Swagger documentation in RESTful web service

# Steps to create swagger documentation

1. Assumed that you have a complete restful web service deployable code.
2. Add the following dependencies in maven pom.xml file.

<!-- Swagger Lib -->

<dependency>

<groupId>io.swagger</groupId>

<artifactId>swagger-jersey2-jaxrs</artifactId>

<version>1.5.9</version>

</dependency>

<dependency>

<groupId>io.swagger</groupId>

<artifactId>swagger-annotations</artifactId>

<version>1.5.9</version>

</dependency>

<dependency>

<groupId>io.swagger</groupId>

<artifactId>swagger-core</artifactId>

<version>1.5.9</version>

</dependency>

<dependency>

<groupId>io.swagger</groupId>

<artifactId>swagger-jaxrs</artifactId>

<version>1.5.9</version>

</dependency>

1. Download the Swagger UI zip file , extract it and copy everything into “webapp” directory.
2. Modify the index.html file with the following contents.

**if** (url && url.length > 1) {

url = decodeURIComponent(url[1]);

} **else** {

//url = "http://petstore.swagger.io/v2/swagger.json";

url = "/jersey-spring-rest-swagger/api/swagger.json"; <!—New Modification -->

}

1. Finally start documenting your web service class and methods with the API provided by Swagger.
2. Add the following two lines in the main resource class which extends Application in jersey

//For swagger

resources.add(io.swagger.jaxrs.listing.ApiListingResource.**class**);

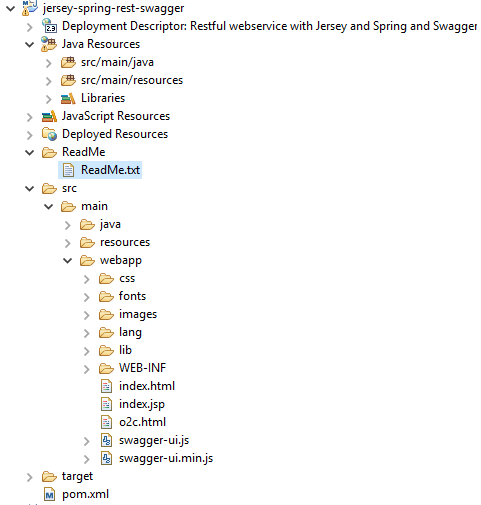
resources.add(io.swagger.jaxrs.listing.SwaggerSerializers.**class**);

1. Finally deploy the war file in Tomcat and access the following URL.

<http://localhost:8090/jersey-spring-rest-swagger/index.html>

The complete code is given below.

# Directory Structure



# Maven Configuration (pom.xml)

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>jersey-spring-rest-swagger</groupId>

<artifactId>jersey-spring-rest-swagger</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>jersey-spring-rest-swagger Maven Webapp</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<org.springframework.version>3.2.1.RELEASE</org.springframework.version>

</properties>

<dependencies>

<!-- Jersey related jar files -->

<dependency>

<groupId>javax.ws.rs</groupId>

<artifactId>javax.ws.rs-api</artifactId>

<version>2.0</version>

</dependency>

<dependency>

<groupId>org.glassfish.jersey.containers</groupId>

<artifactId>jersey-container-servlet</artifactId>

<version>2.6</version>

</dependency>

<dependency>

<groupId>org.glassfish.jersey.core</groupId>

<artifactId>jersey-client</artifactId>

<version>2.6</version>

</dependency>

<dependency>

<groupId>org.glassfish.jersey.media</groupId>

<artifactId>jersey-media-multipart</artifactId>

<version>2.5.1</version>

</dependency>

<dependency>

<groupId>org.glassfish.jersey.media</groupId>

<artifactId>jersey-media-json-jackson</artifactId>

<version>2.5.1</version>

</dependency>

<dependency>

<groupId>org.codehaus.jackson</groupId>

<artifactId>jackson-core-asl</artifactId>

<version>1.9.13</version>

</dependency>

<dependency>

<groupId>org.codehaus.jackson</groupId>

<artifactId>jackson-mapper-asl</artifactId>

<version>1.9.13</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

<!-- <scope>provided</scope> -->

</dependency>

<!-- Jersey Spring Integration -->

<dependency>

<groupId>org.glassfish.jersey.ext</groupId>

<artifactId>jersey-spring3</artifactId>

<version>2.14</version>

</dependency>

<!-- Spring Jar files -->

<!-- Bean Factory and JavaBeans utilities (depends on spring-core) Define

this if you use Spring Bean APIs (org.springframework.beans.\*) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<!-- Swagger Lib -->

<dependency>

<groupId>io.swagger</groupId>

<artifactId>swagger-jersey2-jaxrs</artifactId>

<version>1.5.9</version>

</dependency>

<dependency>

<groupId>io.swagger</groupId>

<artifactId>swagger-annotations</artifactId>

<version>1.5.9</version>

</dependency>

<dependency>

<groupId>io.swagger</groupId>

<artifactId>swagger-core</artifactId>

<version>1.5.9</version>

</dependency>

<dependency>

<groupId>io.swagger</groupId>

<artifactId>swagger-jaxrs</artifactId>

<version>1.5.9</version>

</dependency>

</dependencies>

<build>

<finalName>jersey-spring-rest-swagger</finalName>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.3</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

# Index.html

Here brief code snippet is given below.

$(**function** () {

**var** url = window.location.search.match(/url=([^&]+)/);

**if** (url && url.length > 1) {

url = decodeURIComponent(url[1]);

} **else** {

//url = "http://petstore.swagger.io/v2/swagger.json";

//url = "http://localhost:8090/jersey-spring-rest/api/swagger.json";

**url = "/jersey-spring-rest-swagger/api/swagger.json";**

}

# Deployment Descriptor (web.xml)

<!DOCTYPE web-app PUBLIC

"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

"http://java.sun.com/dtd/web-app\_2\_3.dtd" >

<web-app>

<display-name>Restful webservice with Jersey and Spring and Swagger</display-name>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:applicationContext.xml</param-value>

</context-param>

**<!-- For support for CORS, as per Swagger github specification -->**

**<!-- This should be used only in worst condition -->**

**<filter>**

**<filter-name>ApiOriginFilter</filter-name>**

**<filter-class>com.ddlab.rnd.rest.filter.ApiOriginFilter</filter-class>**

**</filter>**

**<filter-mapping>**

**<filter-name>ApiOriginFilter</filter-name>**

**<url-pattern>/\*</url-pattern>**

**</filter-mapping>**

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<servlet>

<servlet-name>com.ddlab.rest.resources.ApplicationPkgs</servlet-name>

<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>

<init-param>

<param-name>javax.ws.rs.Application</param-name>

<param-value>

com.ddlab.rnd.rest.resources.ApplicationPkgs

</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<!-- Changes for Swagger docs -->

**<servlet>**

**<servlet-name>Jersey2Config</servlet-name>**

**<servlet-class>io.swagger.jersey.config.JerseyJaxrsConfig</servlet-class>**

**<init-param>**

**<param-name>api.version</param-name>**

**<param-value>1.0.0</param-value>**

**</init-param>**

**<init-param>**

**<param-name>swagger.api.basepath</param-name>**

**<!-- <param-value>http://localhost:8090/jersey-spring-rest/api</param-value> -->**

**<param-value>/jersey-spring-rest-swagger/api</param-value>**

**</init-param>**

**<load-on-startup>2</load-on-startup>**

**</servlet>**

<servlet-mapping>

<servlet-name>com.ddlab.rest.resources.ApplicationPkgs</servlet-name>

<url-pattern>/api/\*</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

</welcome-file-list>

</web-app>

# Spring configuration file (applicationContext.xml)

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>

<context:component-scan base-package=*"com.ddlab"* />

<bean id=*"dbHandler"* class=*"com.ddlab.rest.service.EmployeeServiceImpl"* />

</beans>

# Java Classes

## ApplicationPkgs.java

**package** com.ddlab.rnd.rest.resources;

**import** java.util.HashSet;

**import** java.util.Set;

**import** javax.ws.rs.core.Application;

**import** org.glassfish.jersey.jackson.JacksonFeature;

**public** **class** ApplicationPkgs **extends** Application {

@Override

**public** Set<Class<?>> getClasses() {

Set<Class<?>> resources = **new** HashSet<Class<?>>();

resources.add(EmployeeServiceResource.**class**);

resources.add(JacksonFeature.**class**);

//For swagger

**resources.add(io.swagger.jaxrs.listing.ApiListingResource.class);**

**resources.add(io.swagger.jaxrs.listing.SwaggerSerializers.class);**

**return** resources;

}

}

## BaseResource.java

**package** com.ddlab.rnd.rest.resources;

**import** java.nio.file.AccessDeniedException;

**import** javax.ws.rs.NotFoundException;

**import** javax.ws.rs.WebApplicationException;

**import** javax.ws.rs.core.Response;

**import** javax.ws.rs.core.Response.Status;

**public** **class** BaseResource {

**protected** WebApplicationException createWebappException(Exception incomingException) {

Status status;

String string = **null**;

**if** (incomingException **instanceof** SecurityException || incomingException **instanceof** AccessDeniedException) {

status = Status.***FORBIDDEN***;

} **else** **if** (incomingException **instanceof** IllegalArgumentException) {

status = Status.***BAD\_REQUEST***;

string=incomingException.getMessage();

} **else** **if** (incomingException **instanceof** NotFoundException) {

status = Status.***NOT\_FOUND***;

} **else** **if** (incomingException **instanceof** Exception) {

status = Status.***INTERNAL\_SERVER\_ERROR***;

} **else** {

status = Status.***INTERNAL\_SERVER\_ERROR***;

}

**return** **new** WebApplicationException(Response.*status*(status).entity(string).type("text/plain").build());

}

}

## EmployeeServiceResource.java

**package** com.ddlab.rnd.rest.resources;

**import** javax.servlet.ServletConfig;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.ws.rs.Consumes;

**import** javax.ws.rs.DELETE;

**import** javax.ws.rs.GET;

**import** javax.ws.rs.POST;

**import** javax.ws.rs.PUT;

**import** javax.ws.rs.Path;

**import** javax.ws.rs.Produces;

**import** javax.ws.rs.QueryParam;

**import** javax.ws.rs.core.Context;

**import** javax.ws.rs.core.MediaType;

**import** javax.ws.rs.core.Response;

**import** javax.ws.rs.core.Response.ResponseBuilder;

**import** javax.ws.rs.core.Response.Status;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** com.ddlab.rest.entity.ServerResponse;

**import** com.ddlab.rest.entity.User;

**import** com.ddlab.rest.service.EmployeeService;

**import** io.swagger.annotations.Api;

**import** io.swagger.annotations.ApiOperation;

**import** io.swagger.annotations.ApiResponse;

**import** io.swagger.annotations.ApiResponses;

@Path("1/empservice")

@Api(value="/empservice",tags={"empservice"})

**public** **class** EmployeeServiceResource **extends** BaseResource {

@Context

HttpServletRequest request;

@Context

ServletConfig servletConfig;

@Autowired

**private** EmployeeService empService;

@Path("/create")

@POST

@Consumes({ MediaType.***APPLICATION\_XML***, MediaType.***APPLICATION\_JSON*** })

@Produces({ MediaType.***APPLICATION\_XML***, MediaType.***APPLICATION\_JSON*** , MediaType.***TEXT\_PLAIN***})

@ApiOperation(value="Create a user", notes="used to create a user",code=200)

@ApiResponses(value={

@ApiResponse(code=200, message="OK"),

@ApiResponse(code=500, message="Internal Error")

})

**public** Response createUser(User user) {

ResponseBuilder responseBuilder = **null**;

**try** {

**if**(user.getUserName() == **null** || user.getPassword() == **null** )

responseBuilder = Response.*status*(Status.***BAD\_REQUEST***).entity("Incorrect password").header("Access-Control-Allow-Origin", "\*").header("Access-Control-Allow-Methods", "GET, POST, DELETE, PUT");

empService.createUser(user);

responseBuilder = Response.*status*(Status.***CREATED***).entity(**new** ServerResponse("Success", "User created successfully"));

} **catch** (Exception e) {

responseBuilder = Response.*serverError*();

}

**return** responseBuilder.build();

}

@Path("/update")

@PUT

@Consumes({ MediaType.***APPLICATION\_XML***, MediaType.***APPLICATION\_JSON*** })

@Produces({ MediaType.***APPLICATION\_XML***, MediaType.***APPLICATION\_JSON*** })

@ApiOperation(value="Update a user", notes="used to update a user")

**public** Response updateUser(User user) {

ResponseBuilder responseBuilder = **null**;

**if**(user.getUserName() == **null** || user.getPassword() == **null** )

**throw** createWebappException(**new** IllegalArgumentException("Incorrect credentials"));

empService.updateUser(user);

responseBuilder = Response.*ok*(**new** ServerResponse("Success", "User updated successfully"));

**return** responseBuilder.build();

}

@Path("/delete")

@DELETE

@Consumes({ MediaType.***APPLICATION\_XML***, MediaType.***APPLICATION\_JSON*** })

@Produces({ MediaType.***APPLICATION\_XML***, MediaType.***APPLICATION\_JSON*** })

@ApiOperation(value="Delete a user", notes="used to delete a user")

**public** Response deleteUser(User user) {

ResponseBuilder responseBuilder = **null**;

**if**(user.getUserName() == **null** || user.getPassword() == **null** )

**throw** createWebappException(**new** IllegalArgumentException("Incorrect credentials"));

empService.deleteUser(user);

responseBuilder = Response.*ok*(**new** ServerResponse("Success", "User "+user.getUserName()+" has been removed from the system successfully"));

**return** responseBuilder.build();

// return "User "+user.getUserName()+" has been removed from the system successfully";

}

@Path("/userid")

@GET

@Consumes({ MediaType.***APPLICATION\_XML***, MediaType.***APPLICATION\_JSON*** })

@Produces({ MediaType.***APPLICATION\_XML***, MediaType.***APPLICATION\_JSON*** })

@ApiOperation(value="get the user details", notes="Some notes", response=User.**class**)

**public** User getUser(@QueryParam("id") **int** id) { //204 No content

**if**(id == 0 )

**throw** createWebappException(**new** IllegalArgumentException("Incorrect ID"));

**return** empService.getUserById(String.*valueOf*(id));

}

}

## ApiOriginFilter.java

This class is only used in case of CORS support, otherwise it is not required.

**package** com.ddlab.rnd.rest.filter;

**import** java.io.IOException;

**import** javax.servlet.\*;

**import** javax.servlet.http.HttpServletResponse;

**public** **class** ApiOriginFilter **implements** javax.servlet.Filter {

@Override

**public** **void** doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

**throws** IOException, ServletException {

HttpServletResponse res = (HttpServletResponse) response;

res.addHeader("Access-Control-Allow-Origin", "\*");

res.addHeader("Access-Control-Allow-Methods", "GET, POST, DELETE, PUT");

res.addHeader("Access-Control-Allow-Headers", "Content-Type, api\_key, Authorization");

chain.doFilter(request, response);

}

@Override

**public** **void** destroy() {

}

@Override

**public** **void** init(FilterConfig filterConfig) **throws** ServletException {

}

}

## EmployeeService.java

**package** com.ddlab.rest.service;

**import** com.ddlab.rest.entity.User;

**public** **interface** EmployeeService {

**public** **void** createUser(User user);

**public** **void** updateUser(User user);

**public** **void** deleteUser(User user);

**public** User getUserById(String id);

}

## EmployeeServiceImpl.java

**package** com.ddlab.rest.service;

**import** com.ddlab.rest.entity.User;

**public** **class** EmployeeServiceImpl **implements** EmployeeService {

**public** **void** createUser(User user) {

System.***out***.println("User created successfully...");

}

**public** **void** updateUser(User user) {

System.***out***.println("User information updated successfully...");

}

**public** **void** deleteUser(User user) {

System.***out***.println("User information deleted successfully...");

}

**public** User getUserById(String id) {

**if**( id == **null** ) **throw** **new** NullPointerException("User id can not be blank or empty");

User user = **new** User();

user.setPassword("");

user.setUserName("Deb");

user.setId(Integer.*parseInt*(id));

**return** user;

}

}

## User.java

**package** com.ddlab.rest.entity;

**import** javax.xml.bind.annotation.XmlAccessType;

**import** javax.xml.bind.annotation.XmlAccessorType;

**import** javax.xml.bind.annotation.XmlElement;

**import** javax.xml.bind.annotation.XmlRootElement;

**import** javax.xml.bind.annotation.XmlType;

**import** org.codehaus.jackson.annotate.JsonProperty;

**import** org.codehaus.jackson.annotate.JsonPropertyOrder;

**import** io.swagger.annotations.ApiModel;

**import** io.swagger.annotations.ApiModelProperty;

@XmlRootElement(name = "User")

@XmlAccessorType(XmlAccessType.***FIELD***)

@XmlType(propOrder={"userName", "password","id"})

@JsonPropertyOrder(value={"userName", "password", "id"})

@ApiModel(value = "User", description = "User description")

**public** **class** User {

@XmlElement(name = "username")

@JsonProperty("username")

@ApiModelProperty(name="username", value = "username", dataType = "java.lang.String", required = **true**)

**private** String userName;

@XmlElement(name = "password")

@JsonProperty("password")

// @ApiModelProperty(value = "password", required=true)

@ApiModelProperty(name = "password", value = "password", dataType = "java.lang.String", required = **true**)

**private** String password;

@XmlElement(name = "id")

@JsonProperty("id")

@ApiModelProperty(name="id", dataType = "java.lang.Integer", required=**false**)

**private** **int** id;

**public** String getUserName() {

**return** userName;

}

**public** **void** setUserName(String userName) {

**this**.userName = userName;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

}

## ServerResponse.java

**package** com.ddlab.rest.entity;

**import** javax.xml.bind.annotation.XmlAccessType;

**import** javax.xml.bind.annotation.XmlAccessorType;

**import** javax.xml.bind.annotation.XmlElement;

**import** javax.xml.bind.annotation.XmlRootElement;

**import** javax.xml.bind.annotation.XmlType;

**import** org.codehaus.jackson.annotate.JsonProperty;

**import** org.codehaus.jackson.annotate.JsonPropertyOrder;

**import** io.swagger.annotations.ApiModel;

**import** io.swagger.annotations.ApiModelProperty;

@XmlRootElement(name = "ServerResponse")

@XmlAccessorType(XmlAccessType.***FIELD***)

@XmlType(propOrder={"status", "message"})

@JsonPropertyOrder(value={"status", "message"})

@ApiModel(value = "ServerResponse", description = "Custom response from server")

**public** **class** ServerResponse {

@XmlElement(name = "status")

@JsonProperty("status")

@ApiModelProperty(name="status", value = "status", dataType = "java.lang.String")

**private** String status;

@XmlElement(name = "message")

@JsonProperty("message")

@ApiModelProperty(name="message", value = "message", dataType = "java.lang.String")

**private** String message;

**public** ServerResponse(String status,String message) {

**this**.status = status;

**this**.message = message;

}

**public** String getStatus() {

**return** status;

}

**public** String getMessage() {

**return** message;

}

}

**ReadMe details**

Doc URL : http://localhost:8090/jersey-spring-rest-swagger/

POST http://localhost:8090/jersey-spring-rest-swagger/api/1/loginservice/create

{"username":"Deba","password":"deba"}

PUT http://localhost:8090/jersey-spring-rest-swagger/api/1/loginservice/update

{"username":"Deba","password":"deba"}

DELETE http://localhost:8090/jersey-spring-rest-swagger/api/1/loginservice/delete

{"username":"Deba","password":"deba"}

GET http://localhost:8090/jersey-spring-rest-swagger/api/1/loginservice/userid?id=123