Demo Applications

Contents

[Introduction 1](#_Toc488665183)

[1. Spring Boot + Micro Services demo application 1](#_Toc488665184)

[1.1 Build and deployment steps 1](#_Toc488665185)

[1.2 Technical descriptions 2](#_Toc488665186)

[2. MEAN Stack demo application 4](#_Toc488665187)

[2.1 Installation steps 5](#_Toc488665188)

[3. Angular JS + Spring Boot web application demo 6](#_Toc488665189)

[3.1 Build and installation steps 7](#_Toc488665190)

# Introduction

Below demo applications are prepared to show the three different kind of development stacks are used in application development.

1. Spring Boot + Micro Services demo application
2. MEAN Stack demo application
3. Angular JS + Spring Boot web application demo

# Spring Boot + Micro Services demo application

This application explains about spring boot application with different micro services. Below tools and technologies are used to develop this application.

* Spring Boot
* Spring MVC
* Spring Cloud environment
* Spring Eureka Cloud Server
* Spring JPA support
* Thymeleaf
* MySQL

## 1.1 Build and deployment steps

In the command prompt, build the application using maven “**mvn clean package**” command.

To run spring Eureka cloud registration server by using the below command

* **java -jar target/microservice-springboot-application-1.0.0.jar registration**

Open three windows terminal and run the respective micro services and these micro services will be registered in the Eureka registry server.

* **java -jar target/microservice-springboot-application-1.0.0.jar accounts**
* **java -jar target/microservice-springboot-application-1.0.0.jar customers**
* **java -jar target/microservice-springboot-application-1.0.0.jar web**

Below URL’s to access each services:

* Eureka Server – “**localhost:1111**”
* Account Server – “**localhost:2222**”
* Web Server – “**localhost:3333**”
* Customer Server – “**localhost:4444**”

Based on the load, we can increase the instance of the server by running the below commands.

In the new window, run a second set of servers; account-server, customer-server and web-servers by providing different HTTP port number as second argument. Like below,

* **java -jar target/microservice-springboot-application-1.0.0.jar web 2223**
* **java -jar target/microservice-springboot-application-1.0.0.jar accounts 3334**
* **java -jar target/microservice-springboot-application-1.0.0.jar customers 4445**

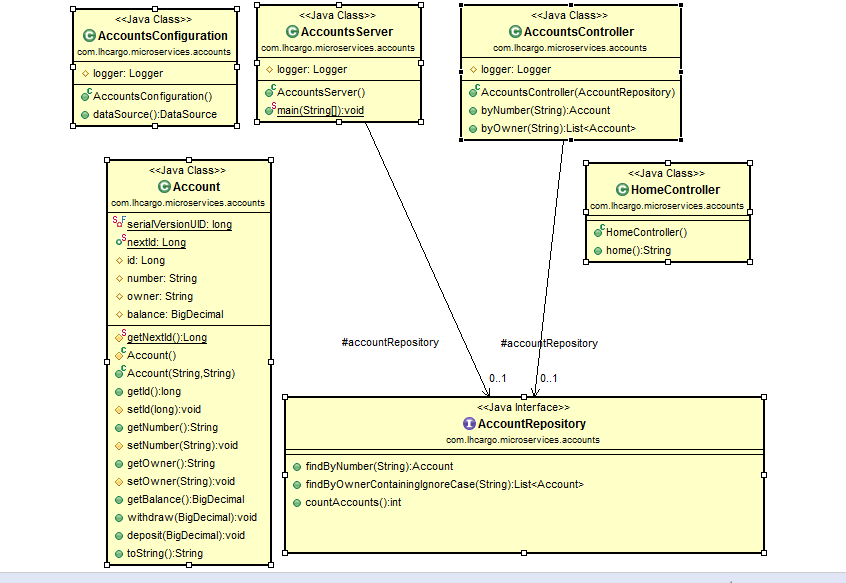
By using the above Service registration and discovery, Load balancing, Circuit breaker and Http client pattern, we can easily maintain the application stability and performance.

## 1.2 Technical descriptions

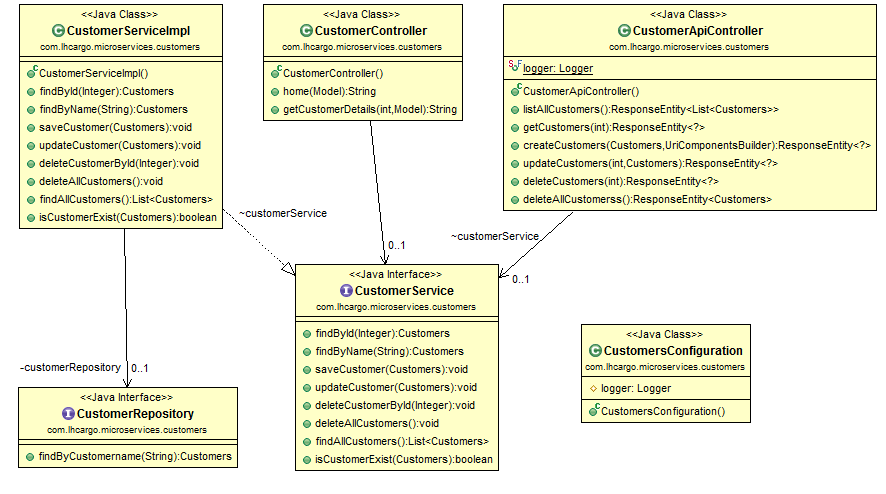
This application developed using Spring boot, Spring MVC and spring cloud environments.

Below UML diagram explains the details of every module.

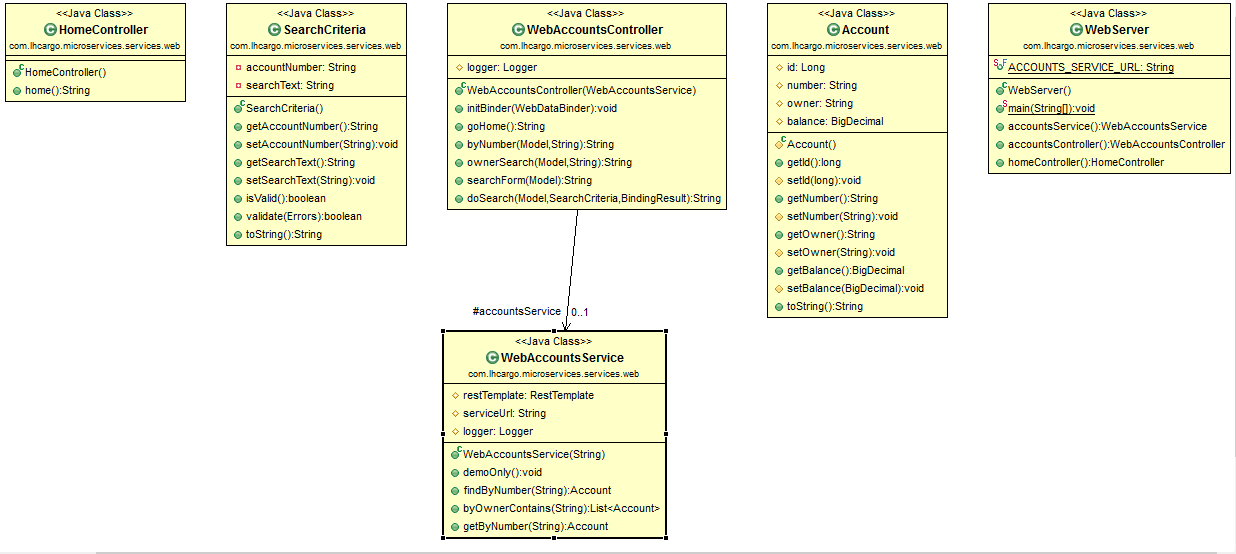
Account Server: This server publishes account Spring boot rest services and registered with Eureka server.



Customer Server: This micro services developed using Spring Boot, Spring MVC, Spring JPA and MySQL database.



Web Server: This micro services developed to consume account web services and this service act as a consumer for the account publisher services.



# MEAN Stack demo application

MEAN is an acronym made up of commonly used technologies for an all JavaScript web stack.

This particular combination of tools has generated a lot of traction in the enterprise area and is framework based, making it a good place to start.

The key components are:

**M**ongoDB (Database)

**E**xpressJS (Web Framework)

**A**ngularJS (Front-end Framework)

**N**odeJS (Application Server)

This application totally developed using JavaScript both frontend(AngularJS) and backend(NodeJS).

**AngularJS** is a very powerful JavaScript Framework. It is a client side JavaScript based MVC Framework.

Let’s have a look at some of the prominent reasons why we use Angular Framework.

* AngularJS helped the developers in quick creation of single page applications (SPA). SPA helps provide a more fluid un-interrupted user experience, similar to a desktop application, by avoiding frequent page reloads.
* AngularJS helped in instant projection of current data wherever required, without the developers writing any additional piece of code for the same.
* AngularJS provided lot of abstractions by automatically handling many of the tasks like DOM manipulation, validations on input, etc, thereby reducing the overhead on the developers for the same and in-turn enabling quick application development.
* AngularJS provided dependency injection, which none of the other frameworks out in the market has given.

**NodeJS** is based on JavaScript, both the client and server side code can be written using the same JavaScript language. It allows the frontend and backend teams to be combined into a single unit. Also since NodeJs uses JavaScript, we can quickly manipulate the JSON data retrieved from external web API sources like MongoDB, hence reduce the processing time need per request.

Node supports Asynchronous/Non-Blocking code by default. This is because, node works on JavaScript Single threaded even loop mechanism which improves the performance of the application. Single threaded event loop mechanism helps in handling multiple requests in your application efficiently.

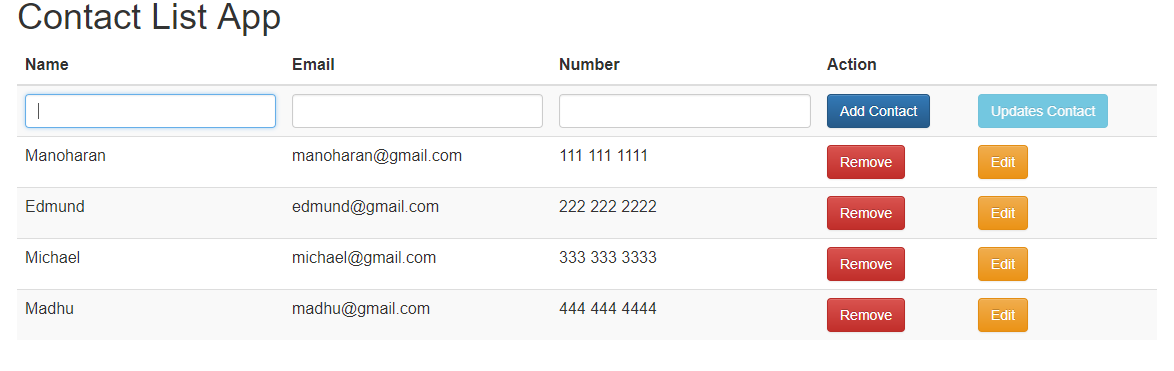
## 2.1 Installation steps

Contact list application is prepared using the MEAN stack and this is a simple CRUD application.

Below **npm** commands are used to install the required node modules for the contact list application

* MingoDB modules – “**npm install mongojs**”
* Express Framework – **“npm install express”**
* Body Parser Module – **“npm install body-parser”**
* Run the application **– “node server.js”**

Bleow single page contact list application used to demonstrate the simple CURD application using angularjs as front end,nodejs and expressjs framework used as business component and MongoDB used as backend.



# Angular JS + Spring Boot web application demo

This application is a full-blown CRUD application using Spring Boot, AngularJS, Spring Data, JPA/Hibernate and MySQL.

Spring-Boot which is build upon Spring foundation itself, takes convention over configuration to the next level by reducing the development time many-fold, enabling your projects up and running as quickly as possible.

It takes an opinionated view by providing the commonly used dependencies out of the box, avoiding version collisions.

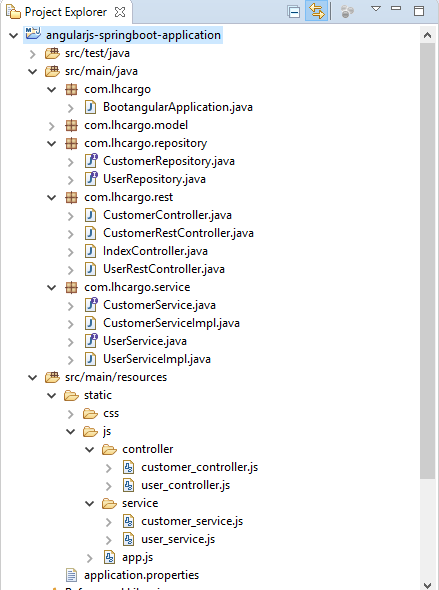
If something gets in your way, you can kick it out by excluding or overriding the version of your choice.

Do more with less code, virtually no xml involved.

Spring Boot apps can be run as standalone [using embedded containers] or can as well be deployed as WAR on external containers.

Provides multiple ways to run your app: “**java –jar**”, new run goal “**mvn spring-boot:run”** or traditional WAR deployment.

Provides out of the box, tons of production-ready non-functional features that are common to large classes of projects (e.g. embedded servers, security, metrics, health checks, externalized configuration).



## 3.1 Build and installation steps

To build the application **“mvn clean package”**

To run the application **“java -jar angularjs-springboot-application-1.0.0.jar”**, spring boot runs the application using internal tomcat server and by default application will be up and running in port 8080.