Spring Tool Suite Link:

<https://cdn.spring.io/spring-tools/release/STS4/4.26.0.RELEASE/dist/e4.33/spring-tool-suite-4-4.26.0.RELEASE-e4.33.0-win32.win32.x86_64.zip>

Java 23: <https://download.oracle.com/java/23/latest/jdk-23_windows-x64_bin.exe>

Video Links:

Cloud Computing: <https://www.youtube.com/watch?v=ae_DKNwK_ms>

Microservices: <https://www.youtube.com/watch?v=ji2MNG5ix44>

Containers: <https://www.youtube.com/watch?v=gFozhTXOx18>

Cloud Native: <https://www.youtube.com/watch?v=9Ik96SBaIvs>

Service Mesh: <https://www.youtube.com/watch?v=2uZmxbGnEqo>

package com.ibm;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ConfigurableApplicationContext;

import com.ibm.model.MyHello;

@SpringBootApplication

public class HelloApplication implements CommandLineRunner{

@Autowired

MyHello mybean;

public static void main(String[] args) {

**ConfigurableApplicationContext context = SpringApplication.run(HelloApplication.class, args);**

**System.out.println("Number of Beans:"+ context.getBeanDefinitionCount());**

**System.out.println("==========Beans=========");**

**for (String bean : context.getBeanDefinitionNames()) {**

**System.out.println(bean);**

**}**

}

@Override

public void run(String... args) throws Exception {

System.out.println(" Printing : "+ mybean.getMessage());

}

}

**MyHello.java**

package com.ibm.model;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Component;

@Component

public class MyHello {

public MyHello() {

System.out.println("MyHello object created!" + this.message);

}

@Value("${message}")

private String message;

public String getMessage() {

return message;

}

}

application.properties

message=This is my IBM Message

======================================================================

java -jar hello-0.0.1-SNAPSHOT.jar

java -jar **-Dmessage="This ROcks!"** hello-0.0.1-SNAPSHOT.jar

===== Profiles ======

application.properties

message=Hi IBM!

application-dev.properties

message=Hi IBM from DEV!

application-test.properties

message=Hi IBM from TEST!

>**java -Dspring.profiles.active=dev -jar hello-0.0.1-SNAPSHOT.jar**

**java "-Dspring.profiles.active=dev" -jar hellospring-0.0.1-SNAPSHOT.jar**

application.yaml

message: Hello IBM!!

---

spring.config.activate.on-profile: dev

message: Hello IBM!! -- DEV

---

spring.config.activate.on-profile: prod

message: Hello IBM!! -- PROD

Download HTTPie: <https://github.com/httpie/desktop/releases/download/v2023.3.6/HTTPie-Setup-2023.3.6.exe>

Create Hello web project and add “Web starter”

For a quick Test

**package** com.ibm.controller;

**import** java.time.LocalDateTime;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** org.springframework.web.bind.annotation.RestController;

@RestController

**public** **class** HelloController {

@GetMapping("hello")

**public** String sayHello() {

**return** "Hello Web! => " + LocalDateTime.*now*();

}

@GetMapping("hellobyname")

**public** String sayHelloByMessage(@RequestParam String name) {

**return** "Hello " + name;

}

@GetMapping("hello/{name}")

**public** String sayHelloByName(@PathVariable String name) {

**return** "Hello " + name;

}

}

Test the below URLs in POSTMAN

localhost:8080/hello

localhost:8080/hellobyname?name=Raj

localhost:8080/hello/Siva

========================================================

package com.ibm.controller;

import java.util.Date;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

import com.ibm.model.Hello;

@RestController

public class HelloController {

@GetMapping("hello")

public String sayHello()

{

return "Hello Web!";

}

@GetMapping("helloobj")

public Hello sayHelloObject()

{

return new Hello("Hi IBM", new Date());

}

**@PostMapping("helloobj")**

**public Hello createHelloObject(@RequestBody Hello hello)**

**{**

**System.*out*.println(hello);**

**return hello;**

**}**

}

Hello.java

package com.ibm.model;

import java.util.Date;

public class Hello {

private String message;

private Date timestamp;

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

public Date getTimestamp() {

return timestamp;

}

public void setTimestamp(Date timestamp) {

this.timestamp = timestamp;

}

@Override

public String toString() {

return "Hello [message=" + message + ", timestamp=" + timestamp + "]";

}

public Hello() {

}

public Hello(String message, Date timestamp) {

super();

this.message = message;

this.timestamp = timestamp;

}

}

With Lombok

**package** com.ibm.model;

**import** java.util.Date;

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.NoArgsConstructor;

@NoArgsConstructor

@AllArgsConstructor

@Data

**public** **class** Hello {

**private** String message;

**private** Date timestamp;

}

package com.ibm.controller;

import java.util.Date;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import com.ibm.model.Hello;

@RestController

public class HelloController {

@GetMapping("hello")

public String sayHello()

{

return "Hello Web!";

}

**@GetMapping("helloobj")**

**public Hello sayHelloObject()**

**{**

**return new Hello("Hi IBM", new Date());**

**}**

**@GetMapping("hellobymessage")**

**public Hello sayHelloByMessage(@RequestParam String message)**

**{**

**return new Hello("Hi "+message, new Date());**

**}**

**@GetMapping("hello/{name}")**

**public String sayHelloByName(@PathVariable String name)**

**{**

**return "Hello "+name;**

**}**

}

localhost:9000/hellobymessage?message=Siva

**application.properties**

logging.level.org.springframework.web=trace

server.port=9000

In case of Java 11:

JAXB Dependencies:

<!-- https://mvnrepository.com/artifact/com.sun.istack/istack-commons-runtime -->

<dependency>

<groupId>com.sun.istack</groupId>

<artifactId>istack-commons-runtime</artifactId>

<version>3.0.8</version>

</dependency>

<!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

</dependency>

<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-impl -->

<dependency>

<groupId>com.sun.xml.bind</groupId>

<artifactId>jaxb-impl</artifactId>

<version>2.4.0-b180830.0438</version>

</dependency>

**For Spring boot 3.2.2 and java 21**

**<!-- https://mvnrepository.com/artifact/com.sun.xml.bind/jaxb-impl -->**

**<dependency>**

**<groupId>com.sun.xml.bind</groupId>**

**<artifactId>jaxb-impl</artifactId>**

**</dependency>**

**Hello.java**

package com.ibm.model;

import java.util.Date;

import javax.xml.bind.annotation.XmlRootElement;

**@XmlRootElement**

public class Hello {

private String message;

private Date timestamp;

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

public Date getTimestamp() {

return timestamp;

}

public void setTimestamp(Date timestamp) {

this.timestamp = timestamp;

}

@Override

public String toString() {

return "Hello [message=" + message + ", timestamp=" + timestamp + "]";

}

public Hello() {

}

public Hello(String message, Date timestamp) {

super();

this.message = message;

this.timestamp = timestamp;

}

}

**Using JACKSON XML binding**

**<dependency>**

**<groupId>com.fasterxml.jackson.dataformat</groupId>**

**<artifactId>jackson-dataformat-xml</artifactId>**

**</dependency>**

**Testing REST Controller**

package com.ibm.controller;

import static org.junit.jupiter.api.Assertions.assertEquals;

**import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.content;**

**import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;**

import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;

import org.springframework.web.client.RestTemplate;

**import org.springframework.http.MediaType;**

@SpringBootTest

**@AutoConfigureMockMvc**

public class HelloControllerTest {

@Autowired

MockMvc mock;

// @Test //unit testing

// public void testHelloByName()

// {

// RestTemplate rt = new RestTemplate();

//

// String result = rt.getForObject("http://localhost:9000/hello/Vikram", String.class);

// System.out.println(result);

// assertEquals(result, "Hello vikram");

// }

@Test //mock testing

public void testHelloByName() throws Exception

{

mock.perform(MockMvcRequestBuilders.get("/hello/Vikram"))

.andExpect(status().isOk())

**.andDo(print())**

.andExpect(content().string("Hello Vikram"));

}

**@Test //mock testing**

**public void testNotFound() throws Exception**

**{**

**mock.perform(MockMvcRequestBuilders.get("/hello123"))**

**.andExpect(status().isNotFound());**

**}**

**@Test**

**public void testSayHelloObj() throws Exception {**

**mock.perform(MockMvcRequestBuilders.get("/helloobj")**

**.accept(MediaType.APPLICATION\_XML))**

**.andDo(print())**

**.andExpect(status().is(200));**

**//.andExpect(content().string("Hello world"));**

**}**

}

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-validation</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

Todo REST API Design

GET /todos => 200 OK

GET /todos/{id} => 200 OK / 404 NOT FOUND

POST /todos => 201 CREATED

PUT /todos/{id} => 200 OK / 404 NOT FOUND

DELETE /todos/{id} => 204 NO CONTENT / 404 NOT FOUND

GET /todos/user/{user} => 200 OK

GET /todos/date/{targetdate} => 200 OK

TODO API

Todo.java

package com.ibm.model;

import java.time.LocalDate;

import java.util.UUID;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Size;

@Entity

public class Todo {

@Id

**@GeneratedValue(strategy = GenerationType.IDENTITY)**

private UUID id;

**@Column(name = "TODOUSER")**

@NotBlank(message = "User cannot be null or empty")

private String user;

@Size(min = 3, max = 200, message = "Task must be b/w 3-200 characters")

private String task;

@Column(name = "TARGETDATE")

private LocalDate targetDate;

private boolean done;

public Todo() {

// TODO Auto-generated constructor stub

}

public UUID getId() {

return id;

}

public void setId(UUID id) {

this.id = id;

}

public String getUser() {

return user;

}

public void setUser(String user) {

this.user = user;

}

public String getTask() {

return task;

}

public void setTask(String task) {

this.task = task;

}

public LocalDate getTargetDate() {

return targetDate;

}

public void setTargetDate(LocalDate targetDate) {

this.targetDate = targetDate;

}

public boolean isDone() {

return done;

}

public void setDone(boolean done) {

this.done = done;

}

public Todo(UUID id, String user, String task, LocalDate targetDate, boolean done) {

super();

this.id = id;

this.user = user;

this.task = task;

this.targetDate = targetDate;

this.done = done;

}

@Override

public String toString() {

return "Todo [id=" + id + ", user=" + user + ", task=" + task + ", targetDate=" + targetDate + ", done=" + done

+ "]";

}

}

========================= with lombok ================================

**package** com.ibm.model;

**import** java.time.LocalDate;

**import** java.util.UUID;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.Id;

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.NoArgsConstructor;

@AllArgsConstructor

@NoArgsConstructor

@Data

@Entity

**public** **class** Todo {

@Id

@GeneratedValue

**private** UUID id;

@Column(name = "TODOUSER")

**private** String user;

**private** String task;

@Column(name = "TARGETDATE")

**private** LocalDate targetDate;

**private** **boolean** done;

}

**application.properties**

server.port=8080

spring.jpa.show-sql=true

spring.h2.console.enabled=true

spring.application.name=Todo-app

logging.level.org.springframework.web=trace

spring.datasource.url=jdbc:h2:mem:tododb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.mvc.format.date=yyyy-MM-dd

**Link:** [**http://localhost:8080/h2-console**](http://localhost:8080/h2-console)

**JDBC URL: jdbc:h2:mem:tododb**

=====================================================================================

package com.ibm;

import java.time.LocalDate;

import java.util.UUID;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import com.ibm.model.Todo;

import com.ibm.repository.TodoRepository;

@SpringBootApplication

public class **TodoApplication** implements CommandLineRunner{

@Autowired

TodoRepository todoRepo;

public static void main(String[] args) {

SpringApplication.run(TodoApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

Todo todo = new Todo(UUID.randomUUID(), "siva", "Learn Spring boot", LocalDate.now(), false);

todoRepo.save(todo);

}

}

package com.ibm.repository;

import java.time.LocalDate;

import java.util.List;

import java.util.UUID;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.ibm.model.Todo;

**@Repository**

public interface **TodoRepository** extends JpaRepository<Todo, UUID>{

public List<Todo> findTodosByUser(String user);

public List<Todo> findTodosByTargetDate(LocalDate date);

}

==============\*\*\*\*\*\*\*\*\*\*\*==================

For h2 e.x there is an problem with UUID

<properties>

<java.version>11</java.version>

<h2.version>1.4.200</h2.version>

</properties>

TodoControoler.java

package com.ibm.controller;

import java.net.URI;

import java.time.LocalDate;

import java.util.List;

import java.util.UUID;

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

import org.springframework.http.ResponseEntity;

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.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import com.ibm.model.Todo;

import com.ibm.service.TodoService;

**@RestController**

**@RequestMapping("/todos")**

**public class TodoController {**

**@Autowired**

**TodoService todoService;**

**@GetMapping**

**public List<Todo> getTodos()**

**{**

**return todoService.getTodos();**

**}**

**@GetMapping("/{id}")**

**public Todo getTodoById(@PathVariable UUID id)**

**{**

**return todoService.getTodoById(id);**

**}**

@GetMapping("/user/{user}")

public List<Todo> getTodosByUser(@PathVariable String user)

{

return todoService.getTodosByUser(user);

}

@GetMapping("/date/{targetdate}")

public List<Todo> getTodosByUser(@PathVariable LocalDate targetdate)

{

return todoService.getTodosByTargetDate(targetdate);

}

@PostMapping

public ResponseEntity<Todo> addTodo(@RequestBody Todo todo)

{

**Todo result = todoService.addTodo(todo);**

**URI location = ServletUriComponentsBuilder**

**.fromCurrentRequest()**

**.path("/{id}")**

**.buildAndExpand(result.getId()).toUri();**

**return ResponseEntity.created(location)**

**.body(result);**

}

**@PutMapping("/{id}")**

**public ResponseEntity<Todo> updateTodo(@PathVariable UUID id, @RequestBody Todo todo)**

**{**

**Todo result = todoService.updateTodo(todo);**

**URI location = ServletUriComponentsBuilder**

**.fromCurrentRequest()**

**.path("/{id}")**

**.buildAndExpand(result.getId()).toUri();**

**return ResponseEntity.created(location)**

**.body(result);**

**}**

**@DeleteMapping("/{id}")**

**public ResponseEntity<?> deleteTodoById(@PathVariable UUID id)**

**{**

**todoService.deleteTodo(id);**

**return ResponseEntity.noContent().build();**

}

}

TodoService.java

package com.ibm.service;

import java.time.LocalDate;

import java.util.List;

import java.util.UUID;

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

import org.springframework.stereotype.Service;

import com.ibm.common.exceptions.TodoNotFoundException;

import com.ibm.model.Todo;

import com.ibm.repository.TodoRepository;

**@Service**

**public class TodoService {**

**@Autowired**

**TodoRepository todoRepo;**

**public List<Todo> getTodos()**

**{**

**return todoRepo.findAll();**

**}**

**public Todo getTodoById(UUID id) {**

**return todoRepo.findById(id)**

**.orElseThrow(TodoNotFoundException::new);**

**}**

public List<Todo> getTodosByUser(String user)

{

return todoRepo.findTodosByUser(user);

}

public List<Todo> getTodosByTargetDate(LocalDate targetDate)

{

return todoRepo.findTodosByTargetDate(targetDate);

}

public Todo addTodo(Todo todo){

return todoRepo.saveAndFlush(todo);

}

public Todo updateTodo(Todo todo) {

getTodoById(todo.getId());

return todoRepo.saveAndFlush(todo);

}

public boolean deleteTodo(UUID id) {

getTodoById(id);

todoRepo.deleteById(id);

return true;

}

}

Updated Todo.java for validation

package com.ibm.model;

import java.time.LocalDate;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.Size;

public class Todo {

private int id;

**@NotBlank(message = "User cannot be null or empty")**

**private String user;**

**@Size(min = 3, max = 200, message = "Task must be b/w 3-200 characters")**

**private String task;**

**@FutureOrPresent(message = "Target date can't be in past")**

private LocalDate targetDate;

private boolean done;

}

@PostMapping

public ResponseEntity<Todo> createTodo(**@Valid** @RequestBody Todo todo)

{

Todo result = todoService.createTodo(todo);

URI location = ServletUriComponentsBuilder

.fromCurrentRequest()

.path("/{id}")

.buildAndExpand(result.getId()).toUri();

return ResponseEntity.created(location).body(result);

}

**package** com.ibm.common.exceptions;

**import** java.time.LocalDateTime;

**import** java.util.List;

**import** org.springframework.validation.FieldError;

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.NoArgsConstructor;

import com.fasterxml.jackson.annotation.JsonInclude;

import com.fasterxml.jackson.annotation.JsonInclude.Include;

@AllArgsConstructor

@NoArgsConstructor

@Data

**public** **class** ErrorDetails {

**private** String errorCode;

**private** String errorDetails;

**private** LocalDateTime timestamp;

@JsonInclude(Include.NON\_NULL)

**private** List<FieldError> errorFields;

}

**package** com.ibm.controller;

**import** java.time.LocalDateTime;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.MethodArgumentNotValidException;

**import** org.springframework.web.bind.annotation.ExceptionHandler;

**import** org.springframework.web.bind.annotation.RestControllerAdvice;

**import** com.ibm.common.ErrorDetails;

**import** com.ibm.common.TodoNotFoundException;

@RestControllerAdvice

**public** **class** TodoControllerAdvice {

@ExceptionHandler(TodoNotFoundException.**class**)

**public** ResponseEntity<ErrorDetails> handleTodoNotFound(TodoNotFoundException e)

{

ErrorDetails ed = **new** ErrorDetails("Todo-404", "Todo Not Found!", LocalDateTime.*now*(),**null**);

**return** ResponseEntity.*status*(HttpStatus.***NOT\_FOUND***).body(ed);

}

**@ExceptionHandler(MethodArgumentNotValidException.class)**

**public ResponseEntity<ErrorDetails> handleInvalidTodo(MethodArgumentNotValidException e)**

**{**

**ErrorDetails ed = new ErrorDetails("Todo-400", "Todo Not Found!", LocalDateTime.*now*(), e.getFieldErrors());**

**return ResponseEntity.*status*(HttpStatus.*BAD\_REQUEST*).body(ed);**

**}**

}

To document REST API we can use Spring doc library

[OpenAPI 3 Library for spring-boot (springdoc.org)](https://springdoc.org/)

<dependency>

<groupId>org.springdoc</groupId>

<artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>

<version>2.6.0</version>

</dependency>

Goto: <http://localhost:8080/swagger-ui.html>

package com.ibm.controller.advice;

import java.util.Date;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.ExceptionHandler;

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

import org.springframework.web.bind.annotation.RestControllerAdvice;

import com.ibm.exception.TodoIDMisMatchException;

import com.ibm.exception.TodoNotFoundException;

@RestControllerAdvice

public class TodoControllerAdvice {

@ExceptionHandler(TodoNotFoundException.class)

@ResponseStatus(code = HttpStatus.NOT\_FOUND)

public ResponseEntity<ErrorResponse> handleTodoNotFound(TodoNotFoundException e)

{

return ResponseEntity

.status(HttpStatus.NOT\_FOUND)

.body(new ErrorResponse("todo-404", "Todo not found", new Date()));

}

@ExceptionHandler(TodoIDMisMatchException.class)

@ResponseStatus(code = HttpStatus.BAD\_REQUEST)

public ResponseEntity<ErrorResponse> handleTodoIDMismatch(TodoIDMisMatchException e)

{

System.out.println(e.getMessage());

return ResponseEntity

.status(HttpStatus.BAD\_REQUEST)

.body(new ErrorResponse("todo-400", "Todo ID mismatch", new Date()));

}

}

Todo Application Class

@Bean

public OpenAPI springTodoOpenAPI() {

return new OpenAPI()

.info(new Info().title("Todo API")

.description("Todo sample application")

.version("v0.0.1")

.license(new License().name("Apache 2.0").url("http://springdoc.org")))

.externalDocs(new ExternalDocumentation()

.description("Todo Wiki Documentation")

.url("https://Todo.wiki.github.org/docs"));

}

Query By Example

<https://docs.spring.io/spring-data/jpa/reference/repositories/query-methods-details.html>

TodoRepository.java

package com.ibm.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.ibm.model.Todo;

@Repository

public interface TodoRepository extends JpaRepository<Todo, Integer> {

**List<Todo> findAllByUser(String username);**

}

In todo service

**public List<Todo> getTodosByUser(String user)**

**{**

**return todoRepo.findAllByUser(user);**

**}**

In controller

**@GetMapping("/user/{user}")**

**public ResponseEntity<List<Todo>> getTodosByUser(@PathVariable String user){**

**return ResponseEntity.ok(todoService.getTodosByUser(user));**

**}**

Employee **CRUD** API with H2 database

**Empid, name, age[18-65], mobile, aadhar [Pattern validation], Designation**

**Person, Employee extends Person and then create a REST API**

**@MappedSuperClass**

Adding Actuator

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

application.properties

**management.endpoints.web.exposure.include=\***

Link: <http://localhost:8080/actuator>

<http://localhost:9000/actuator/metrics/jvm.memory.used>

Custom End point for endpoint

**package com.ibm.controller;**

**import org.springframework.boot.actuate.endpoint.web.annotation.RestControllerEndpoint;**

**import org.springframework.http.HttpStatus;**

**import org.springframework.http.ResponseEntity;**

**import org.springframework.stereotype.Component;**

**import org.springframework.web.bind.annotation.GetMapping;**

**import org.springframework.web.bind.annotation.ResponseBody;**

**@Component**

**@RestControllerEndpoint(id = "rest-end-point")**

**public class RestCustomEndPoint {**

**@GetMapping("/custom")**

**public @ResponseBody ResponseEntity<?> customEndPoint(){**

**return new ResponseEntity<>("REST end point", HttpStatus.OK);**

**}**

**}**

To only allow info and health

management.endpoints.web.exposure.include=info,health

management.info.env.enabled = true

info.app.name=Todo API Application

info.app.description=This is my first spring boot application

info.app.version=1.0.0

Replace the server:

<https://docs.spring.io/spring-boot/docs/2.1.13.RELEASE/reference/html/howto-embedded-web-servers.html>

CORS: [Getting Started | Enabling Cross Origin Requests for a RESTful Web Service (spring.io)](https://spring.io/guides/gs/rest-service-cors/)

Index.html

<html>

<head>

<script type="text/javascript">

function getTodos(){

var requestOptions = {

method: 'GET',

redirect: 'follow'

};

fetch("http://localhost:8080/todos", requestOptions)

.then(response => response.text())

.then(result => console.log(result))

.catch(error => console.log('error', error));

}

</script>

</head>

<body>

<button onclick="getTodos()">Click</button>

</body>

</html>

<https://martinfowler.com/articles/richardsonMaturityModel.html>

Todo DATA REST

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

**<dependency>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-data-rest</artifactId>**

**</dependency>**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-validation</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

**<dependency>**

**<groupId>org.springframework.data</groupId>**

**<artifactId>spring-data-rest-hal-explorer</artifactId>**

**</dependency>**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

application.properties

*server.port=8080*

*spring.jpa.show-sql=true*

*spring.h2.console.enabled=true*

*spring.application.name=Todo-rest*

*spring.datasource.url=jdbc:h2:mem:todorestdb*

*spring.datasource.driverClassName=org.h2.Driver*

*spring.datasource.username=sa*

*spring.jpa.database-platform=org.hibernate.dialect.H2Dialect*

*management.endpoints.web.exposure.include=\**

*TodoRepository.java*

*package com.ibm.repository;*

*import java.util.List;*

*import org.springframework.data.jpa.repository.JpaRepository;*

*import org.springframework.data.rest.core.annotation.RepositoryRestResource;*

*import org.springframework.stereotype.Repository;*

*import com.ibm.model.Todo;*

***@RepositoryRestResource****(collectionResourceRel = "todos", itemResourceRel = "todo")*

*@Repository*

*public interface TodoRepository extends JpaRepository<Todo, Integer> {*

*List<Todo> findAllByUser(String username);*

*}*

*package com.ibm;*

*import org.springframework.data.rest.core.event.AbstractRepositoryEventListener;*

*import org.springframework.stereotype.Component;*

*import com.ibm.model.Todo;*

*@Component*

*public class BeforeSaveEventListener extends AbstractRepositoryEventListener<****Todo****> {*

*@Override*

*protected void onAfterCreate(Todo entity) {*

*// TODO Auto-generated method stub*

*super.onAfterCreate(entity);*

*System.out.println("========================");*

*System.out.println(entity);*

*System.out.println("========================");*

*}*

*}*

***Install visual studio code from:*** [***Visual Studio Code - Code Editing. Redefined***](https://code.visualstudio.com/)

***We need to add one extension***

1. ***Docker by Microsoft***

******

Docker

**docker info**

**docker**

A self-sufficient runtime for containers

**docker ps**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

=========

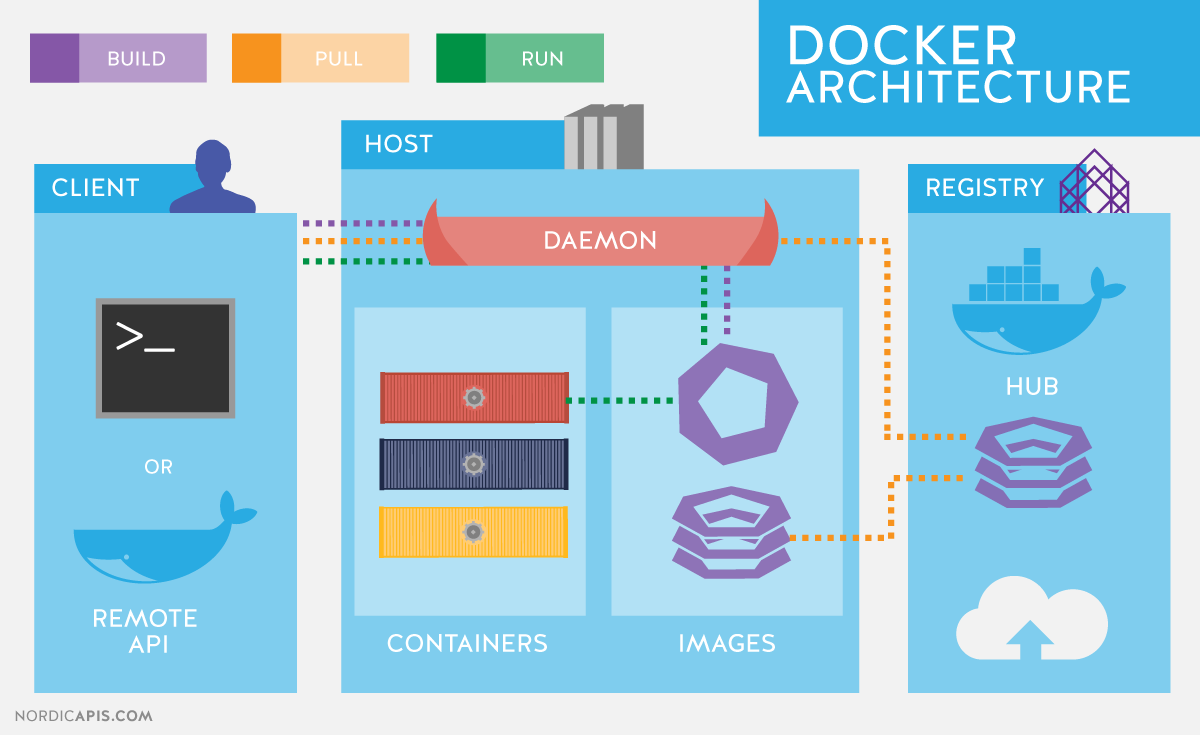
**docker run hello-world**

**.\mvnw spring-boot:build-image**

**docker run -d -p5000:8080 todoapp:0.0.1-SNAPSHOT**

**docker ps**

**docker ps -a**



[Nginx - Official Image | Docker Hub](https://hub.docker.com/_/nginx)

**docker pull nginx**

**docker pull shivaspk/todorest**

**docker run --name ibmnginx -p 5000:80 -d nginx**

docker run --name ibmnginx1 -p 5001:80 -d nginx

docker run --name ibmnginx2 -p 5002:80 -d nginx

docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

95213054e8f9 nginx "/docker-entrypoint.…" 5 minutes ago Up 5 minutes 0.0.0.0:5000->80/tcp, :::5000->80/tcp ibmnginx

**docker exec -it ibmnginx bash**

**ls**

**echo "<h1>hello IBM</h1>" > /usr/share/nginx/html/index.html**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset='utf-8'>**

**<meta http-equiv='X-UA-Compatible' content='IE=edge'>**

**<title>My IBM</title>**

**<meta name='viewport' content='width=device-width, initial-scale=1'>**

**</head>**

**<body>**

**<h1> IBM Microservices!</h1>**

**</body>**

**</html>**

docker inspect nginx

docker container inspect ibmnginx

# for RabbitMQ 3.13, the latest series

**docker run -it --name rabbitmq -p 5672:5672 -p 15672:15672 -d rabbitmq:3.13-management**

Link: [RabbitMQ Management](http://localhost:15672/)

guest

guest

package com.ibm.controller;

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

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import com.ibm.service.HelloService;

@RestController

public class HelloController {

@Autowired

HelloService helloService;

@GetMapping

public void publish(@RequestParam String queuename, @RequestParam String message)

{

System.out.println(queuename);

System.out.println(message);

helloService.publish(queuename, message);

}

}

**HelloService.java**

package com.ibm.service;

import org.springframework.amqp.rabbit.core.RabbitTemplate;

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

import org.springframework.stereotype.Service;

@Service

public class HelloService {

@Autowired

RabbitTemplate rabbitTemplate;

public void publish(String queuename, String message )

{

rabbitTemplate.convertAndSend(queuename, message);

}

}

http://**localhost:8080?queuename=tasks&message=Testing Rabbitmq**

**application.properties**

server.port=9000

spring.rabbitmq.host=localhost

spring.rabbitmq.port=5672

spring.rabbitmq.username=guest

spring.rabbitmq.password=guest

spring.rabbitmq.virtual-host=/

Consumer:

package com.ibm;

import org.springframework.amqp.rabbit.annotation.RabbitListener;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class RabbitmqconsumerApplication {

public static void main(String[] args) {

SpringApplication.run(RabbitmqconsumerApplication.class, args);

}

@RabbitListener(queues = "tasks")

public void consume(String message)

{

System.out.println("=========\*\*\*\*=========");

System.out.println(message);

System.out.println("=========\*\*\*\*=========");

}

}

Create your own Image

Dockerfile

FROM busybox

CMD [ "date" ]

**docker build -t <<username>>/ibmdate .**

**docker build -t shivaspk/ibmdate .**

docker run shivaspk/ibmdate

docker login -u shivaspk

docker push **shivaspk/ibmdate**

In Spring Boot they have added a Docker support without any docker file from your project folder

>**mvnw.cmd spring-boot:build-image**

Run As > Maven Install and then run Docker build

Custom Dockerfile

*FROM adoptopenjdk:16.0.1\_9-jre-openj9-0.26.0*

*ADD target/todorest-0.0.1-SNAPSHOT.jar todorest.jar*

*EXPOSE 9000*

*ENTRYPOINT [ "java", "-jar" , "todorest.jar"]*

=========

**FROM eclipse-temurin:11-jre-alpine**

**ADD target/todoapp-0.0.1-SNAPSHOT.jar todo.jar**

**EXPOSE 8080**

**ENTRYPOINT [ "java", "-jar" , "todo.jar"]**

========

FROM **eclipse-temurin:17.0.10\_7-jre-alpine**

ADD target/todoapp-0.0.1-SNAPSHOT.jar todo.jar

EXPOSE 8080

ENTRYPOINT [ "java", "-jar" , "todo.jar"]

====

FROM **eclipse-temurin:23-jre-alpine**

ADD target/todoapp-0.0.1-SNAPSHOT.jar todo.jar

EXPOSE 8080

ENTRYPOINT [ "java", "-jar" , "todo.jar"]

docker build -t shivaspk/ibmtodo .

docker login

docker tag mytodrest <<your name>>/todorest

docker push <<your name>>/ibmtodo:latest

docker run -p 9080:8080 -d shivaspk/ibmtodo:2.7.2

<https://www.docker.com/get-started/>

Persist Data in Volumes!

docker volume create mydata

**docker run --name ibmmysql -v mydata:/var/lib/mysql -p3306:3306 -e MYSQL\_ROOT\_PASSWORD=rootroot -e MYSQL\_DATABASE=tododb -d mysql**

Create a HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Hell Container</title>

</head>

<body>

<h1>This is from HOST Machine!!!</h1>

<h3>We rock!</h3>

</body>

</html>

docker run -v C:/Users/vmware19/Downloads/html/mydata:/usr/share/nginx/html -p 7500:80 -d nginx

Add mysql dependency

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

application.properties

server.port=9000

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://localhost:3306/tododb

spring.datasource.username=root

spring.datasource.password=rootroot

spring.datasource.driver-class-name =com.mysql.cj.jdbc.Driver

spring.application.name=Todo-rest

management.endpoints.web.exposure.include=\*

**Employee API**

**package com.ibm.model;**

**import javax.persistence.Embedded;**

**import javax.persistence.MappedSuperclass;**

**import javax.validation.constraints.NotBlank;**

**import lombok.AllArgsConstructor;**

**import lombok.Data;**

**import lombok.NoArgsConstructor;**

**@NoArgsConstructor**

**@AllArgsConstructor**

**@Data**

**@MappedSuperclass**

**public class Person {**

**@NotBlank**

**private String name;**

**private int age;**

**@Embedded**

**private Address address;**

**}**

**package com.ibm.model;**

**import lombok.AllArgsConstructor;**

**import lombok.Data;**

**import lombok.NoArgsConstructor;**

**@NoArgsConstructor**

**@AllArgsConstructor**

**@Data**

**public class Address {**

**private String street;**

**private String city;**

**private int pincode;**

**}**

**package com.ibm.model;**

**import javax.persistence.Entity;**

**import javax.persistence.GeneratedValue;**

**import javax.persistence.Id;**

**import lombok.AllArgsConstructor;**

**import lombok.Data;**

**import lombok.EqualsAndHashCode;**

**import lombok.NoArgsConstructor;**

**@NoArgsConstructor**

**@AllArgsConstructor**

**@Data**

**@EqualsAndHashCode(callSuper = true)**

**@Entity**

**public class Employee extends Person {**

**@Id**

**@GeneratedValue**

**private int empid;**

**private String department;**

**}**

**package com.ibm.service;**

**import java.util.List;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Service;**

**import com.ibm.model.Employee;**

**import com.ibm.repository.EmployeeRepository;**

**@Service**

**public class EmployeeService {**

**@Autowired**

**EmployeeRepository employeeRepository;**

**public List<Employee> getEmployees(){**

**return employeeRepository.findAll();**

**}**

**public Employee addEmployee(Employee employee) {**

**return employeeRepository.save(employee);**

**}**

**}**

**package com.ibm.repository;**

**import org.springframework.data.jpa.repository.JpaRepository;**

**import com.ibm.model.Employee;**

**public interface EmployeeRepository extends JpaRepository<Employee, Integer> {**

**}**

**package com.ibm.service;**

**import java.util.List;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Service;**

**import com.ibm.model.Employee;**

**import com.ibm.repository.EmployeeRepository;**

**@Service**

**public class EmployeeService {**

**@Autowired**

**EmployeeRepository employeeRepository;**

**public List<Employee> getEmployees(){**

**return employeeRepository.findAll();**

**}**

**public Employee addEmployee(Employee employee) {**

**return employeeRepository.save(employee);**

**}**

**}**

**Any new concept of Spring Boot ever :** [**Spring | Guides**](https://spring.io/guides)

**Let’s try a simple mono and flux**

[**Getting Started | Building a Reactive RESTful Web Service (spring.io)**](https://spring.io/guides/gs/reactive-rest-service/)

**Our example**

**package com.ibm.model;**

**import lombok.AllArgsConstructor;**

**import lombok.Data;**

**import lombok.NoArgsConstructor;**

**@NoArgsConstructor**

**@AllArgsConstructor**

**@Data**

**public class Greeting {**

**private String message;**

**}**

**package com.ibm.controller;**

**import org.springframework.http.ResponseEntity;**

**import org.springframework.web.bind.annotation.GetMapping;**

**import org.springframework.web.bind.annotation.RestController;**

**import com.ibm.model.Greeting;**

**import reactor.core.publisher.Flux;**

**import reactor.core.publisher.Mono;**

**@RestController**

**public class GreetingController {**

**@GetMapping("/hello")**

**public ResponseEntity<Mono<Greeting>> hello() {**

**return ResponseEntity.*ok*(Mono.*just*(new Greeting("Hello Reactive!!!")));**

**}**

**@GetMapping("/hellos")**

**public ResponseEntity<Flux<Greeting>> hellos() {**

**return ResponseEntity.*ok*(Flux.*just*(new Greeting("Hello Reactive!!!"),new Greeting("Hello Reactive!!!"),new Greeting("Hello Reactive!!!"),new Greeting("Hello Reactive!!!"),new Greeting("Hello Reactive!!!"),new Greeting("Hello Reactive!!!"),new Greeting("Hello Reactive!!!")));**

**}**

**}**

**package com.ibm;**

**import org.springframework.boot.SpringApplication;**

**import org.springframework.boot.autoconfigure.SpringBootApplication;**

**@SpringBootApplication**

**public class HelloReactiveApplication {**

**public static void main(String[] args) {**

**SpringApplication.run(HelloReactiveApplication.class, args);**

**}**

**}**

**New Employee API**

**package com.ibm.model;**

**import jakarta.persistence.Embedded;**

**import jakarta.persistence.MappedSuperclass;**

**import jakarta.validation.constraints.NotBlank;**

**import jakarta.validation.constraints.Pattern;**

**import lombok.AllArgsConstructor;**

**import lombok.Data;**

**import lombok.NoArgsConstructor;**

**@NoArgsConstructor**

**@AllArgsConstructor**

**@Data**

**@MappedSuperclass**

**public class Person {**

**@NotBlank**

**private String name;**

**private int age;**

**@Pattern(regexp = "(\\+91|91)\\s?\\-?\\d+")**

**private String mobile;**

**@Pattern(regexp = "^[2-9]{1}[0-9]{3}\\s[0-9]{4}\\s[0-9]{4}$")**

**private String aadhar;**

**@Embedded**

**private Address address;**

**}**

**package com.ibm.model;**

**import jakarta.persistence.Entity;**

**import jakarta.persistence.GeneratedValue;**

**import jakarta.persistence.Id;**

**import lombok.AllArgsConstructor;**

**import lombok.Data;**

**import lombok.EqualsAndHashCode;**

**import lombok.NoArgsConstructor;**

**@NoArgsConstructor**

**@AllArgsConstructor**

**@Data**

**@EqualsAndHashCode(callSuper = true)**

**@Entity**

**public class Employee extends Person {**

**@Id**

**@GeneratedValue**

**private int empid;**

**private String department;**

**}**

**package com.ibm.model;**

**import lombok.AllArgsConstructor;**

**import lombok.Data;**

**import lombok.NoArgsConstructor;**

**@NoArgsConstructor**

**@AllArgsConstructor**

**@Data**

**public class Address {**

**private String street;**

**private String city;**

**private int pincode;**

**}**

**package com.ibm.repository;**

**import org.springframework.data.jpa.repository.JpaRepository;**

**import com.ibm.model.Employee;**

**public interface EmployeeRepository extends JpaRepository<Employee, Integer> {**

**}**

**package com.ibm;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.boot.CommandLineRunner;**

**import org.springframework.boot.SpringApplication;**

**import org.springframework.boot.autoconfigure.SpringBootApplication;**

**import com.ibm.model.Address;**

**import com.ibm.model.Employee;**

**import com.ibm.repository.EmployeeRepository;**

**@SpringBootApplication**

**public class EmployeeApiApplication implements CommandLineRunner{**

**@Autowired**

**EmployeeRepository empRepo;**

**public static void main(String[] args) {**

**SpringApplication.run(EmployeeApiApplication.class, args);**

**}**

**@Override**

**public void run(String... args) throws Exception {**

**Employee emp = new Employee(1,"IT");**

**emp.setAadhar("7919 0230 0189"); //with spaces**

**emp.setMobile("+919662332335");**

**emp.setAge(30);**

**emp.setName("Raj");**

**Address address = new Address("MG ROAD","Bangalore",560001);**

**emp.setAddress(address);**

**empRepo.save(emp);**

**}**

**}**

**application.properties**

**spring.application.name=employee-api**

**server.port=9080**

**spring.jpa.show-sql=true**

**spring.h2.console.enabled=true**

**logging.level.org.springframework.web=trace**

**spring.datasource.url=jdbc:h2:mem:empdb**

**spring.datasource.driverClassName=org.h2.Driver**

**spring.datasource.username=sa**

**spring.jpa.database-platform=org.hibernate.dialect.H2Dialect**

**spring.mvc.format.date=yyyy-MM-dd**