REST WEB SERVICE BY USING SPRING MVC FRAMEWORK

|  |
| --- |
| POM dependency for JSON Response NO need to do for Xml its present under JRE rt.jar |
| <!-- Jackson JSON -->  <dependency>  <groupId>com.fasterxml.jackson.core</groupId>  <artifactId>jackson-databind</artifactId>  <version>2.8.5</version>  </dependency> |

|  |  |
| --- | --- |
| **public** **class** UserNotFoundException **extends** Exception {  **private** **static** **final** **long** ***serialVersionUID*** = 1L;  **private** String errorMessage;  **public** UserNotFoundException() {  **super**();  }  **public** UserNotFoundException(String errorMessage) {  **super**(errorMessage);  **this**.errorMessage = errorMessage;  }  **public** String getErrorMessage() {  **return** errorMessage;  }  } | **public** **class** UserErrorResponse {  **private** **int** errorCode;  **private** String errorMessage;  **public** **int** getErrorCode() {  **return** errorCode;  }  **public** **void** setErrorCode(**int** errorCode) {  **this**.errorCode = errorCode;  }  **public** String getErrorMessage() {  **return** errorMessage;  }  **public** **void** setErrorMessage(String errorMessage) {  **this**.errorMessage = errorMessage;  }  } |

@ExceptionHandler

@ExceptionHandler works at the **Controller level** and it is only active for that **particular Controller,**not globally for the entire application.

|  |
| --- |
| @Controller //Exception Handling  @RequestMapping("/user/controllerExceptionHandler")  **public** **class** RestServiceControllerExceptionHandling {  @RequestMapping(value = "/getSpecificUser/{id}", method = RequestMethod.***GET***)  **public** @ResponseBody ResponseEntity<User> getUserForId(@PathVariable("id") **int** id) **throws** UserNotFoundException {  User user = **new** User();  user.setId(id);  user.setName("John");  user.setAge(45);  **if** (id != 1) {  **throw** **new** UserNotFoundException("User not found");  }  **return** **new** ResponseEntity<User>(user, HttpStatus.***OK***);  }  @ExceptionHandler(UserNotFoundException.**class**)  **public** ResponseEntity<UserErrorResponse> handleUserNotFoundException(Exception ex) {  UserErrorResponse errorResponse = **new** UserErrorResponse();  errorResponse.setErrorCode(HttpStatus.***PRECONDITION\_FAILED***.value());  errorResponse.setErrorMessage(ex.getMessage());  **return** **new** ResponseEntity<UserErrorResponse>(errorResponse, HttpStatus.***OK***);  }  } |

HandlerExceptionResolver

This will resolve any exception thrown by the application. It is used to resolve standard Spring exceptions to their **corresponding HTTP Status Codes**. It does not have control over the body of the response, means **it does not set anything to the body of the Response**.It does map the status code on the response but the **body is null**.

@ControllerAdvice

@ControllerAdvice used for global error handling in the Spring MVC application.It also has full control over the body of the response and the status code.

|  |
| --- |
| @Controller  @RequestMapping("/user/globalExceptionHandler")  **class** RestServiceGlobalExceptionHandling {  @RequestMapping(value = "/getSpecificUser/{id}", method = RequestMethod.***GET***)  **public** @ResponseBody ResponseEntity<User> getUserForId(@PathVariable("id") **int** id) **throws** Exception {  User user = **new** User();  user.setId(id);  user.setName("John");  user.setAge(45);  **if** (id != 1) {  **throw** **new** UserNotFoundException("User not found");  }  // Intentionally throwing exception  **int** ageByZero = user.getAge() / 0;  user.setAge(ageByZero);  **return** **new** ResponseEntity<User>(user, HttpStatus.***OK***);  }  } |

|  |
| --- |
| @ControllerAdvice  **public** **class** UserExceptionHandler {    @ExceptionHandler(UserNotFoundException.**class**)  **public** ResponseEntity<UserErrorResponse> handleUserNotFoundException(Exception ex) {  UserErrorResponse errorResponse = **new** UserErrorResponse();  errorResponse.setErrorCode(HttpStatus.***PRECONDITION\_FAILED***.value());  errorResponse.setErrorMessage(ex.getMessage());  **return** **new** ResponseEntity<UserErrorResponse>(errorResponse, HttpStatus.***OK***);  }    @ExceptionHandler(Exception.**class**)  **public** ResponseEntity<UserErrorResponse> handleGenericException(Exception ex) {  System.***out***.println("Runtime exception");  UserErrorResponse errorResponse = **new** UserErrorResponse();  errorResponse.setErrorCode(HttpStatus.***INTERNAL\_SERVER\_ERROR***.value());  errorResponse.setErrorMessage("There is some techncal issue");  **return** **new** ResponseEntity<UserErrorResponse>(errorResponse, HttpStatus.***OK***);  }  } |

|  |
| --- |
| //fully tested =========================Without Exception Handling== JAXB and JSON Format Result===========  @Controller  @RequestMapping("/user")  **public** **class** RestController {  @RequestMapping(value = "/getSpecificUser/{id}", method = RequestMethod.***GET***)  **public** @ResponseBody User getUserForId(@PathVariable ("id") **int** id) {  User user = **new** User();  user.setId(id);  user.setName("John");  user.setAge(45);  **if**(id !=1){  **throw** **new** RuntimeException();  }    **return** user;  }    @RequestMapping(value = "/mobile/getcomp", method = RequestMethod.***GET***)  @ResponseBody  **public** List<Company> listforCompanies() {  List<Company> listOfCompanies= **new** ArrayList<Company>();  Company cmp = **new** Company();  cmp.setCid("100");  cmp.setCname("TCS");  Company cmp1 = **new** Company();  cmp1.setCid("200");  cmp1.setCname("Wipro");  listOfCompanies.add(cmp);  listOfCompanies.add(cmp1);  **return** listOfCompanies;  }    @RequestMapping(value = "/coffee/{name}", method = RequestMethod.***GET***)  **public** @ResponseBody Coffee getCoffeeInXML(@PathVariable String name) {  Coffee coffee = **new** Coffee(name, 100);  **return** coffee;  }  } |

Web.xml spring-mvc.xml

|  |  |
| --- | --- |
| <web-app id=*"WebApp\_ID"* version=*"2.4"*  xmlns=*"http://java.sun.com/xml/ns/j2ee"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://java.sun.com/xml/ns/j2ee*  *http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd"*>  <display-name>Archetype Created Web Application</display-name>  <!-- This is the root application context for whole web application. Spring  loads this file and creates the ApplicationContext for whole application.  File applicationContext.xml is loaded by ContextLoaderLoaderLinstner which  is configured into web.xml file as the context configuration.There will be  only one application context per web application. We can pass multiple files  in the context configuration by commas or space separation. e.g.  “WEB-INF/applicationContext.xml,WEB-INF/applicationContext-security.xml” -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/spring-mvc.xml</param-value>  </context-param>  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <servlet>  <servlet-name>mvc-dispatcher</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/spring-mvc.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>mvc-dispatcher</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  <!--  <servlet>  <servlet-name>webapp1</servlet-name>  <servlet-class>  org.springframework.web.servlet.DispatcherServlet  </servlet-class>  We require this configuration when we want to change the default  name / location of the servlet specific configuration files  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/app1-servlet.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet>  <servlet-name>webapp2</servlet-name>  <servlet-class>  org.springframework.web.servlet.DispatcherServlet  </servlet-class>  We require this configuration when we want to change the default  name / location of the servlet specific configuration files  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/app2-servlet.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>webapp1</servlet-name>  <url-pattern>/webapp1</url-pattern>  </servlet-mapping>  <servlet-mapping>  <servlet-name>webapp2</servlet-name>  <url-pattern>/webapp2</url-pattern>  </servlet-mapping>  </web-app> -->  </web-app> | <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-4.2.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context-4.2.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd"*>  <!-- Below declaration in the spring application configuration file would  scan the classes inside the specified package and create the beans instance. -->  <context:component-scan base-package=*"com.kb.rest"* />  <!-- This tag would register the HandlerMapping and HandlerAdapter required  to dispatch requests to your @Controllers. In addition, it also applies some  defaults based on what is present in your classpath. -->  <mvc:annotation-driven />  </beans> |