

Program for Spring MVC Rest API using Hibernate with MySQL

SpringRestHibernateApplication.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
```

Test.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <form action="http://localhost:7070/api/cars/">
        click here
    </form>
</body>
</html>
```







OUTPUT

```
8
9 • create database springdb;
10 • use springdb;
11 • select * from car;
```

Result Grid				
Filter Rows: <input type="text"/>				
Edit:   				
Export/Import:  				
Wrap Cell Content: 				
car_id	car_brand	car_engine	car_model	horsepower
1	Lamborghini	cxzy	X7	2000
2	Ferrari	ghtg	MS9	1500
3	RollsRoyce	xxcd	RoyalX1	1400
4	NULL	NULL	NULL	NULL


```
Problems Javadoc Declaration Console Progress
SpringRestHibernate - SpringRestHibernateApplication [Spring Boot App] D:\Training Workspace\Project\sts-4.13.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64.17.0.1.v20211116-1657\jre\bin\javaw.exe (11-Mar-2022, 5:10:39)
2022-03-11 17:11:05 INFO o.h.e.t.j.p.i.JtaPlatformInitiator - HH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2022-03-11 17:11:06 INFO o.s.o.j.LocalContainerEntityManagerFactoryBean - Initialized JPA EntityManagerFactory for persistence unit 'default'
2022-03-11 17:11:06 INFO o.s.b.d.a.OptionalLiveReloadServer - LiveReload server is running on port 35729
2022-03-11 17:11:07 WARN o.s.b.a.o.j.JpaBaseConfiguration$JpaWebConfiguration - spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering.
2022-03-11 17:11:07 INFO o.s.s.c.ThreadPoolTaskExecutor - Initializing ExecutorService 'applicationTaskExecutor'
2022-03-11 17:11:09 INFO s.d.s.w.p.DocumentationPluginsBootstrapper - Context refreshed
2022-03-11 17:11:09 INFO s.d.s.w.p.DocumentationPluginsBootstrapper - Found 1 custom documentation plugin(s)
2022-03-11 17:11:09 INFO s.d.s.w.s.ApiListingReferenceScanner - Scanning for api listing references
2022-03-11 17:11:10 INFO o.s.b.w.e.tomcat.TomcatWebServer - Tomcat started on port(s): 7070 (http) with context path ''
2022-03-11 17:11:10 INFO c.t.SpringRestHibernateApplication - Started SpringRestHibernateApplication in 24.732 seconds (JVM running for 27.437)
2022-03-11 17:11:58 INFO o.a.c.c.C.[Tomcat].[localhost].[/] - Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-03-11 17:11:58 INFO o.s.web.servlet.DispatcherServlet - Initializing Servlet 'dispatcherServlet'
2022-03-11 17:11:58 INFO o.s.web.servlet.DispatcherServlet - Completed initialization in 136 ms
2022-03-11 17:12:10 DEBUG org.hibernate.SQL - select car0_.car_id as car_id1_0_, car0_.car_brand as car_bran2_0_, car0_.car_engine as car_engi3_0_, car0_.car_model as car_mode4_0_, car0_.horsepower as horsepower5_0_ from car
Hibernate: select car0_.car_id as car_id1_0_, car0_.car_brand as car_bran2_0_, car0_.car_engine as car_engi3_0_, car0_.car_model as car_mode4_0_, car0_.horsepower as horsepower5_0_ from car
2022-03-11 17:13:52 INFO com.test.controller.CarController - Lamborghini
2022-03-11 17:13:52 INFO com.test.controller.CarController - hxxl
2022-03-11 17:13:52 INFO com.test.controller.CarController - X7
2022-03-11 17:13:52 INFO com.test.controller.CarController - 8000
2022-03-11 17:13:52 DEBUG org.hibernate.SQL - select car0_.car_id as car_id1_0_, car0_.car_brand as car_bran2_0_, car0_.car_engine as car_engi3_0_, car0_.car_model as car_mode4_0_, car0_.horsepower as horsepower5_0_ from car
Hibernate: select car0_.car_id as car_id1_0_, car0_.car_brand as car_bran2_0_, car0_.car_engine as car_engi3_0_, car0_.car_model as car_mode4_0_, car0_.horsepower as horsepower5_0_ from car
2022-03-11 17:13:52 DEBUG org.hibernate.SQL - select next_val as id_val from car_seq for update
Hibernate: select next_val as id_val from car_seq for update
2022-03-11 17:13:52 DEBUG org.hibernate.SQL - update car_seq set next_val= ? where next_val=?
Hibernate: update car_seq set next_val= ? where next_val=?
2022-03-11 17:13:52 DEBUG org.hibernate.SQL - insert into car (car_brand, car_engine, car_model, horsepower, car_id) values (?, ?, ?, ?, ?)
Hibernate: insert into car (car_brand, car_engine, car_model, horsepower, car_id) values (?, ?, ?, ?, ?)
2022-03-11 17:14:07 DEBUG org.hibernate.SQL - select car0_.car_id as car_id1_0_, car0_.car_brand as car_bran2_0_, car0_.car_engine as car_engi3_0_, car0_.car_model as car_mode4_0_, car0_.horsepower as horsepower5_0_ from car
2022-03-11 17:16:23 DEBUG org.hibernate.SQL - select car0_.car_id as car_id1_0_, car0_.car_brand as car_bran2_0_, car0_.car_engine as car_engi3_0_, car0_.car_model as car_mode4_0_, car0_.horsepower as horsepower5_0_ from car
2022-03-11 17:16:23 DEBUG org.hibernate.SQL - update car set car_brand=?, car_engine=?, car_model=?, horsepower=? where car_id=?
2022-03-11 17:16:23 DEBUG org.hibernate.SQL - update car set car_brand=?, car_engine=?, car_model=?, horsepower=? where car_id=?
2022-03-11 17:17:07 DEBUG org.hibernate.SQL - select car0_.car_id as car_id1_0_, car0_.car_brand as car_bran2_0_, car0_.car_engine as car_engi3_0_, car0_.car_model as car_mode4_0_, car0_.horsepower as horsepower5_0_ from car
2022-03-11 17:17:07 DEBUG org.hibernate.SQL - delete from car where car_id=?
Hibernate: delete from car where car_id=?
```

- 8
- 9 • create database springdb;
- 10 • use springdb;
- 11 • select * from car;

Result Grid					
Filter Rows: <input type="text"/>					
Edit:    Export/Import:   Wrap Cell Content: 					
	car_id	car_brand	car_engine	car_model	horsepower
▶	1	Lamborghini	cxzy	X7	2000
	2	TaTa	abcd	Safari	800
	3	RollsRoyce	xxcd	RoyalX1	1400
✱	NULL	NULL	NULL	NULL	NULL