Demo of Vehicle REST API and cucumber Java

V 1.0

Table of Contents

[Objective and Tools 2](#_Toc496465472)

[Code Base 2](#_Toc496465473)

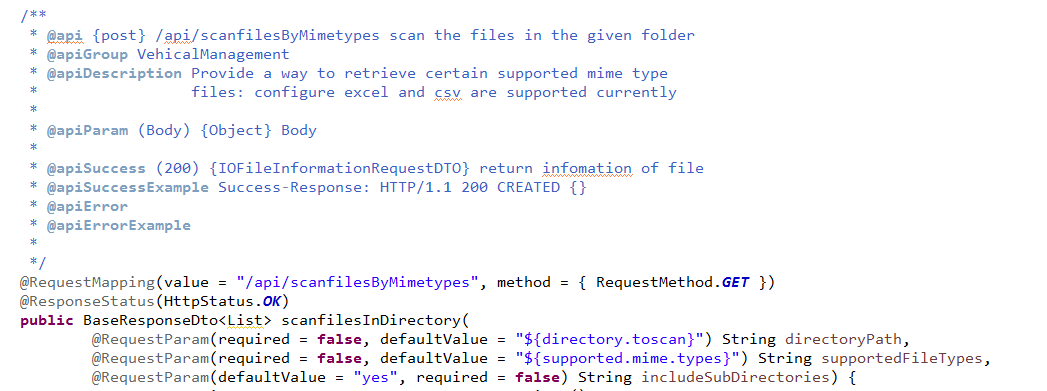
[PART-1 3](#_Toc496465474)

[Setting the Project and dev-env 3](#_Toc496465475)

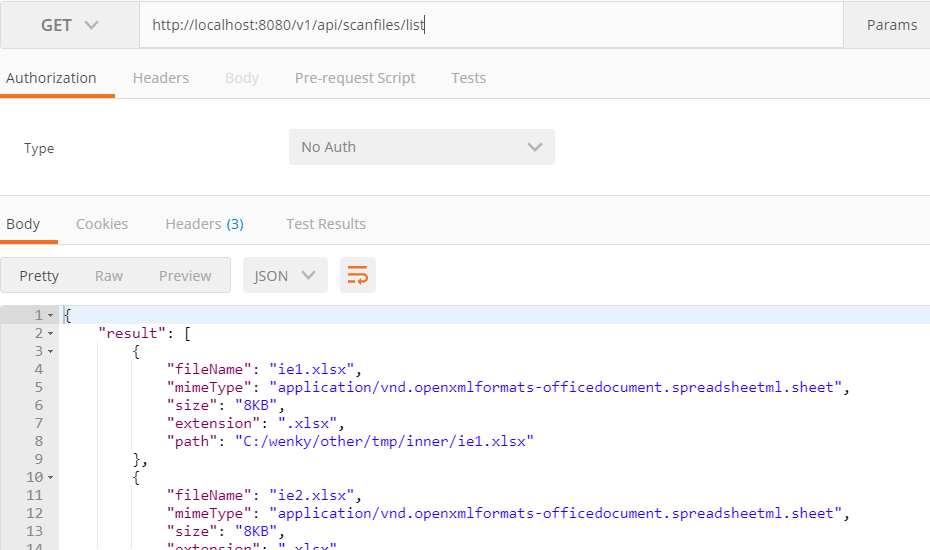
[Project Skelton 3](#_Toc496465476)

[Create logic: 4](#_Toc496465477)

[API #1:Retreive the files based on mimeTypes provided. 4](#_Toc496465478)

[ 4](#_Toc496465479)

[API #2:Retrieves all the files irrespective of the mimeTypes. 5](#_Toc496465480)

[ 6](#_Toc496465481)

[Code WalkThrough: 6](#_Toc496465482)

[PART -2 7](#_Toc496465483)

# Code Base

PART-1 REST apis

https://github.com/krimohshu/filescanner-RESTapis

PART-2 Automation

https://github.com/krimohshu/automation-springboot-cucumber

git clone

# Objective and Tools

1. Demo to write a REST POST api to get files information in a set.

Used: Springboot, Hibernate, File-IO , InteliJ, Maven,Postman

1. Automate using cucumber

# PART-1

## Run project

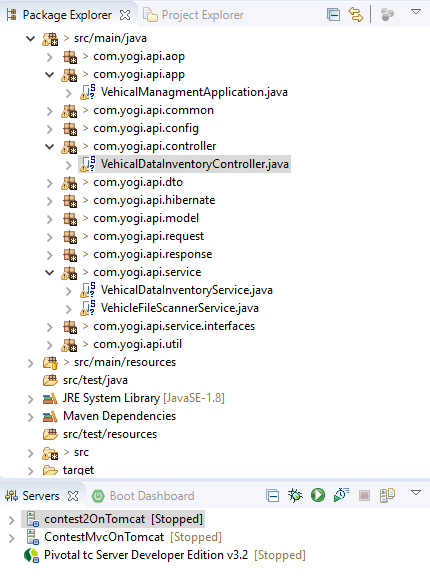
clean spring-boot:run -Dspring.profiles.active="dev" test -Dtest=com.yogi.testrunner.CucumberRunner

## Setting the Project and dev-env

* 1. Use maven compiler plugin to set J1.8 while creating a maven Project in IntelliJ
  2. Create controller-model-view package
  3. Insert commonly used dependencies such as spring, guava, joda-time, commons, junit , poi, mysql ( enable auto import in IntelliJ)
  4. Make spring-boot-starter-parent in parent tag and use property for version

## Project Skelton

Creating the controller, service, request response structure (I skipped, repository, business delegate layer)



* 1. Create the Base controller in common so it can be utilize by all controllers
  2. Map the @RestController.
  3. Provide the property using @PropertySource, the same can achieved by value annotation of spring.

@PropertySource(value = {**"classpath:properties.${ENV\_SYSTEM:dev}/props-for-api-tests.properties"**, **"classpath:properties.${ENV\_SYSTEM:dev}/props-for-ui-tests.properties"**})

* 1. Provide the request mapping
     1. http://<Server:port> /v1/api/scanfiles
     2. method.POST
  2. @Autowired Environment to gte property and pass the property while creating the bean

## Create logic:

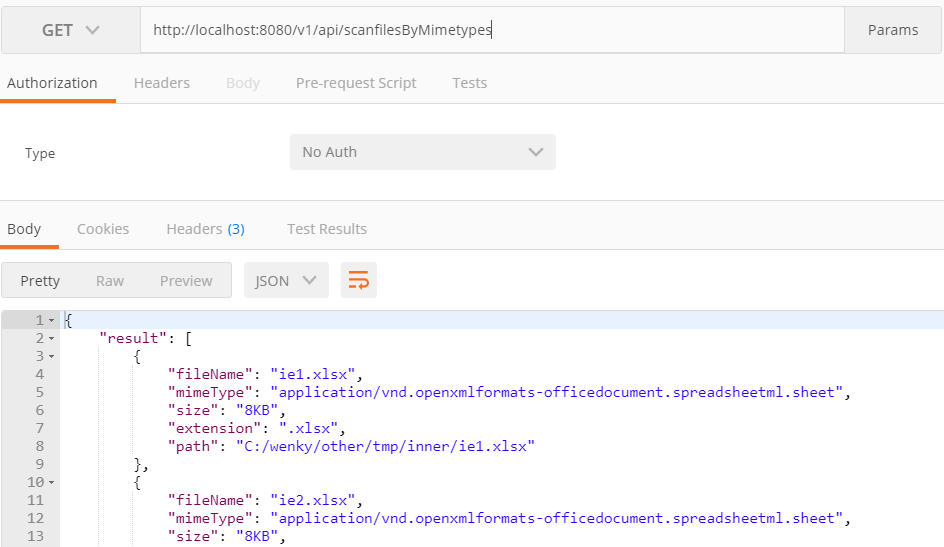
# API #1:Retreive the files based on mimeTypes provided.

Case 1: If no MimeTypes are provided it will return the default files of mimeTypes excel & csv.

Case 2: If mimeTypes are not valied,it will return with the status error.

Case 3: It will return the files with mimeTypes from the user input(multiple mimeTypes are separated by comma)

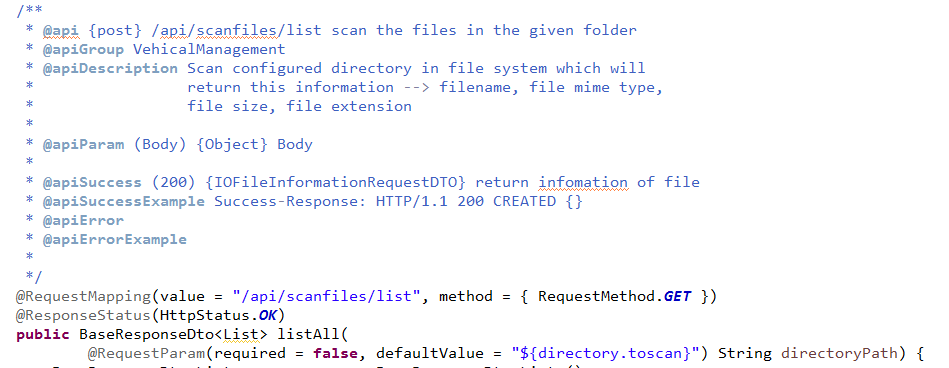
# 



# API #2:Retrieves all the files irrespective of the mimeTypes.

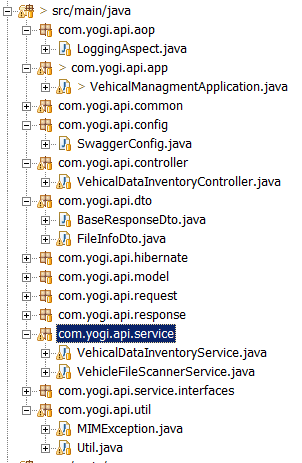
Case 1: Retrieves all the files in the preconfigured directory.

Case 2: Retrieves all the files in the directory specified



# 

# Code WalkThrough:



Control flow will be from the controller(com.yogi.api.controller) to the service layer(com.yogi.api.service).

controller & service layers will make use of util (com.yogi.api.util) classes for processing the code and

The controller will return response (com.yogi.api.dto.BaseResponseDto) with the status field and the result.

**Swagger is integrated and can be accessed at below url:**

http://localhost:8080/swagger-ui.html

**UI for API #1:**

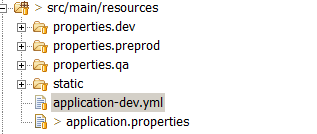
http://localhost:8080/scanner.html

Preconfigured properties can be found under application-dev.yml

directory.toscan: C:\Automation\vehicleManagement

supported.mime.types:

valied.mime.types:



# PART -2

git clone

Framework demonstrate following features:

1. Spring dependency injection

2. Singleton pattern to manage single webdriver instance and global wait properties(Fluent/implicit)

3. Maven using maven-compiler plugin for1.8 and cuke/web/guava/common dependencies

4. Cucumber using cucumber junit & cucumber options e.g. glue,-dryrun and tags.

5. Factory pattern to increase reusability of common components (e.g. Topic in Page and Post)

6. Builder pattern to manage navigation and flexibility to build steps for tester using "." operator

7. Tags to run multiple Scenarios 8. Property file for reusable element,env,url,database property

9. Combine all library in page object model and behavior in command builders

10.Custom exception handling using user define exceptions