

Spring boot, jersey, swagger-2017

Spring, Boot, Jersey,swagger



May 7, 2017

DDLAB INC

Debadatta Mishra

[Spring Boot 3](#_Toc481951836)

[Features 3](#_Toc481951837)

[Spring Boot Basic Web Application 4](#_Toc481951838)

[Project Structure 4](#_Toc481951839)

[Maven Configuration (pom.xml) 4](#_Toc481951840)

[WEB-INF/pages 5](#_Toc481951841)

[Index.jsp 5](#_Toc481951842)

[Login.jsp 5](#_Toc481951843)

[Show.jsp 5](#_Toc481951844)

[Application Configuration (application.properties) 6](#_Toc481951845)

[Java Code 6](#_Toc481951846)

[Application.java 6](#_Toc481951847)

[IndexPageController.java 6](#_Toc481951848)

[LoginPageController.java 6](#_Toc481951849)

[LoginSubmitController.java 7](#_Toc481951850)

[Spring Boot With Jersey 8](#_Toc481951851)

[Project Structure 8](#_Toc481951852)

[Maven Configuration (pom.xml) 8](#_Toc481951853)

[Application Configuration (application.properties) 9](#_Toc481951854)

[Java Code 9](#_Toc481951855)

[Spring Boot and Config specific java classes 9](#_Toc481951856)

[SpringBootJerseyApplication.java 9](#_Toc481951857)

[Jersey Configuration Java File 10](#_Toc481951858)

[ApplicationPkgs.java 10](#_Toc481951859)

[Resource Layer 11](#_Toc481951860)

[BaseResource.java 11](#_Toc481951861)

[ITCResources.java 12](#_Toc481951862)

[ITCUploadService.java 15](#_Toc481951863)

[ITCDownloadService.java 18](#_Toc481951864)

[Service Layer 19](#_Toc481951865)

[IUserService.java 19](#_Toc481951866)

[UserServiceImpl.java 19](#_Toc481951867)

[Util Layer 20](#_Toc481951868)

[Constants.java 20](#_Toc481951869)

[FileUtil.java 20](#_Toc481951870)

[Entity Layer 21](#_Toc481951871)

[Xternalapp.properties 23](#_Toc481951872)

[Spring Boot with Rest and Swagger (No Jersey) 24](#_Toc481951873)

[Project Structure 24](#_Toc481951874)

[Maven Configuration (pom.xml) 24](#_Toc481951875)

[Application Configuration (src/main/resources/application.properties) 25](#_Toc481951876)

[Java Code 25](#_Toc481951877)

[SpringBootApp.java 25](#_Toc481951878)

[SwaggerConfig.java 25](#_Toc481951879)

[ITCResources.java 26](#_Toc481951880)

[How to Access 26](#_Toc481951881)

[Spring Boot with Jersy and Swagger 27](#_Toc481951882)

[Project Structure 27](#_Toc481951883)

[Maven Configuration (pom.xml) 27](#_Toc481951884)

[Spring Boot Configuration (src.main/resources/application.properties) 29](#_Toc481951885)

[Java Classes 29](#_Toc481951886)

[Application.java 29](#_Toc481951887)

[ApplicationPkgs.java 30](#_Toc481951888)

[IndexPageController.java 30](#_Toc481951889)

[ITCResources.java 31](#_Toc481951890)

[Changes in Index.html of swagger doc 32](#_Toc481951891)

[URLS to Use 32](#_Toc481951892)

Spring Boot

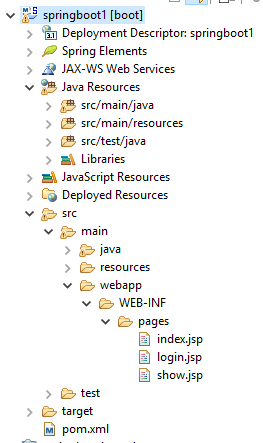
Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

# Features

* Create stand-alone Spring applications
* Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
* Provide opinionated 'starter' POMs to simplify your Maven configuration
* Automatically configure Spring whenever possible
* Provide production-ready features such as metrics, health checks and externalized configuration
* Absolutely **no code generation** and **no requirement for XML** configuration

Spring Boot Basic Web Application

# **Project 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>springboot1</groupId>

<artifactId>springboot1</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>springboot1 Maven Webapp</name>

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

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.2.7.RELEASE</version>

</parent>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

</dependencies>

<build>

<finalName>springboot1</finalName>

<plugins>

<plugin>

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

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

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

# **WEB-INF/pages**

## **Index.jsp**

<!DOCTYPE html>

<%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"*%>

<html>

<body>

<h2>Welcome to the World of Spring Boot</h2>

<h2>Message From Server : ${message}</h2>

<a href=*"login"*> Login </a>

<p>

</body>

</html>

## **Login.jsp**

<!DOCTYPE html>

<%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"*%>

<html>

<body>

<h2>Login with your credentials</h2>

<form action=*"loginSubmit"* method=*"post"*>

Username: <input type=*"text"* name=*"username"* value=*""*>

<input type=*"submit"* value=*"Submit"* name=*"submit"*>

</form>

</body>

</html>

## **Show.jsp**

<!DOCTYPE html>

<%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"*%>

<html>

<body>

<h2>You have successfully logged in ${name}</h2>

</body>

</html>

# **Application Configuration (application.properties)**

server.contextPath=/springboot1

spring.view.prefix: /WEB-INF/pages/

spring.view.suffix: .jsp

server.port=8090

#application.message: Hello World

# **Java Code**

## **Application.java**

**package** com.ddlab.rnd.spring.boot;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** Application {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(Application.**class**, args);

}

}

## **IndexPageController.java**

**package** com.ddlab.rnd.spring.boot;

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.servlet.ModelAndView;

@Controller

**public** **class** IndexPageController {

@RequestMapping("/")

**public** ModelAndView index() {

Map<String, String> map = **new** HashMap<String, String>();

map.put("message", "A warm welcome to you.");

**return** **new** ModelAndView("index", map);

}

}

## **LoginPageController.java**

**package** com.ddlab.rnd.spring.boot;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.RequestMapping;

@Controller

**public** **class** LoginPageController {

@RequestMapping("/login")

**public** String login() {

**return** "login";

}

}

## **LoginSubmitController.java**

**package** com.ddlab.rnd.spring.boot;

**import** java.io.UnsupportedEncodingException;

**import** java.net.URLDecoder;

**import** java.util.HashMap;

**import** java.util.LinkedHashMap;

**import** java.util.Map;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.servlet.ModelAndView;

@Controller

**public** **class** LoginSubmitController {

@RequestMapping(value="/loginSubmit", method=RequestMethod.***POST***)

~~// public ModelAndView loginSubmit( @ModelAttribute ModelMap modelMap ) { //~~

**public** ModelAndView loginSubmit(@RequestBody String body) **throws** Exception {

Map<String,String> paramMap = *splitQuery*(body);

System.***out***.println(paramMap);

System.***out***.println(body);

Map<String, String> map = **new** HashMap<String, String>();

// map.put("name", "Piku");

map.put("name", paramMap.get("username"));

**return** **new** ModelAndView("show", map);

// return "index";

}

**public** **static** Map<String, String> splitQuery(String query) **throws** UnsupportedEncodingException {

Map<String, String> query\_pairs = **new** LinkedHashMap<String, String>();

String[] pairs = query.split("&");

**for** (String pair : pairs) {

**int** idx = pair.indexOf("=");

query\_pairs.put(URLDecoder.*decode*(pair.substring(0, idx), "UTF-8"), URLDecoder.*decode*(pair.substring(idx + 1), "UTF-8"));

}

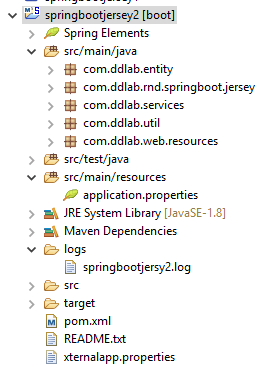
**return** query\_pairs;

}

}

Spring Boot With Jersey

# **Project 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/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>springbootjersey2</groupId>

<artifactId>springbootjersey2</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>springbootjersey2</name>

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

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.2.7.RELEASE</version>

</parent>

<properties>

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

</properties>

<dependencies>

<dependency>

<!-- only dependency needed for spring boot jersey -->

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jersey</artifactId>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<scope>test</scope>

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

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

<version>2.14</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

# **Application Configuration (application.properties)**

server.contextPath=/springbootjersey2/api

server.port=8090

#logging.path=E:/springboot-2017/springbootjersey2/logs

#logging.path does not work, hence give the full path with file name

logging.file=E:/springboot-2017/springbootjersey2/logs/springbootjersy2.log

# **Java Code**

## **Spring Boot and Config specific java classes**

Make sure that both the java classes are in the same package.

## **SpringBootJerseyApplication.java**

**package** com.ddlab.rnd.springboot.jersey;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.annotation.ComponentScan;

@ComponentScan(basePackages={"com.ddlab.\*"})

@SpringBootApplication

**public** **class** SpringBootJerseyApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringBootJerseyApplication.**class**, args);

}

}

## **Jersey Configuration Java File**

It is not always necessary that the file name should be JerseyConfig.java as mentioned in various websites. The Java class name can be anything.

## **ApplicationPkgs.java**

**package** com.ddlab.rnd.springboot.jersey;

**import** java.util.HashSet;

**import** java.util.Set;

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

**import** org.glassfish.jersey.media.multipart.MultiPartFeature;

**import** org.glassfish.jersey.server.ResourceConfig;

**import** org.springframework.stereotype.Component;

**import** com.ddlab.web.resources.ITCDownloadService;

**import** com.ddlab.web.resources.ITCResources;

**import** com.ddlab.web.resources.ITCUploadService;

//To know about WADl : GET http://localhost:8090/springbootjersey2/api/application.wadl

**@Component** **//~~ This is mandatory**

**public** **class** ApplicationPkgs **extends** ResourceConfig {

**public** ApplicationPkgs() {

// super( ITCResources.class,ITCUploadService.class,ITCDownloadService.class,JacksonFeature.class,MultiPartFeature.class);

// OR you can write below

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

classes.add(ITCResources.**class**);

classes.add(ITCUploadService.**class**);

classes.add(ITCDownloadService.**class**);

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

classes.add(MultiPartFeature.**class**);

registerClasses(classes);

}

}

# **Resource Layer**

## **BaseResource.java**

**package** com.ddlab.web.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***;

string=incomingException.getMessage();

} **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());

}

}

## **ITCResources.java**

**package** com.ddlab.web.resources;

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

**import** javax.ws.rs.FormParam;

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

**import** javax.ws.rs.HeaderParam;

**import** javax.ws.rs.MatrixParam;

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

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

**import** javax.ws.rs.PathParam;

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

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

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

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

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

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

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

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

**import** com.ddlab.services.IUserService;

@Path("/itc")

**public** **class** ITCResources **extends** BaseResource {

@Autowired @Qualifier("userService")

**private** IUserService userService;

@GET

**public** String message() {

**return** "Hello " + "Piku, you got spring with Jersey";

}

//GET http://localhost:8090/springbootjersey2/api/itc/user/0

//Status Code : 400 Bad Request

//Response body : Id cannot be 0, enter a valid id

@Path("/user/{id}")

@GET

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

**public** Response getUserById(@PathParam("id") **int** id) {

**if**( id == 0 ) **throw** createWebappException(**new** IllegalArgumentException("Id cannot be 0, enter a valid id"));

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

user.setFirstName("Deb");

user.setLastName("Mishra");

user.setId(id);

**return** Response.*ok*().entity(user).build();

}

//GET http://localhost:8090/springbootjersey2/api/itc/address/USA

//GET http://localhost:8090/springbootjersey2/api/itc/address/europe

//Status Code : 200 OK

@Path("/address/{areaCode}") //USA,EUROPE

@GET

@Produces(MediaType.***TEXT\_PLAIN***)

**public** Response getAddressByCode(@PathParam("areaCode") String areaCode) {

String addressString = **null**;

**try** {

**if** (areaCode.equalsIgnoreCase("USA"))

addressString = "12 North State, Route 17,Suite 303,Paramus,1234";

**else** **if** (areaCode.equalsIgnoreCase("Europe"))

addressString = "Newell Consulting Oy,P.O. Box 16 , Olari,02211 ";

**else** **if** (areaCode.equalsIgnoreCase("Africa"))

addressString = "Johannesburg,2nd Floor, West Tower,Nelson Mandela Square,Maude Street, Sandton,Johannesburg, 2196";

**else** **if** (areaCode.equalsIgnoreCase("Asia"))

addressString = "ITC Infotech India Limited, 18, Banaswadi Main Rd, Maruthi Sevanagar, Bangalore, 560005";

**else** {

addressString = "No such area code exists for ITC";

}

} **catch** (Exception e) {

**throw** createWebappException( **new** IllegalArgumentException("No such area code exists for ITC"));

}

**return** Response.*ok*().entity(addressString).build();

}

// ~~~~~~~~~~~~~~~~~~~~~~~ @QueryParam ~~~~~~~~~~~~~~~~~~~~~~~~~~~~

//GET http://localhost:8090/springbootjersey2/api/itc/regionaladdress/Europe?country=SE

//Status Code : 200 OK

@Path("/regionaladdress/{areaCode}") //USA,EUROPE

@GET

@Produces(MediaType.***TEXT\_PLAIN***)

**public** Response getAddressByCountry(@PathParam("areaCode") String areaCode, @QueryParam("country") String country) {

String adrsByCountry = **null**;

**try** {

**if** (areaCode.equalsIgnoreCase("USA")

&& country.equalsIgnoreCase("NJ"))

adrsByCountry = "12 North State, Route 17,Suite 303,Paramus,New Jersey,NJ-07652";

**else** **if** (areaCode.equalsIgnoreCase("Europe")

&& country.equalsIgnoreCase("FI"))

adrsByCountry = "Newell Consulting Oy,P.O. Box 16 , Olari,02211 Espoo, Helsinki";

**else** **if** (areaCode.equalsIgnoreCase("Europe")

&& country.equalsIgnoreCase("SE"))

adrsByCountry = "C/o Matrisen AB,Box 22059 , 104 22 Stockholm";

**else** **if** (areaCode.equalsIgnoreCase("Europe")

&& country.equalsIgnoreCase("DK"))

adrsByCountry = "Havnegade 39, 3. sal,1058 Copenhagen K";

**else** **if** (areaCode.equalsIgnoreCase("Asia")

&& country.equalsIgnoreCase("IN"))

adrsByCountry = "ITC Infotech India Limited, 18, Banaswadi Main Rd, Maruthi Sevanagar, Bangalore, 560005";

**else**

adrsByCountry = "No such area code exists for ITC";

} **catch** (Exception e) {

**throw** createWebappException( **new** IllegalArgumentException("No such area code exists for ITC"));

}

**return** Response.*ok*().entity(adrsByCountry).build();

}

// ~~~~~~~~~~~~~~~~~~~~~~~ @MatrixParam ~~~~~~~~~~~~~~~~~~~~~~~~~~~~

//GET http://localhost:8090/springbootjersey2/api/itc/itcaddress;country=FI;areacode=europe

//Status Code : 200 OK

@Path("/itcaddress")

@GET

@Produces(MediaType.***TEXT\_PLAIN***)

**public** Response getITCAddress(@MatrixParam("country") String country, @MatrixParam("areacode") String areaCode) {

**return** getAddressByCountry(areaCode, country);

}

// ~~~~~~~~~~~~~~~~~~~~~~~ @FormParam ~~~~~~~~~~~~~~~~~~~~~~~~~~~~

//POST http://localhost:8090/springbootjersey2/api/itc/postaladdress

//Content-Type : application/x-www-form-urlencoded

//Body : country=FI&areacode=Europe

//Status Code : 200 OK

@Path("/postaladdress")

@POST

@Produces(MediaType.***TEXT\_PLAIN***)

**public** Response postNGetITCAddress(@FormParam("country") String country,@FormParam("areacode") String areaCode) {

**return** getAddressByCountry(areaCode, country);

}

/\*

\* POST http://localhost:8090/springbootjersey2/api/itc/createuser

\* Content-Type : application/json

\* Body :

\* {

\* "firstName" : "Deb",

\* "lastName" : "Mishra",

\* "id" : 1

\* }

\*

\* Response Status Code: 201 Created

\*/

@Path("/createuser")

@POST

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

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

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

userService.createUser(user);

**return** Response.*status*(Status.***CREATED***).entity("User " + user.getFirstName()+" created successfully...").build();

}

/\*

\* To know the user-agent header

\* GET http://localhost:8090/springjersey/api/headers/get

\*

\* Response will be

\* In FireFox

\* addUser is called, userAgent : Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0

\*

\* In Chrome

\* addUser is called, userAgent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36

\*/

@GET

@Path("/headers/get")

**public** Response addUser(@HeaderParam("user-agent") String userAgent) {

String msg = "addUser is called, userAgent : " + userAgent;

**return** Response.*status*(200).entity(msg).build();

}

}

## **ITCUploadService.java**

**package** com.ddlab.web.resources;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.util.List;

**import** java.util.Map;

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

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

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

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

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

**import** org.glassfish.jersey.media.multipart.BodyPart;

**import** org.glassfish.jersey.media.multipart.BodyPartEntity;

**import** org.glassfish.jersey.media.multipart.ContentDisposition;

**import** org.glassfish.jersey.media.multipart.FormDataBodyPart;

**import** org.glassfish.jersey.media.multipart.FormDataContentDisposition;

**import** org.glassfish.jersey.media.multipart.FormDataMultiPart;

**import** org.glassfish.jersey.media.multipart.FormDataParam;

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

**import** com.ddlab.util.Constants;

**import** com.ddlab.util.FileUtil;

@Path("itc/upload")

**public** **class** ITCUploadService **extends** BaseResource {

//POST http://localhost:8090/springbootjersey2/api/itc/upload/image

//In case of postman client

//Choose form-data, file : c.jpg

@POST

@Path("/image")

@Consumes(MediaType.***MULTIPART\_FORM\_DATA***)

**public** Response uploadFile(FormDataMultiPart form) {

FormDataBodyPart filePart = form.getField("file");

ContentDisposition headerOfFilePart = filePart.getContentDisposition();

InputStream fileInputStream = filePart.getValueAs(InputStream.**class**);

String filePath = Constants.***SERVER\_UPLOAD\_LOCATION\_FOLDER*** + headerOfFilePart.getFileName();

// save the file to the server

FileUtil.*saveFile*(fileInputStream, filePath);

String output = "File saved to server location using FormDataMultiPart : " + filePath;

**try** {

fileInputStream.close();

} **catch** (IOException e) {

e.printStackTrace();

}

**return** Response.*status*(200).entity(output).build();

}

/\*

\* Another way to upload file

\* POST http://localhost:8090/springbootjersey2/api/itc/upload/image1

\* In case of postman client, choose form-data file : a.jpg

\*

\*/

@POST

@Path("/image1")

@Consumes(MediaType.***MULTIPART\_FORM\_DATA***)

**public** Response uploadFile(

@FormDataParam("file") InputStream fileInputStream,

@FormDataParam("file") FormDataContentDisposition contentDispositionHeader) {

String filePath = Constants.***SERVER\_UPLOAD\_LOCATION\_FOLDER***

+ contentDispositionHeader.getFileName();

// save the file to the server

String output = "";

**try** {

FileUtil.*saveFile*(fileInputStream, filePath);

output = "File saved to server location : " + filePath

+ " successfully";

} **catch** (Exception e) {

e.printStackTrace();

output = "Unexpected server exception while uploading the file.";

}

**return** Response.*status*(200).entity(output).build();

}

/\*

\* Posting image with data

\* POST http://localhost:8090/springbootjersey2/api/itc/upload/imageWithData

\* In case of postman client choose form-data

\* file: a.jpg

\* user: { "firstName" : "Deb", "lastName" : "Mishra","id" : 1 }

\*/

@POST

@Path("/imageWithData")

@Consumes(MediaType.***MULTIPART\_FORM\_DATA***)

**public** Response uploadFile(FormDataMultiPart form, @FormDataParam("user") String user) {

System.***out***.println("User-------->" + user.toString());

FormDataBodyPart filePart = form.getField("file");

ContentDisposition headerOfFilePart = filePart.getContentDisposition();

InputStream fileInputStream = filePart.getValueAs(InputStream.**class**);

String filePath = Constants.***SERVER\_UPLOAD\_LOCATION\_FOLDER***

+ headerOfFilePart.getFileName();

// save the file to the server

FileUtil.*saveFile*(fileInputStream, filePath);

String output = "File saved to server location using FormDataMultiPart : "

+ filePath;

**return** Response.*status*(200).entity(output).build();

}

/\*

\* Posting image with data

\* POST http://localhost:8090/springbootjersey2/api/itc/upload/imageWithData1

\* In case of postman client choose form-data

\* file: a.jpg

\* user: { "firstName" : "Deb", "lastName" : "Mishra","id" : 1 }

\*/

@POST

@Path("/imageWithData1")

@Consumes(MediaType.***MULTIPART\_FORM\_DATA***)

**public** Response uploadFileWithData(FormDataMultiPart form) {

Map<String, List<FormDataBodyPart>> params = form.getFields();

System.***out***.println(params);

String myJson = params.get("user").get(0).getValue();

System.***out***.println("myjson---------->" + myJson);

User user = User.*toUser*(myJson);

System.***out***.println("User-------->" + user.toXML());

FormDataBodyPart filePart = form.getField("file");

ContentDisposition headerOfFilePart = filePart.getContentDisposition();

InputStream fileInputStream = filePart.getValueAs(InputStream.**class**);

String filePath = Constants.***SERVER\_UPLOAD\_LOCATION\_FOLDER***

+ headerOfFilePart.getFileName();

// save the file to the server

FileUtil.*saveFile*(fileInputStream, filePath);

String output = "File saved to server location using FormDataMultiPart : "

+ filePath;

**try** {

fileInputStream.close();

} **catch** (IOException e) {

e.printStackTrace();

}

**return** Response.*status*(200).entity(output).build();

}

/\*

\* Posting multiple images

\* POST http://localhost:8090/springbootjersey2/api/itc/upload/multiupload

\* In case of postman client choose form-data

\* file1: a.jpg

\* file2: b.jpg

\* You can also use the html file called form.html

\*/

@POST

@Path("/multiupload")

@Consumes(MediaType.***MULTIPART\_FORM\_DATA***)

**public** Response uploadMultipleFile(FormDataMultiPart form) {

BodyPartEntity bodyPartEntity;

**for** (BodyPart part : form.getBodyParts()) {

bodyPartEntity = (BodyPartEntity) part.getEntity();

String fileName = part.getContentDisposition().getFileName();

System.***out***

.println("part.getContentDisposition().getFileName()---------->"

+ fileName);

InputStream fileInputStream = bodyPartEntity.getInputStream();

String filePath = Constants.***SERVER\_UPLOAD\_LOCATION\_FOLDER*** + fileName;

FileUtil.*saveFile*(fileInputStream, filePath);

}

**return** Response.*status*(200)

.entity("All Files uploaded successfully ... ").build();

}

}

## **ITCDownloadService.java**

**package** com.ddlab.web.resources;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.InputStream;

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

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

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

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

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

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

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

**import** com.ddlab.util.FileUtil;

@Path("itc/download")

**public** **class** ITCDownloadService **extends** BaseResource {

/\*

\* You can directly hit the url in the browser

\* GET http://localhost:8090/springbootjersey2/api/itc/download/image

\*/

@Path("/image")

@GET

@Produces("image/jpg")

**public** Response getFile() {

**byte**[] docStream = FileUtil.*readContents*("D:/temp/k.jpg");

**return** Response.*ok*(docStream, MediaType.***APPLICATION\_OCTET\_STREAM***)

.header("content-disposition", "attachment; filename = k.jpg").build();

}

/\*

\* You can directly hit the url in the browser

\* GET http://localhost:8090/springbootjersey2/api/itc/download/image1?fileName=k.jpg

\*/

@Path("/image1")

@GET

@Produces(MediaType.***APPLICATION\_OCTET\_STREAM***)

**public** Response getFile(@QueryParam("fileName") String fileName) {

// Do not provide path param

Response response = **null**;

String dirPath = "D:/temp";

String filePath = dirPath + File.***separator*** + fileName;

**try** {

**if** (!**new** File(filePath).exists())

**throw** **new** FileNotFoundException();

**byte**[] docStream = FileUtil.*readContents*(dirPath + File.***separator***

+ fileName);

String attachment = "attachment; filename = " + fileName;

response = Response

.*ok*(docStream, MediaType.***APPLICATION\_OCTET\_STREAM***)

.header("content-disposition", attachment).build();

} **catch** (FileNotFoundException e) {

response = Response.*serverError*().status(Status.***BAD\_REQUEST***)

.build();

}

**return** response;

}

/\* You can directly hit the url in the browser

\* GET http://localhost:8090/springbootjersey2/api/itc/download/showImage?fileName=k.jpg

\*/

@Path("/showImage")

@GET

@Produces("image/jpg")

**public** Response showImage(@QueryParam("fileName") String fileName) {

Response response = **null**;

String dirPath = "D:/temp";

InputStream inStream = **null**;

String filePath = dirPath + File.***separator*** + fileName;

**try** {

**if** (!**new** File(filePath).exists())

**throw** **new** FileNotFoundException();

inStream = **new** FileInputStream(filePath);

response = Response.*ok*().entity(inStream).build();

} **catch** (FileNotFoundException e) {

e.printStackTrace();

response = Response.*serverError*().status(Status.***BAD\_REQUEST***)

.build();

}

**return** response;

}

}

# **Service Layer**

## **IUserService.java**

**package** com.ddlab.services;

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

**public** **interface** IUserService {

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

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

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

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

}

## **UserServiceImpl.java**

**package** com.ddlab.services;

**import** org.springframework.stereotype.Component;

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

@Component(value="userService")

**public** **class** UserServiceImpl **implements** IUserService {

**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.setFirstName("Deb");

user.setLastName("Mishra");

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

**return** user;

}

}

# Util Layer

## **Constants.java**

**package** com.ddlab.util;

**public** **interface** Constants {

**public** **static** **final** String ***SERVER\_UPLOAD\_LOCATION\_FOLDER*** = "D:/temp/";

**public** **static** **final** String ***DATA\_LOCATION\_FOLDER*** = "D:/temp/data";

}

## **FileUtil.java**

**package** com.ddlab.util;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.OutputStream;

**public** **class** FileUtil {

// save uploaded file to a defined location on the server

**public** **static** **void** saveFile(InputStream uploadedInputStream, String serverLocation) {

OutputStream outpuStream = **null**;

**try** {

**int** read = 0;

**byte**[] bytes = **new** **byte**[1024];

outpuStream = **new** FileOutputStream(**new** File(serverLocation));

**while** ((read = uploadedInputStream.read(bytes)) != -1) {

outpuStream.write(bytes, 0, read);

}

} **catch** (IOException e) {

e.printStackTrace();

} **finally** {

**try** {

outpuStream.flush();

outpuStream.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

**public** **static** **byte**[] readContents(String filePath) {

File file = **new** File(filePath);

**byte**[] buffer = **new** **byte**[(**int**) file.length()];

InputStream inStream = **null**;

**try** {

inStream = **new** FileInputStream(file);

inStream.read(buffer);

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

} **finally** {

**try** {

**if** (inStream != **null**)

inStream.close();

} **catch** (Exception e2) {

}

}

**return** buffer;

}

}

## **Entity Layer**

**package** com.ddlab.entity;

**import** java.io.IOException;

**import** java.io.StringWriter;

**import** javax.xml.bind.JAXBContext;

**import** javax.xml.bind.JAXBException;

**import** javax.xml.bind.Marshaller;

**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.JsonParseException;

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

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

**import** org.codehaus.jackson.map.JsonMappingException;

**import** org.codehaus.jackson.map.ObjectMapper;

@XmlRootElement(name = "User")

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

@XmlType(propOrder = { "firstName", "lastName", "id" })

@JsonPropertyOrder(value = { "firstName", "lastName", "id" })

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

@XmlElement(name = "firstName")

@JsonProperty("firstName")

**private** String firstName;

@XmlElement(name = "lastName")

@JsonProperty("lastName")

**private** String lastName;

@XmlElement(name = "id")

@JsonProperty("id")

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

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

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

**return** id;

}

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

**this**.id = id;

}

@Override

**public** String toString() {

**return** "User [firstName=" + firstName + ", lastName=" + lastName + ", id=" + id + "]";

}

**public** String toJSON() {

ObjectMapper mapper = **new** ObjectMapper();

String toJson = **null**;

**try** {

toJson = mapper.writerWithDefaultPrettyPrinter().writeValueAsString(**this**);

} **catch** (IOException e) {

e.printStackTrace();

}

**return** toJson;

}

**public** String toXML() {

String xmlString = "";

**try** {

JAXBContext context = JAXBContext.*newInstance*(**this**.getClass());

Marshaller m = context.createMarshaller();

m.setProperty(Marshaller.***JAXB\_FORMATTED\_OUTPUT***, Boolean.***TRUE***);

StringWriter sw = **new** StringWriter();

m.marshal(**this**, sw);

xmlString = sw.toString();

} **catch** (JAXBException e) {

e.printStackTrace();

}

**return** xmlString;

}

**public** **static** User toUser(String jsonString) {

ObjectMapper mapper = **new** ObjectMapper();

User user = **null**;

**try** {

user = (User) mapper.readValue(jsonString, User.**class**);

} **catch** (JsonParseException e) {

e.printStackTrace();

} **catch** (JsonMappingException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

}

**return** user;

}

}

You can the entire application from Eclipse.

You can also run using maven command given below.

Mvn clean package

Cd /target

Java -jar springbootjersey2-0.0.1-SNAPSHOT.jar

You can also externalize the spring boot configuration file. The properties file is given below.

# **Xternalapp.properties**

server.contextPath=/jerseyboot/api

server.port=8090

#logging.path=E:/springboot-2017/springbootjersey2/logs

#logging.path does not work, hence give the full path with file name

logging.file=E:/springboot-2017/springbootjersey2/logs/springbootjersy2.log

You have to use the following option

**Java -jar <springbootApplication>.jar –spring.config.location=<actual path of file>**

use below the command to run as externalized properties file.

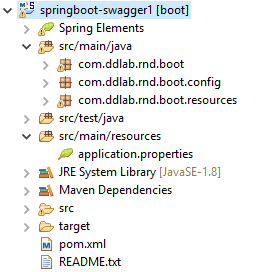
java -jar springbootjersey2-0.0.1-SNAPSHOT.jar **--spring.config.location**=E:/springboot-2017/springbootjersey2/xternalapp.properties

You can also run like this but make sure the properties file in the same directory where the springapplication.java file resides.

java -jar springbootjersey2-0.0.1-SNAPSHOT.jar **--spring.config.location=./xternalapp.properties**

Spring Boot with Rest and Swagger (No Jersey)

# **Project 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/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>springboot-swagger1</groupId>

<artifactId>springboot-swagger1</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>springboot-swagger1</name>

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

**<parent>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-parent</artifactId>**

**<version>1.4.2.RELEASE</version>**

**</parent>**

<properties>

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

</properties>

<dependencies>

**<dependency>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-web</artifactId>**

**</dependency>**

**<dependency>**

**<groupId>io.springfox</groupId>**

**<artifactId>springfox-swagger-ui</artifactId>**

**<version>2.6.1</version>**

**<scope>compile</scope>**

**</dependency>**

**<dependency>**

**<groupId>io.springfox</groupId>**

**<artifactId>springfox-swagger2</artifactId>**

**<version>2.6.1</version>**

**<scope>compile</scope>**

**</dependency>**

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

<plugin>

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

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

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

# **Application Configuration (src/main/resources/application.properties)**

server.contextPath=/springbootswagger1

server.port=8090

# **Java Code**

## **SpringBootApp.java**

**package** com.ddlab.rnd.boot;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** SpringBootApp {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringBootApp.**class**, args);

}

}

## **SwaggerConfig.java**

**package** com.ddlab.rnd.boot.config;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** springfox.documentation.builders.PathSelectors;

**import** springfox.documentation.builders.RequestHandlerSelectors;

**import** springfox.documentation.service.ApiInfo;

**import** springfox.documentation.service.Contact;

**import** springfox.documentation.spi.DocumentationType;

**import** springfox.documentation.spring.web.plugins.Docket;

**import** springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration

@EnableSwagger2

**public** **class** SwaggerConfig {

@Bean //best way to use

**public Docket resourcesApi() {**

**return new Docket(DocumentationType.*SWAGGER\_2*)**

**.select()**

**.apis(RequestHandlerSelectors.*any*())**

**.paths(PathSelectors.*any*())**

**.build()**

**.apiInfo(metaData());**

**}**

**private** ApiInfo metaData() {

ApiInfo apiInfo = **new** ApiInfo(

"Spring Boot REST API",

"Spring Boot REST API for Online Store",

"1.0",

"Terms of service",

**new** Contact("Debadatta Mishra", "https://springframework.org", "deba@ddlab.com"),

"Apache License Version 2.0",

"https://www.apache.org/licenses/LICENSE-2.0");

**return** apiInfo;

}

}

## **ITCResources.java**

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

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RestController;

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

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

@RestController

@RequestMapping("/itc")

@Api(value="onlinestore", ~~description~~="Operations pertaining to products in Online Store")

**public** **class** ITCResources {

**@ApiOperation(value = "Delete a product")**

@RequestMapping(value="/delete/{id}", method = RequestMethod.***DELETE***, produces = "application/json")

**public** ResponseEntity delete(@PathVariable **int** id){

System.***out***.println("Deleting some ids");

**return** **new** ResponseEntity("Product deleted successfully", HttpStatus.***OK***);

}

}

# **How to Access**

To see only json data

<http://localhost:8090/springbootswagger1/v2/api-docs>

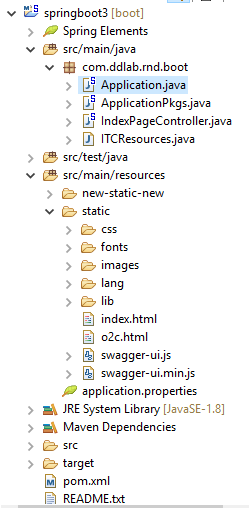
To see the Swagger UI

<http://localhost:8090/springbootswagger1/swagger-ui.html>

Spring Boot with Jersy and Swagger

Springfox does not support Jersey framework for swagger documentation.

# **Project 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/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>springboot3</groupId>

<artifactId>springboot3</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>springboot3</name>

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

**<parent>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-parent</artifactId>**

**<!-- <version>1.2.7.RELEASE</version> -->**

**<version>1.4.1.RELEASE</version>**

**</parent>**

<properties>

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

</properties>

<dependencies>

**<dependency>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-web</artifactId>**

**</dependency>**

<!-- Spring Jersey Integration -->

**<dependency>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-jersey</artifactId>**

**</dependency>**

<!-- Jersey Specific -->

**<dependency> <!-- Very important -->**

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

**<artifactId>jersey-server</artifactId>**

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

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

<version>2.14</version>

</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>springboot3</finalName>

<plugins>

<plugin>

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

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

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

# **Spring Boot Configuration (src.main/resources/application.properties)**

server.contextPath=/springboot3

server.port=8090

spring.mvc.view.prefix: /

spring.mvc.view.suffix: .html

# **Java Classes**

## **Application.java**

**package** com.ddlab.rnd.boot;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** Application {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(Application.**class**, args);

}

}

## **ApplicationPkgs.java**

**package** com.ddlab.rnd.boot;

**import** java.util.HashSet;

**import** java.util.Set;

**import** javax.ws.rs.ApplicationPath;

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

**import** org.glassfish.jersey.media.multipart.MultiPartFeature;

**import** org.glassfish.jersey.server.ResourceConfig;

**import** org.springframework.stereotype.Component;

**import** io.swagger.jaxrs.config.BeanConfig;

//To know about WADl : GET http://localhost:8090/springboot3/myitc/application.wadl

@Component // ~~ This is mandatory

@ApplicationPath("myitc") // ~~ Very very important

**public** **class** ApplicationPkgs **extends** ResourceConfig {

**public** ApplicationPkgs() {

// super(

// ITCResources.class,ITCUploadService.class,ITCDownloadService.class,JacksonFeature.class,MultiPartFeature.class);

// OR you can write below

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

classes.add(ITCResources.**class**);

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

classes.add(MultiPartFeature.**class**);

//For swagger docs

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

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

swaggerConfiguration();

registerClasses(classes);

}

**private** **void** swaggerConfiguration() {

BeanConfig beanConfig = **new** BeanConfig();

// beanConfig.setHost("localhost:8090");

beanConfig.setBasePath("springboot3/myitc");

beanConfig.setResourcePackage("com.ddlab.rnd.boot");

beanConfig.setScan(**true**);

}

}

## **IndexPageController.java**

**package** com.ddlab.rnd.boot;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.RequestMapping;

@Controller

**public** **class** IndexPageController {

@RequestMapping("/")

**public** String login() {

**return** "index";

}

}

## **ITCResources.java**

**package** com.ddlab.rnd.boot;

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

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

**import** javax.ws.rs.PathParam;

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

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

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

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

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

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

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

**import** io.swagger.annotations.Contact;

**import** io.swagger.annotations.ExternalDocs;

**import** io.swagger.annotations.Info;

**import** io.swagger.annotations.License;

**import** io.swagger.annotations.SwaggerDefinition;

**import** io.swagger.annotations.Tag;

@Path("/itc")

@Api(value="/myitc/itc",tags={"BasicService: This is the basic service.\n It is useful now."})

@SwaggerDefinition(

info = @Info( description = "A basic service for testing", version = "v1",

title = "Basic service api", termsOfService = "http://github.com/debjava",

contact = @Contact(name = "Debadatta Mishra", email = "deba.java@gmail.com",

url = "http://github.com/debjava"

),

license=@License(name = "Apache 2.0", url = "http://www.apache.org/licenses"),

consumes = {"text/plain,","application/json", "application/xml"},

produces = {"text/plain,","application/json", "application/xml"},

schemes = {SwaggerDefinition.Scheme.***HTTP***, SwaggerDefinition.Scheme.***HTTPS***},

tags = {

@Tag(name = "Private", description = "Tag used to denote operations as private")

},

externalDocs = @ExternalDocs(value = "No external doc", url = "http://nodoc.html")

)

**public** **class** ITCResources {

@GET

**public** String message() {

**return** "Hello " + "Piku, you got spring with Jersey";

}

@Path("/user/{id}")

@GET

@Produces({ MediaType.***TEXT\_PLAIN*** })

@ApiOperation(value = "get user details by id as path param", notes = "get the user details from id as path param", code = 200)

@ApiResponses(value={

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

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

})

**public** Response getUserById(@PathParam("id") **int** id) {

**if**( id == 0 ) **throw** **new** IllegalArgumentException("Id cannot be 0, enter a valid id");

System.***out***.println("Got the user response");

**return** Response.*ok*().entity("Got the user response").build();

}

}

# **Changes in Index.html of swagger doc**

<script type=*"text/javascript"*>

$(**function** () {

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

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

url = decodeURIComponent(url[1]);

} **else** {

/\* url = "/springboot3/myitc/swagger.json"; \*/

url = "/springboot3/myitc/swagger.json";

}

# **URLS to Use**

<http://localhost:8090/springboot3/myitc/application.wadl>

For Swagger UI : <http://localhost:8090/springboot3>

Only Swagger json : <http://localhost:8090/springboot3/myitc/swagger.json>