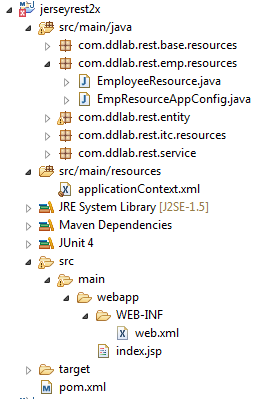
Restful Web service using Jersey 2.14 and Spring

# Project Structure



# Maven Config (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>jerseyrest2x</groupId>

<artifactId>jerseyrest2x</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>jerseyrest2x Maven Webapp</name>

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

<dependencies>

<!-- servlet 3.0 -->

<dependency>

<groupId>javax.servlet</groupId>

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

<version>3.0.1</version>

<scope>provided</scope>

</dependency>

<!-- jax-rs 2.0 -->

<dependency>

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

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

<version>2.0</version>

</dependency>

<!-- jersey implementation library -->

<dependency>

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

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

<!-- <version>2.11</version> -->

<version>2.14</version>

</dependency>

<!-- jersey media multipart for file upload -->

<dependency>

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

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

<!-- <version>2.11</version> -->

<version>2.14</version>

</dependency>

<!-- Spring Jersey Integration -->

<dependency>

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

<artifactId>jersey-spring3</artifactId>

<version>2.14</version>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.2</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>

<!-- The following is required for object to json -->

<dependency>

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

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

<version>2.14</version>

</dependency>

</dependencies>

<build>

<finalName>jerseyrest2x</finalName>

<plugins>

<plugin>

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

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

<version>3.0</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

</configuration>

</plugin>

<plugin>

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

<artifactId>maven-war-plugin</artifactId>

<version>2.4</version>

<configuration>

<failOnMissingWebXml>false</failOnMissingWebXml>

</configuration>

</plugin>

</plugins>

</build>

</project>

# WEB.xml

<web-app xmlns=*"http://java.sun.com/xml/ns/javaee"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"*

version=*"3.0"*>

<display-name>JAX-RS 2 File Upload Example Web Application</display-name>

<listener>

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

</listener>

<context-param>

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

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

</context-param>

</web-app>

# Spring IOC Configuration (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

## BaseResource.java

package com.ddlab.rest.base.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());

}

}

## EmployeeResource.java

package com.ddlab.rest.emp.resources;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.InputStream;

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.base.resources.BaseResource;

import com.ddlab.rest.entity.User;

import com.ddlab.rest.service.EmployeeService;

@Path("1/empresource")//Vesrion/Resources

public class EmployeeResource 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 })

public Response createUser(User user) {

//DELETE http://localhost:8080/jerseyrest2x/empresources/1/empresource/create

//Content-Type:application/json

//Payload : {"username":"Deba","password":"deba","id":"123"}

ResponseBuilder responseBuilder = null;

try {

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

responseBuilder = Response.status(Status.BAD\_REQUEST).entity("Incorrect password");

empService.createUser(user);

responseBuilder = Response.status(Status.CREATED).entity("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 })

public void updateUser(User user) { //204 No content

//DELETE http://localhost:8080/jerseyrest2x/empresources/1/empresource/update

//Content-Type:application/json

//Payload : {"username":"Deba","password":"deba","id":"123"}

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

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

empService.updateUser(user);

}

@Path("/delete")

@DELETE

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

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

public String deleteUser(User user) {

//DELETE http://localhost:8080/jerseyrest2x/empresources/1/empresource/delete

//Content-Type:application/json

//Payload : {"username":"Deba","password":"deba","id":"123"}

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

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

empService.deleteUser(user);

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

}

@Path("/userid")

@GET

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

public User getUser(@QueryParam("id") int id) {

//GET http://localhost:8080/jerseyrest2x/empresources/1/empresource/userid?id=123

//Accept:application/json or application/xml

if(id == 0 )

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

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

}

@Path("/show/image")

@GET

@Produces("image/png")

public Response showImage() {

//GET http://localhost:8080/jerseyrest2x/empresources/1/empresource/show/image

//You can directly hit in the browser

InputStream inStream = null;

try {

String SERVER\_DOWNLOAD\_LOCATION\_FOLDER = "D:"+File.separator+"temp"+File.separator;

String IMG\_FILE\_NAME = "r.jpg";

inStream = new FileInputStream(SERVER\_DOWNLOAD\_LOCATION\_FOLDER+IMG\_FILE\_NAME);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

return Response.ok().entity(inStream).build();

}

}

## EmpResourceAppConfig.java

package com.ddlab.rest.emp.resources;

import java.util.HashSet;

import java.util.Set;

import javax.ws.rs.ApplicationPath;

import javax.ws.rs.core.Application;

import org.glassfish.jersey.jackson.JacksonFeature;

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

//@ApplicationPath("/\*")

@ApplicationPath("empresources")

public class EmpResourceAppConfig extends Application {

//There can be multiple Classes which extend javax.ws.rs.core.Application

//This is used for logical functional separation

@Override

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

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

resources.add(EmployeeResource.class);

resources.add(MultiPartFeature.class);

resources.add(JacksonFeature.class);

return resources;

}

}

## ITCInfotechServices.java

package com.ddlab.rest.itc.resources;

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;

import javax.servlet.ServletConfig;

import javax.servlet.http.HttpServletRequest;

import javax.ws.rs.Consumes;

import javax.ws.rs.DELETE;

import javax.ws.rs.FormParam;

import javax.ws.rs.GET;

import javax.ws.rs.MatrixParam;

import javax.ws.rs.POST;

import javax.ws.rs.PUT;

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.Context;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

import org.codehaus.jackson.JsonParseException;

import org.codehaus.jackson.map.JsonMappingException;

import org.codehaus.jackson.map.ObjectMapper;

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.rest.entity.Constants;

import com.ddlab.rest.entity.Employee;

import com.ddlab.rest.entity.FileUtil;

@Path("itc")

public class ITCInfotechServices {

@Context

HttpServletRequest request;

@Context

ServletConfig servletConfig;

@GET

@Produces(MediaType.TEXT\_PLAIN)

public String getOrganisationName() {

//GET http://localhost:8080/jerseyrest2x/itcresources/itc

return "ITC Infotech, Bangalore, Karnataka";

}

@Path("/address")

@GET

@Produces(MediaType.TEXT\_PLAIN)

public String getAddress() {

// URL : http://localhost:8080/jerseyrest2x/itcresources/itc/address

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

}

// ~~~~~~~~~~~~~~~~~~~~~~~ @PathParam ~~~~~~~~~~~~~~~~~~~~~~~~~~~~

@Path("/address/{areaCode}")

@GET

@Produces(MediaType.TEXT\_PLAIN)

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

// GET http://localhost:8080/jerseyrest2x/itcresources/itc/address/USA

try {

if (areaCode.equalsIgnoreCase("USA"))

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

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

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

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

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

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

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

else

return "No such area code exists for ITC";

} catch (Exception e) {

return "No such area code exists for ITC";

}

}

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

@Path("/regionaladdress/{areaCode}")

// USA,EUROPE,ASIA

@GET

@Produces(MediaType.TEXT\_PLAIN)

public String getAddressByCountry(@PathParam("areaCode") String areaCode,

@QueryParam("country") String country) {

// GET http://localhost:8080/jerseyrest2x/itcresources/itc/regionaladdress/Europe?country=FI

try {

if (areaCode.equalsIgnoreCase("USA")

&& country.equalsIgnoreCase("NJ"))

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

else if (areaCode.equalsIgnoreCase("Europe")

&& country.equalsIgnoreCase("FI"))

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

else if (areaCode.equalsIgnoreCase("Europe")

&& country.equalsIgnoreCase("SE"))

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

else if (areaCode.equalsIgnoreCase("Europe")

&& country.equalsIgnoreCase("DK"))

return "Havnegade 39, 3. sal,1058 Copenhagen K";

else if (areaCode.equalsIgnoreCase("Asia")

&& country.equalsIgnoreCase("IN"))

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

else

return "No such area code exists for ITC";

} catch (Exception e) {

return "No such area code exists for ITC";

}

}

@Path("/itcaddress")

@GET

@Produces(MediaType.TEXT\_PLAIN)

public String getITCAddress(@MatrixParam("country") String country,

@MatrixParam("areacode") String areaCode) {

// GET http://localhost:8080/jerseyrest2x/itcresources/itc/itcaddress;country=FI;areacode=europe

return getAddressByCountry(areaCode, country);

}

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

@Path("/postaddress")

@POST

@Produces(MediaType.TEXT\_PLAIN)

public String postNGetITCAddress(@FormParam("country") String country,

@FormParam("areacode") String areaCode) {

// http://localhost:8080/jerseyrest2x/itcresources/itc/postaddress

return getAddressByCountry(areaCode, country);

}

/\*

\* In case of Firefox Rest Client

Set in the header

“name” = “Content-Type” and “value” = “application/x-www-form-urlencoded”

Then set the header as

country - FI

areacode - europe

In case of Chrome Postman

Click on x-www-form-urlencoded in Postman

country - FI

areacode - europe

\*/

// ~~~~~~~~~~~~~~~~~~~~~~~ @PathParam with Object ~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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

@GET

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

public Response getEmployee(@PathParam("id") String id) {

//GET http://localhost:8080/jerseyrest2x/itcresources/itc/emp/123

ObjectMapper mapper = new ObjectMapper();

Employee emp = null;

try {

File file = new File(Constants.DATA\_LOCATION\_FOLDER+File.separator+id+".json");

if( ! file.exists() )

return Response.status(404).entity("No information found for the given employee id ...").build();

emp = mapper.readValue(file, Employee.class);

} catch (JsonParseException e) {

e.printStackTrace();

} catch (JsonMappingException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return Response.ok(emp).build();

}

/\*\*

\* POST http://localhost:8080/jerseyrest2x/itcresources/itc/createemp

\*

\* JSON Structure

\* {

"Name": "Deba",

"Age": 23,

"Email": "deba@gmail.com",

"Adrs": {

"DoorNo": "12-A",

"Street": "Street-11",

"City": "Bangalore",

"Country": "Karnataka"

}

}

XML Structure

<Emp>

<Name>Deba</Name>

<Age>23</Age>

<Email>deba@gmail.com</Email>

<Adrs>

<DoorNo>12-A</DoorNo>

<Street>Street-11</Street>

<City>Bangalore</City>

<Country>Karnataka</Country>

</Adrs>

</Emp>

\*/

@Path("/createemp")

@POST

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

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

public Response createEmp(Employee emp) {

System.out.println("Employee object received successfully .........");

//store into file system

if( new File(Constants.DATA\_LOCATION\_FOLDER+File.separator+emp.getName()+".json").exists()) {

return Response.status(409).entity("Employee already exists in the system ...").build();

}

FileUtil.saveFile(Constants.DATA\_LOCATION\_FOLDER+File.separator+emp.getName()+".json", emp.toJSON());

return Response.ok("Employee created successfully...").build();

}

/\*\*

\* PUT http://localhost:8080/jerseyrest2x/itcresources/itc/updateemp

\*

\* JSON Structure

\* {

"Name": "Deba",

"Age": 23,

"Email": "deba@gmail.com",

"Adrs": {

"DoorNo": "12-A",

"Street": "Street-11",

"City": "Bangalore",

"Country": "Karnataka"

}

}

XML Structure

<Emp>

<Name>Deba</Name>

<Age>23</Age>

<Email>deba@gmail.com</Email>

<Adrs>

<DoorNo>12-A</DoorNo>

<Street>Street-11</Street>

<City>Bangalore</City>

<Country>Karnataka</Country>

</Adrs>

</Emp>

\*/

@Path("/updateemp")

@PUT

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

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

public Response updateEmp(Employee emp) {

System.out.println("Employee object received successfully .........");

//store into file system

if( new File(Constants.DATA\_LOCATION\_FOLDER+File.separator+emp.getName()+".json").exists()) {

FileUtil.saveFile(Constants.DATA\_LOCATION\_FOLDER+File.separator+emp.getName()+".json", emp.toJSON());

return Response.ok("Employee info updated successfully...").build();

}

else

return Response.status(404).entity("Employee information not found, hence can not be updated ...").build();

}

/\*\*

\* PUT http://localhost:8080/jerseyrest2x/itcresources/itc/deleteemp

\*

\* JSON Structure

\* {

"Name": "Deba",

"Age": 23,

"Email": "deba@gmail.com",

"Adrs": {

"DoorNo": "12-A",

"Street": "Street-11",

"City": "Bangalore",

"Country": "Karnataka"

}

}

XML Structure

<Emp>

<Name>Deba</Name>

<Age>23</Age>

<Email>deba@gmail.com</Email>

<Adrs>

<DoorNo>12-A</DoorNo>

<Street>Street-11</Street>

<City>Bangalore</City>

<Country>Karnataka</Country>

</Adrs>

</Emp>

\*/

@Path("/deleteemp")

@DELETE

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

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

public Response deleteEmp(Employee emp) {

System.out.println("Employee object received successfully .........");

//store into file system

File file = new File(Constants.DATA\_LOCATION\_FOLDER+File.separator+emp.getName()+".json");

if( file.exists()) {

file.delete();

return Response.ok("Employee info deleted successfully...").build();

}

else

return Response.status(404).entity("Employee information not found, hence can not be deleted ...").build();

}

@POST

@Path("/upload")

@Consumes(MediaType.MULTIPART\_FORM\_DATA)

public Response uploadFile(FormDataMultiPart form) {

//POST http://localhost:8080/jerseyrest2x/itcresources/itc/upload

//Content-Type:multipart/form-data

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

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();

}

@POST

@Path("/upload1")

@Consumes(MediaType.MULTIPART\_FORM\_DATA)

public Response uploadFile(

@FormDataParam("file") InputStream fileInputStream,

@FormDataParam("file") FormDataContentDisposition contentDispositionHeader) {

//POST http://localhost:8080/jerseyrest2x/itcresources/itc/upload1

//Content-Type:multipart/form-data

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

// save the file to the server

String output = "";

try {

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();

}

@POST

@Path("/upload3")

@Consumes({MediaType.MULTIPART\_FORM\_DATA})

public Response uploadFileWithData1(

@FormDataParam("file") InputStream fileInputStream,

@FormDataParam("file") FormDataContentDisposition cdh,

@FormDataParam("emp") String emp) throws Exception {

//http://localhost:8080/jerseyrest2x/itcresources/itc/upload3

//Content-Type:multipart/form-data

//form data as emp:abcd

try {

System.out.println("Emp data as String :::"+emp);

//System.out.println("--->"+URLDecoder.decode(emp,"UTF-8"));

} catch (Exception e) {

e.printStackTrace();

}

String output = "";

try {

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

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();

}

@Path("/download")

@GET

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

@Produces("image/jpg")

public Response getFile() {

//GET http://localhost:8080/jerseyrest2x/itcresources/itc/download

//You can invoke this url directly in the browser

byte[] docStream = readContents("D:/temp/d.doc");

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

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

.build();

}

private 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;

}

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

private void saveFile(InputStream uploadedInputStream, String serverLocation) {

try {

OutputStream outpuStream = new FileOutputStream(new File(

serverLocation));

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);

}

outpuStream.flush();

outpuStream.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

## ITCAppConfig.java

package com.ddlab.rest.itc.resources;

import java.util.HashSet;

import java.util.Set;

import javax.ws.rs.ApplicationPath;

import javax.ws.rs.core.Application;

import org.glassfish.jersey.jackson.JacksonFeature;

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

@ApplicationPath("itcresources") //It becomes optional if you provide <url-pattern>/\*</url-pattern> in web.xml

public class ITCAppConfig extends Application {

@Override

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

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

resources.add(ITCInfotechServices.class);

resources.add(MultiPartFeature.class);

resources.add(JacksonFeature.class);

return resources;

}

}

## Service Layer

## 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;

}

}

## Entity Layer

## Address.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** org.codehaus.jackson.annotate.JsonProperty;

@XmlRootElement(name = "Adrs")

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

**public** **class** Address {

@XmlElement(name = "DoorNo")

@JsonProperty("DoorNo")

**private** String doorNo;

@XmlElement(name = "Street")

@JsonProperty("Street")

**private** String streetId;

@XmlElement(name = "City")

@JsonProperty("City")

**private** String city;

@XmlElement(name = "Country")

@JsonProperty("Country")

**private** String country;

**public** String getDoorNo() {

**return** doorNo;

}

**public** **void** setDoorNo(String doorNo) {

**this**.doorNo = doorNo;

}

**public** String getStreetId() {

**return** streetId;

}

**public** **void** setStreetId(String streetId) {

**this**.streetId = streetId;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

**public** String getCountry() {

**return** country;

}

**public** **void** setCountry(String country) {

**this**.country = country;

}

}

## Constants.java

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

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

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

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

}

## Employee.java

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

**import** java.io.IOException;

**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** org.codehaus.jackson.JsonGenerationException;

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

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

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

@XmlRootElement(name = "Emp")

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

**public** **class** Employee {

@XmlElement(name = "Name")

@JsonProperty("Name")

**private** String name;

@XmlElement(name = "Age")

@JsonProperty("Age")

**private** **int** age;

@XmlElement(name = "Email")

@JsonProperty("Email")

**private** String emailId;

@XmlElement(name = "Adrs")

@JsonProperty("Adrs")

**private** Address adrs;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

**public** String getEmailId() {

**return** emailId;

}

**public** **void** setEmailId(String emailId) {

**this**.emailId = emailId;

}

**public** Address getAdrs() {

**return** adrs;

}

**public** **void** setAdrs(Address adrs) {

**this**.adrs = adrs;

}

**public** String toJSON() {

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

**try** {

**return** mapper.writeValueAsString(**this**);

} **catch** (JsonGenerationException e) {

e.printStackTrace();

} **catch** (JsonMappingException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

}

**return** **null**;

}

}

## FileUtil.java

package com.ddlab.rest.entity;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.OutputStream;

public class FileUtil {

public static void saveFile( String filePath , String contents ) {

OutputStream out = null;

try {

out = new FileOutputStream(filePath);

out.write(contents.getBytes());

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

finally {

if( out != null ) {

try {

out.flush();

out.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

}

## 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;

@XmlRootElement(name = "User")

@XmlAccessorType(XmlAccessType.FIELD)

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

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

public class User {

@XmlElement(name = "username")

@JsonProperty("username")

private String userName;

@XmlElement(name = "password")

@JsonProperty("password")

private String password;

@XmlElement(name = "id")

@JsonProperty("id")

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;

}

}