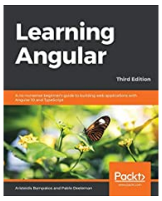
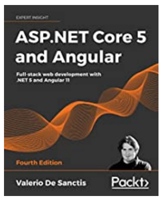
grokOnez START HERE

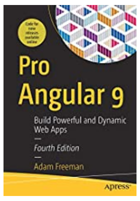
Spring Boot + Vue.js example | Spring Data MongoDB + RestApi CRUD

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


Learning Angular: A no-nonsense beginner's guide to building web a...
$34.99
(35)


ASP.NET Core 5 and Angular: Full-stack web development with .NET...
$17.19
(25)

Angular Projects: Build modern web apps by exploring Angular 12 with...
$34.99
(2)


Pro Angular 9: Build Powerful and Dynamic Web Apps
$45.95 ~~$69.99~~
(46)

Ads by Amazon

Related Posts:
– [Spring MongoOperations to access MongoDB]
– [How to use SpringData MongoRepository to interact with MongoDB]
– [How to build SpringBoot MongoDb RestfulApi]

– [Vue Router example – with Nav Bar, Dynamic Route & Nested Routes]

Technologies

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

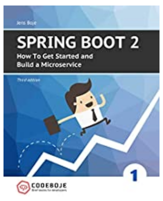
– Vue 2.5.17
– Vue Router 3
– Axios 0.18.0

Overview

This is full-stack Architecture:

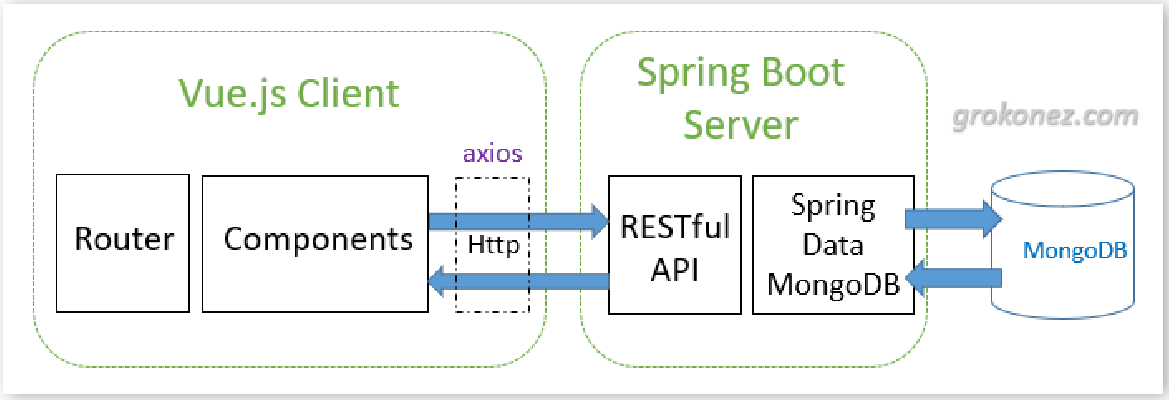
Shop Related Products

Learning Spring Boot 2.0 - Second Edition: Simplify the developmen…
$22.39
(53)


Spring Boot 2: How To Get Started and Build a Microservice - Thir…
$6.99
(11)

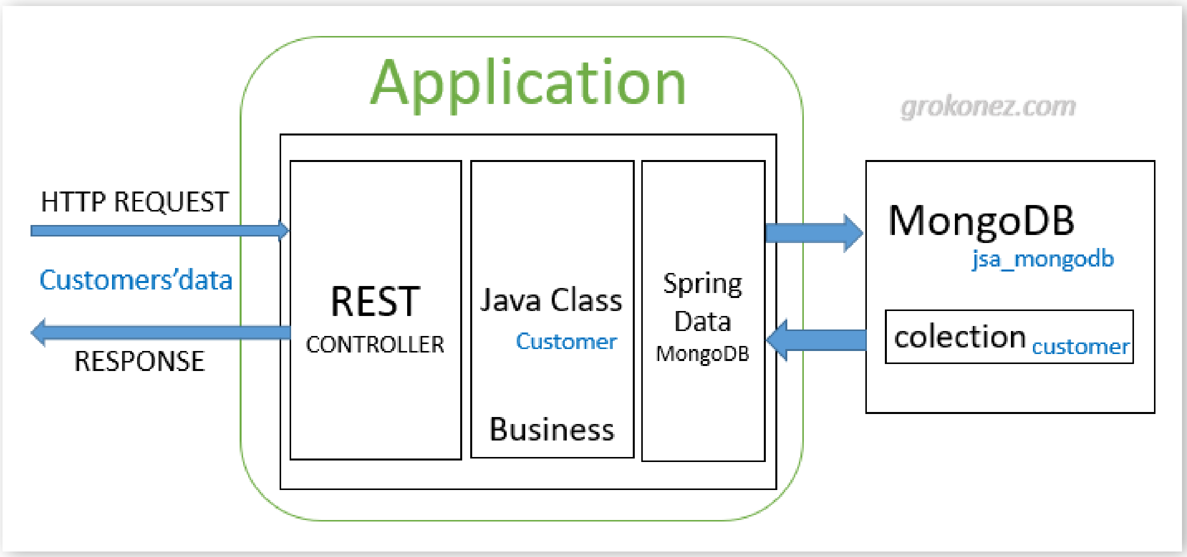
SpringBoot揭秘：快速构建微服务体系 (Chinese Edition)
$7.08

Bebe Girls' Microsuede Winter Boots with Back Lace Up, Size…
$19.29
(119)

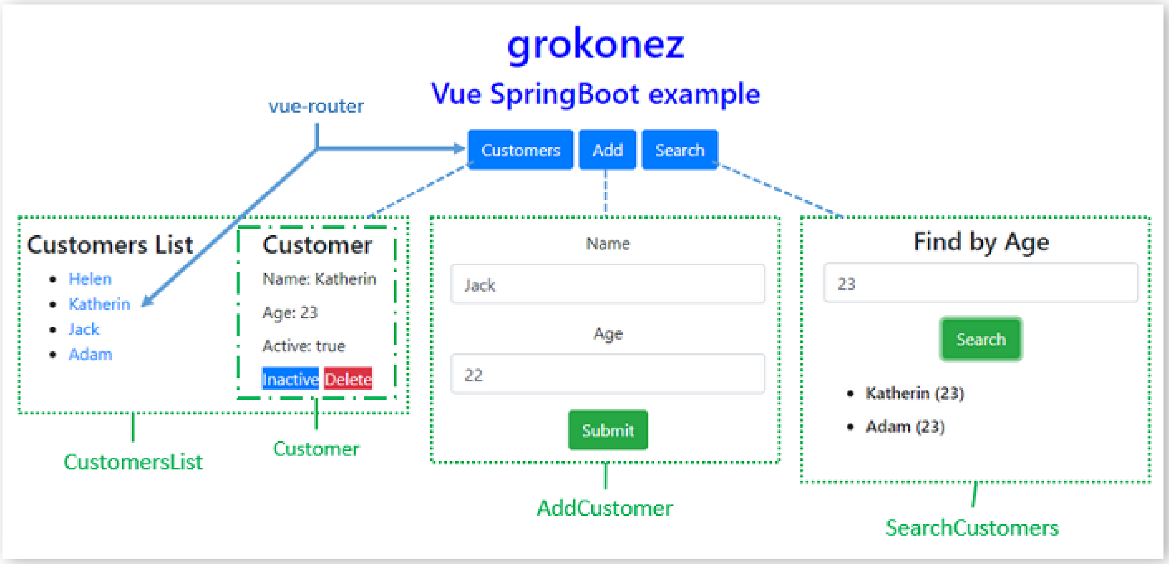


Demo

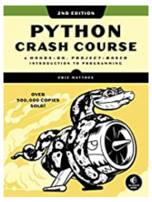
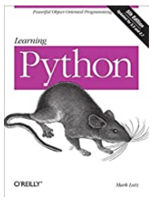
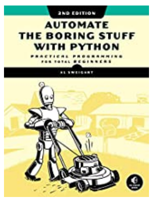
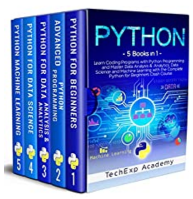
Spring Boot Server

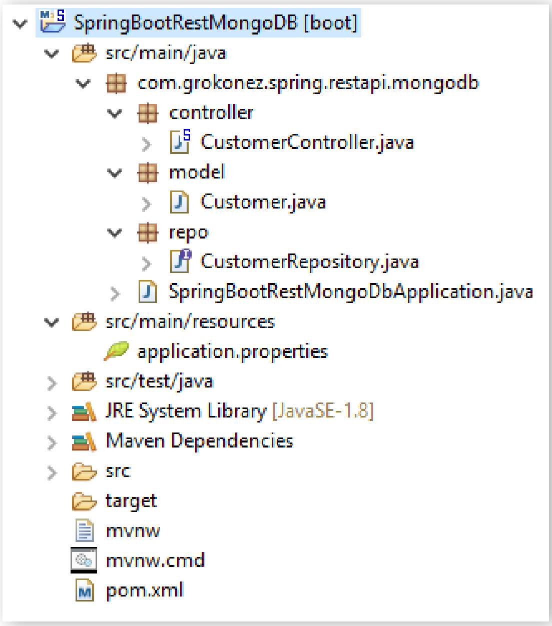


Vue.js Client



Spring Boot Server



```
SpringBootRestMongoDB [boot]
  src/main/java
    com.grokonez.spring.restapi.mongodb
      controller
        CustomerController.java
      model
        Customer.java
      repo
        CustomerRepository.java
      SpringBootRestMongoDbApplication.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 **MongoRepository**, 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 Data properties in **application.properties**
– **Dependencies** for **Spring Boot** and **MongoDb** in **pom.xml**

Dependency

```
 <dependency>
     <groupId>org.springframework.boot</groupId>
     <artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency>

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

Data Model

model/Customer.java

```java
package com.grokonez.spring.restapi.mongodb.model;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "customer")
public class Customer {
    @Id
    private String id;

    private String name;
    private int age;
    private boolean active;

    public Customer() {
    }

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

    public String 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=" + acti
    }
}
```

SpringJPA Repository

repo/CustomerRepository.java

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

import java.util.List;

import org.springframework.data.mongodb.repository.MongoRepository;

import com.grokonez.spring.restapi.mongodb.model.Customer;

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

SpringBoot REST Controller

controller/CustomerController.java

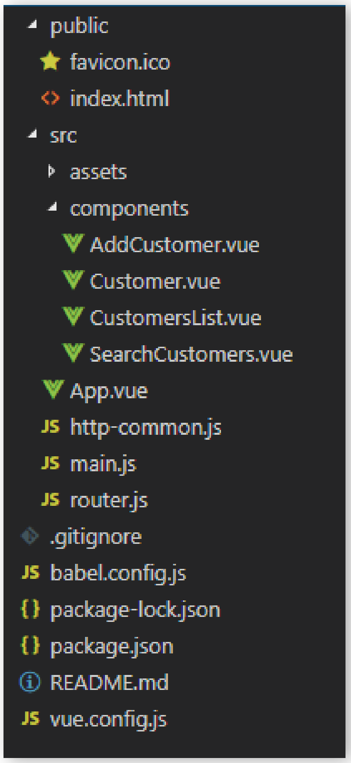
```
package com.grokonez.spring.restapi.mongodb.controller;

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.mongodb.model.Customer;
import com.grokonez.spring.restapi.mongodb.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.getAg
        return _customer;
    }

    @DeleteMapping("/customer/{id}")
    public ResponseEntity deleteCustomer(@PathVariable("id") String 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") String id, @RequestBody Custome
        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);
        }
    }
}
```

Configuration for Spring Datasource & Data MongoDb properties

application.properties

```
spring.data.mongodb.database=jsa_mongodb
spring.data.mongodb.port=27017
```

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
```

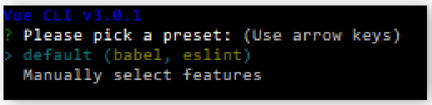
– **package.json** with 3 main modules: `vue`, `vue-router`, `axios`.
– 4 components: ***CustomersList***, ***Customer***, ***AddCustomer***, ***SearchCustomer***.
– **router.js** defines `routes`, each route has a path and maps to a component.
– **http-common.js** initializes HTTP Client with `baseUrl`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

Init Vue 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 Vue 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>
```

Initialize Vue 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",
  }
});
```

Vuejs 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>
```

Vue 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>
```

Vue 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"
        </div>

        <div class="form-group">
          <label for="age">Age</label>
          <input type="number" class="form-control" id="age" required v-model="customer.age"
        </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);
        });
    }
    /* eslint-enable no-console */
  }
};
</script>

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

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

– SpringBootRestMongoDB
– vue-springboot

By grokonez | September 18, 2018.
Last updated on **March 1, 2021**.

Related Posts

- Angular 8 Spring WebFlux MongoDB CRUD RestAPI
- Angular 9 Spring WebFlux CRUD RestAPI
- Angular 10 Spring WebFlux CRUD RestAPI
- Angular 11 Spring WebFlux MongoDB CRUD RestAPI
- Angular 12 Spring WebFlux MongoDB CRUD RestAPI
- Angular 9 + Nodejs/Express + Mongoose CRUD MongoDB – Get/Post/Put/Delete
- Angular 10 + Nodejs/Express + Mongoose CRUD MongoDB – Get/Post/Put/Delete
- Angular 11 + Nodejs/Express + Mongoose CRUD MongoDB – Get/Post/Put/Delete
- Angular 12 + Nodejs/Express + Mongoose CRUD MongoDB – Get/Post/Put/Delete
- React Node.js MongoDB CRUD Example – MERN Stack

Post Tags

MongoDB | mongodb crud | spring boot mongodb | spring boot vue 2 example | spring boot vue crud | spring boot vue example | spring boot vue tutorial | spring data | spring data mongodb | vue spring boot mongodb

2 thoughts on “Spring Boot + Vue.js example | Spring Data MongoDB + RestApi CRUD”

Gujju
September 20, 2018 at 1:18 pm

Hi Grokonez,
When I issued a command npm create , it did not do anything , instead it prompted me “Didi you mean this? : Update”. was there a typos error ?

Gujju
September 20, 2018 at 4:00 pm

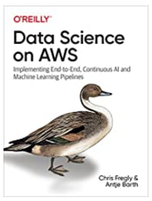
Never mind, I got it after dealing some configuration issue on Windows 10.
instead of “npm create vue-springboot” , it should be “vue create vue-springboot”.

(28) (417) (275) (740)

Python Data Science Handbook: Essential Tools for Working with…

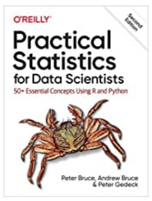
$33.50 ~~$69.99~~

(428)


Data Science on AWS: Implementing End-to-End, Cont…

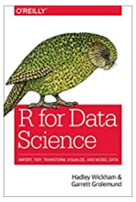
$37.56 ~~$69.99~~

(36)


Practical Statistics for Data Scientists: 50+ Essential Concept…

$30.18 ~~$69.99~~

(261)


R for Data Science: Import, Tidy, Transform, Visualize, and Model…

$44.99 ~~$49.99~~

(1057)

Ads by Amazon

grokonez

Home | Privacy Policy | Contact Us | Our Team

FOLLOW US

ABOUT US

We are passionate engineers in software development by Java Technology & Spring Framework. We believe that creating little good thing with specific orientation everyday can make great influence on the world someday.