SpringBoot-SpringDataJPA-MYSQL-SOAPUI Sample Application

Prerequisites

We must have installed the following

- JDK 1.7 or later
- Maven 3 or later

Project Structure

```
├── src
│   └── main
│       ├── java
│       │   └── com
│       │       └── springboot
│       │           └── springdatajpa
│       │               ├──Controller
│       │               │    └──BookingController.java
│       │               ├──models
│       │               │    ├── Booking.java
│       │               │    └── BookingRepository.java
│       │               SpringBootJpaSpringDataApplication.java
│       └── resources
│           ├── application.properties
│           └── hibernate.properties
│
└── pom.xml
```

Project dependencies

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>

	<groupId>com.springdatajpa.sdjpa</groupId>
	<artifactId>springdb</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>jar</packaging>

	<name>SpringBootJPASpringData</name>
	<description>SpringBootJPASpringData project for Spring Boot with Spring Data JPA
implementation</description>

	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
```

```
		<version>1.4.0.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>

	<properties>
		<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
		<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
		<java.version>1.8</java.version>
	</properties>

	<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-web</artifactId>
		</dependency>
		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
			<scope>runtime</scope>
		</dependency>

	</dependencies>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>

</project>
```

Application Configuration

SpringBootJpaSpringDataApplication.java

Path → src/main/java/com/test/sdjpa/ SpringBootJpaSpringDataApplication.java

```
package com.test.sdjpa;

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

@SpringBootApplication
public class SpringBootJpaSpringDataApplication {

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

```
                SpringApplication.run(SpringBootJpaSpringDataApplication.class, args);
        }
}
```

Controller File

BookingController.java

Path → src/main/java/com/test/sdjpa/ BookingController.java

```
package com.springboot.springdatajpa.controller;

import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.springboot.springdatajpa.models.Booking;
import com.springboot.springdatajpa.models.BookingRepository;


/**
 * @author Dinesh.Rajput
 *
 */

@RestController
@RequestMapping("/booking")
public class BookingController {

        @Autowired
        BookingRepository bookingRepository;
        /**
         * GET /create  --> Create a new booking and save it in the database.
         */


        @RequestMapping("/create")

        public Booking create(@Valid @RequestBody Booking booking) {
                booking.setTravelDate(new Date());
                booking = bookingRepository.save(booking);
           return booking;
        }
```

```
/**
 * GET /read  --> Read a booking by booking id from the database.
 */


@RequestMapping("/read")
public Booking read(@Valid @RequestBody Booking booking) {


        System.out.println("bookingId"+booking.getBookingId());
        Booking book = bookingRepository.findOne(booking.getBookingId());
   return book;
}

@RequestMapping("/readAll")
public List<Booking> readAll() {


        Iterable<Booking> book = bookingRepository.findAll();
        List<Booking> list = new ArrayList<Booking>();
   if(book != null) {
    for(Booking e: book) {
      list.add(e);
    }
   }
   return list;
}


/**
 * GET /update  --> Update a booking record and save it in the database.
 */


@RequestMapping("/update")
public Booking update(@Valid @RequestBody Booking book) {
        Booking booking = bookingRepository.findOne(book.getBookingId());
        booking.setPsngrName(book.getPsngrName());
        booking.setDeparture(book.getDeparture());
        booking.setDestination(book.getDestination());
        booking.setTravelDate(new Date());
        booking = bookingRepository.save(booking);
   return booking;
}

/**
 * GET /delete  --> Delete a booking from the database.
 */
```

```
	@RequestMapping("/delete")
	public Map<String, Object> delete(@Valid @RequestBody Booking book) {
		Map<String, Object> dataMap = new HashMap<String, Object>();

		bookingRepository.delete(book.getBookingId());
		dataMap.put("bookingId", book);
		dataMap.put("Status", "Successfully Deleted");
	   return dataMap;
	}

}
```

Model File

Booking.java

Path → src/main/java/com/test/sdjpa/ Booking.java

```
package com.test.sdjpa.models;

import java.io.Serializable;
import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "BOOKING")
public class Booking implements Serializable{

	/**
	 *
	 */
	private static final long serialVersionUID = 1L;
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	Long bookingId;
	@Column
	String psngrName;
	@Column
	String departure;
	@Column
	String destination;
	@Column
	Date travelDate;

	public Long getBookingId() {
		return bookingId;
```

```
	}

	public void setBookingId(Long bookingId) {
		this.bookingId = bookingId;
	}

	public String getPsngrName() {
		return psngrName;
	}

	public void setPsngrName(String psngrName) {
		this.psngrName = psngrName;
	}

	public String getDeparture() {
		return departure;
	}

	public void setDeparture(String departure) {
		this.departure = departure;
	}

	public String getDestination() {
		return destination;
	}

	public void setDestination(String destination) {
		this.destination = destination;
	}

	public Date getTravelDate() {
		return travelDate;
	}

	public void setTravelDate(Date travelDate) {
		this.travelDate = travelDate;
	}

}
```

BookingRepository.java

Path → src/main/java/com/test/sdjpa/ BookingRepository.java

```
package com.test.sdjpa.models;

import org.springframework.data.repository.CrudRepository;
import org.springframework.transaction.annotation.Transactional;

@Transactional
```

```
public interface BookingRepository extends CrudRepository<Booking, Long> {

        /**
          * This method will find an Boooking instance in the database by its departure.
          * Note that this method is not implemented and its working code will be
          * automatically generated from its signature by Spring Data JPA.
          */
         public Booking findByDeparture(String departure);
}
```

Application.properties

application.properties

Path → src/main/resources/application.properties

```
# DataSource settings: set here your own configurations for the database
# connection. In this example we have "dojsb" as database name and
# "root" as username and password.
spring.datasource.url = jdbc:mysql://localhost:3306/test
spring.datasource.username = root
spring.datasource.password = root

# Keep the connection alive if idle for a long time (needed in production)
spring.datasource.testWhileIdle = true
spring.datasource.validationQuery = SELECT 1

# Show or not log for each sql query
spring.jpa.show-sql = true

# Hibernate ddl auto (create, create-drop, update)
spring.jpa.hibernate.ddl-auto = create

# Naming strategy
spring.jpa.hibernate.naming-strategy = org.hibernate.cfg.ImprovedNamingStrategy

# Use spring.jpa.properties.* for Hibernate native properties (the prefix is
# stripped before adding them to the entity manager)

# The SQL dialect makes Hibernate generate better SQL for the chosen database
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect

server.port = 8082
```

Steps to Test the Application

D:\Karthik-ws\code\SpringBoot-SpringDataJPA-Mysql >mvn spring-boot:run

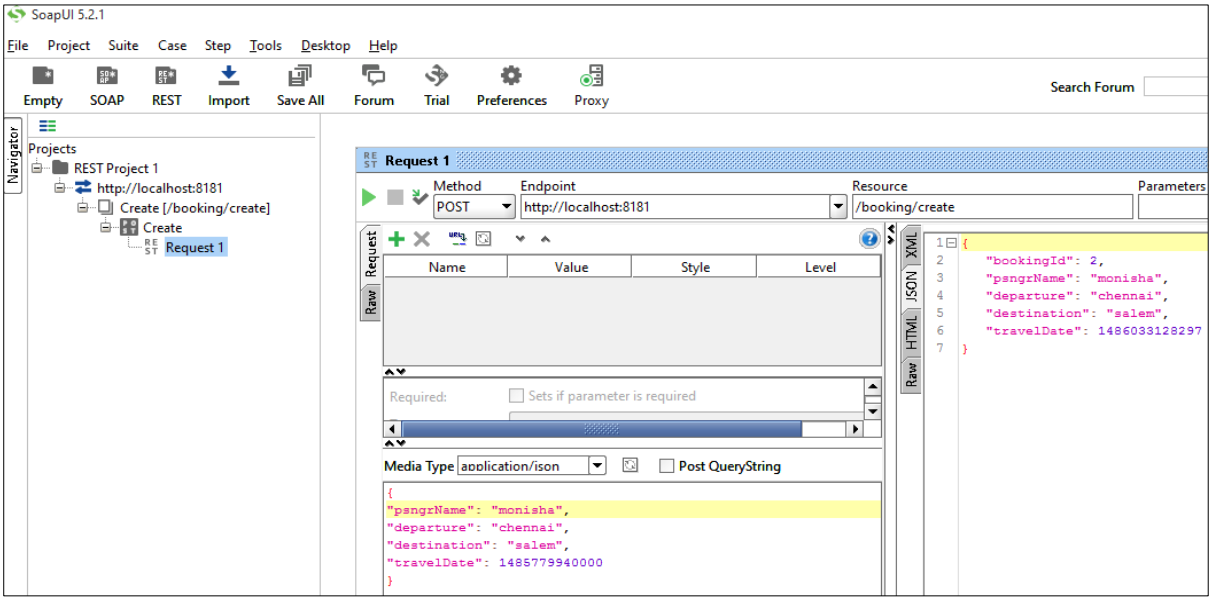
Once the server has been started. Enter the below url in the SOAPUI and select POST method.

Insert

http://localhost:8082/booking/create

JSON Input

{
"psngrName": "monisha",
"departure": "chennai",
"destination": "salem",
"travelDate": 1485779940000
}

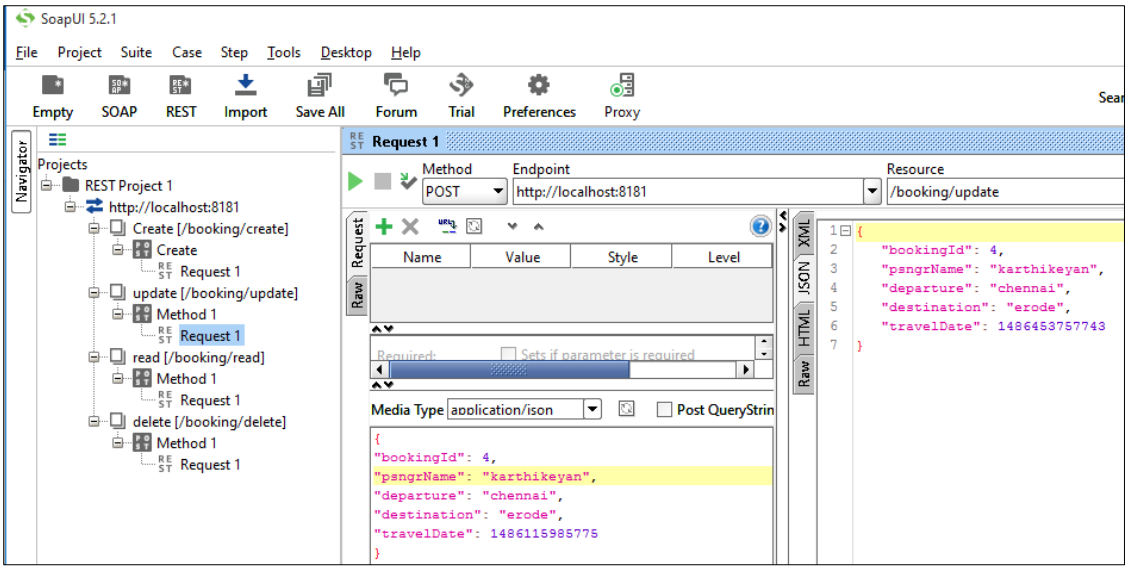


Update

http://localhost:8082/booking/update

JSON Input

{
"bookingId": 5,
"psngrName": "karthik",
"departure": "chennai",
"destination": "erode",
"travelDate": 1486115985775
}

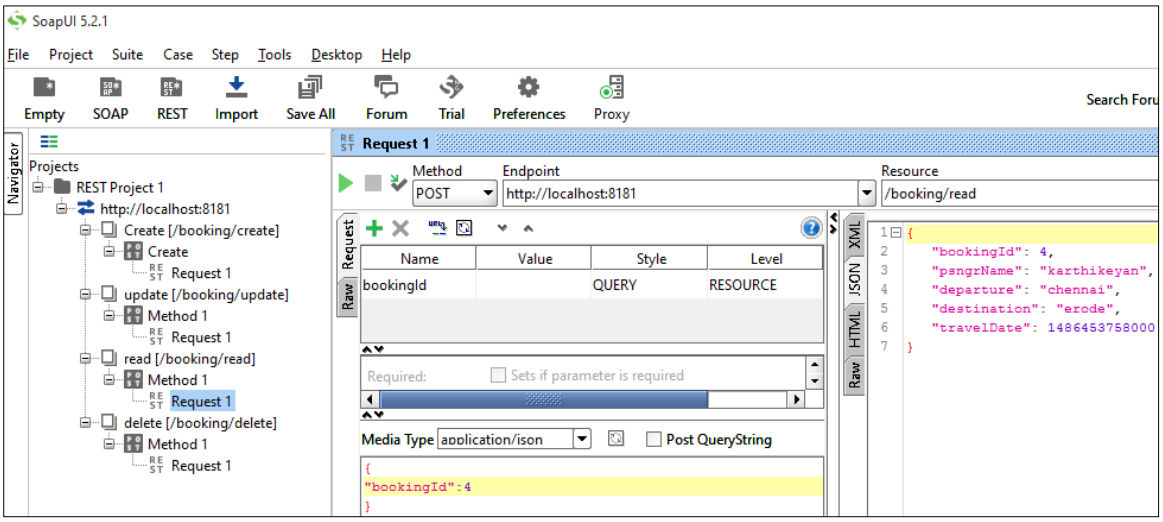


View

http://localhost:8082/booking/read

JSON Input

{
"bookingId":4
}

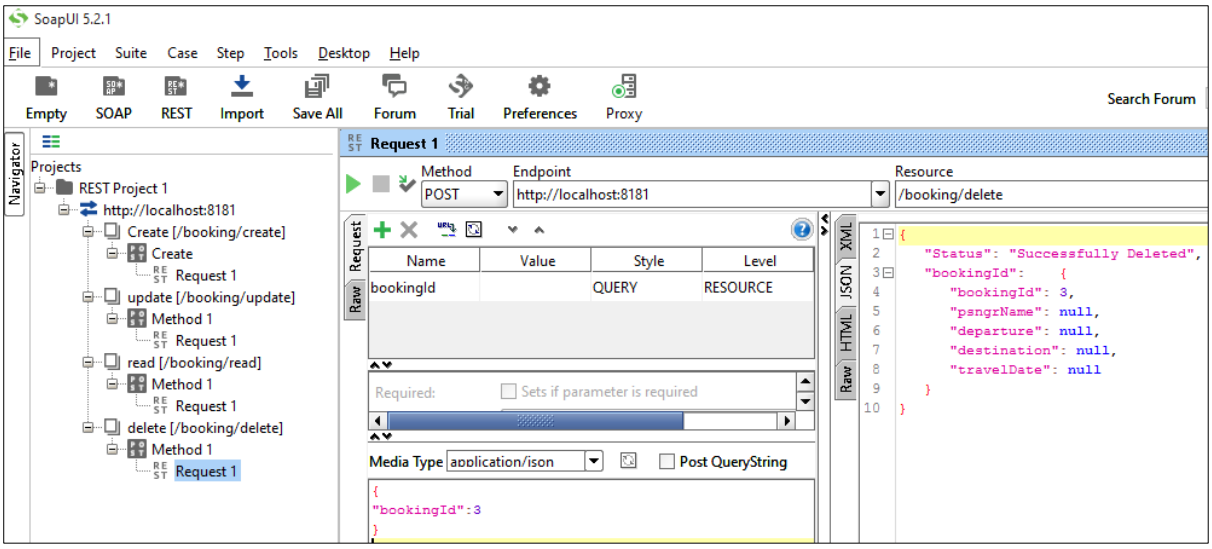


Delete

http://localhost:8082/booking/delete

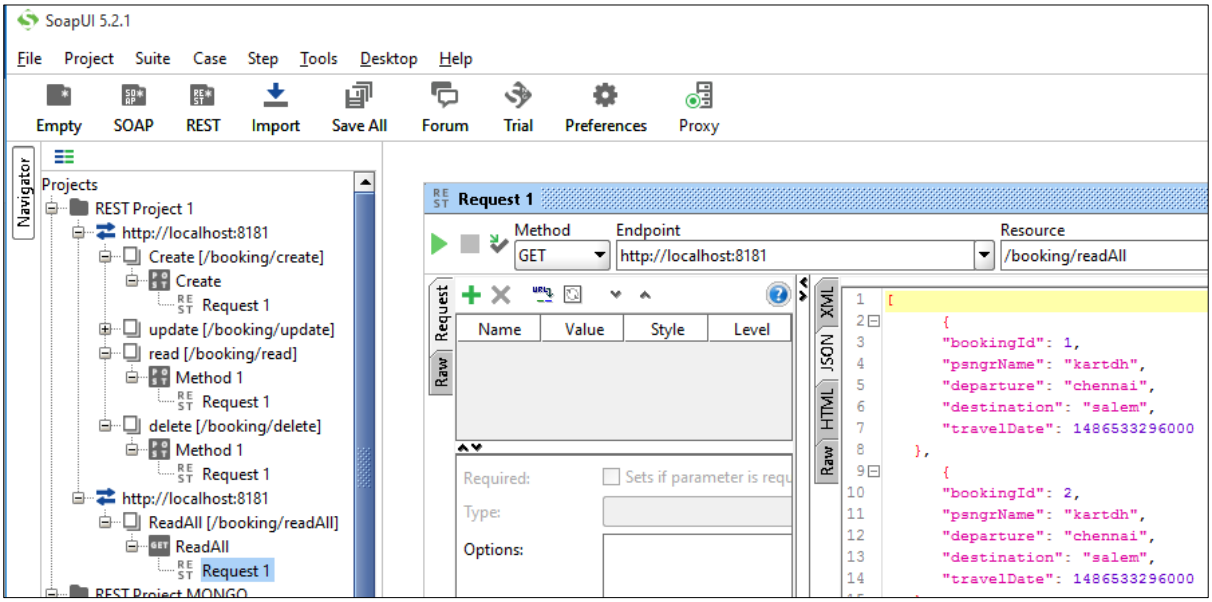
JSON Input

{
"bookingId":3
}



View All

http://localhost:8082/booking/readAll



Test in mysql Client