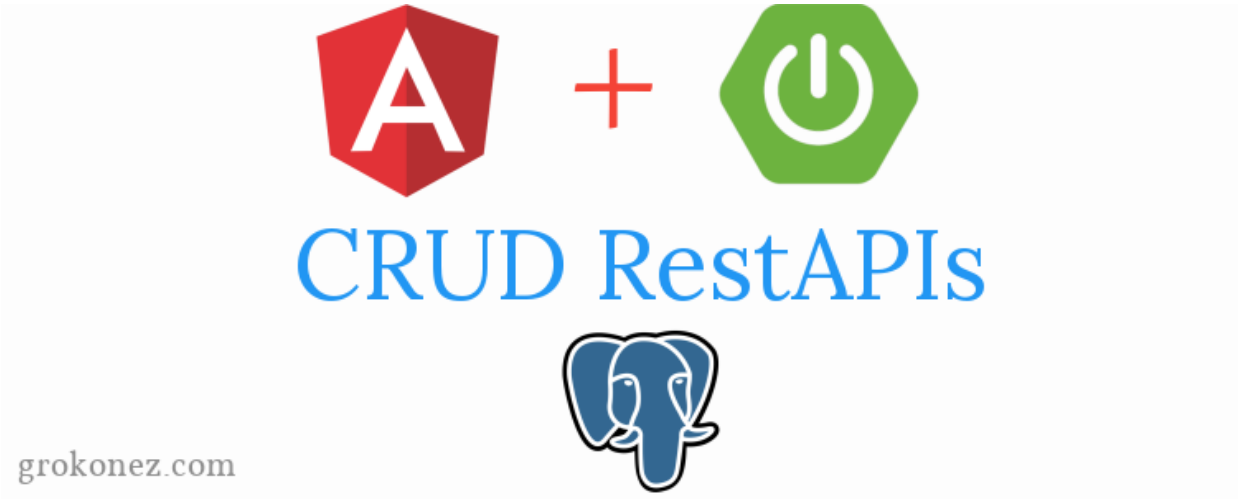
grokonez START HERE

# Spring Boot + Angular 6 example | Spring Data JPA + REST + PostgreSQL CRUD example




In this tutorial, we show you Angular 6 Http Client & Spring Boot Server example that uses Spring JPA to do CRUD with PostgreSQL and Angular 6 as a front-end technology to make request and receive response.

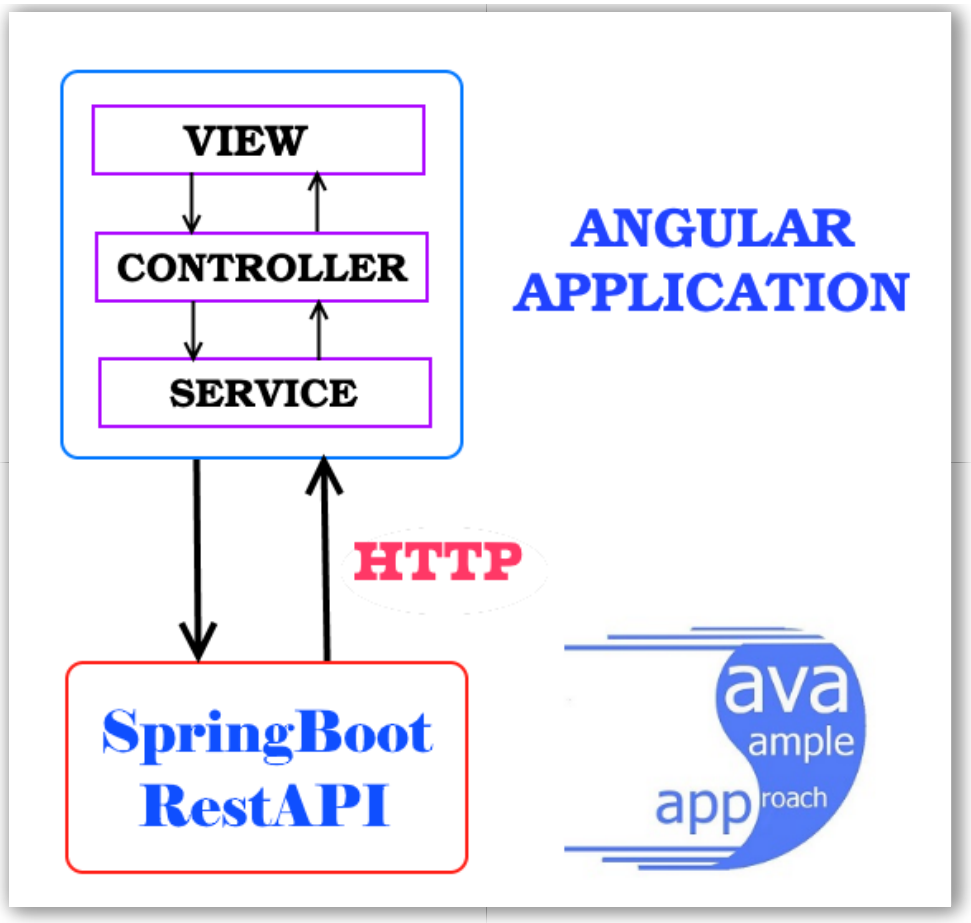
Related Posts:
– How to use Spring JPA with PostgreSQL | Spring Boot
– Spring JPA + PostgreSQL + AngularJS example | Spring Boot

**Contents** [hide]

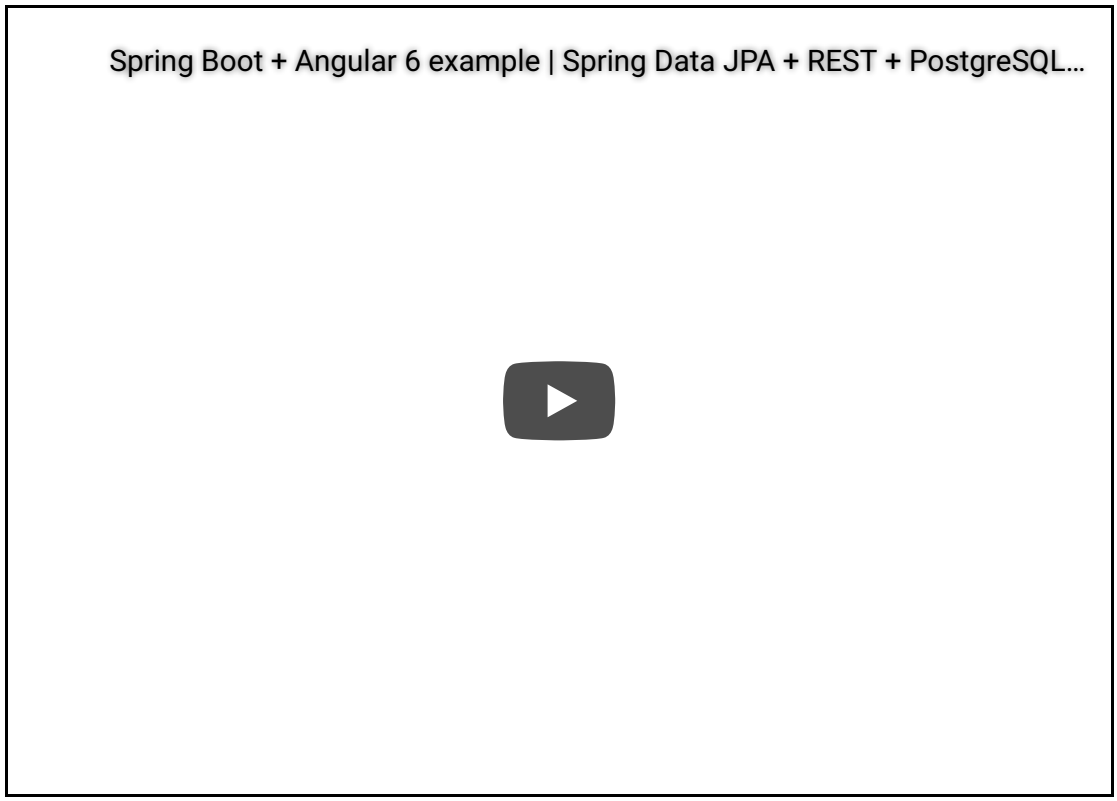
## I. Technologies

– Java 1.8
– Maven 3.3.9
– Spring Tool Suite – Version 3.8.4.RELEASE
– Spring Boot: 2.0.3.RELEASE
– Angular 6
– RxJS 6

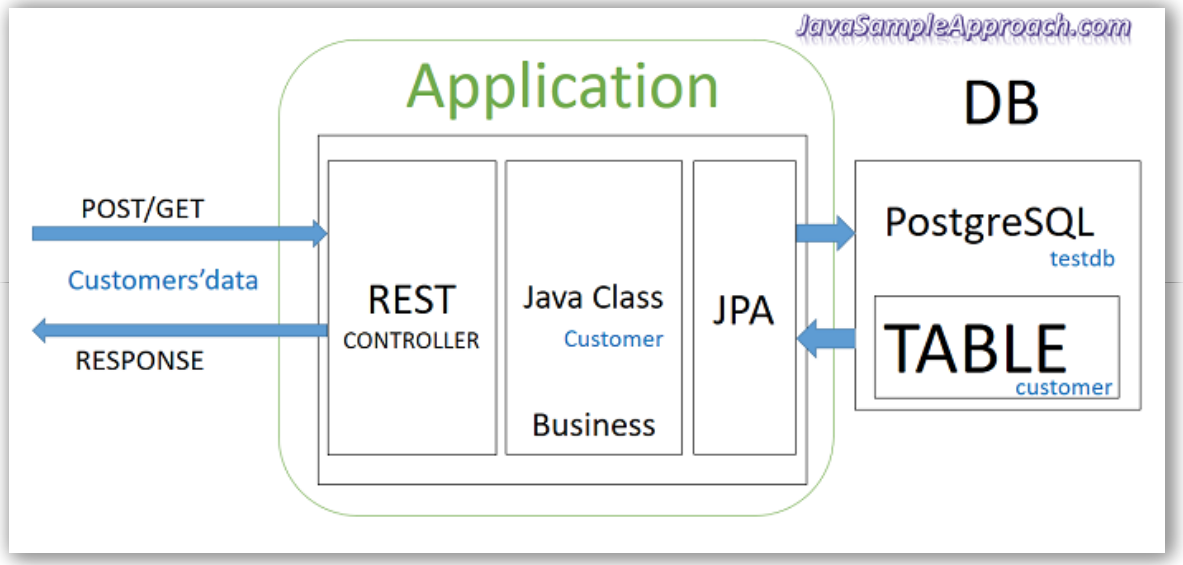
## II. Overview


VIEW
CONTROLLER
SERVICE
ANGULAR
APPLICATION
HTTP
SpringBoot
RestAPI
ava
ample
app
roach

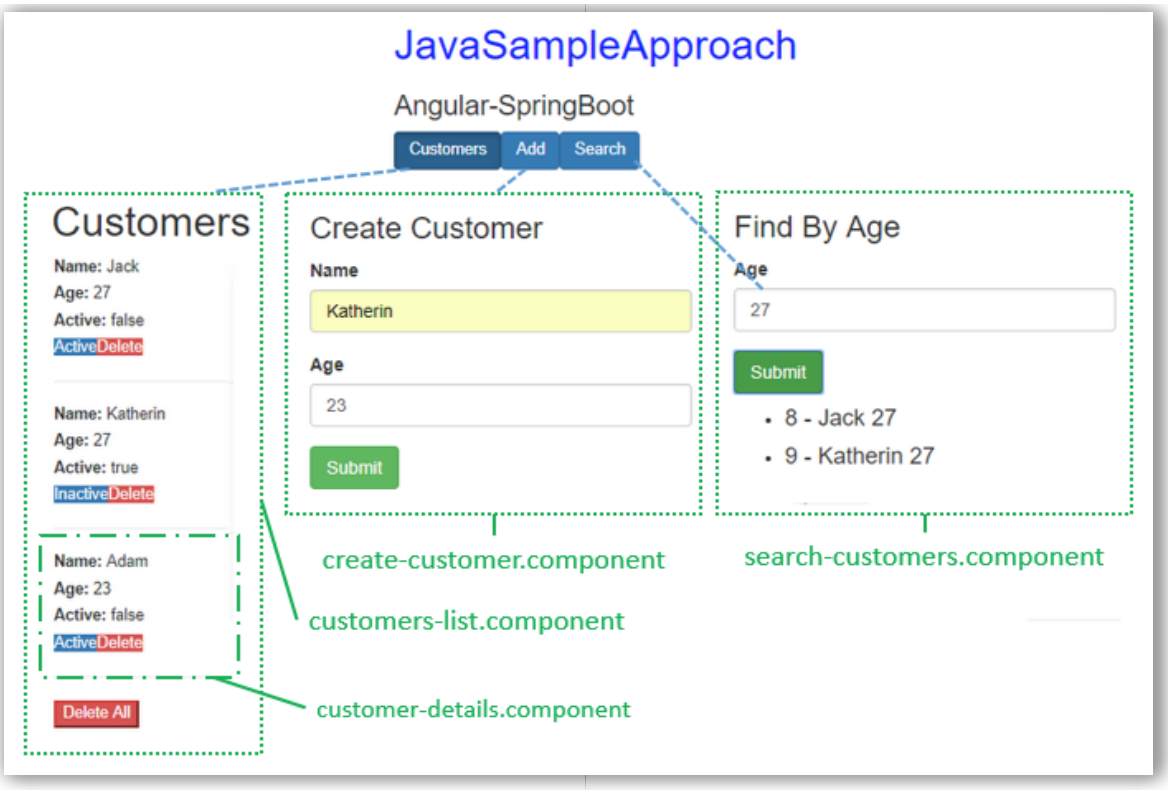
**Demo**




**1. Spring Boot Server**



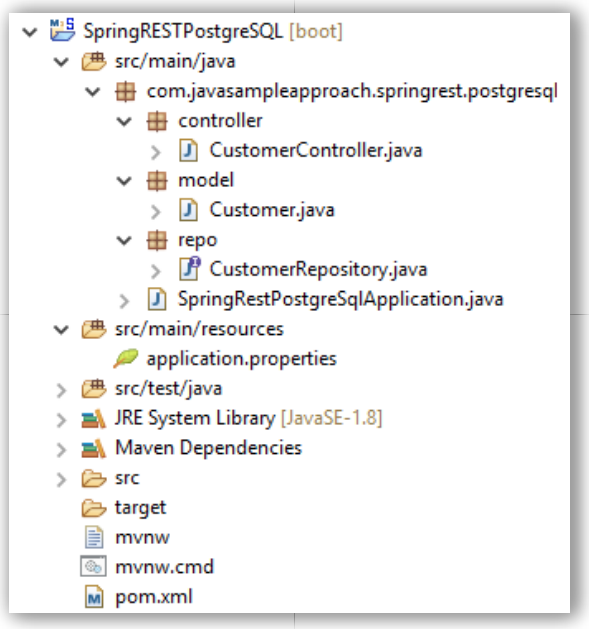

**2. Angular 6 Client**




## III. Practice

### 1. Project Structure

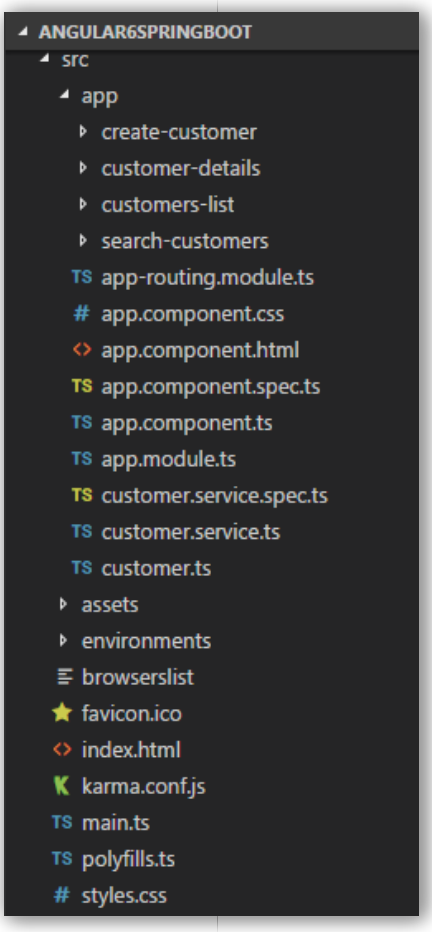
#### 1.1 Spring Boot Server




– **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`, `deleteAllCustomers`, `findByAge`, `updateCustomer`.
– Configuration for Spring Datasource and Spring JPA properties in **application.properties**
– **Dependencies** for **Spring Boot** and **PostgreSQL** in **pom.xml**

#### 1.2 Angular 6 Client




In this example, we focus on:
– 4 components: ***customers-list***, ***customer-details***, ***create-customer***, ***search-customer***.
– 3 modules: ***FormsModule***, ***HttpClientModule***, ***AppRoutingModule***.
– ***customer.ts***: class Customer (id, firstName, lastName)
– ***customer.service.ts***: Service for Http Client methods

### 2. How to do

#### 2.1 Spring Boot Server

#### 2.1.1 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>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency>
```

### 2.1.2 Customer – Data Model

*model/Customer.java*

```
package com.javasampleapproach.springrest.postgresql.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 + "]"
    }
}
```

### 2.1.3 JPA Repository

*repo/CustomerRepository.java*

```
package com.javasampleapproach.springrest.postgresql.repo;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import com.javasampleapproach.springrest.postgresql.model.Customer;

public interface CustomerRepository extends CrudRepository<Customer, Long> {
    List<Customer> findByAge(int age);
}
```

### 2.1.4 REST Controller

*controller/CustomerController.java*

```
package com.javasampleapproach.springrest.postgresql.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.javasampleapproach.springrest.postgresql.model.Customer;
import com.javasampleapproach.springrest.postgresql.repo.CustomerRepository;

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

    @Autowired
    CustomerRepository repository;

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

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

        return customers;
    }

    @PostMapping(value = "/customers/create")
    public Customer postCustomer(@RequestBody Customer customer) {

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

    @DeleteMapping("/customers/{id}")
    public ResponseEntity<String> 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);
    }

    @DeleteMapping("/customers/delete")
    public ResponseEntity<String> deleteAllCustomers() {
        System.out.println("Delete All Customers...");

        repository.deleteAll();

        return new ResponseEntity<>("All customers have been deleted!", HttpStatus.OK);
    }

    @GetMapping(value = "customers/age/{age}")
    public List<Customer> findByAge(@PathVariable int age) {

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

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

        Optional<Customer> 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);
        }
    }
}
```

#### 2.1.5 Configuration for Spring Datasource & JPA properties

*application.properties*

```
spring.datasource.url=jdbc:postgresql://localhost/testdb
spring.datasource.username=postgres
spring.datasource.password=123
spring.jpa.generate-ddl=true
```

### 2.2 Angular 6 Client

### 2.2.0 Create Service & Components

Run commands below:

- `ng g s customer`
- `ng g c create-customer`
- `ng g c customer-details`
- `ng g c customers-list`
- `ng g c search-customers`

On each Component selector, delete `app-` prefix, then change **tslint.json** `rules - "component-selector"` to **false**.

### 2.2.1 Model

*customer.ts*

```
export class Customer {
    id: number;
    name: string;
    age: number;
    active: boolean;
}
```

### 2.2.2 CustomerService

*customer.service.ts*

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class CustomerService {

  private baseUrl = 'http://localhost:8080/api/customers';

  constructor(private http: HttpClient) { }

  getCustomer(id: number): Observable<Object> {
    return this.http.get(`${this.baseUrl}/${id}`);
  }

  createCustomer(customer: Object): Observable<Object> {
    return this.http.post(`${this.baseUrl}` + `/create`, customer);
  }
```

```
  updateCustomer(id: number, value: any): Observable<Object> {
    return this.http.put(`${this.baseUrl}/${id}`, value);
  }

  deleteCustomer(id: number): Observable<any> {
    return this.http.delete(`${this.baseUrl}/${id}`, { responseType: 'text' });
  }

  getCustomersList(): Observable<any> {
    return this.http.get(`${this.baseUrl}`);
  }

  getCustomersByAge(age: number): Observable<any> {
    return this.http.get(`${this.baseUrl}/age/${age}`);
  }

  deleteAll(): Observable<any> {
    return this.http.delete(`${this.baseUrl}` + `/delete`, { responseType: 'text' });
  }
}
```

### 2.2.3 Components

– CustomerDetailsComponent:
*customer-details/customer-details.component.ts*

```
import { Component, OnInit, Input } from '@angular/core';
import { CustomerService } from '../customer.service';
import { Customer } from '../customer';

import { CustomersListComponent } from '../customers-list/customers-list.component';

@Component({
  selector: 'customer-details',
  templateUrl: './customer-details.component.html',
  styleUrls: ['./customer-details.component.css']
})
export class CustomerDetailsComponent implements OnInit {

  @Input() customer: Customer;

  constructor(private customerService: CustomerService, private listComponent: CustomersListComponen

  ngOnInit() {
  }

  updateActive(isActive: boolean) {
    this.customerService.updateCustomer(this.customer.id,
      { name: this.customer.name, age: this.customer.age, active: isActive })
      .subscribe(
        data => {
```

```
          console.log(data);
          this.customer = data as Customer;
        },
        error => console.log(error));
  }

  deleteCustomer() {
    this.customerService.deleteCustomer(this.customer.id)
      .subscribe(
        data => {
          console.log(data);
          this.listComponent.reloadData();
        },
        error => console.log(error));
  }
}
```

*customer-details/customer-details.component.html*

```
<div *ngIf="customer">
  <div>
    <label>Name: </label> {{customer.name}}
  </div>
  <div>
    <label>Age: </label> {{customer.age}}
  </div>
  <div>
    <label>Active: </label> {{customer.active}}
  </div>

  <span class="button is-small btn-primary" *ngIf='customer.active' (click)='updateActive(false)'>In

  <span class="button is-small btn-primary" *ngIf='!customer.active' (click)='updateActive(true)'>Ac

  <span class="button is-small btn-danger" (click)='deleteCustomer()'>Delete</span>

  <hr/>
</div>
```

– CustomersListComponent:
*customers-list/customers-list.component.ts*

```
import { Component, OnInit } from '@angular/core';
import { Observable } from 'rxjs';

import { CustomerService } from '../customer.service';
import { Customer } from '../customer';

@Component({
  selector: 'customers-list',
  templateUrl: './customers-list.component.html',
```

```
  styleUrls: ['./customers-list.component.css']
})
export class CustomersListComponent implements OnInit {

  customers: Observable<Customer[]>;

  constructor(private customerService: CustomerService) { }

  ngOnInit() {
    this.reloadData();
  }

  deleteCustomers() {
    this.customerService.deleteAll()
      .subscribe(
        data => {
          console.log(data);
          this.reloadData();
        },
        error => console.log('ERROR: ' + error));
  }

  reloadData() {
    this.customers = this.customerService.getCustomersList();
  }
}
```

*customers-list/customers-list.component.html*

```
<h1>Customers</h1>

<div *ngFor="let customer of customers | async" style="width: 300px;">
  <customer-details [customer]='customer'></customer-details>
</div>

<div>
  <button type="button" class="button btn-danger" (click)='deleteCustomers()'>Delete All</button>
</div>
```

– CreateCustomerComponent:
*create-customer/create-customer.component.ts*

```
import { Component, OnInit } from '@angular/core';

import { Customer } from '../customer';
import { CustomerService } from '../customer.service';

@Component({
  selector: 'create-customer',
  templateUrl: './create-customer.component.html',
  styleUrls: ['./create-customer.component.css']
```

```
})
export class CreateCustomerComponent implements OnInit {

  customer: Customer = new Customer();
  submitted = false;

  constructor(private customerService: CustomerService) { }

  ngOnInit() {
  }

  newCustomer(): void {
    this.submitted = false;
    this.customer = new Customer();
  }

  save() {
    this.customerService.createCustomer(this.customer)
      .subscribe(data => console.log(data), error => console.log(error));
    this.customer = new Customer();
  }

  onSubmit() {
    this.submitted = true;
    this.save();
  }
}
```

*create-customer/create-customer.component.html*

```
<h3>Create Customer</h3>
<div [hidden]="submitted" style="width: 300px;">
  <form (ngSubmit)="onSubmit()">
    <div class="form-group">
      <label for="name">Name</label>
      <input type="text" class="form-control" id="name" required [(ngModel)]="customer.name" name="n
    </div>

    <div class="form-group">
      <label for="age">Age</label>
      <input type="text" class="form-control" id="age" required [(ngModel)]="customer.age" name="age
    </div>

    <button type="submit" class="btn btn-success">Submit</button>
  </form>
</div>

<div [hidden]="!submitted">
  <h4>You submitted successfully!</h4>
  <button class="btn btn-success" (click)="newCustomer()">Add</button>
</div>
```

– SearchCustomersComponent:
*search-customers/search-customers.component.ts*

```
import { Component, OnInit } from '@angular/core';
import { Customer } from '../customer';
import { CustomerService } from '../customer.service';

@Component({
  selector: 'search-customers',
  templateUrl: './search-customers.component.html',
  styleUrls: ['./search-customers.component.css']
})
export class SearchCustomersComponent implements OnInit {

  age: number;
  customers: Customer[];

  constructor(private dataService: CustomerService) { }

  ngOnInit() {
    this.age = 0;
  }

  private searchCustomers() {
    this.dataService.getCustomersByAge(this.age)
      .subscribe(customers => this.customers = customers);
  }

  onSubmit() {
    this.searchCustomers();
  }
}
```

*search-customers/search-customers.component.html*

```
<h3>Find By Age</h3>
<div style="width: 300px;">
  <form (ngSubmit)="onSubmit()">
    <div class="form-group">
      <label for="lastname">Age</label>
      <input type="text" class="form-control" id="age" required [(ngModel)]="age" name="age">
    </div>

    <div class="btn-group">
      <button type="submit" class="btn btn-success">Submit</button>
    </div>
  </form>
</div>
<ul>
  <li *ngFor="let customer of customers">
    <h4>{{customer.id}} - {{customer.name}} {{customer.age}}</h4>
```

```
  </li>
</ul>
```

### 2.2.4 AppRoutingModule

*app-routing.module.ts*

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { CustomersListComponent } from './customers-list/customers-list.component';
import { CreateCustomerComponent } from './create-customer/create-customer.component';
import { SearchCustomersComponent } from './search-customers/search-customers.component';

const routes: Routes = [
    { path: '', redirectTo: 'customer', pathMatch: 'full' },
    { path: 'customer', component: CustomersListComponent },
    { path: 'add', component: CreateCustomerComponent },
    { path: 'findbyage', component: SearchCustomersComponent },
];

@NgModule({
    imports: [RouterModule.forRoot(routes)],
    exports: [RouterModule]
})

export class AppRoutingModule { }
```

And **AppComponent** HTML for routing:
*app.component.html*

```
<div style="padding: 20px;">
  <h1 style="color: blue">{{title}}</h1>
  <h3>{{description}}</h3>
  <nav>
    <a routerLink="customer" class="btn btn-primary active" role="button" routerLinkActive="active">
    <a routerLink="add" class="btn btn-primary active" role="button" routerLinkActive="active">Add</
    <a routerLink="findbyage" class="btn btn-primary active" role="button" routerLinkActive="active"
  </nav>
  <router-outlet></router-outlet>
</div>
```

### 2.2.5 AppModule

*app.module.ts*

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
```

```
import { CreateCustomerComponent } from './create-customer/create-customer.component';
import { CustomerDetailsComponent } from './customer-details/customer-details.component';
import { CustomersListComponent } from './customers-list/customers-list.component';
import { SearchCustomersComponent } from './search-customers/search-customers.component';
import { AppRoutingModule } from './app-routing.module';
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [
    AppComponent,
    CreateCustomerComponent,
    CustomerDetailsComponent,
    CustomersListComponent,
    SearchCustomersComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

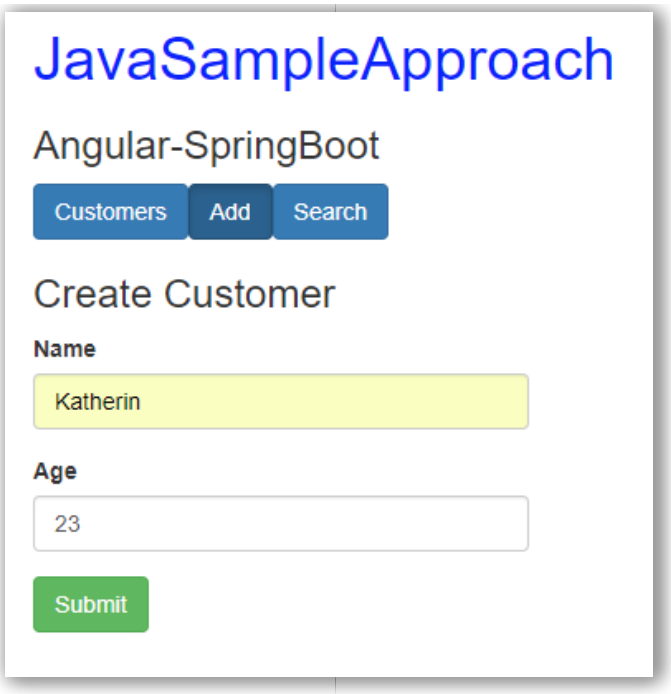
#### 3. Run & Check Result

– **Build** and **Run Spring Boot** project with commandlines: `mvn clean install` and `mvn spring-boot:run`.

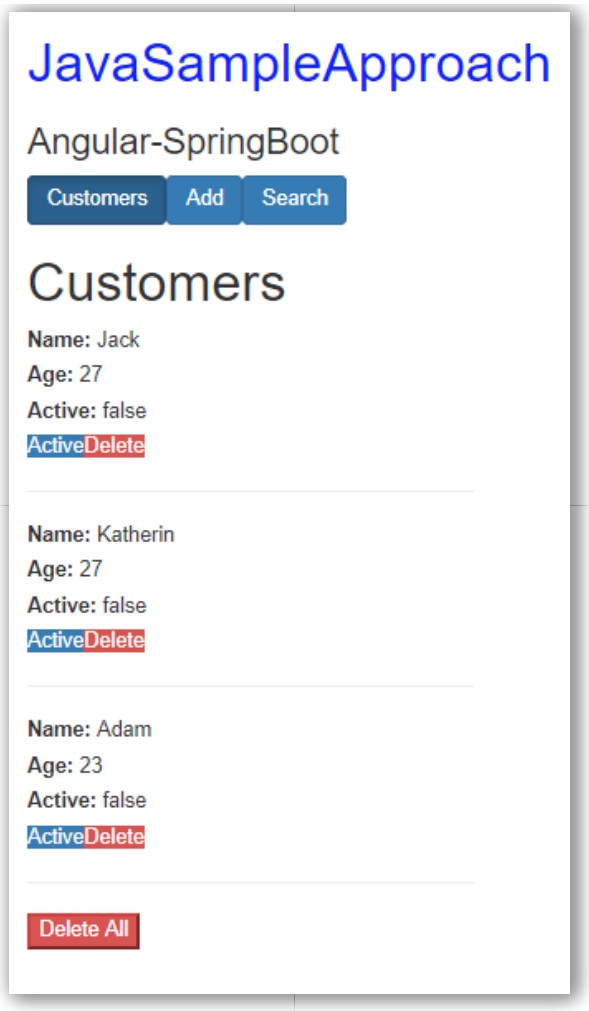
– Run the **Angular App** with command: `ng serve`.

– Open browser for url `http://localhost:4200/` :

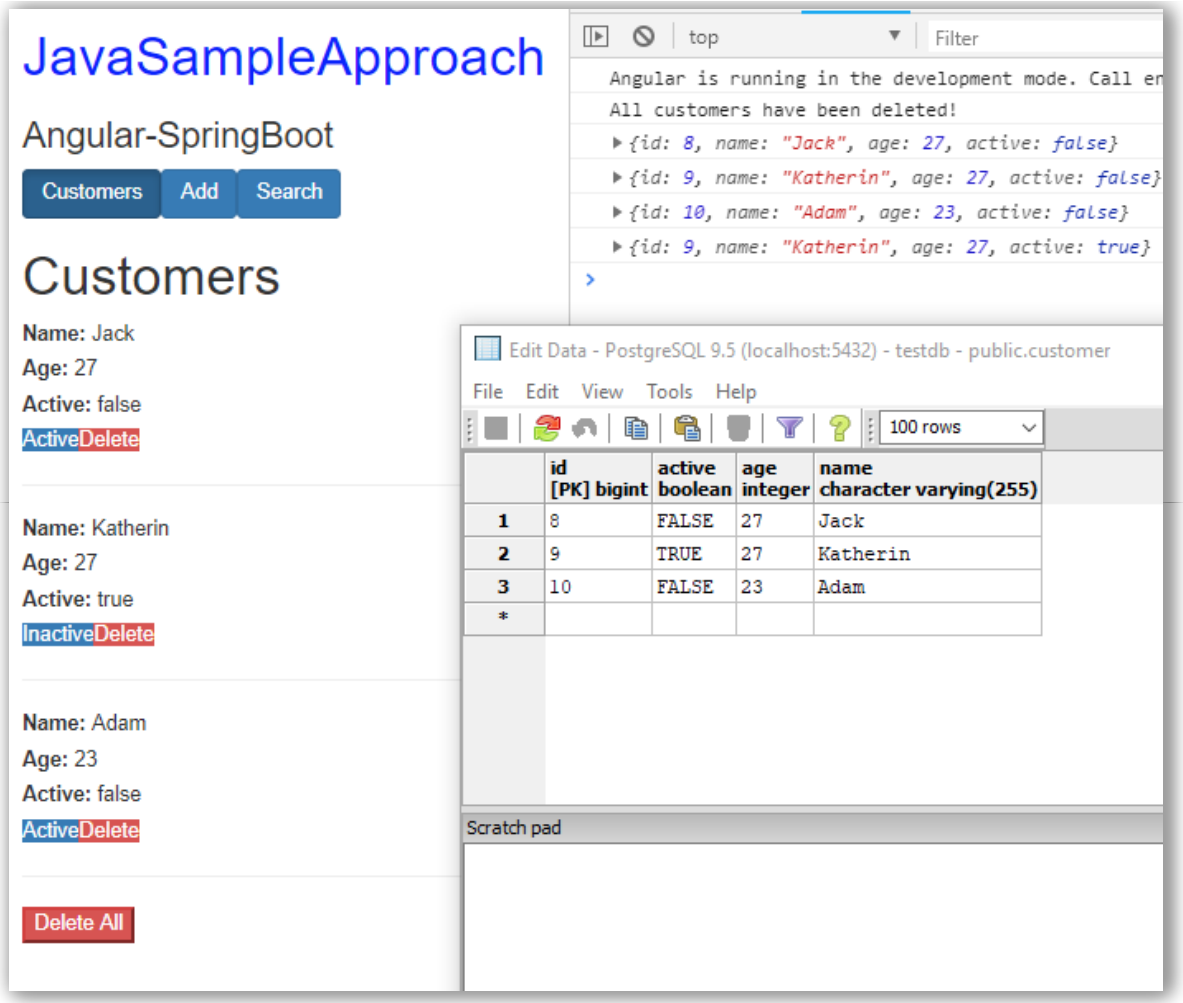
Add Customer:



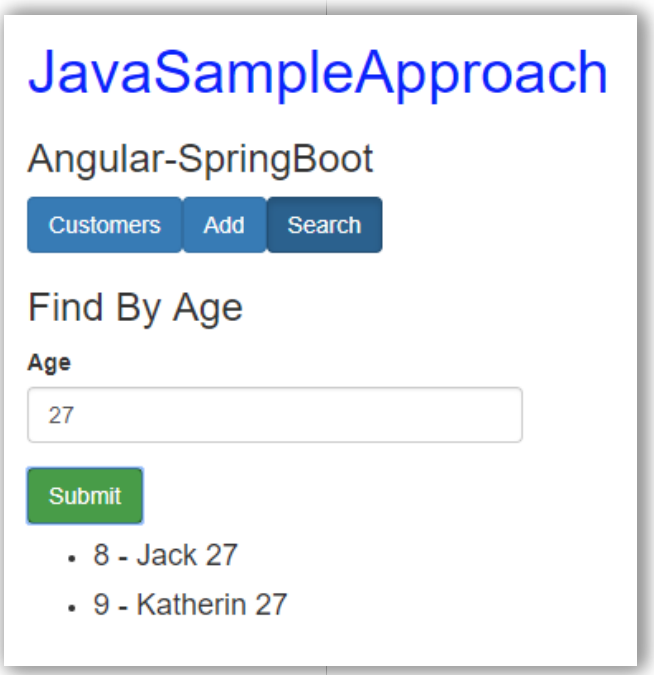

Show Customers:



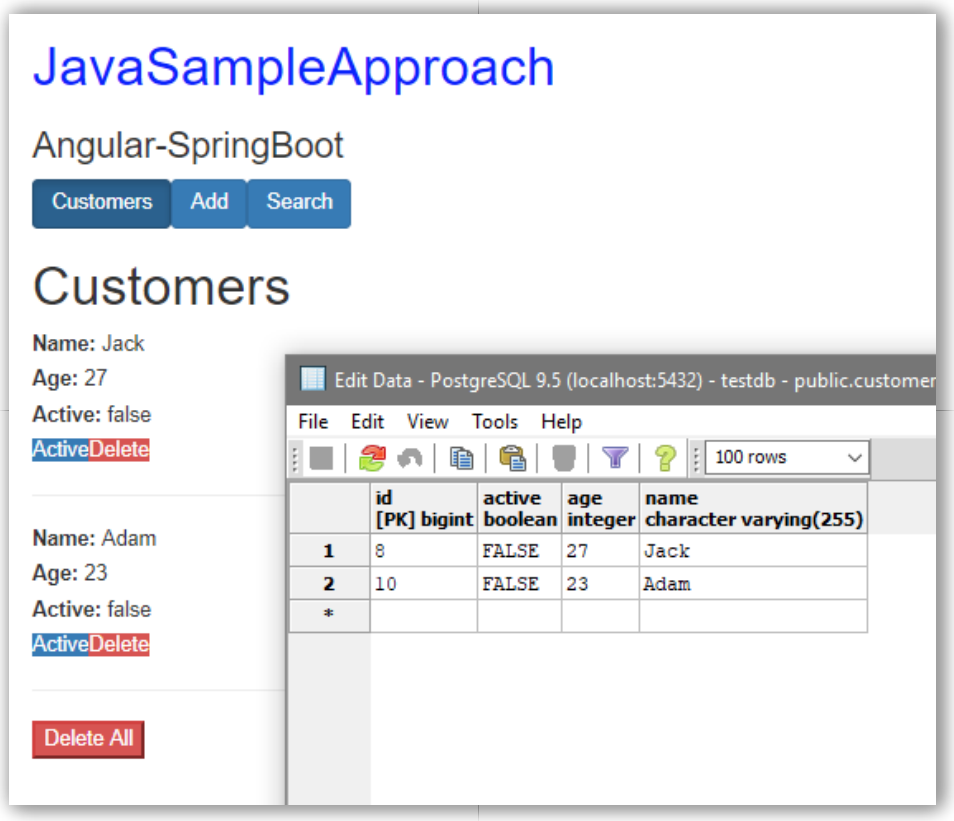

Click on **Active** button to update Customer status:



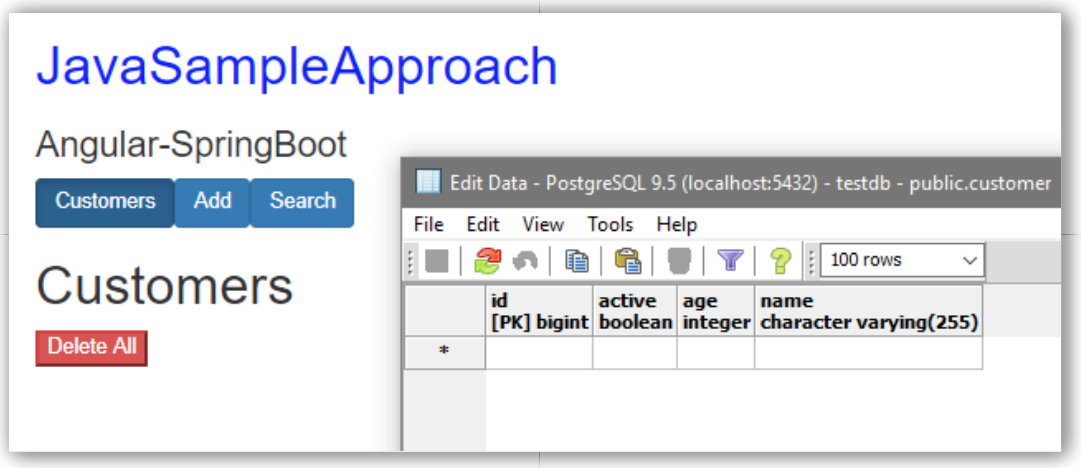

Search Customers by Age:



Delete a Customer:




Delete All Customers:




## IV. Source Code

- Angular6SpringBoot-Client
- SpringRESTPostgreSQL-Server

By grokonez | June 21, 2018.

Last updated on **November 2, 2018**.

## Related Posts

- Angular built-in Slice Pipe | Array SlicePipe + String SlicePipe Example
- Angular 6 Custom Pipe | with Parameterizing a pipe | Angular Pure pipes + Impure pipes
- Angular Built-in DatePipe Example | Pre-defined Format + Timezone + Locale + Custom Format
- Angular 6 – KeyValue Pipe – *ngFor Loop through Object, Map example
- Angular 6 WebSocket example with Spring Boot WebSocket Server | SockJS + STOMP
- Angular 6 + Node.js + Amazon S3 | Upload Files + Download Files + List Files | using Express RestAPI, Multer, AWS-SDK
- Angular 6 NGXS example – Angular State Management
- Angular 6 NgRx Store example – Angular State Management
- Angular 6 HttpClient + Node.js/Express RestAPIs + MariaDB example | Sequelize ORM CRUD APIs example
- Angular 6 ElasticSearch example – simple Full Text Search

## Post Tags

angular 6 | angular httpclient | angular postgresql | angular spring mvc | crud | rest api | rest client | rest service | spring boot | spring boot angular | spring data | spring jpa | spring jpa postgresql | spring mvc | spring rest api

## 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.