

Program for Swagger with Spring MVC Rest API using JPA CRUD operations with MySQL

SpringRestHibernationApplication.java

```
package com.test;

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

@SpringBootApplication

public class SpringRestHibernateApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringRestHibernateApplication.class, args);
    }

}
```

CarController.java

```
package com.test.controller;

import java.util.List;
import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import com.test.entities.Car;
import com.test.service.CarService;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;

@RestController
@Api(value = "This is spring boot rest api and hibernate Car mngnt application")
@RequestMapping(value = "/api/cars")
public class CarController {

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

    CarService carService;

    @Autowired
    public CarController(CarService carService) {
        this.carService = carService;
    }

    @ApiOperation(value = "it will add CAR DATA ")
    @RequestMapping(value = "", method = RequestMethod.POST)
    public ResponseEntity<Void> create(@RequestBody Car car) {
        try {
            logger.info(car.getCarBrand());
        }
    }
}
```

```

        logger.info(car.getCarEngine());
        logger.info(car.getCarModel());
        logger.info(car.getHorsepower());

        carService.add(car);

        return ResponseEntity.status(HttpStatus.OK).build();
    } catch (Exception e) {
        e.printStackTrace();
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
    }
}

@ApiOperation(value = "it will display all CAR DATA ")
@RequestMapping(value = "", method = RequestMethod.GET)
public ResponseEntity<List<Map<String, Object>>> getAll() {
    try {
        List<Map<String, Object>> result = carService.findAll();
        return ResponseEntity.status(HttpStatus.OK).body(result);
    } catch (Exception e) {
        logger.error(e.getMessage(), e);
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
    }
}

@ApiOperation(value = "it will display CAR DATA based on id")
@RequestMapping(value =("/{id}", method = RequestMethod.GET)
public ResponseEntity<Car> getById(@PathVariable("id") int id) {
    try {
        Car car = carService.findById(id);
        if (car != null) {
            return ResponseEntity.status(HttpStatus.OK).body(car);
        } else {

```

```

        return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
    }
} catch (Exception e) {
    logger.error(e.getMessage(), e);
    return ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
}
}

@ApiOperation(value = "it will update CAR DATA based on ID")
@RequestMapping(value =("/{id}", method = RequestMethod.PUT)
public ResponseEntity<Void> update(@PathVariable("id") int id, @RequestBody Car car) {
    try {
        carService.update(id, car);
        return ResponseEntity.status(HttpStatus.OK).build();
    } catch (Exception e) {
        e.printStackTrace();
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
    }
}

@ApiOperation(value = "it will delete CAR DATA based on ID")
@RequestMapping(value =("/{id}", method = RequestMethod.DELETE)
public ResponseEntity<Void> delete(@PathVariable("id") int id) {
    try {
        carService.remove(id);
        return ResponseEntity.status(HttpStatus.OK).build();
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
    }
}
}

```

CarDao.java

```
package com.test.dao;

import java.util.List;
import java.util.Map;

import com.test.entities.Car;

public interface CarDao {

    public Car findById(int id);

    public void remove(int id);

    public void add(Car car);

    public void update(int id, Car car);

    public List<Map<String, Object>> findAll();
}
```

Car.java

```
package com.test.entities;

import java.io.Serializable;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
```

```
import javax.persistence.Id;

import javax.persistence.SequenceGenerator;

import javax.persistence.Table;


import org.slf4j.Logger;

import org.slf4j.LoggerFactory;


import com.fasterxml.jackson.annotation.JsonIgnoreProperties;


import lombok.Getter;

import lombok.Setter;


@Entity
@Table(name="CAR")
@JsonIgnoreProperties(ignoreUnknown=true)
public class Car implements Serializable {


    private static final long serialVersionUID = 1L;

    public static final Logger logger = LoggerFactory.getLogger(Car.class);


    @Id

    @GeneratedValue(generator = "CAR_SEQ", strategy = GenerationType.SEQUENCE)

    @SequenceGenerator(name = "CAR_SEQ", sequenceName = "CAR_SEQ", allocationSize=1)

    @Column(name="CAR_ID", unique=true, nullable=false, precision=10, scale=0)

    @Getter @Setter

    private Integer carId;


    @Column(name="CAR_BRAND", nullable = true, length = 50)

    @Getter @Setter

    private String carBrand;
```

```
@Column(name="CAR_MODEL", nullable = true, length = 50)
```

```
@Getter @Setter
```

```
private String carModel;
```

```
@Column(name="HORSEPOWER", nullable = true, length = 6)
```

```
@Getter @Setter
```

```
private String horsepower;
```

```
@Column(name="CAR_ENGINE", nullable = true, length = 6)
```

```
@Getter @Setter
```

```
private String carEngine;
```

```
public Car(){}
```

```
public Car(String carBrand, String carModel, String horsepower, String carEngine) {
```

```
    this.carBrand = carBrand;
```

```
    this.carModel = carModel;
```

```
    this.horsepower = horsepower;
```

```
    this.carEngine = carEngine;
```

```
}
```

```
public Integer getCarId() {
```

```
    return carId;
```

```
}
```

```
public void setCarId(Integer carId) {
```

```
    this.carId = carId;
```

```
}
```

```
public String getCarBrand() {  
    return carBrand;  
}
```

```
public void setCarBrand(String carBrand) {  
    this.carBrand = carBrand;  
}
```

```
public String getCarModel() {  
    return carModel;  
}
```

```
public void setCarModel(String carModel) {  
    this.carModel = carModel;  
}
```

```
public String getHorsepower() {  
    return horsepower;  
}
```

```
public void setHorsepower(String horsepower) {  
    this.horsepower = horsepower;  
}
```

```
public String getCarEngine() {  
    return carEngine;  
}
```

```
public void setCarEngine(String carEngine) {  
    this.carEngine = carEngine;  
}
```



```
        public static long getSerialVersionUID() {  
            return serialVersionUID;  
        }  
  
        public static Logger getLogger() {  
            return logger;  
        }  
    }  
}
```

CarDaoImpl.java

```
package com.test.impl;  
  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Repository;  
  
import com.test.dao.CarDao;  
import com.test.entities.Car;  
import com.test.repository.CarRepository;  
  
@Repository  
public class CarDaoImpl implements CarDao {  
  
    private static final Logger logger = LoggerFactory.getLogger(CarDaoImpl.class);
```

@Autowired

private CarRepository carRepository;

public CarDaoImpl() {

}

@Override

public Car findById(int id) {

try {

return carRepository.findById(id);

} catch (Exception e) {

logger.error(e.getMessage(), e);

}

return null;

}

@Override

public void remove(int id) {

try {

Car car = new Car();

car.setCarId(id);

carRepository.delete(car);

} catch (Exception e) {

logger.error(e.getMessage(), e);

}

}

@Override

```
public void add(Car car) {  
    try {  
        carRepository.save(car);  
    } catch (Exception e) {  
        logger.error(e.getMessage(), e);  
    }  
}
```

@Override

```
public void update(int id, Car car) {  
    try {  
        car.setCarId(id);  
        carRepository.save(car);  
    } catch (Exception e) {  
        logger.error(e.getMessage(), e);  
    }  
}
```

@Override

```
public List<Map<String, Object>> findAll() {  
    try {  
        List<Map<String, Object>> list = new ArrayList<>();  
        Map<String, Object> map = new HashMap<>();  
        List<Car> result = carRepository.findAll();  
        for (Car car : result) {  
            map = new HashMap<>();  
            map.put(car.getCarId().toString(), car);  
            list.add(map);  
        }  
    }  
}
```

```

        return list;
    } catch (Exception e) {
        logger.error(e.getMessage(), e);
    }
    return null;
}
}

```

CarRepository.java

```
package com.test.repository;
```

```
import java.util.List;
```

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

```
import org.springframework.transaction.annotation.Transactional;
```

```
import com.test.entities.Car;
```

```
@Transactional(readOnly = true)
```

```
public interface CarRepository extends JpaRepository<Car, Integer> {
```

```
    @Transactional(timeout = 10)
```

```
    Car findByCarId(Integer carId);
```

```
    @Transactional(timeout = 10)
```

```
    List<Car> findAll();
```

```
    @Transactional
```

```
    <S extends Car> S save(Car car);
```

```
        void delete(Car car);  
    }  
}
```

CarService.java

```
package com.test.service;
```

```
import java.util.List;
```

```
import java.util.Map;
```

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

```
import org.springframework.stereotype.Service;
```

```
import org.springframework.transaction.annotation.Transactional;
```

```
import com.test.dao.CarDao;
```

```
import com.test.entities.Car;
```

```
@Service
```

```
@Transactional
```

```
public class CarService {
```

```
    @Autowired
```

```
    private CarDao carDao;
```

```
    public CarService() {
```

```
    }
```

```
    public Car findById(int id) {
```

```
        if (id <= 0) {
```

```

        throw new IllegalArgumentException("ID cannot be 0 or < 0");
    }
    return carDao.findById(id);
}

public void remove(int id) {
    if (id <= 0) {
        throw new IllegalArgumentException("ID cannot be 0 or < 0 or this id do not exist");
    }
    carDao.remove(id);
}

public List<Map<String, Object>> findAll() {

    List<Map<String, Object>> result = carDao.findAll();
    if (result.size() > 0) {
        return result;
    } else {
        return null;
    }
}

public void add(Car car) {
    if (car == null) {
        throw new IllegalArgumentException("The passed object cannot be null.");
    }
    carDao.add(car);
}

public void update(int id, Car car) {
    if (id <= 0 && car == null) {

```

```

        throw new IllegalArgumentException("The passed object cannot be null.");
    }

    carDao.update(id, car);
}

}

```

ApplicationProperties

```

server.port=7070
spring.main.banner-mode=off

#server
#server.error.whitelabel.enabled=false

# create and drop tables and sequences
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true

# database settings
spring.datasource.url=jdbc:mysql://localhost:3306/springdb
spring.datasource.username=root
spring.datasource.password=Password@123
spring.datasource.driver-class-name=com.mysql.jdbc.Driver

# HikariCP settings
# spring.datasource.hikari.*

spring.datasource.hikari.connection-timeout=60000
spring.datasource.hikari.maximum-pool-size=5

# logging
logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n
logging.level.org.hibernate.SQL=debug
#logging.level.org.hibernate.type.descriptor.sql=trace
logging.level.=info

```

SwaggerConfig.java

```
package com.test.swaggerCfg;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

import java.nio.file.Path;

@Configuration
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket api(){
        return new Docket(DocumentationType.SWAGGER_2)
            .select()
            .apis(RequestHandlerSelectors.any())
            .paths(PathSelectors.any())
            .build().apiInfo(apiInfo());
    }

    private ApiInfo apiInfo() {
        return new ApiInfoBuilder()
```



```

        .title("CVRINDIA API")

        .description("CVRINDIA API's")

        .version("1.0.0")

        .termsOfServiceUrl("http://cvrindia.com")


        .license("CVRINDIA License")

        .licenseUrl("http://cvrindia.com")

        .build();
    }
}

```

OUTPUT

 **swagger**

default (v2/api-docs)
api_key
Explore

CVRINDIA API

CVRINDIA API's

[CVRINDIA License](#)

basic-error-controller : Basic Error Controller

Show/Hide | List Operations | Expand Operations

car-controller : Car Controller

Show/Hide | List Operations | Expand Operations

| | | |
|--------|----------------|--------------------------------------|
| GET | /api/cars | it will display all CAR DATA |
| POST | /api/cars | it will add CAR DATA |
| DELETE | /api/cars/{id} | it will delete CAR DATA based on ID |
| GET | /api/cars/{id} | it will display CAR DATA based on id |
| PUT | /api/cars/{id} | it will update CAR DATA based on ID |

[BASE URL: / , API VERSION: 1.0.0]

Request URL

http://localhost:7070/api/cars

Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body

```
[
  {
    "1": {
      "carId": 1,
      "carBrand": "Lamborghini",
      "carModel": "X7",
      "horsepower": "8000",
      "carEngine": "hxxl"
    }
  }
]
```

Response Code

200

Response Headers

```
{
  "connection": "keep-alive",
}
```

POST /api/cars

it will add CAR DATA

Parameters

| Parameter | Value | Description | Parameter Type | Data Type |
|-----------|---|-------------|----------------|---|
| car | <pre>{ "carBrand": "Lamborghini", "carEngine": "hxxl", "carId": 1, "carModel": "X7", "horsepower": "8000" }</pre> | car | body | <div>Model Model Schema</div> <pre>{ "carBrand": "string", "carEngine": "string", "carId": 0, "carModel": "string", "horsepower": "string" }</pre> <div>Click to set as parameter value</div> |

Response Messages

| HTTP Status Code | Reason | Response Model | Headers |
|------------------|--------------|----------------|---------|
| 200 | OK | | |
| 201 | Created | | |
| 401 | Unauthorized | | |
| 403 | Forbidden | | |
| 404 | Not Found | | |

Try it out!

[Hide Response](#)

PUT

/api/cars/{id}

it will update CAR DATA based on ID

Parameters

| Parameter | Value | Description | Parameter Type | Data Type |
|-----------|--|-------------|----------------|---|
| id | 1 | id | path | integer |
| car | <pre>{ "carBrand": "Ferrari", "carEngine": "VGGF", "carId": 1, "carModel": "JJ8", "horsepower": "6500" }</pre> <div>Parameter content type: application/json</div> | car | body | <div>Model Model Schema</div> <div><pre>{ "carBrand": "string", "carEngine": "string", "carId": 0, "carModel": "string", "horsepower": "string" }</pre><div>Click to set as parameter value</div></div> |

Response Messages

| HTTP Status Code | Reason | Response Model | Headers |
|------------------|--------------|----------------|---------|
| 200 | OK | | |
| 201 | Created | | |
| 401 | Unauthorized | | |
| 403 | Forbidden | | |
| 404 | Not Found | | |

DELETE

/api/cars/{id}

it will delete CAR DATA based on ID

Parameters

| Parameter | Value | Description | Parameter Type | Data Type |
|-----------|-------|-------------|----------------|-----------|
| id | 1 | id | path | integer |

Response Messages

| HTTP Status Code | Reason | Response Model | Headers |
|------------------|--------------|----------------|---------|
| 200 | OK | | |
| 204 | No Content | | |
| 401 | Unauthorized | | |
| 403 | Forbidden | | |

Try it out!

[Hide Response](#)

Curl

```
curl -X DELETE --header 'Accept: */*' 'http://localhost:7070/api/cars/1'
```

Request URL

```
http://localhost:7070/api/cars/1
```