Sample Spring Boot project API document

Introduction

This is a sample spring boot project those who are seeking to learn about how to create a simple spring boot crud application. This project has implemented all the crud operations in a simple way. You can learn how to do crud operations in spring boot.

Technologies Used

Spring Boot

MySQL

Hibernate

IDE used

IntelliJ IDEA

API Signatures

Here I have run this application in localhost and 8883 port. You can change the port as you wish.

API signature	Method	Description
localhost:8883/employee/save	POST	This API will save an employee in employee database. (a record will insert to employee table).
localhost:8883/employee/getall	GET	This API will give details of all the employees as a List.
localhost:8883/employee/getone/{employeeId}	GET	This API will give details of the employee, that we send the employee id in the url.
localhost:8883/employee/delete/{employeeId}	DELETE	This API will give delete the employee, that we send the employee id in the url.(delete the record of that employee from employee table)
localhost:8883/employee/update	PUT	This API will update the details of an employee. (update the employee table)

Let’s test the API

Import project to IDE

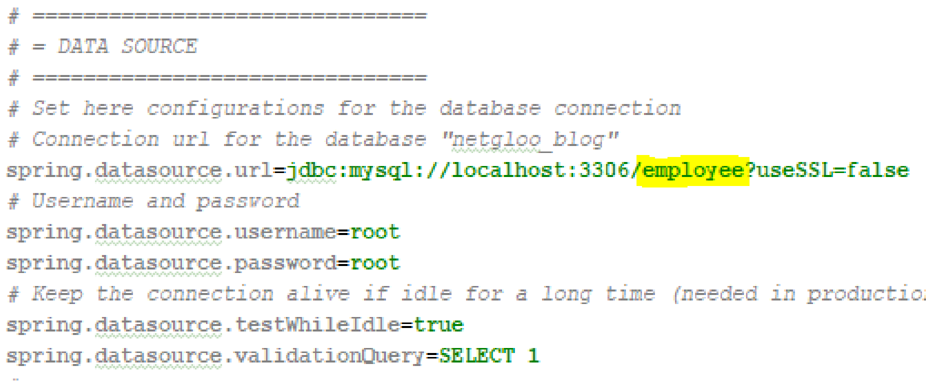
First you must clone the project from the GitHub repository and then open the project using IntelliJ as below.

File -> Open -> *select your project*

Create the Database

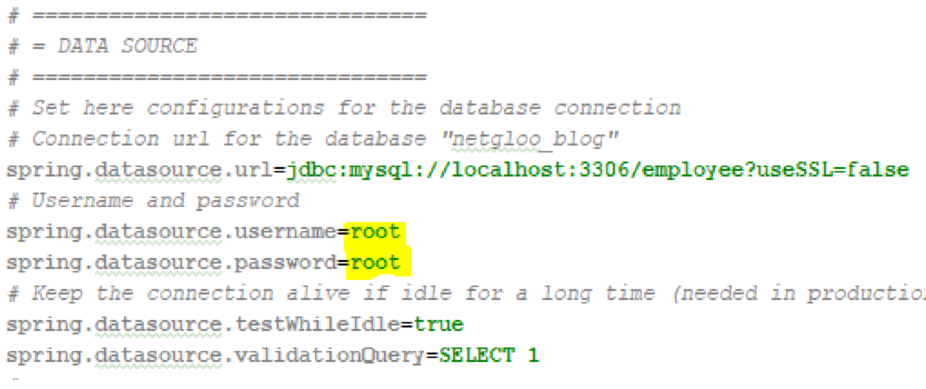
You must create your database.

Create a database as ***employee*** in your MySQL workbench. You can create your own database. it doesn’t have to be the name as ***employee*** in my case I used the database as ***employee****.* You can create your oen database. If you are creating your own database, then you must change the database name in property file to the name of your changed database. You can find it in below image, the database name is highlighted.



```
# ===============================
# = DATA SOURCE
# ===============================
# Set here configurations for the database connection
# Connection url for the database "netgloo_blog"
spring.datasource.url=jdbc:mysql://localhost:3306/employee?useSSL=false
# Username and password
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
```

And, database ***username*** and ***password*** must change according to your MySQL configuration.



```
# ===============================
# = DATA SOURCE
# ===============================
# Set here configurations for the database connection
# Connection url for the database "netgloo_blog"
spring.datasource.url=jdbc:mysql://localhost:3306/employee?useSSL=false
# Username and password
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
```

NOTE: -

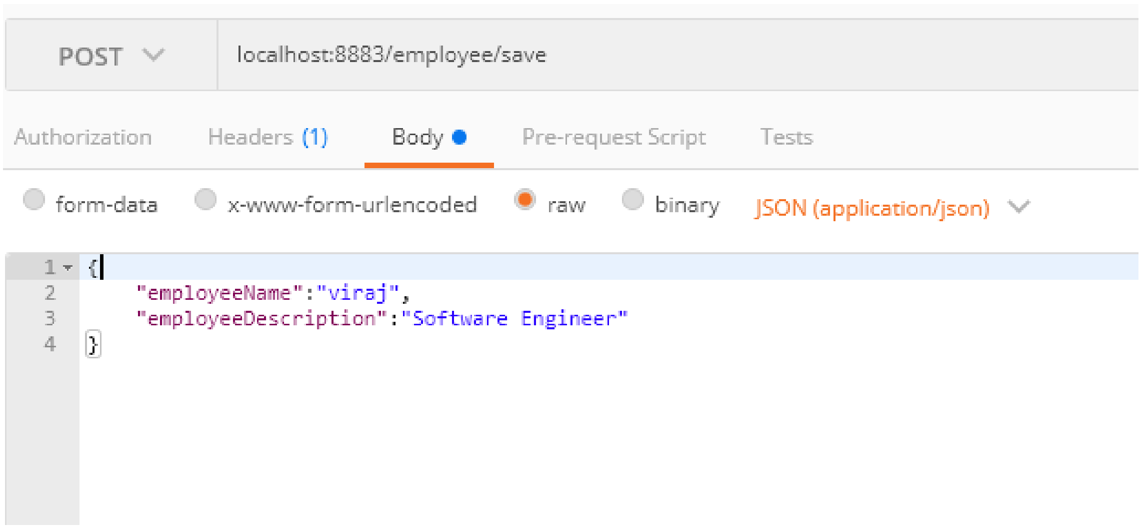
You only need to create the database , you must not create the tables. when you run the application Hibernate will create the tables in the ***employee*** database for you.

Then Build and Run your project.

You can test all the routes of the API using postman. Here I have include all the screen shots of API calls using postman.

POST
localhost:8883/employee/save

Request

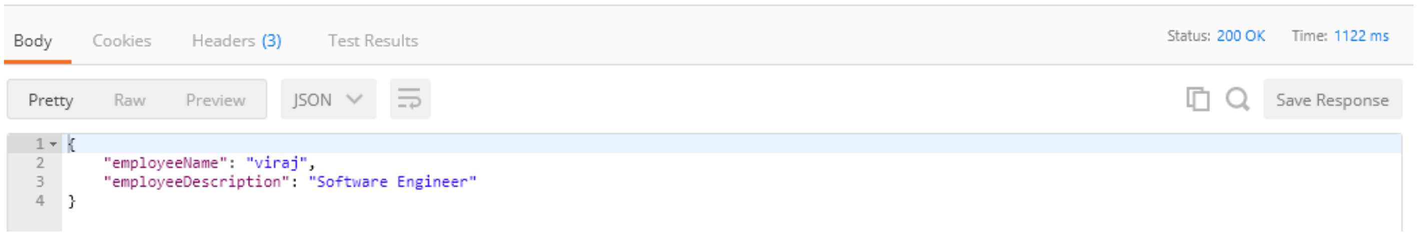


localhost:8883/employee/save

raw JSON (application/json)

```
{
    "employeeName":"viraj",
    "employeeDescription":"Software Engineer"
}
```

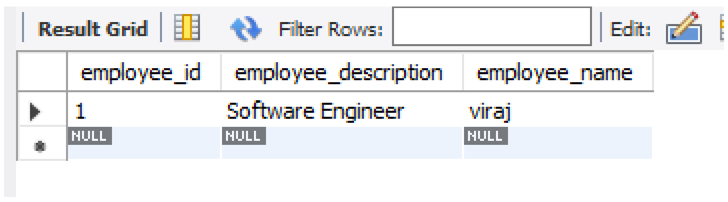
Response



Status: 200 OK Time: 1122 ms

```
{
    "employeeName": "viraj",
    "employeeDescription": "Software Engineer"
}
```

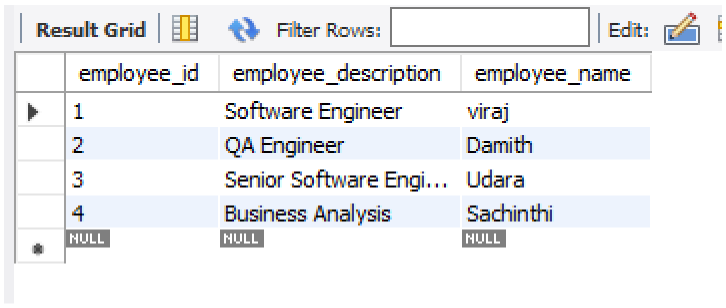
Database



	employee_id	employee_description	employee_name
▶	1	Software Engineer	viraj
*	NULL	NULL	NULL

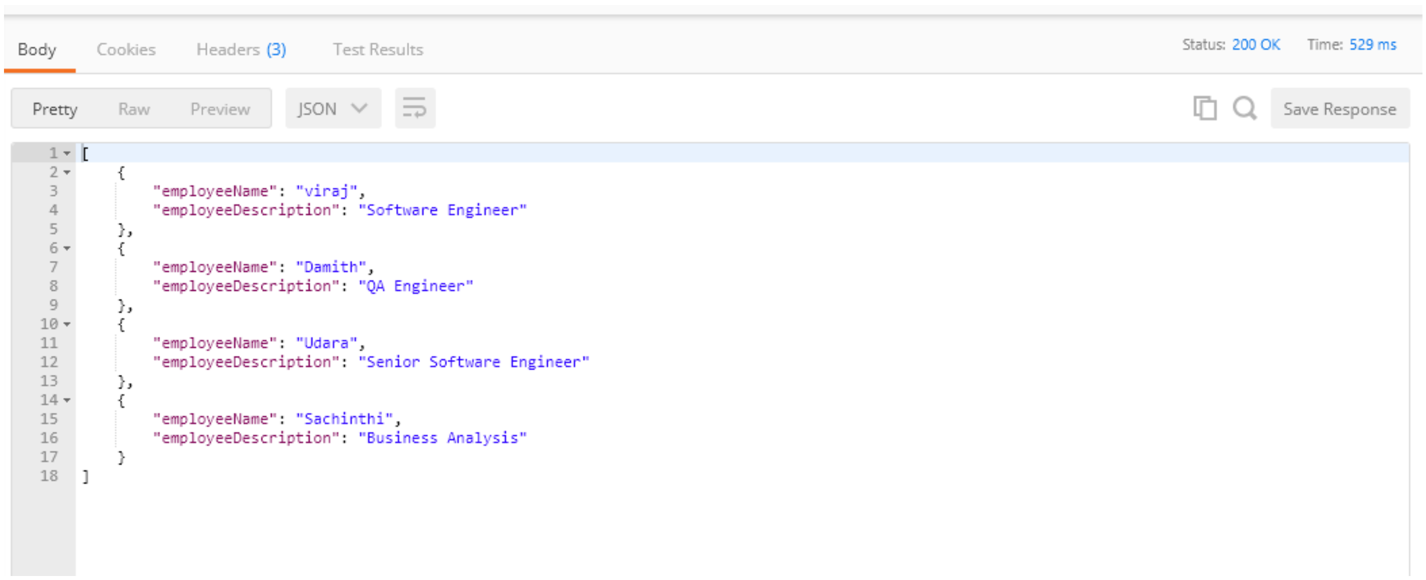
GET
localhost:8883/employee/getall

Database



	employee_id	employee_description	employee_name
▶	1	Software Engineer	viraj
	2	QA Engineer	Damith
	3	Senior Software Engi...	Udara
	4	Business Analysis	Sachinthi
*	NULL	NULL	NULL

Response

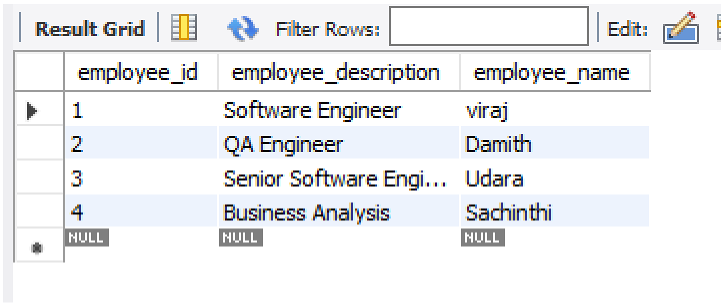


```
[
    {
        "employeeName": "viraj",
        "employeeDescription": "Software Engineer"
    },
    {
        "employeeName": "Damith",
        "employeeDescription": "QA Engineer"
    },
    {
        "employeeName": "Udara",
        "employeeDescription": "Senior Software Engineer"
    },
    {
        "employeeName": "Sachinthi",
        "employeeDescription": "Business Analysis"
    }
]
```

GET

localhost:8883/employee/getone/2

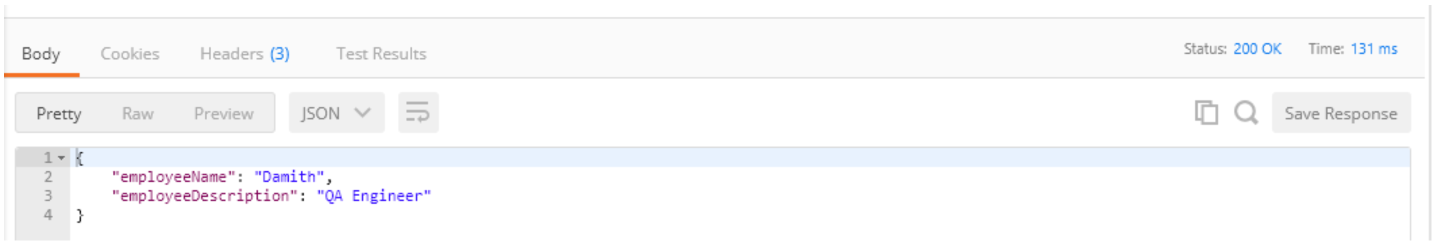
Database



Result Grid | Filter Rows: | Edit:

	employee_id	employee_description	employee_name
▶	1	Software Engineer	viraj
	2	QA Engineer	Damith
	3	Senior Software Engi...	Udara
	4	Business Analysis	Sachinthi
✱	NULL	NULL	NULL

Response



Body | Cookies | Headers (3) | Test Results | Status: 200 OK | Time: 131 ms

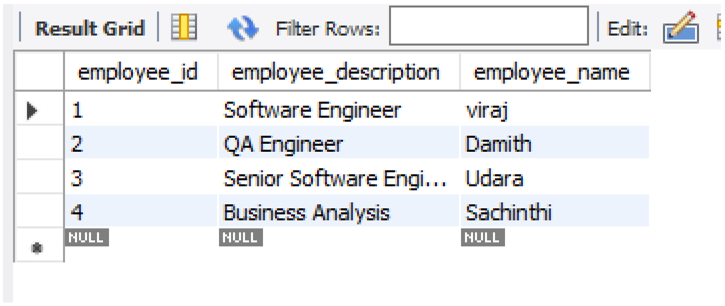
Pretty | Raw | Preview | JSON | Save Response

```
{
    "employeeName": "Damith",
    "employeeDescription": "QA Engineer"
}
```

DELETE

localhost:8883/employee/delete/2

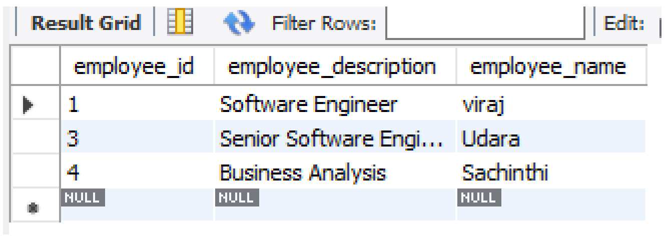
Database (Before delete the record which has the employee Id is 2)



Result Grid | Filter Rows: | Edit:

	employee_id	employee_description	employee_name
▶	1	Software Engineer	viraj
	2	QA Engineer	Damith
	3	Senior Software Engi...	Udara
	4	Business Analysis	Sachinthi
✱	NULL	NULL	NULL

Database (After delete the record which has the employee Id as 2)



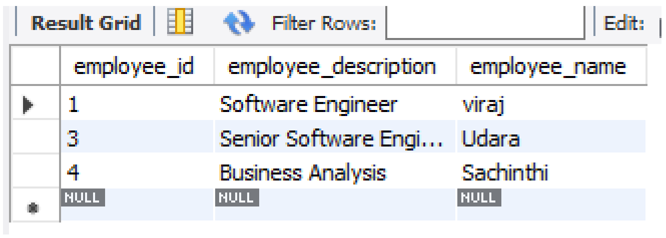
Result Grid | Filter Rows: | Edit:

employee_id	employee_description	employee_name
1	Software Engineer	viraj
3	Senior Software Engi...	Udara
4	Business Analysis	Sachinthi
NULL	NULL	NULL

PUT

localhost:8883/employee/update

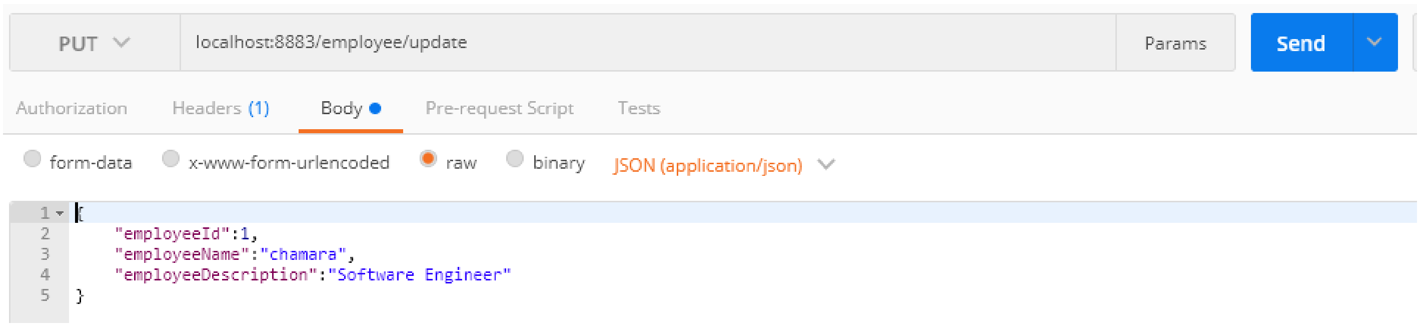
Database (Before update the record which has the id as 1)



Result Grid | Filter Rows: | Edit:

employee_id	employee_description	employee_name
1	Software Engineer	viraj
3	Senior Software Engi...	Udara
4	Business Analysis	Sachinthi
NULL	NULL	NULL

Request

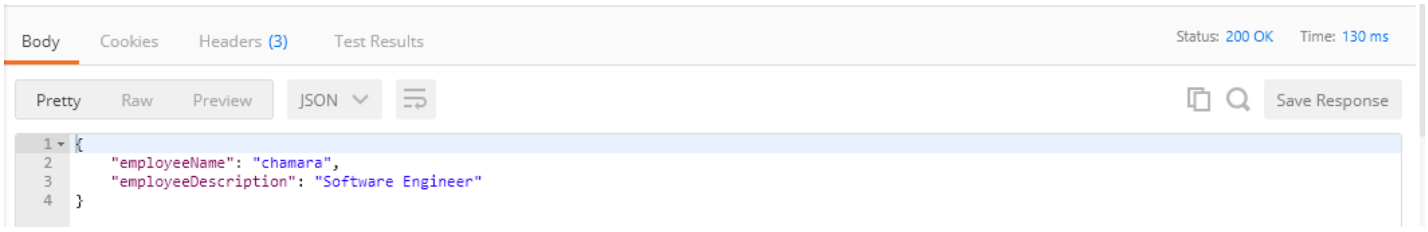


PUT localhost:8883/employee/update

Body: raw, JSON (application/json)

```
{
    "employeeId":1,
    "employeeName":"chamara",
    "employeeDescription":"Software Engineer"
}
```

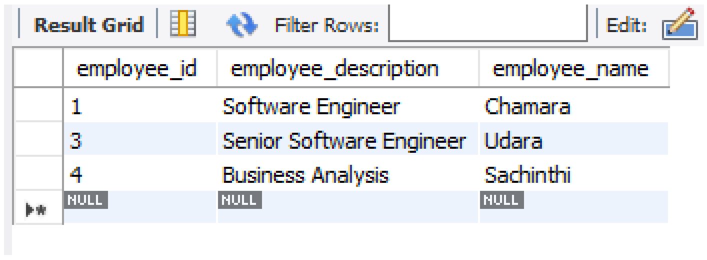
Response



Status: 200 OK Time: 130 ms

```
{
    "employeeName": "chamara",
    "employeeDescription": "Software Engineer"
}
```

Database (After update the record which has the id as 1)



employee_id	employee_description	employee_name
1	Software Engineer	Chamara
3	Senior Software Engineer	Udara
4	Business Analysis	Sachinthi
NULL	NULL	NULL