me-2" onClick={(e) => saveStudent(e)}>Save</button>

                      <Link to = "/" className="btn btn-danger">Cancel</Link>

                    </form>

                  </div>

              </div>

          </div>

        </div>

    </div>

  )

Step 11:

🡪Here we will be able to change the values of the records and save them

🡪Example:

A screenshot of a computer

Description automatically generated

Backend code:

@PutMapping("{id}")  
public ResponseEntity<Student> updateEmployeeById(@PathVariable long id,@RequestBody Student employee){  
 Student updateemployee = studentRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Not Found"));  
 updateemployee.setFirstName(employee.getFirstName());  
 updateemployee.setLastName(employee.getLastName());  
 updateemployee.setDob(employee.getDob());  
 updateemployee.setSrn(employee.getSrn());  
 studentRepository.save(updateemployee);  
 return ResponseEntity.*ok*(updateemployee);  
}

🡪This backed code helps to change the values

Front end api:

updateEmployeeById(studid,stud){

            return axios.put(url+"/"+studid,stud)

        }

🡪UI will be same, logic will be different.

Logic of the code is

const history = useNavigate();

const saveStudent = (e) =>{

            e.preventDefault(); //stop the refresh

            const student = {firstName,lastName,dob,srn}

            if(id){

              Student.updateEmployeeById(id,student).then((response)=>{

                history("/")

            }).catch((error)=>{

                console.log(error)

            })

            }

            else{

            Student.createStudent(student).then((response)=>{

                console.log(response.data)

                history("/")

            }).catch((error)=>{

                console.log(error)

            })

          }

        }

The logic of the code is if we click update button it has to show the values of every field and we should be able to modify the code. i.e. if we click and entries, we will be assigned an id. If we get the id, will be able to modify the entries, else will be ending up creating a new entry and direct to the landing page.

Step 12:(Deletion)

Backend code:

@DeleteMapping("{id}")  
public ResponseEntity<HttpStatus> deleteemployeeById(@PathVariable long id){  
 Student employee = studentRepository.findById(id).orElseThrow(() -> new ResourceNotFoundException("Not Found"));  
 studentRepository.delete(employee);  
 return ResponseEntity.*noContent*().build();  
}

Api

deleteemployee(employeeid){

            return axios.delete(url+"/"+employeeid);

        }

Front end code

<button className="btn btn-danger" onClick={() => deleteemployee(stud.id)} style={{marginLeft:"10px"}}>Delete</button>

Logic of code whose main aim is to delete all the content

const getAllEmplouess = () =>{

    Student.getStudent().then((response) =>{

        setfirst(response.data)

    }).catch((error) =>{

        console.log(error)

    })

  }

  const deleteemployee = (empid) => {

    Student.deleteemployee(empid).then((response) =>{

        getAllEmplouess();

    })

}

The code for pop up

const deleteemployee = (empid) => {

    Swal.fire({

        title: 'Are you sure you want to delete?',

        icon: 'warning',

        showCancelButton: true,

        confirmButtonText: 'Yes, delete it!',

        cancelButtonText: 'No, keep it'

    }).then((result) => {

        if (result.isConfirmed) {

            Student.deleteemployee(empid).then((response) => {

                getAllEmplouess();

            });

        }

    });

}

A screenshot of a computer

Description automatically generated

How to upload project in git hub

🡪git init

🡪git add README.md

🡪git commit -m "first commit"

🡪git branch -M main

🡪git remote add origin <https://github.com/Varcoder-123/CURD_Student.git>

error: remote origin already exists.

🡪git push -u origin main

Repository not found.

🡪git remote –verbose

origin https://github.com/Varcoder-123/CURD.git (fetch)

origin https://github.com/Varcoder-123/CURD.git (push)

🡪 git remote set-url origin <https://github.com/Varcoder-123/CURD_Student.git>

🡪git push -u origin main

Successfully project will be added to git hub