**UPRAiZAL – 3.0 (Using Microservices)**

This documentation explains how we planning to implement microservices for UPRAiZAL application.

**Version: 1.0**

**Date: 15-JUN-2021**

*All rights reserved. No part of this publication may be reproduced, transcribed, transmitted, stored in a retrieval system or translated into any language, in any form or by any means mechanical, manual, electronic, magnetic, chemical, optical, including photocopying or otherwise without the prior written permission of UPRAiZAL, Inc.*

Contents

[Revision History 3](#_Toc74672267)

[Requirement/Objective 4](#_Toc74672268)

[Pre-requisites 4](#_Toc74672269)

[STS and Spring Boot 4](#_Toc74672270)

[Application Architecture 5](#_Toc74672271)

[How to create microservices 5](#_Toc74672272)

[What is Eureka Server And How to Create? 7](#_Toc74672273)

[Holiday Calendar Implementation using microservice. 8](#_Toc74672274)

[OKR/OBJECTIVE/TASKS and Standard Task Steps Implementation using microservice. 9](#_Toc74672275)

[GIT Repository 9](#_Toc74672276)

[Docker – for containerization 10](#_Toc74672277)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author | Date | Description |
| 1.0 | HYD0176SD | 15/June/2021 | Initial Version |

**Confidentiality**

The contents of this document are considered confidential and proprietary information of UPRAiZAL, Inc. No part of this document may be reproduced, fully or in part, or divulged to any other party without prior written permission from UPRAiZAL, Inc. All logos of respective owner companies used in this presentation are recognized.

# Requirement/Objective

1.To Bring a revamped version of UPRAiZAL which is

* Technologically advanced
* Ease the development and deployment.
* Robust in security
* Better performance
* User Friendly

2. Functionally, should suffice the current market requirements and provide a solution approach with a user-friendly design which that could help in attracting the clients.

# Pre-requisites

To fulfil the above-mentioned requirement, following are the tools and technologies needed:

* STS- Spring Tool Suite (IDE)
* Spring Boot framework
* GIT repository
* VS Code setup for React UI.
* ReactJS - Front end framework
* Docker - for containerization

# STS and Spring Boot

STS is an Eclipse-based development environment that is customized for the development of Spring applications.

It provides a ready-to-use environment to implement, debug, run and deploy your applications. It also includes integration for Pivotal tc Server, Pivotal Cloud Foundry, Git, Maven and AspectJ. STS is built as an addition on top of the latest Eclipse releases. Spring Boot Framework can be easily implemented using STS.

# Application Architecture

Following is the architectural diagram of the Application –

Diagram

Description automatically generated

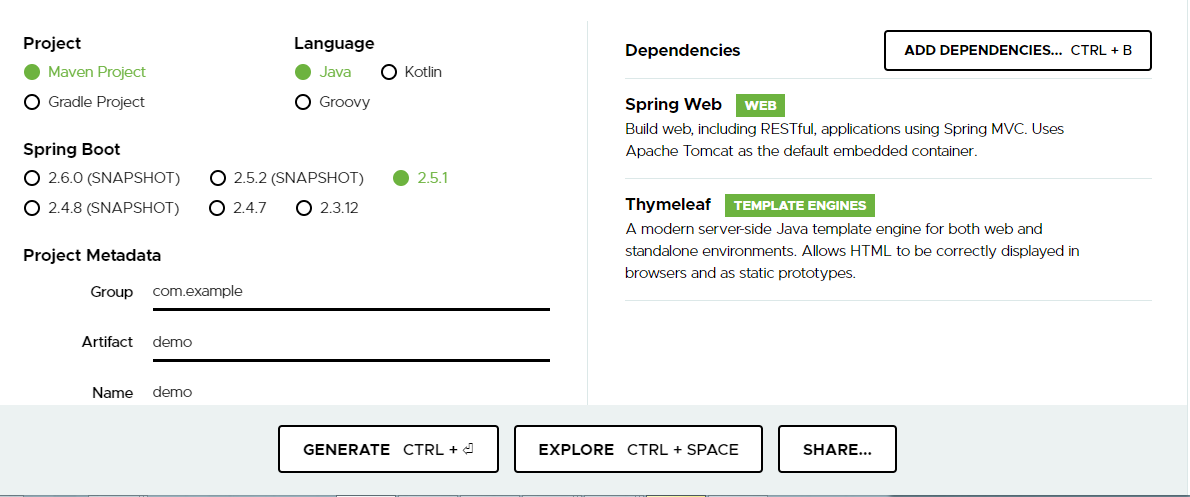
# How to create microservices

To create the microservices, we need to follow the following steps:

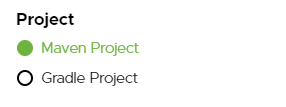
1.We need to open below URL on any browser.

[**URL:https://start.spring.io/**](URL:https://start.spring.io/)

Below screen appears on window

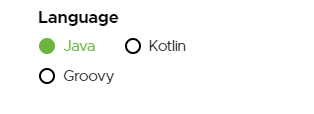


2.Select the Project type i.e. Maven project/Gradle project



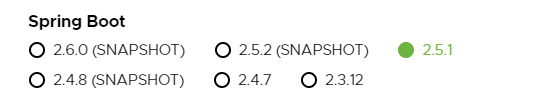
3.Once we decide the project type then select the which language, we going to use for microservice development, Spring cloud supports below languages

Languages: Java, Kotlin, Groovy

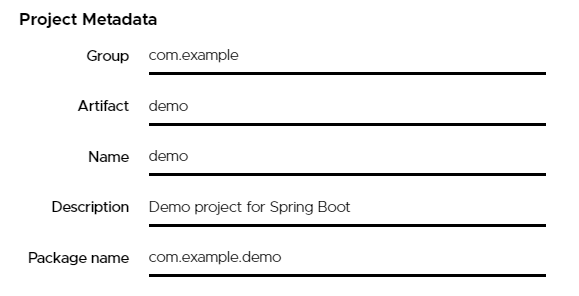


4.Then we need to select Spring Boot version

Try to select latest version of Spring Boot.



5.Add Project Metadata for related to Microservice



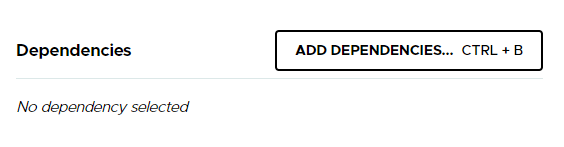
6. select the packaging, since we are developing micro-service so packaging should be jar



7. select version of Java, Microservices can be developed from java 8 onwards,



8. select required dependencies for microservice, click on ADD DEPENDENCIES Button, and add the dependencies



9.Finally click GENERATE button, so it will create microservice with above specified options

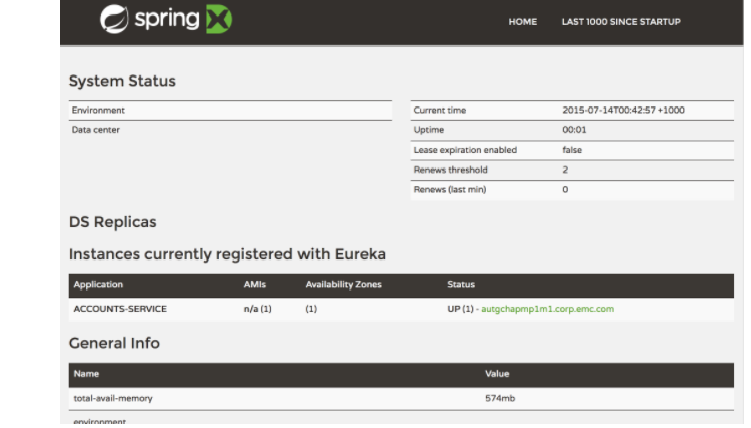
10.Now required microservice is ready, Unzip and import into eclipse as maven project.

# What is Eureka Server And How to Create?

**Eureka Server** is an application that holds the information about all client-service applications. Every Micro service will register into the **Eureka server** and **Eureka server** knows all the client applications running on each port and IP address. **Eureka Server** is also known as Discovery Server.

We no need to worry to create Eureka server, Spring provides it is also a micro service, so it all about to select the dependency for Eureka server, and enable Eureka server annotation in Main class, So the micro service automatically acts as Eureka server

Sample screen for Eureka-Server console.



# Holiday Calendar Implementation using microservice.

1.Create a microservice called as “upraizal-configuration”. In the future this microservice responsible for entire configuration related things for application.

2.Need to develop services for implement holiday calendar in the” upraizal-configuration” microservice.

3.Need to prepare independent DB for this microservices and in the future it will handle data related to this microservices.

DB Name: configurations

Table: holidays

4.Need to establish intercommunication between existing upraizal application to this microservice.

To implement communication with the existing application we can use the Spring Rest Template Approach.

String uri = "http://{contextpath}/save-holidays.json";

RestTemplate restTemplate = new RestTemplate();

Holiday holiday = restTemplate.getForObject(uri, Holidays.class);

Eg:

5.Deploy microservice with cloud environment/ in Eureka server.

# OKR/OBJECTIVE/TASKS and Standard Task Steps Implementation using microservice.

1. Create a microservice called as “upraizal-task”. In the future this microservice responsible for task related things for application.
2. Need to develop services for Task Steps and OKR/Objective features implementation.
3. Need to create separate DB and this DB responsible to Task related information.

DB Name: Tasks

Required Tables:

Task-Steps

Task-KPIs

Task Types

1. Establish the communication with existing UPRAiZAL application.
2. Similar way we can establish intercommunication with “upraizal-configuration”, microservice.

Note: DB table names and number of tables might change while developing the feature.

# GIT Repository

Git is better than SVN at branching — and it can be better than SVN for access control and auditability. It is a great tool for automation and DevOps. Following are the steps to setup and commit the code to repository:

1. Install GIT Bash to your local.
2. Using the GIT dev account credentials, update the latest version of the branch team is working on.

$ git init

$ git pull

1. Commit the changes to the repository using below command:

$ git status

$ git status  
  
$ git Add -files  
  
$ git commit -m "Message"  
  
$ git push  
  
$ git pull

# Docker – for containerization

Docker is a container-based virtualization and infrastructure technology whose distributed nature gives developers greater control over resources and features. It simplifies app development, deployment, and maintenance through containerization. It is an excellent fit for a microservices architecture as it enables encasing the developed software in containers so that one container would support one service or functionality. Docker Machine, Docker composer, and Docker Swarm can then also be used to manage those containers in the most efficient and scalable way.