RESTful web service with Spring and Jersey

# Maven(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</groupId>

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

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>jersey-spring-rest 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>

</dependency>

**<!-- Jersey Spring Integration -->**

**<dependency>**

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

**<artifactId>jersey-spring3</artifactId>**

**<version>2.14</version>**

**</dependency>**

<!-- Spring Jar files -->

<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>

</dependencies>

<build>

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

</build>

</project>

# Spring 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.LoginServiceImpl"* />**

</beans>

# 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</display-name>

**<context-param>**

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

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

**</context-param>**

<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.rest.resources.ApplicationPkgs</param-value>**

**</init-param>**

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

**</servlet>**

<servlet-mapping>

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

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

</servlet-mapping>

<welcome-file-list>

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

</welcome-file-list>

</web-app>

# Index.jsp

<html>

<body>

<h2>Hello World!</h2>

</body>

</html>

# Java Code

## ApplicationPkgs.java

package com.ddlab.rest.resources;

import org.glassfish.jersey.jackson.JacksonFeature;

import org.glassfish.jersey.server.ResourceConfig;

public class ApplicationPkgs extends ResourceConfig {

**public ApplicationPkgs() {**

**super(LoginServiceResource.class,JacksonFeature.class);**

**}**

}

## BaseResource.java

package com.ddlab.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());**

}

}

## LoginServiceResource.java

package com.ddlab.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.User;

import com.ddlab.rest.service.LoginService;

@Path("1/loginservice")

public class LoginServiceResource extends BaseResource {

@Context

HttpServletRequest request;

@Context

ServletConfig servletConfig;

@Autowired

private LoginService loginService;

@Path("/create")

@POST

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

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

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");

loginService.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

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

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

loginService.updateUser(user);

}

@Path("/delete")

@DELETE

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

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

public String deleteUser(User user) { //204 No content

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

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

loginService.deleteUser(user);

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 })

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

if(id == 0 )

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

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

}

}

## LoginService.java

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

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

**public** **interface** LoginService {

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

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

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

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

}

## LoginServiceImpl.java

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

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

**public** **class** LoginServiceImpl **implements** LoginService {

**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;

@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;

}

}

## Project Structure

