Step-By-Step guide for creating a basic Spring Boot Rest API

1. ***Setup Development Environment***
2. ***Create a Spring Boot Project***
3. ***Setting configuration data-source***
4. ***Create Entity Class***
5. ***Create Repository Interface***
6. ***Create Service Interface***
7. ***Create Service Class Implement Service Interface***
8. ***Handler Exception***
9. ***Create Controller Class***

Step 1: Setup Development Environment

Install the following software:

- Java Development Kit (JDK)
- IDE: Eclipse/Intellij
- MariaDB

Step 2: Create a Spring Boot Project

1. Using STS's Eclipse
2. Or Spring boot of Intellij
3. Package name: ***iuh.fit.se***
4. Add dependencies:
   - Spring Web
   - Rest Repository
   - Spring Data JPA
   - Validation
   - MariaDB Driver
   - Spring DevTools

Step 3: Setting configuration data-source

src/main/resources/application.properties

```
# Setting port
server.port=9998

# Setting mariaDB
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.url=jdbc:mariadb://localhost:3306/employees
spring.datasource.username=root
spring.datasource.password=Aa@123456
spring.jpa.hibernate.ddl-auto=none
spring.jpa.show-sql=true

# Setting Spring Rest API
spring.data.rest.base-path=/api
```

Step 4: Create Entity Class

Right-click on the ***"iuh.fit.se"*** package and create a package called entities inside it.

- Enter "***Employee***" as the class name in the "***entities***" package and click on the "Finish" button.

```
package iuh.fit.se.entities;

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

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int id;
```

```
	@Column(name = "first_name")
	@NotNull(message = "First Name must not be Null")
	@NotEmpty(message = "First Name must not be Empty")
	private String firstName;

	@Column(name = "last_name")
	@NotNull(message = "Last Name must not be Null")
	@NotEmpty(message = "Last Name must not be Empty")
	private String lastName;

	private String gender;

	@Column(name = "email")
	@NotEmpty(message = "Email must not be Empty")
	@Email(message = "Email should be valid")
	private String emailAddress;

	@Column(name = "phone_number")
	@Pattern(regexp = "\\(\\d{3}\\)\\d{3}-\\d{4}", message = "Please input phone number with format: (NNN)NNN-NNNN")
	private String phoneNumber;

	@Past(message = "Date of birth must be less than today")
	@DateTimeFormat(pattern = "yyyy-MM-dd")
	private Date dob;

	@CreationTimestamp
	@Temporal(TemporalType.TIMESTAMP)
	@Column(name = "created_date")
	private Date createdDate;

	@UpdateTimestamp
	@Temporal(TemporalType.TIMESTAMP)
	@Column(name = "modified_date")
	private Date modifiedDate;

	@OneToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL, orphanRemoval = true)
	@JoinColumn(name = "address_id", referencedColumnName = "id")
//	@NotNull(message="addresses attributes are required")
//	@Valid
	@JsonIgnore
	private Address address;

	public Employee() {
	}

	public Employee(String firstName, String lastName, String gender, String emailAddress, String phoneNumber, Date dob,
Address address) {
		this.firstName = firstName;
		this.lastName = lastName;
		this.gender = gender;
		this.emailAddress = emailAddress;
		this.phoneNumber = phoneNumber;
		this.dob = dob;
		this.address = address;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getFirstName() {
		return firstName;
	}

	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}

	public String getLastName() {
		return lastName;
	}
```

```
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}

	public String getGender() {
		return gender;
	}

	public void setGender(String gender) {
		this.gender = gender;
	}

	public String getEmailAddress() {
		return emailAddress;
	}

	public void setEmailAddress(String emailAddress) {
		this.emailAddress = emailAddress;
	}

	public String getPhoneNumber() {
		return phoneNumber;
	}

	public void setPhoneNumber(String phoneNumber) {
		this.phoneNumber = phoneNumber;
	}

	public Date getDob() {
		return dob;
	}

	public void setDob(Date dob) {
		this.dob = dob;
	}

	public Date getCreatedDate() {
		return createdDate;
	}

	public void setCreatedDate(Date createdDate) {
		this.createdDate = createdDate;
	}

	public Date getModifiedDate() {
		return modifiedDate;
	}

	public void setModifiedDate(Date modifiedDate) {
		this.modifiedDate = modifiedDate;
	}

	public Address getAddress() {
		return address;
	}

	public void setAddress(Address address) {
		this.address = address;
	}

	@Override
	public String toString() {
		return "Employee [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName + ", gender=" + gender
				+ ", emailAddress=" + emailAddress + ", phoneNumber=" + phoneNumber + ", dob=" + dob + ",
createdDate="
				+ createdDate + ", modifiedDate=" + modifiedDate + "]";
	}

}
```

- Enter “***Address***” as the class name in the “***entities***” package and click on the “Finish” button.

```
package iuh.fit.se.entities;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.*;
import jakarta.validation.constraints.Size;

@Entity
@Table(name = "address")
public class Address {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int id;
	@Size(max = 5, message = "Address cannot exceed 255 characters")
	private String address;

	@OneToOne(mappedBy = "address", fetch = FetchType.LAZY)
	@JsonIgnore
	private Employee employee;

	public Address() {
	}

	public Address(String address) {
		this.address = address;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getAddress() {
		return address;
	}

	public void setAddress(String address) {
		this.address = address;
	}

	public Employee getEmployee() {
		return employee;
	}

	public void setEmployee(Employee employee) {
		this.employee = employee;
	}

	@Override
	public String toString() {
		return "Address [id=" + id + ", address=" + address + "]";
	}

}
```

Step 5: Create Repository Interface

```
package iuh.fit.se.repositories;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

import iuh.fit.se.entities.Employee;

//@Repository
//@ResResoucre
@RepositoryRestResource
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {
            @Query(value = "SELECT e FROM Employee e WHERE e.firstName LIKE  %:keyword%"
          + " OR e.lastName LIKE %:keyword%"
          + " OR e.emailAddress LIKE  %:keyword%"
          + " OR e.phoneNumber LIKE  %:keyword%")
    List<Employee> search(@Param("keyword") String keyword);
}
```

```
package iuh.fit.se.repositories;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;
import iuh.fit.se.entities.Address;

@RepositoryRestResource(collectionResourceRel = "address", path = "address", exported = false)
public interface AddressRepository extends JpaRepository<Address, Integer>{

}
```

Step 6: Create Service Interface

```
package iuh.fit.se.services;

import java.util.List;

import org.springframework.data.domain.Page;

import iuh.fit.se.entities.Employee;

public interface EmployeeService {

        public Employee findById(int id);

        public List<Employee> findAll();

        public Page<Employee> findAllWithPaging(int pageNo, int pageSize, String sortBy, String sortDirection);

        public Employee save(Employee employee);

        public Employee update(int id, Employee employee);

        public boolean delete(int id);

        public List<Employee> search(String keyword);
}
```

```
package iuh.fit.se.services;

import iuh.fit.se.entities.Address;
```

```
public interface AddressService {
    public Address save(Address address);
}
```

Step 7: Create Service Class Implement Service Interface

```
package iuh.fit.se.services.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import iuh.fit.se.entities.Employee;
import iuh.fit.se.exceptions.ItemNotFoundException;
import iuh.fit.se.repositories.EmployeeRepository;
import iuh.fit.se.services.EmployeeService;

@Service
public class EmployeeServiceImpl implements EmployeeService{
	@Autowired
	EmployeeRepository employeeRepository;

	@Override
	public Employee findById(int id) {
		return employeeRepository.findById(id)
					.orElseThrow(()-> new ItemNotFoundException("Can not find Employee with id: " + id));
	}

	@Override
	public List<Employee> findAll() {
		return employeeRepository.findAll();
	}

	@Override
	public Page<Employee> findAllWithPaging(int pageNo, int pageSize, String sortBy, String sortDirection) {
		Sort sort = sortDirection.equalsIgnoreCase(Sort.Direction.ASC.name()) ? Sort.by(sortBy).ascending()
					: Sort.by(sortBy).descending();

		Pageable pageable = PageRequest.of(pageNo, pageSize, sort);
		return employeeRepository.findAll(pageable);
	}

	@Transactional
	@Override
	public Employee save(Employee employee) {
		return employeeRepository.save(employee);
	}

	@Override
	public Employee update(int id, Employee employee) {

		// Check id exists or not
		this.findById(id);

		// Update
		employeeRepository.save(employee);
		return employee;
	}

	@Override
	public boolean delete(int id) {

		Employee employee = this.findById(id);
		employeeRepository.delete(employee);
		return true;
	}
```

```
	@Override
	public List<Employee> search(String keyword) {
		return employeeRepository.search(keyword);
	}
}
```

```
package iuh.fit.se.services.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import iuh.fit.se.entities.Address;
import iuh.fit.se.repositories.AddressRepository;
import iuh.fit.se.services.AddressService;

@Service
public class AddressServiceImpl implements AddressService{
	@Autowired
   private AddressRepository addressRepository;

	@Override
   public Address save(Address address) {
      return this.addressRepository.save(address);
   }
}
```

Step 8: Handler Exception

```
package iuh.fit.se.exceptions;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(value = HttpStatus.NOT_FOUND)
public class ItemNotFoundException extends RuntimeException {

	private static final long serialVersionUID = 1L;

	public ItemNotFoundException(String message) {
		super(message);
	}
}
```

```
package iuh.fit.se.exceptions;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

@ControllerAdvice
public class GlobalExceptionHandler extends ResponseEntityExceptionHandler {

	@ExceptionHandler(ItemNotFoundException.class)
	public ResponseEntity<Map<String, Object>> userNotFoundException(ItemNotFoundException ex) {
		Map<String, Object> errors = new LinkedHashMap<String, Object>();
		errors.put("status", HttpStatus.NOT_FOUND.value());
		errors.put("message", ex.getMessage());
		return new ResponseEntity<Map<String, Object>>(errors, HttpStatus.NOT_FOUND);
	}

	@ExceptionHandler(Exception.class)
	public ResponseEntity<Map<String, Object>> globleExcpetionHandler(Exception ex) {
		Map<String, Object> errors = new LinkedHashMap<String, Object>();
		errors.put("status", HttpStatus.INTERNAL_SERVER_ERROR.value());
		errors.put("message", ex.getMessage());
		return new ResponseEntity<Map<String, Object>>(errors, HttpStatus.INTERNAL_SERVER_ERROR);
```

```
	}
}
```

Step 9: Create Controller Class

```
package iuh.fit.se.controllers;

import java.util.LinkedHashMap;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.rest.webmvc.RepositoryRestController;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestParam;

import iuh.fit.se.entities.Employee;
import iuh.fit.se.services.EmployeeService;
import jakarta.validation.Valid;

@RestController
// @RepositoryRestController
public class EmployeeController {

	@Autowired
	private EmployeeService employeeService;

	@GetMapping("/employees/{id}")
	public ResponseEntity<Map<String, Object>> getEmmployeeById(@PathVariable int id) {
		Map<String, Object> response = new LinkedHashMap<String, Object>();
		response.put("status", HttpStatus.OK.value());
		response.put("data", employeeService.findById(id));
		return ResponseEntity.status(HttpStatus.OK).body(response);
	}

	@PostMapping("/employees")
	public ResponseEntity<Map<String, Object>> saveEmployee(@Valid @RequestBody EmployeeDTO employeeDTO,
			BindingResult bindingResult) {
		Map<String, Object> response = new LinkedHashMap<String, Object>();

		if (bindingResult.hasErrors()) {
			Map<String, Object> errors = new LinkedHashMap<String, Object>();
			bindingResult.getFieldErrors().stream().forEach(result -> {
				errors.put(result.getField(), result.getDefaultMessage());
			});

			System.out.println(bindingResult);
			response.put("status", HttpStatus.BAD_REQUEST.value()); // 400
			response.put("errors", errors);
			return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(response);
		}
		else {
			response.put("status", HttpStatus.OK.value());
			response.put("data", employeeService.save(employee));
			return ResponseEntity.status(HttpStatus.OK).body(response);
		}

	}




	@PutMapping("/employees/{id}")
```

```
	public ResponseEntity<Map<String, Object>> updateEmployee(@PathVariable int id, @Valid @RequestBody Employee
employee, BindingResult bindingResult) {

		Map<String, Object> response = new LinkedHashMap<String, Object>();

		if (bindingResult.hasErrors()) {
			Map<String, Object> errors = new LinkedHashMap<String, Object>();
			bindingResult.getFieldErrors().stream().forEach(result -> {
				errors.put(result.getField(), result.getDefaultMessage());
			});

			response.put("status", HttpStatus.BAD_REQUEST.value());
			response.put("errors", errors);
			return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(response);
		}
		else {
			response.put("status", HttpStatus.OK.value());
			response.put("data", employeeService.update(id, employee));
			return ResponseEntity.status(HttpStatus.OK).body(response);
		}
	}

	@DeleteMapping("/employees/{id}")
	public ResponseEntity<Map<String, Object>> deleteEmployee(@PathVariable int id) {
		Map<String, Object> response = new LinkedHashMap<String, Object>();
		response.put("status", HttpStatus.OK.value());
		response.put("data", employeeService.delete(id));
		return ResponseEntity.status(HttpStatus.OK).body(response);
	}

	@GetMapping("/employees")
	public ResponseEntity<Map<String, Object>> getEmployees(@RequestParam(required = false) String keyword) {

		Map<String, Object> response = new LinkedHashMap<String, Object>();
		response.put("status", HttpStatus.OK.value());

		if (keyword == null || keyword.isEmpty()) {
			response.put("data", employeeService.findAll());
		}
		else {
			response.put("data", employeeService.search(keyword));
		}

		return ResponseEntity.status(HttpStatus.OK).body(response);
	}
}
```

Step 10: The HAL (Hypertext Application Language) Explorer

- Dependency: Rest Repository HAL Explorer

```
<dependency>
	<groupId>org.springframework.data</groupId>
	<artifactId>spring-data-rest-hal-explorer</artifactId>
</dependency>
```

- Access URL: http://localhost:9998/api

Step 11: Springdoc – openapi

- Dependency:

```
<!--https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-starter-webmvc-ui -->
<dependency>
	<groupId>org.springdoc</groupId>
	<artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
	<version>2.6.0</version>
</dependency>
```

- Config springdoc:

```
package iuh.fit.se.configs;

import java.util.List;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import io.swagger.v3.oas.models.OpenAPI;
import io.swagger.v3.oas.models.info.Info;
import io.swagger.v3.oas.models.servers.Server;

@Configuration
public class OpenAPIConfiguration {
	@Bean
	public OpenAPI defineOpenApi() {
		Server server = new Server();
		server.setUrl("http://localhost:9998");
		server.setDescription("Employee Management REST API Documentation");

		Info information = new Info()
				.title("Employee Management REST API Documentation")
				.version("1.0")
				.description("This API exposes endpoints to manage employees.");

		return new OpenAPI().info(information).servers(List.of(server));
	}
}
```

- Add setting springdoc at application.properties

```
# Paths to include
springdoc.pathsToMatch=/**
springdoc.paths-to-exclude=/api/profile/**
springdoc.swagger-ui.operationsSorter=method
```

- Access URL: http://localhost:9998/swagger-ui/index.html

Step 12: Config Logging:

- Create: src/main/resources/logback-spring.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
	<include resource="org/springframework/boot/logging/logback/defaults.xml"/>
	<include resource="org/springframework/boot/logging/logback/console-appender.xml" />
	<include resource="org/springframework/boot/logging/logback/file-appender.xml" />
	<root level="INFO">
		<appender-ref ref="CONSOLE" />
		<appender-ref ref="FILE" />
	</root>
</configuration>
```

- Add setting logging at application.properties

```
# Logging
logging.level.org.springframework.web=debug
logging.level.org.hibernate=error
logging.file.name=logs/myapplication.log
logging.config=classpath:logback-spring.xml
```

- How to Use:

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

private final static Logger logger = LoggerFactory.getLogger(EmployeeController.class.getName());

logger.info("info");
logger.trace("trace");
logger.debug("debug");
logger.warn("warn");
logger.error("error");
```