Mailer web application – Developers Guide

Version v1

Author: Jammula Vishnuvardhan Reddy

Date: 12/07/2018

Contents

[1. Objective](#_Toc511660611)

[2. Target Audience](#_Toc511660612)

[3. Architecture](#_Toc511660613)

[3.1 Security](#_Toc511660614)

[4. Custom Web Service Creation](#_Toc511660618)

[4.1 Prerequisites](#_Toc511660619)

[4.2 Maven POM](#_Toc511660620)

[4.3 Main Application Implementation](#_Toc511660621)

[4.4 Application Properties](#_Toc511660623)

[4.5 Build Process](#_Toc511660625)

[5. Deployment](#_Toc511660626)

[5.1 Stand Alone Tomcat](#_Toc511660627)

[5.2 Embedded Tomcat](#_Toc511660628)

# Objective

This document provides information about the Mailer application to allow developers to extend the application and understand various functionalities of the application in detail. This document is based on the first version (v1) of the web application.

# Target Audience

This document is targeted at all Web Application Developers, including the Business Services API developers, Architects, Engineering Leads, and Security personnel, who will be contributing to any part of web Application.

# Architecture

The Mailer application uses Spring Boot version 1.5.3 and JDK version 1.8 along with some other open source libraries to support the application features. The Web app in this phase (Phase I) concentrates on the areas of

* Security – Authentication using the Spring security.
* Build pipeline: using Jenkins.
* Code quality analysis : Using sonarQube software.

The Web app consists of a utility library and a Maven parent POM. The utility library is intended to help developers to create Rest services. The parent POM defines a minimum set of required and security scanned libraries for building Spring Rest services.

The first iteration supports application deployed to stand alone Tomcat application server . Only the users authenticated will be able to access the web application.

## Security

In this release, the application uses Spring security with JDBC authentication as the authentication mechanism to validate user requests for the web application hosted in the Tomcat application server. Spring Security is used to satisfy the authentication requirement of this phase. All the Security modules have been included in the *com.vishnu.project package in the web app,* and there is no need for developers to directly deal with them.

To access the secured web services deployed in a Tomcat application server, users must be valid registered users and have been authenticated with the Spring security .

## Prerequisites

* Maven 3.2+
* Eclipse Oxygen or Spring Tool Suite 3.9.1 with the m2e plugin
* JDK 1.8.0\_101 or later
* Tomcat 8.0.36
* MySQL Database

## Maven POM

Apache Maven has been chosen as the build and dependency management tool to develop Mailer application. As described in Section 3, a Maven parent POM that contains all necessary libraries for extending the Mailer application has been constructed to allow developers to create functionalities with concentration on business logic rather than worrying about the plumbing underneath.

## Main Application Implementation

The main application class is implemented to instantiate all required Spring beans and to start the application with embedded Tomcat or to bind the servlet, filters, and beans from the application context to a stand-alone Tomcat server.

To implement the main application class, simply use the following code (replacing the class name with the preferred name):

package com.vishnu.project;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.builder.SpringApplicationBuilder;

import org.springframework.boot.web.support.SpringBootServletInitializer;

@SpringBootApplication

public class MailerApplication extends SpringBootServletInitializer

{

@Override

protected SpringApplicationBuilder configure(SpringApplicationBuilder application)

{

return application.sources(MailerApplication.class);

}

public static void main(String[] args)

{

SpringApplication.run(MailerApplication.class, args);

}

}

*SpringBootServletInitializer* is used here to allow the Spring Boot application to be deployed to a servlet container, such as a Tomcat application server.

## Application Properties

Application properties should be defined in properties format. Use the *application.properties* file in the src/main/resources directory of the Mailer project as a starting point.

The following properties are required:

spring.mvc.view.prefix: /

spring.mvc.view.suffix: .jsp

spring.messages.basename=validation

spring.datasource.url = jdbc:mysql://localhost:<port>/<database name>

spring.datasource.username = <user name>

spring.datasource.password = <user password>

spring.jpa.hibernate.ddl-auto = update

spring.jpa.hibernate.naming-strategy = org.hibernate.cfg.ImprovedNamingStrategy

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect

Change the url,username and password of the SQL database with the that of your system.

## Build Process

Build project: mvn clean install

# Deployment

## Stand Alone Tomcat

Copy the generate mailer.war from the target folder of the Mailer project folder to the webapps folder of the Tomcat in your system.

## Embedded Tomcat

To run applications with embedded Tomcat ,

Right *click on the app in the Eclipse and choose to run as Spring boot App.*

To run applications with embedded Tomcat in a production environment, use the command

*java –jar mailer.war --server.port=8090*

You can access various end points from an instance started with the second command above from a browser:

* For the login end point, use <http://localhost:8080/mailer/login>
* For the register end point, use http://localhost:8080/mailer /register