Recipe CookBook Design

Proof of Concept of Spring Boot Micro-services exposing REST APIs for a Recipe Cookbook.

Schemas

Recipe

recipeID	string
recipeName	string
recipeType	string
instructions	string
prepTime	number($double)
cookTime	number($double)

Ingredient

ingredientID	string
ingredientName	string
ingredientDescription	string
isAllergen	boolean

RecipeIngredients

id	integer($int32)
recipe	Recipe{...}
ingredient	Ingredient{...}
quantity	number($double)

IngredientNutrition

ingredientID	string
servingSize	integer($int64)
servingUnit	string Enum: Array [9]
servingSizeDescription	string
calories	number($double)
energy	number($double)

protein	number($double)
totalFat	number($double)
carbohydrates	number($double)
cholesterol	number($double)
sodium	number($double)
potassium	number($double)
calcium	number($double)
vitaminA	number($double)
vitaminC	number($double)
vitaminD	number($double)

Key spring services used:

Spring Web	Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
Spring Web DevTools	Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
Spring Web Actuator	Supports built in (or custom) endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.
Spring Data JPA	Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
Eureka Server	spring-cloud-netflix Eureka Server.
Eureka Discovery Client	A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.
Gateway	Provides a simple, yet effective way to route to APIs and provide cross cutting concerns to them such as security, monitoring/metrics, and resiliency.
H2 Database	Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

URLs:

H2 console
http://localhost:8900/h2-console

Eureka Server
http://localhost:8761

API Gateway - API Invocation via Eureka
http://localhost:8765/recipeserver/ingredient
http://localhost:8765/recipeserver/ingredient/get/name/Milk
http://localhost:8765/recipeserver/ingredient/get/id/Ingredient102

http://localhost:8765/recipeserver/ingredient-nutrition/get/id/Ingredient105
http://localhost:8765/recipeserver/ingredient-nutrition/get/id/Ingredient105
http://localhost:8765/recipeserver/ingredient-nutrition

http://localhost:8765/recipeserver/recipes/get/id/recipe101
http://localhost:8765/recipeserver/recipes/get/name/Chicken%20Curry
http://localhost:8765/recipeserver/recipes

http://localhost:8765/recipeserver/recipe-ingredients

Swagger or OpenAPI Documentation
http://localhost:8900/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config
http://localhost:8900/v3/api-docs

Actuator
http://localhost:8900/actuator

HAL explorer:
http://localhost:8900

Data Used:

Pre-configured in SQL which populates in-memory H2 as RecipeServer is launched

SELECT * FROM INGREDIENT;

INGREDIENTID	INGREDIENT_DESCRIPTION	INGREDIENT_NAME	IS_ALLERGEN
Ingredient101	Cloves from Zanzibar	Clove	FALSE
Ingredient102	Potatoes from Russia	Potatoes	FALSE
Ingredient103	Milk from Australia	Milk	TRUE
Ingredient104	Groundnuts from Bolivia	Groundnuts	TRUE
Ingredient105	Lamb from New Zealand	Lamb	FALSE
Ingredient106	Chicken from Brazil	Chicken	FALSE

SELECT * FROM INGREDIENT_NUTRITION;

INGREDIENTID	CALCIUM	CALORIES	CARBOHYDRATES	CHOLESTEROL	ENERGY	POTASSIUM	PROTEIN	SERVING_SIZE	SERVING_SIZE_DESCRIPTION	SERVING_UNIT	SODIUM	TOTAL_FAT	VITAMINA	VITAMINC	VITAMIND
Ingredient101	1.3	6.0	1.4	0.0	0.0	21.4	0.1	1	1 tsp of Cloves	4	5.8	0.3	0.1	0.0	0.0
Ingredient102	1.0	77.0	17.0	0.0	77.0	421.0	2.0	100	100g of Potatoes	5	6.0	0.1	0.0	32.0	0.0
Ingredient103	30.0	150.0	12.0	30.0	150.0	350.0	8.0	150	150ml of Milk	4	120.0	8.0	4.0	2.0	0.0
Ingredient104	0.0	523.0	52.0	0.0	523.0	0.0	15.0	100	100g of Groundnuts	5	0.0	29.0	0.0	0.0	0.0
Ingredient105	1.0	250.0	0.0	0.0	250.0	264.0	21.0	100	100g of Lamb	5	61.0	18.0	0.0	0.0	0.0
Ingredient106	2.0	100.0	1.0	50.0	100.0	0.0	22.0	100	100g of Chicken	5	110.0	2.0	0.0	2.0	0.0

SELECT * FROM RECIPE;

RECIPEID	COOK_TIME	INSTRUCTIONS	PREP_TIME	RECIPE_NAME	RECIPE_TYPE
recipe101	60.0	Boil peel serve	45.0	Something not tasty	Indian
recipe102	10.0	Fry and fry and fry	5.0	Tasty	Indian
recipe103	15.0	Put oil in a wok and fry	35.0	Chicken Curry	Thai

```
SELECT * FROM RECIPE_INGREDIENTS;
```

ID	QUANTITY	INGREDIENTID	RECIPEID
1	2.0	Ingredient102	recipe103
2	3.0	Ingredient103	recipe103
3	3.0	Ingredient104	recipe103
4	2.0	Ingredient101	recipe101
5	4.0	Ingredient102	recipe101

Next Steps:

Event publish onto a monitoring tool like Kibana or Zipkin

- Have exposed basic GET functions, expose POST/DELETE as well that should be easy
- Set up a centralised Enterprise Database rather than H2, which will allow for breaking up into smaller micro services. This was a challenge as all entities were required to be referred from in-memory H2 database.
- Value add services on top
    - Instead of storing Ingredients and RecipeIngredients in a database, can call APIs such as those of MyFitnessPal - https://myfitnesspalapi.com/docs/
    - Based on the quantity of ingredients chosen, calculate nutrition information for the recipe
    - Expose more utility APIs such as finding all recipes by ingredients
    - Introduce a fuzzy search to accept ingredients from users and recommend recipes that they can attempt to cook
- Add security elements to access APIs
- Separate roles for Admin and Users
- Setting up of a monitoring too like Kibana/ElasticSearch/Zipkin to be able to publish events for easy traceability

Appendix: Exception Handling

```
package com.abhishek.recipebook.recipebookrecipeserver.exception

/ 20211101235341
// http://localhost:8765/recipeserver/ingredient/get/name/Milky

{
  "timestamp": "2021-11-01T15:53:40.164+00:00",
  "message": "IngredientsNotFoundException: Ingredient name
Milky not found",
  "details": "uri=/ingredient/get/name/Milky"
}

// 20211101235416
// http://localhost:8765/recipeserver/ingredient/get/id/
Ingredient107

{
  "timestamp": "2021-11-01T15:54:15.992+00:00",
  "message": "IngredientsNotFoundException: Ingredient id
Ingredient107 not found",
  "details": "uri=/ingredient/get/id/Ingredient107"
}

// 20211102000356
// http://localhost:8765/recipeserver/ingredient-nutrition/get/
id/Ingredient1057

{
  "timestamp": "2021-11-01T16:03:56.215+00:00",
  "message": "IngredientNutritionNotFoundException: Ingredient
isn't registered",
  "details": "uri=/ingredient-nutrition/get/id/Ingredient1057"
}

// 20211101234804
// http://localhost:8765/recipeserver/recipes/get/id/recipe1012
```

```
{
  "timestamp": "2021-11-01T15:48:03.504+00:00",
  "message": "RecipeNotFoundException: Unable to find any data
with this id recipe1012 ",
  "details": "uri=/recipes/get/id/recipe1012"
}
```


```
// 20211102000514
// http://localhost:8765/recipeserver/recipes/get/name/
Chicken%20Currys
```


```
{
  "timestamp": "2021-11-01T16:05:14.408+00:00",
  "message": "RecipeNotFoundException: Unable to find any data
with this name Chicken Currys ",
  "details": "uri=/recipes/get/name/Chicken%20Currys"
}
```