SpringRest
==========
  Hanlding Exception in SpringRest(GlobalExceptionHandling)

To handle the exception Globally in SpringRest we use the following annotations
      a. @RestControllerAdvice
      b. @ExceptioniHandler
      c. Return Type of the method should be public ResponseEntity<ErrorEntity>.

APICreation/Development
=======================
 Developing RestController having different methods/opertios for various httpmethods like
       a. GET b. POST c. PATCH d. PUT e. DELETE
  is called APICreation/Devleopment.

 => We can generatlly take one @RestController per 1 module, so developing each RestController is called "APICreation".

Accounts Module   => AccountsController(@RestController) is called AccountsAPI Creation.
TransactionModule => TransactionController(@RestController) is called TransctionAPI Creation.

Note::
API(Corejava,AdvancedJava,Framework like hiberante,spring,springboot)
      Collection of .class file zipped in the form of jar and given to the endusers

RestAPI(Generating the RestEndPoints to establish cross communication b/w 2 programms where programms can be of same
            language/different langauge)

EndPoints:: Providing mulitple details or collection of details that is required to call methods/operations of RestController from ClientApp
               or to send the request from different tools like POSTMAN is called "ENDPOINTS".
                  eg:: AccountController nothing but AccountAPI then the endpoints are
                                 BASEURL :: http:://lh:9999/RestProj/api/accounts
                                           register()   :: /register   -> POST
                                           findById()   :: /find/{id}   -> GET
                                           deleteById() :: /delete/{id} ->DELETE

In EndPoints of any API, we need to provide multiple details like URL,,methodNames,requestpath,httpmethod types, content type and etc...
                              refer:: .png image

API Documentation
=================
   => We can write documentationfor java classes in mulitple ways
            a. Using seperate text docs
            b. Using API documentation comments and javadoc tool

1.
/* text */

The compiler ignores everything from /* to */.

2.
//text
The compiler ignores everything from // to the end of the line.

3
/** documentation */
This is a documentation comment and in general its called doc comment. The JDK javadoc tool uses doc comments when preparing automatically generated documentation.

Note: Both these approaches are non-responsive documentation.we can read about the java class/methods but we cannot test the immediately.

After developing REST apis we can provide API documentation for RestAPI in the following ways
a. Using seperate text docs
b. Using API documentation comments and javadoc tool
c. Using Swagger/Swagger API(Creates Responsive API Documentation)
d. OpenAPI(it is an alternative to Swagger,but still comapnies are using Swagger only)

SwaggerAPI
==========
=> It is an open source thirdparty library to provide Responsive API documentation for RestController and its methods
=> For All RestControllers of the project we can create API documentation from Single place while working swagger api
=> Responsive Documentation means not only we get docs about API and its methods(EndPoints) we can test immediately by providing inputs and getting outputs.
=> It this is used POSTMAN tool is not needed seperately and it is alos useful to provide documentation based TestingEnvironemnt.
=> SpringFox+Swagger together released libraries that are required to use swagger in "SpringBoot Application".

Swagger Documentation provides the following details
a. API info(company,title,license url,....)
b. End Points Info
c. Model class info
;;;

=> While working with Swagger documentation for reading and testing we need not remember and give URL,HttpMethods types,contenttype and etc.
we just need to give inputs and get the ouptus from the Application.

Procedure to work with Swagger api
==================================
1. Keep RestController/RestAPI project ready
2. Add the following two jars in pom.xml file related to swagger api
a. Springfox-swagger2
b. springfor-swagger-ui

```
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger2 -->
<dependency>
    <groupId>io.springfox</groupId>
```

```
		<artifactId>springfox-swagger2</artifactId>
		<version>2.9.2</version>
	</dependency>

	<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui -->
	<dependency>
		<groupId>io.springfox</groupId>
		<artifactId>springfox-swagger-ui</artifactId>
		<version>2.9.2</version>
	</dependency>
```

3. Develops seperate Configuration class enbaling Swagger api
   => Create a Docket object having
      a. Documentation type
      b. specificying the base package of restcontroller
      c. specifying the requestpath info
      d. other details of API(AP info object having companyname,license url,...)

```
@Configuration
@EnableSwagger2
public class SwaggerDocsConfig {

	@Bean
	public Docket createDocket() {
		return new Docket(DocumentationType.SWAGGER_2)// UI Screen type
				.select() // to specify RestControllers
				.apis(RequestHandlerSelectors.basePackage("in.ineuron.restcontroller"))// base packages for// RestController
				.paths(PathSelectors.regex("/api/tourist.*"))// To specify the request paths
				.build()// build the docket object
				.useDefaultResponseMessages(true)
				.apiInfo(getApiInfo());
	}

	private ApiInfo getApiInfo() {
		Contact contact = new Contact("nitin", "http://www.ineuron.ai/course", "nitin@ineuron.ai@gmail.com");

		return new ApiInfo("TouristInfo",
					"Gives information about tourist activities",
					"3.4.RELEASE",
					"http:www.hcl.com/license",
					contact,
					"GNU PUBLIC",
					"http://apache.org/license/guru",
					 Collections.emptyList());
	}
}
```

specify this key in application.properties

```
=======================================
#Configuring the information about swagger to display the matching expression
spring.mvc.pathmatch.matching-strategy = ANT_PATH_MATCHER
```

Send the request to the following url to get the swagger ui

http://localhost:9999/RestMiniProject/swagger-ui.html

@ApiOperation :: It can be applied on the RestAPI/Controller methods to provider our choice description and that reflects in swagger docs.

Eg::

```
@RestController
@RequestMapping("/api/tourist")
public class TouristController {

	@Autowired
	private ITouristMgmtService service;

	@PostMapping("/register")
	@ApiOperation("For Tourist Registration")
	public ResponseEntity<String> enrollTourist(@RequestBody Tourist tourist) {

		String resultMsg = service.registerTourist(tourist);
		return new ResponseEntity<String>(resultMsg, HttpStatus.OK);

	}
}
```

refer :: .png

Developing Consumer App using RestTemplate
==========

=> it alllows to devleop to Consumer/ClientApp RestFulWebService as a Programable client in java environment.

=> We need to take seperate WebService/MVC Project for this having logics to consume the Webservice/Api by calling methods.

=> The object RestTemplate would not come through AutoConfiguration process, so it should be created either using

new operator :: RestTemplate template = new RestTemplate()

=> In Configuration class

```
@Bean
public RestTemplate createTemplate(){
		return new RestTemplate();
}
```

=> RestTemplate Object provides methods to generate different modes request like GET/POST/PUT/Delete to Consume the Restful Service

=> While using this methods to consume Restful WebService/API we need details inputs(nothing but endpoints) like baseurl,httpmethod type,http header info like content type and etc...

=> It also provides XXXforEntity() methods like getForEntity,postForeEntity() and etc taking url,requestobj(body,header) to send different modes of http requests as method calls to consume the restultFul Webservices(Consumer)

Note:: Do not forget Webservices is given to link differnt apps that are developed either in same langauge or in different language and running in same server or different server belonging to same machine or different machine.

=> So far we have developed only RestFulWebservice(Provider app) and we tested that Server app using tools like "POSTMAN/Swagger"

=> Instead of using these tools we can develop programmable realclient apps with the support of RestTemplate in SpringEnvironment.