grokonez START HERE

Spring Boot + Vue.js example | Spring Data JPA + REST + MySQL CRUD

Spring Boot + Vue.js example | Spring Data JPA + REST + MySQL CRUD

In this tutorial, we show you Vue.js Http Client & Spring Boot Server example that uses Spring JPA to do CRUD with MySQL and Vue.js as a front-end technology to make request and receive response.

Related Posts:
– How to use Spring JPA MySQL | Spring Boot
– Vue Router example – with Nav Bar, Dynamic Route & Nested Routes

Contents [hide]

Technologies
Overview
    Demo
    1. Spring Boot Server
    2. Vue.js Client
Practice
    1. Spring Boot Server
        1.1 Dependency
        1.2 Data Model
        1.3 JPA Repository
        1.4 REST Controller
        1.5 Configuration for Spring Datasource & JPA properties
    2. Vue.js Client
        2.0 Setup Vue Project & Router
            Init Project
            Add Vue Router to Project
            Define Routes
            App template with Navbar and router-view
        2.1 Initialize HTTP Client
        2.2 Components
            List of Items
            Item Details
            Add Item
            Search Items
        2.3 Configure Port for Vue App
    Run
Source Code

Technologies

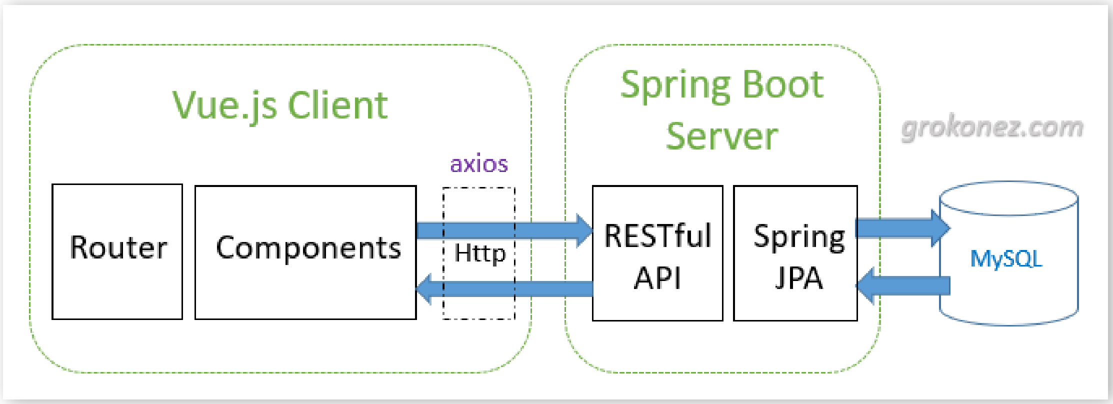
– Java 1.8
– Maven 3.3.9
– Spring Tool Suite – Version 3.8.4.RELEASE
– Spring Boot: 2.0.5.RELEASE

– Vue 2.5.17
– Vue Router 3

– Axios 0.18.0

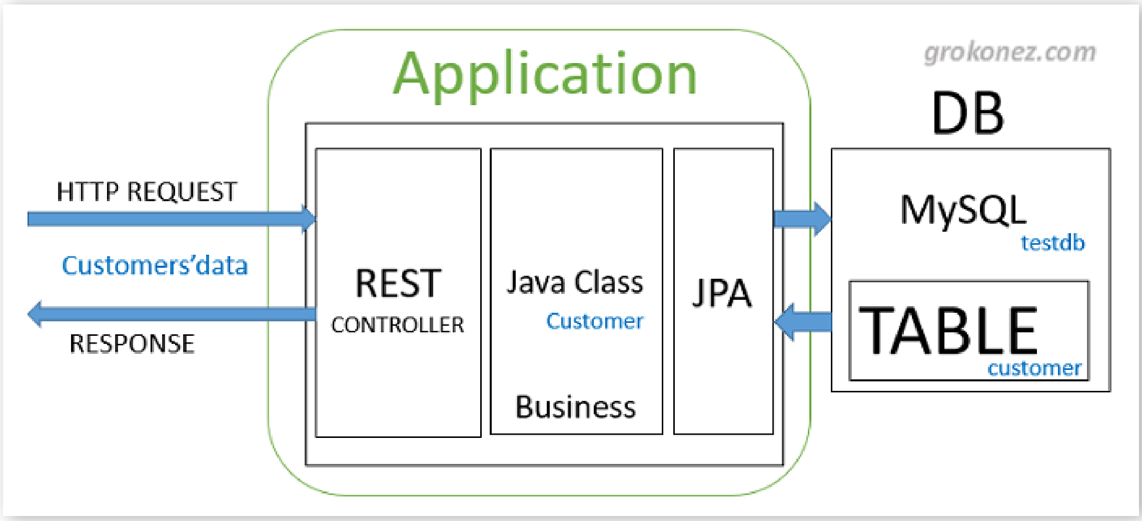
Overview

This is full-stack Architecture:

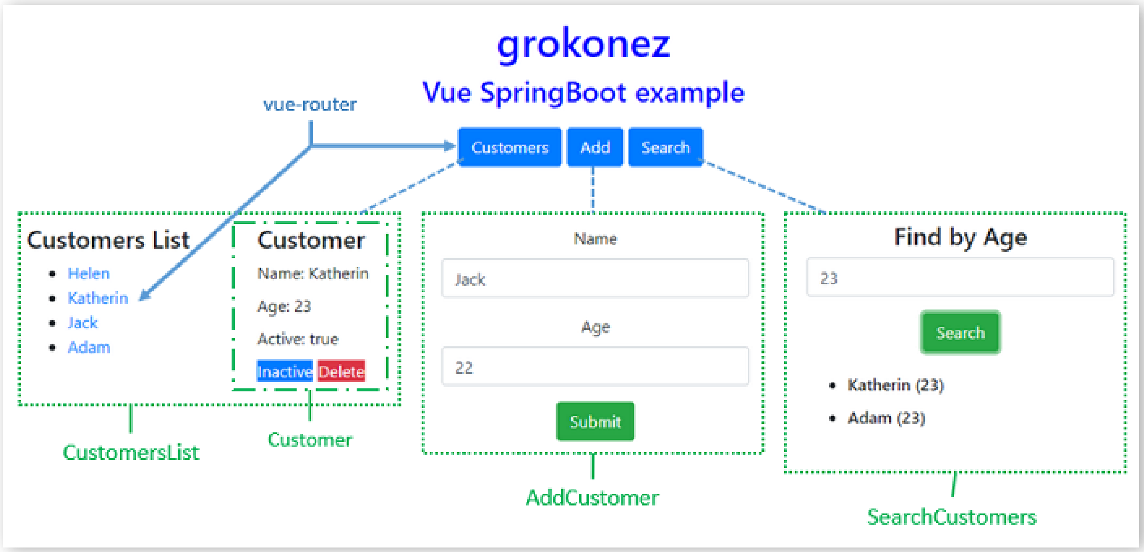


Demo

1. Spring Boot Server

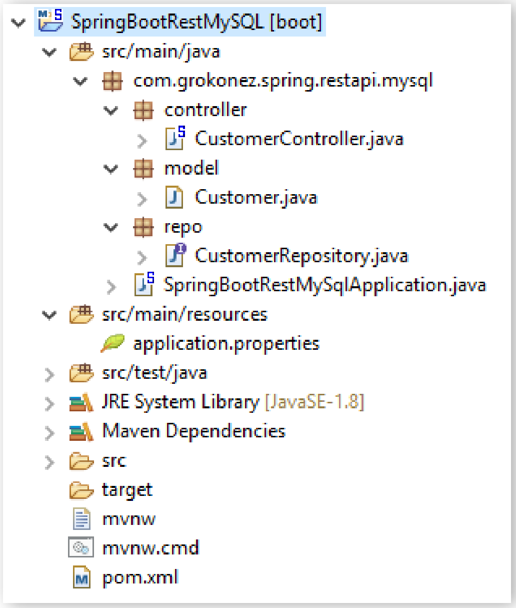


2. Vue.js Client



Practice

1. Spring Boot Server


```
SpringBootRestMySQL [boot]
  src/main/java
    com.grokonez.spring.restapi.mysql
      controller
        CustomerController.java
      model
        Customer.java
      repo
        CustomerRepository.java
      SpringBootRestMySqlApplication.java
  src/main/resources
    application.properties
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  mvnw
  mvnw.cmd
  pom.xml
```

– **Customer** class corresponds to entity and table **customer**.
– **CustomerRepository** is an interface extends **CrudRepository**, will be autowired in **CustomerController** for implementing repository methods and custom finder methods.
– **CustomerController** is a REST Controller which has request mapping methods for RESTful requests such as: `getAllCustomers`, `postCustomer`, `deleteCustomer`, `findByAge`, `updateCustomer`.

– Configuration for Spring Datasource and Spring JPA properties in **application.properties**
– **Dependencies** for **Spring Boot** and **MySQL** in **pom.xml**

1.1 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-web</artifactId>
</dependency>

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

1.2 Data Model

model/Customer.java

```
package com.grokonez.spring.restapi.mysql.model;

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 = "customer")
public class Customer {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;

    @Column(name = "name")
    private String name;

    @Column(name = "age")
    private int age;

    @Column(name = "active")
    private boolean active;

    public Customer() {
    }

    public Customer(String name, int age) {
        this.name = name;
        this.age = age;
        this.active = false;
    }

    public long getId() {
        return id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getAge() {
        return this.age;
    }

    public boolean isActive() {
        return active;
    }

    public void setActive(boolean active) {
        this.active = active;
    }

    @Override
    public String toString() {
        return "Customer [id=" + id + ", name=" + name + ", age=" + age + ", active=" + active
    }
}
```

1.3 JPA Repository

repo/CustomerRepository.java

```
package com.grokonez.spring.restapi.mysql.repo;

import java.util.List;

import org.springframework.data.repository.CrudRepository;
import com.grokonez.spring.restapi.mysql.model.Customer;

public interface CustomerRepository extends CrudRepository {
    List findByAge(int age);
}
```

1.4 REST Controller

controller/CustomerController.java

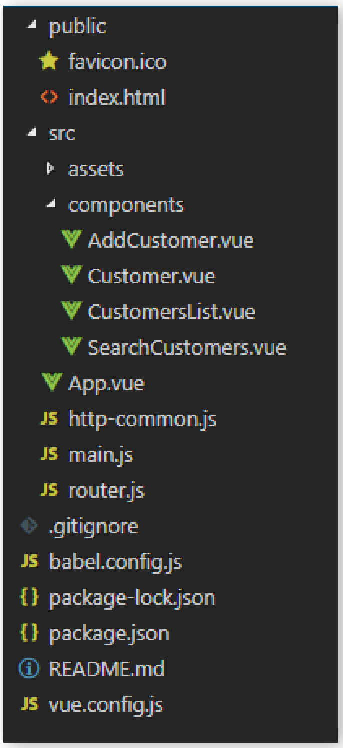
```
package com.grokonez.spring.restapi.mysql.controller;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
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.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.grokonez.spring.restapi.mysql.model.Customer;
import com.grokonez.spring.restapi.mysql.repo.CustomerRepository;

@CrossOrigin(origins = "http://localhost:4200")
@RestController
@RequestMapping("/api")
public class CustomerController {

    @Autowired
    CustomerRepository repository;

    @GetMapping("/customers")
    public List getAllCustomers() {
        System.out.println("Get all Customers...");

        List customers = new ArrayList<>();
        repository.findAll().forEach(customers::add);

        return customers;
    }

    @PostMapping("/customer")
    public Customer postCustomer(@RequestBody Customer customer) {

        Customer _customer = repository.save(new Customer(customer.getName(), customer.getAge(
        return _customer;
    }

    @DeleteMapping("/customer/{id}")
    public ResponseEntity deleteCustomer(@PathVariable("id") long id) {
        System.out.println("Delete Customer with ID = " + id + "...");

        repository.deleteById(id);

        return new ResponseEntity<>("Customer has been deleted!", HttpStatus.OK);
    }

    @GetMapping("customers/age/{age}")
    public List findByAge(@PathVariable int age) {

        List customers = repository.findByAge(age);
        return customers;
    }

    @PutMapping("/customer/{id}")
    public ResponseEntity updateCustomer(@PathVariable("id") long id, @RequestBody Customer cu
        System.out.println("Update Customer with ID = " + id + "...");

        Optional customerData = repository.findById(id);

        if (customerData.isPresent()) {
            Customer _customer = customerData.get();
            _customer.setName(customer.getName());
            _customer.setAge(customer.getAge());
            _customer.setActive(customer.isActive());
            return new ResponseEntity<>(repository.save(_customer), HttpStatus.OK);
        } else {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }
}
```

1.5 Configuration for Spring Datasource & JPA properties

application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/testdb?useSSL=false
spring.datasource.username=root
spring.datasource.password=123456
spring.jpa.generate-ddl=true
```

2. Vue.js Client



```
public
  favicon.ico
  index.html
src
  assets
  components
    AddCustomer.vue
    Customer.vue
    CustomersList.vue
    SearchCustomers.vue
  App.vue
  http-common.js
  main.js
  router.js
.gitignore
babel.config.js
package-lock.json
package.json
README.md
vue.config.js
```

– **package.json** with 3 main modules: `vue`, `vue-router`, `axios`.
– 4 components: ***CustomersList***, ***Customer***, ***AddCustomer***, ***SearchCustomer***.
– **router.js** defines `routes`, each route has a path and maps to a component.
– **http-common.js** initializes HTTP Client with `baseUrl` and `headers` for axios HTTP methods.
– **vue.config.js** configures `port` for Vue App.

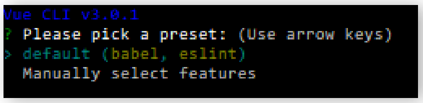
For more details about how to use Vue Router in this example, please visit:
Vue Router example – with Nav Bar, Dynamic Route & Nested Routes

2.0 Setup Vue Project & Router

Init Project

Point cmd to the folder you want to save Project folder, run command:
`vue create vue-springboot`

You will see 2 options, choose **default**:



```
Vue CLI v3.0.1
? Please pick a preset: (Use arrow keys)
> default (babel, eslint)
  Manually select features
```

Add Vue Router to Project

– Run command: `npm install vue-router`.
– Import `router` to **src/main.js**:

```
import Vue from "vue";
import App from "./App.vue";
import router from './router'

Vue.config.productionTip = false;

new Vue({
  router, // inject the router to make whole app router-aware
  render: h => h(App)
}).$mount("#app");
```

Define Routes

src/router.js:

```
import Vue from "vue";
import Router from "vue-router";
import CustomersList from "./components/CustomersList.vue";
import AddCustomer from "./components/AddCustomer.vue";
import SearchCustomers from "./components/SearchCustomers.vue";
import Customer from "./components/Customer.vue";

Vue.use(Router);

export default new Router({
  mode: "history",
  routes: [
    {
      path: "/",
      name: "customers",
      alias: "/customer",
      component: CustomersList,
      children: [
        {
          path: "/customer/:id",
          name: "customer-details",
          component: Customer,
          props: true
        }
      ]
    },
    {
      path: "/add",
      name: "add",
      component: AddCustomer
    },
    {
      path: "/search",
      name: "search",
      component: SearchCustomers
    }
  ]
});
```

App template with Navbar and router-view

src/App.vue:

```
<template>
    <div id="app" class="container-fluid">
```

```
        <div class="site-info">
            <h1>grokonez</h1>
            <h3>Vue SpringBoot example</h3>
        </div>
        <nav>
            <router-link class="btn btn-primary" to="/">Customers</router-link>
            <router-link class="btn btn-primary" to="/add">Add</router-link>
            <router-link class="btn btn-primary" to="/search">Search</router-link>
        </nav>
        <br/>
        <router-view/>
    </div>
</template>

<script>
export default {
  name: "app"
};
</script>

<style>
.site-info {
  color: blue;
  margin-bottom: 20px;
}

.btn-primary {
  margin-right: 5px;
}

.container-fluid {
  text-align: center;
}
</style>
```

2.1 Initialize HTTP Client

Install **axios** with command: `npm install axios`.
Then create *http-common.js* file:

```
import axios from "axios";

export default axios.create({
  baseURL: "http://localhost:8080/api",
  headers: {
    "Content-type": "application/json",
  }
});
```

2.2 Components

List of Items

components/CustomersList.vue

```
<template>
    <div class="list row">
        <div class="col-md-6">
            <h4>Customers List</h4>
            <ul>
                <li v-for="(customer, index) in customers" :key="index">
                    <router-link :to="{
                            name: 'customer-details',
                            params: { customer: customer, id: customer.id }
                        }">
                            {{customer.name}}
                    </router-link>
                </li>
            </ul>
        </div>
        <div class="col-md-6">
            <router-view @refreshData="refreshList"></router-view>
        </div>
    </div>
</template>

<script>
import http from "../http-common";

export default {
  name: "customers-list",
  data() {
    return {
      customers: []
    };
  },
  methods: {
    /* eslint-disable no-console */
    retrieveCustomers() {
      http
        .get("/customers")
        .then(response => {
          this.customers = response.data; // JSON are parsed automatically.
          console.log(response.data);
        })
        .catch(e => {
          console.log(e);
        });

    },
    refreshList() {
      this.retrieveCustomers();
    }
    /* eslint-enable no-console */
  },
  mounted() {
    this.retrieveCustomers();
  }
};
</script>

<style>
.list {
  text-align: left;
  max-width: 450px;
  margin: auto;
}
</style>
```

Item Details

components/Customer.vue

```
<template>
  <div v-if="this.customer">
    <h4>Customer</h4>
    <div>
      <label>Name: </label> {{this.customer.name}}
    </div>
    <div>
      <label>Age: </label> {{this.customer.age}}
    </div>
    <div>
      <label>Active: </label> {{this.customer.active}}
    </div>

    <span v-if="this.customer.active"
      v-on:click="updateActive(false)"

      class="button is-small btn-primary">Inactive</span>
    <span v-else
      v-on:click="updateActive(true)"
      class="button is-small btn-primary">Active</span>

    <span class="button is-small btn-danger" v-on:click="deleteCustomer()">Delete</span>
  </div>
  <div v-else>
    <br/>
    <p>Please click on a Customer...</p>
  </div>
</template>

<script>
import http from "../http-common";

export default {
  name: "customer",
  props: ["customer"],
  methods: {
    /* eslint-disable no-console */
    updateActive(status) {
      var data = {
        id: this.customer.id,
        name: this.customer.name,
        age: this.customer.age,
        active: status
      };

      http
        .put("/customer/" + this.customer.id, data)
        .then(response => {
          this.customer.active = response.data.active;
          console.log(response.data);
        })
        .catch(e => {
          console.log(e);
        });
    },
    deleteCustomer() {
      http
        .delete("/customer/" + this.customer.id)
        .then(response => {
          console.log(response.data);
          this.$emit("refreshData");
          this.$router.push('/');
        })
        .catch(e => {
          console.log(e);
        });
    }
    /* eslint-enable no-console */
  }

};
</script>
```

Add Item

components/AddCustomer.vue

```
<template>
  <div class="submitform">
    <div v-if="!submitted">
        <div class="form-group">
          <label for="name">Name</label>
          <input type="text" class="form-control" id="name" required v-model="customer.name" n
        </div>

        <div class="form-group">
          <label for="age">Age</label>
          <input type="number" class="form-control" id="age" required v-model="customer.age" n
        </div>

        <button v-on:click="saveCustomer" class="btn btn-success">Submit</button>
    </div>

    <div v-else>
      <h4>You submitted successfully!</h4>
      <button class="btn btn-success" v-on:click="newCustomer">Add</button>
    </div>
  </div>
</template>

<script>
import http from "../http-common";

export default {
  name: "add-customer",
  data() {
    return {
      customer: {
        id: 0,
        name: "",
        age: 0,
        active: false
      },
      submitted: false
    };
  },
  methods: {
    /* eslint-disable no-console */
    saveCustomer() {
      var data = {
        name: this.customer.name,
        age: this.customer.age
      };

      http
        .post("/customer", data)
        .then(response => {
          this.customer.id = response.data.id;
          console.log(response.data);
        })
        .catch(e => {
          console.log(e);
        });

      this.submitted = true;
    },
    newCustomer() {
      this.submitted = false;
      this.customer = {};
    }
    /* eslint-enable no-console */
  }
};
</script>
```

```
<style>
.submitform {
  max-width: 300px;
  margin: auto;
}
</style>
```

Search Items

components/SearchCustomers.vue

```
<template>
  <div class="searchform">
    <h4>Find by Age</h4>
    <div class="form-group">
      <input type="number" class="form-control" id="age" required v-model="age" name="age">
    </div>

    <div class="btn-group">
      <button v-on:click="searchCustomers" class="btn btn-success">Search</button>
    </div>

    <ul class="search-result">
      <li v-for="(customer, index) in customers" :key="index">
        <h6>{{customer.name}} ({{customer.age}})</h6>
      </li>
    </ul>
  </div>
</template>

<script>
import http from "../http-common";

export default {
  name: "search-customer",
  data() {
    return {
      age: 0,
      customers: []
    };
  },
  methods: {
    /* eslint-disable no-console */
    searchCustomers() {
      http
        .get("/customers/age/" + this.age)
        .then(response => {
          this.customers = response.data; // JSON are parsed automatically.
          console.log(response.data);
        })
        .catch(e => {
          console.log(e);
```

```
            console.log(e);
          });
      }
      /* eslint-enable no-console */
    }
  };
</script>

<style>
.searchform {
  max-width: 300px;
  margin: auto;
}
.search-result {
  margin-top: 20px;
  text-align: left;
}
</style>
```

2.3 Configure Port for Vue App

vue.config.js

```
module.exports = {
  devServer: {
    port: 4200
  }
}
```

Run

– Spring Boot Server: `mvn clean install` and `mvn spring-boot:run`.
– Vue.js Client: `npm run serve`.

Open Browser with Url: `http://localhost:4200/`.

Source Code

– SpringBootRestMySQL
– vue-springboot

By grokonez | September 16, 2018.
Last updated on **April 3, 2021**.

Related Posts

- Spring Boot + Vue.js example | Spring Data JPA + REST + PostgreSQL CRUD
- Vue.js + Spring Boot + H2 Database [embedded mode] example | Spring Data JPA + RestAPIs CRUD example
- Vue.js + Spring Boot example | Spring Data JPA + REST + MariaDB CRUD
- Vue.js + Spring Boot example | Spring Data Cassandra + RestApi CRUD example
- Spring Boot + Vue.js example | Spring Data MongoDB + RestApi CRUD
- Django CRUD Application with VueJs as front-end | VueJs + Django Rest Framework + MySQL example – Part 3: VueJs Client
- Django CRUD Application with VueJs as front-end | VueJs + Django Rest Framework + MySQL example – Part 1: Overview
- Vue.js + Nodejs/Express RestAPIs – Sequelize ORM + MySQL CRUD example

Post Tags

integrate spring boot vue | spring boot vue 2 example | spring boot vue crud | spring boot vue example | spring boot vue tutorial | vue http client | vue mysql | vue rest client | vue router spring boot vue spring data jpa mysql | vue spring jpa | vue spring jpa mysql

13 thoughts on “Spring Boot + Vue.js example | Spring Data JPA + REST + MySQL CRUD”

xuanning meng
December 5, 2018 at 6:20 am

This tutorial doesn’t contain the css configuration like “class = button is-small btn-danger”, class=“btn btn-primary”. Could you tell me how to make them, cause I can’t show the button and click them to delete…

grokonez
December 5, 2018 at 11:17 am

Hi xuanning meng,

You can add Bootstrap to **public/index.html** with one line of code:

```
...
```

Regards,
grokonez.

mariano
December 20, 2018 at 8:59 pm

hi,

your tutorial is real good,but i got a problem,wath kind a project should i create in STS,becouse i created a spring boot project with web,jpa,mysql,but i still i get not added the hibernate.jpa dependency,and even so cant make it run. Your example i did make it run ,and i was able to persiste on the mysql db,but when i create a spring project for my own i can’t make functioned,can you help me

thanks,
Mariano

grokonez
December 21, 2018 at 2:07 am

Hi Mariano,

Using STS, you can create new Project with **Spring Starter Project** ☺

Regards,
grokonez.

Mariano
December 30, 2018 at 12:44 am

Hi, Grokonez, I was finally able to create a project and make it run locally like your example!!
change this:

org.springframework.boot
spring-boot-starter-parent
2.1.1.RELEASE

to this:

org.springframework.boot
spring-boot-starter-parent
2.0.5.RELEASE

Could you make a tutorial to deploy this example on heroku,please? you already got one but with postgres db instead mysql,thanks again,
happy new year!!

Andy
January 17, 2019 at 5:40 am

Thank you for this sir
can we use vuetify instead??

Michael
February 14, 2019 at 3:24 pm

Thank you very much for this detailed tutorial. You helped me a lot.

grokonez
February 16, 2019 at 2:47 am

Hi Michael, we’re so glad to hear that ☺

Samuel
March 22, 2019 at 4:32 am

Thanks for the tutorial. Its really nice.

Question- How would you save the customer if the customer had a composite property say Contact as an object with OneToOne mapping. (is it possible to create the contact object with fields like Tel, email, adress in view and the before save, say customer.contact= contact? contact ==object of type Contact?

chayma
September 23, 2019 at 7:34 pm

Hi,
Your tutorial was very interesting but when i run the project i got two problems, the first one related to the CORS policy and the second is: Failed to load resource: the server responded with a status of 404 (Not Found) 8080/api/customers
How can i fix that ? thank you

bbaahhmvgo
April 4, 2020 at 6:08 am

yszkovuorgegwitdfpfiuujacmdnug

nztkdfrbbs
May 10, 2020 at 6:34 pm

rkknwleeadxgnldcogifbbskpiwsps

vvqqeyybde
May 29, 2020 at 1:53 pm

sjfgbwrlmyhujawbjpuxdnndauzwmm