|  |
| --- |
|  |
| Swagger Grails Application Integration |
|  |
| Rabindra Kumar Panigrahy |
| **1/15/2018** |

|  |
| --- |
|  |

# Configuration

1. Dependency :

Below are the dependencies’ required for the Swagger documentation for the grails REST APIs. This should be mentioned in build.gradle and downloaded from the Maven repo before starting the development.

|  |
| --- |
| compile group: 'io.swagger', name: 'swagger-models', version: '1.5.17'  compile group: 'io.swagger', name: 'swagger-servlet', version: '1.5.17'  compile group: 'org.webjars', name: 'swagger-ui', version: '3.5.0' |

1. Defining the Swagger Meta data Information:

The below code snippet should be present under grails-app/conf/ application.groovy for defining the Swagger Documentation description, version and etc.

|  |
| --- |
| swagger {  info {  description = "Error Log Service provide a platform for capturing the Application Error Details. Move your app forward with the Swagger API Documentation for providing the API Specification !!!"  version = "ttn-swagger-2.0.0"  title = "Error Log Service API Documentation "  termsOfServices = "http://swagger.io/"  contact {  name = "Contact Us"  url = "http://swagger.io"  email = "rabindrakumar.panigrahy@timeinc.com"  }  license {  name = "Licence Under"  url = "http://swagger.io"  }  }  schemes = []  consumes = ["application/json"]  } |

1. Retrieving the Swagger Meta data Information :

For retrieving and populating the above mentioned information in the Swagger Document the below plugin class is needed. This should be the part of grails-app/services.

|  |
| --- |
| package com.synapsegroupinc.swagger.swaggerplugin  import grails.plugins.Plugin  import io.swagger.models.\*  import org.apache.commons.lang.StringUtils  class SwaggerGrailsPlugin extends Plugin {  // the version or versions of Grails the plugin is designed for  def grailsVersion = "3.1.9 > \*"  // resources that are excluded from plugin packaging  def pluginExcludes = [  "grails-app/views/error.gsp",  "\*\*/services.swaggerResources\*\*",  "\*\*/logback.groovy",  "\*\*/logback.xml",  "\*\*/logback\*\*",  "\*\*/logback/\*\*"  ]  def title = "Grails Swagger"  def author = "Rabindra Kumar Panigrahy"  def authorEmail = "rabindrakumar.panigrahy@timeinc.com"  def description = '''\  Grails Plugin For Swagger Documentation  '''  def profiles = ['rest-api']  // URL to the plugin's documentation  def documentation = "http://grails.org/plugin/swagger"  def license = "Time Inc."  // Details of company behind the plugin  def organization = [ name: "Time Inc.", url: "http://www.timeinc.com/" ]  def observe = ['controllers', 'services']  def loadAfter = ['controllers', 'services']  Closure doWithSpring() {  { ->  swagger(Swagger) {  Map swaggerConfig = (config.swagger as Map) ?: [:]  Map infoConfig = swaggerConfig.info ?: [:]  Info swaggerInfo = new Info(  description: infoConfig.description ?: StringUtils.EMPTY,  version: infoConfig.version ?: StringUtils.EMPTY,  title: infoConfig.title ?: StringUtils.EMPTY,  termsOfService: infoConfig.termsOfServices ?: StringUtils.EMPTY  )  Map contactConfig = infoConfig.contact ?: [:]  swaggerInfo.setContact(new Contact(  name: contactConfig.name ?: StringUtils.EMPTY,  url: contactConfig.url ?: StringUtils.EMPTY,  email: contactConfig.email ?: StringUtils.EMPTY)  )  Map licenseConfig = infoConfig.license ?: [:]  swaggerInfo.license(new License(  name: licenseConfig.name ?: StringUtils.EMPTY,  url: licenseConfig.url ?: StringUtils.EMPTY)  )  info = swaggerInfo  //host = swaggerAsMap.host ?: "localhost:8080"  schemes = swaggerConfig.schemes ?: [Scheme.HTTP]  consumes = swaggerConfig.consumes ?: ["application/json"]  }  }  }  } |

1. Dynamically Scan and Generate the Swagger Resource :

For dynamically scanning the Swagger configured URI’s and generating the swagger resources (swagger.json files) the below service class is required.

|  |
| --- |
| package com.synapsegroupinc.swagger  import com.fasterxml.jackson.core.JsonProcessingException  import grails.web.mapping.LinkGenerator  import io.swagger.annotations.Api  import io.swagger.models.Swagger  import io.swagger.servlet.Reader  import io.swagger.util.Json  import org.apache.commons.lang.StringUtils  import org.springframework.context.ApplicationContext  import org.springframework.context.ApplicationContextAware  import groovy.util.logging.Slf4j  @Slf4j(value='logger')  class SwaggerService implements ApplicationContextAware {  Swagger swagger  ApplicationContext applicationContext  String generateSwaggerDocument() {  return getJsonDocument(scanSwaggerResources())  }  Swagger scanSwaggerResources() {  // Below code is written to support multi-module project.  LinkGenerator linkGenerator = applicationContext.getBean(LinkGenerator.class)  String host = linkGenerator.getServerBaseURL()  host = host.replace($/http:///$, StringUtils.EMPTY)  host = host.replace($/https:///$, StringUtils.EMPTY)  swagger.setHost(host)  Map<String, Object> swaggerResourcesAsMap = applicationContext.getBeansWithAnnotation(Api.class)  List<Class> swaggerResources = swaggerResourcesAsMap.collect { it.value?.class }  if (swaggerResources) {  Reader.read(swagger, new HashSet<Class<?>>(swaggerResources))  }  return swagger  }  static String getJsonDocument(Swagger swagger) {  String swaggerJson = null  if (swagger != null) {  try {  swaggerJson = Json.mapper().writeValueAsString(swagger)  } catch (JsonProcessingException e) {  logger.error('Exception Occurred while Swagger Json Specification Processing', e)  }  }  return swaggerJson  }  } |

1. Swagger Document Integration:

For integrating the Swagger resource files with Swagger-UI module the below ‘ ApiDocController’ controller class is required. This should be the part of grails-app/controller package.

|  |
| --- |
| package swagger  import org.springframework.beans.factory.annotation.Value  import org.springframework.core.io.Resource  import org.springframework.http.HttpStatus  import org.springframework.http.MediaType  import com.synapsegroupinc.swagger.SwaggerService  import groovy.util.logging.Slf4j  @Slf4j(value='logger')  class ApiDocController {  static responseFormats = ['json']  static namespace = 'v1'  static allowedMethods = [getDocuments: "GET"]  SwaggerService swaggerService  @Value("classpath\*:\*\*/webjars/swagger-ui/\*\*/index.html")  Resource[] swaggerUiResources  def getDocuments() {  if (request.getHeader('accept') && request.getHeader('accept').indexOf(MediaType.APPLICATION\_JSON\_VALUE) > -1) {  try {  String swaggerJson = swaggerService.generateSwaggerDocument()  render contentType: MediaType.APPLICATION\_JSON\_UTF8\_VALUE,  text: swaggerJson  } catch (Exception e) {  logger.error('Exception Occurred during swagger specification document generation', e)  render status: HttpStatus.INTERNAL\_SERVER\_ERROR,  text: 'Some error occurred'  }  } else {  redirect uri: "/webjars/swagger-ui${getSwaggerUiFile()}?url=${request.getRequestURI()}"  }  }  protected String getSwaggerUiFile() {  try {  (swaggerUiResources.getAt(0) as Resource).getURI().toString().split("/webjars/swagger-ui")[1]  } catch (Exception e) {  throw new Exception("Unable to find swagger ui.. Please make sure that you have added swagger ui dependency eg:-\n compile group: 'org.webjars', name: 'swagger-ui', version: '3.5.0' \nin your build.gradle file", e)  }  }  } |

1. Triggering Swagger Document Controller:

The below mapping should be mentioned in the UrlMappings.groovy for triggering the Swagger document Controller.

|  |
| --- |
| '/errorLogServiceDocApi/$action?/$id?'(controller: 'apiDoc', action: 'getDocuments') |

The above mentioned steps are one time activity for setting up the Swagger Documentation of Grails REST APIs.

1. Developing the Swagger Enabled Grails REST API:

Below are the some set of basic annotations that we need to follow while designing the REST APIs.

* 1. Controller Level Annotations : Below are the controller level annotations
     1. @Api : This annotation should provide above the controller class name with the value and description parameter of the controller class. For example :

|  |
| --- |
| @Api(value = "/v1.0", description = "Operations about ErrorLog Service" , tags = ["Errorlog"]) |

* + 1. @ApiOperation : This annotation should provide above the REST API method name along with value, notes, nickname, consumes, httpMethod and response params. For example :

|  |
| --- |
| @ApiOperation(  value = "Posting Error Message To Queue.",  notes = "Posting the Application Error Details into Amazon SQS.",  nickname = "/processError",  consumes = "application/json",  httpMethod = "POST",  response = ErrorLog.class  ) |

* + 1. @ApiResponses : This annotation should provide for defining the error code and error messages related to the purpose and behavior of the method mentioned in the controller. For Example:

|  |
| --- |
| @ApiResponses([  @ApiResponse(code = 405,  message = "Method Not Allowed. Only POST is allowed."),  @ApiResponse(code = 404,  message = "Method Not Found."),    @ApiResponse(code = 200,  message = "App Configured Error Message Injected Successfully into Amazon SQS.")  ]) |

* + 1. @ApiImplicitParams : This annotation is require for mentioning the input parameters to the method mentioned in the controller class along with the paramType, value, datatype, required flag. For example:

|  |
| --- |
| @ApiImplicitParams([  @ApiImplicitParam(name = "Application Error Details",  paramType = "body",  required = true,  value = "Requires Application Error Details. Please Check the Model name under Models section for the request input type description details.",  dataType = "com.synapsegroupinc.ErrorLog")  ]) |

* 1. Domain/Model /Entity Level Annotations: Below are the domain level annotations

1. @ApiModel : This annotation is require for mentioning the Model name and its data structure along with the value and description params and mentioned above the Domain class name. For example:

|  |
| --- |
| @ApiModel(value="EX\_ERROR\_LOG", description="ErrorLog Service Model is used for Capturing the Application Error Details. Below are the Model Property details require for forming the request Json input type") |

1. @JsonIgnoreProperties: This annotation is used for removing irrelevant json property tags that are generated as part of the Swagger resource generation from the Swagger UI document params and mentioned above the Domain class name. For example:

|  |
| --- |
| @JsonIgnoreProperties(['dirtyPropertyNames', 'errors', 'dirty', 'attached', 'version' , 'properties' ]) |

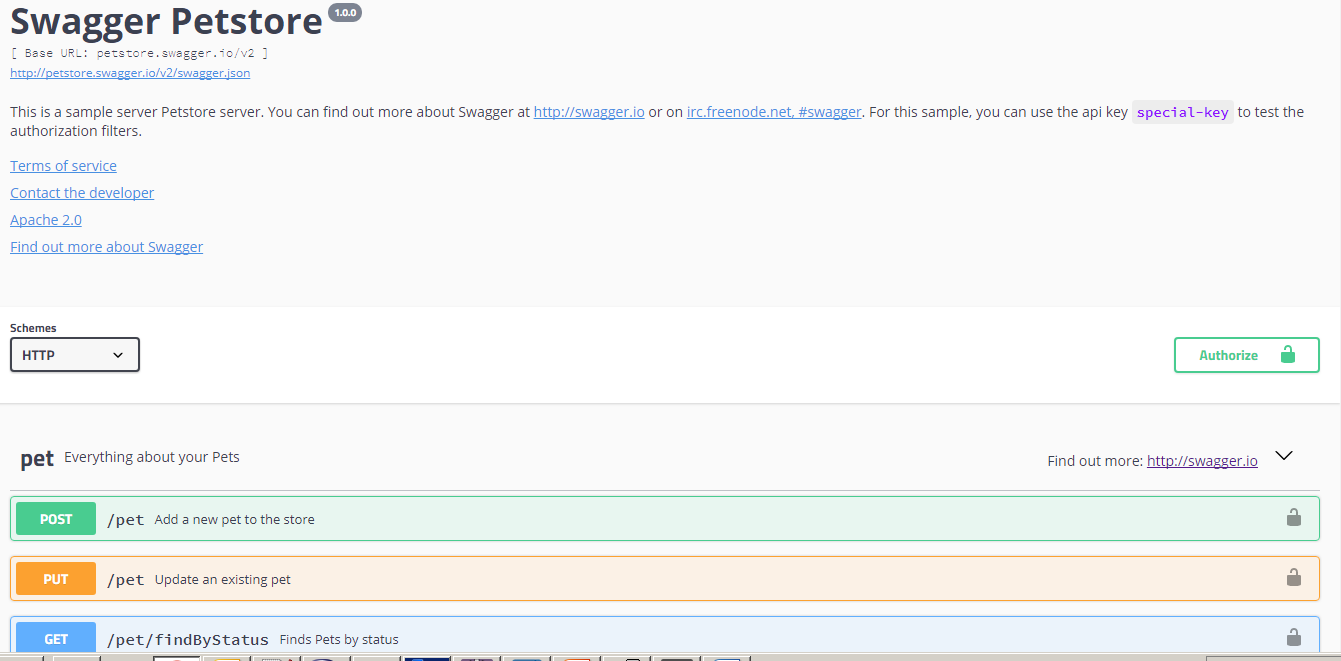
# Swagger Document Invocation:

After successful integration and deployment of the swagger enabled code base we can invoke the Swagger REST API Documentation by proving the below formatted URL in the browser.

URL: http://<hostname>:<port>/<configured url pattern path>/ <mapped ApiDocController path in Url mappings file>

For Example: <http://localhost:8080/v1.0/errorLogServiceDocApi>

Sample Swagger UI:



# Swagger Editor Local Set-Up:

This Step is not part of the development process. If the individual wants to set up the Swagger Editor locally then we should follow these steps. With this one can import the swagger resource files into the editor and view the content.

This step is ideally required for the Manual /Technical writer Approach of writing the Open API Specification for REST Web APIs.

1. Install node.js

Install node.js to your local machine as an admin. Follow the steps in installer. Once installed successfully, cross check the path variable updated with node installed location e.g. “C:\NODE\_INSTALLED\_DIR\nodejs”

1. Install node.js dependency

After nodejs installation, open new nodejs command prompt and prefebly close the one opened.

Run following command to install npm dependency

–> npm install -g npm

installed version by running command

–> npm -v

if any issue or error installing npm , reinstall node and then try to install npm again.

1. Install swagger dependency

This is not mandatory but could be useful when creating project from command prompt using swagger commands.

-> npm install -g swagger

1. Download Swagger Editor

Download latest version of Swagger-Editor from git-hub by providing the below command.

git clone <https://github.com/swagger-api/swagger-editor.git>

1. Swagger Editor Run

Follow the below commands sequentially to run the Swagger editor locally

* cd swagger-editor
* npm install sudo -i / npm install sudo -H (for run as admin/root we need to pass sudo with i/h)
* npm start

Success message after starting the npm local server (http server):

|  |
| --- |
| Starting up http-server, serving ./  Available on:  http://10.177.225.122:3001  http://127.0.0.1:3001 |