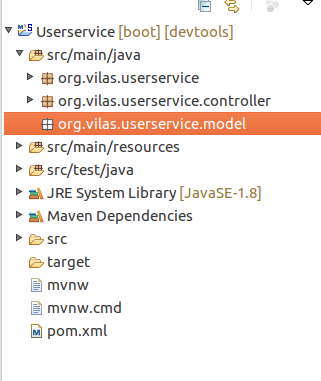
1) Create Spring STS Project

2) Choose the components as\

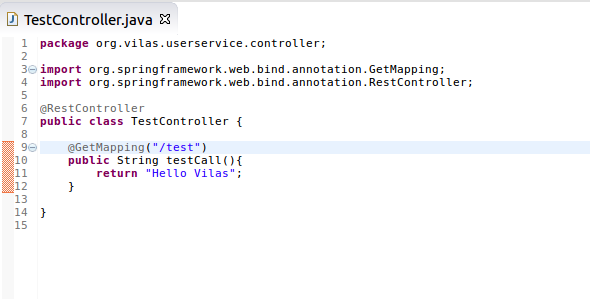
* Devtools
* H2
* Web

3)Once the project is created we create the following project strucutre

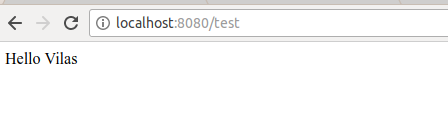


**Creating an Test Controller:**

**Under the package org.vilas.userservice.controller**



When we start the Spring boot application and test the applicaiton we get the following in the browser as it retuns the string



Creating an Model in the org.vilas.userservice.model

**package** org.vilas.userservice.model;

**import** java.util.Date;

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

**private** Integer id;

**private** String name;

**private** Date date;

**public** User(){

}

**public** User (**int** id, String name, Date date){

**this**.id = id;

**this**.name = name;

**this**.date = date;

}

**public** Integer getId() {

**return** id;

}

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

**this**.id = id;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** Date getDate() {

**return** date;

}

**public** **void** setDate(Date date) {

**this**.date = date;

}

@Override

**public** String toString() {

**return** "User [id=" + id + ", name=" + name + ", date=" + date + "]";

}

}

**Create an USERDAO**

**package org.vilas.userservice.dao;**

**import java.util.ArrayList;**

**import java.util.Date;**

**import java.util.Iterator;**

**import java.util.List;**

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

**import org.vilas.userservice.model.User;**

**@Component**

**public class Userservicedao {**

**private static List<User> users = new ArrayList<User>();**

**private static int userCount =3;**

**static{**

**users.add(new User(1, "Adam", new Date()));**

**users.add(new User(2, "Eve", new Date()));**

**users.add(new User(3, "Vilas", new Date()));**

**}**

**//find all the users**

**public List<User> getAllUsers(){**

**return users;**

**}**

**//save the user**

**public User save(User user){**

**System.out.println(user.getId());**

**if(user.getId() == 0){**

**user.setId(++userCount);**

**}**

**users.add(user);**

**return user;**

**}**

**//find one user**

**public User findUser(int id){**

**for(User user : users){**

**if( user.getId() == id){**

**return user;**

**}**

**}**

**return null;**

**}**

**public User deleteUser(int id){**

**Iterator<User> itr =users.iterator();**

**while(itr.hasNext()){**

**User user = itr.next();**

**if(user.getId() == id){**

**itr.remove();**

**}**

**return user;**

**}**

**return null;**

**}**

**}**

**Create an Controller for the User Model**

**package org.vilas.userservice.controller;**

**import java.net.URI;**

**import java.util.List;**

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

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

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

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

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

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

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

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

**import org.springframework.web.servlet.support.ServletUriComponentsBuilder;**

**import org.vilas.userservice.dao.Userservicedao;**

**import org.vilas.userservice.model.User;**

**@RestController**

**public class UserController {**

**@Autowired**

**Userservicedao userservice;**

**@GetMapping("/all")**

**public List<User> getAllUsers(){**

**return userservice.getAllUsers();**

**}**

**@GetMapping("/all/{id}")**

**public User getUser(@PathVariable int id){**

**return userservice.findUser(id);**

**}**

**@PostMapping("/all")**

**public ResponseEntity<?> saveUser(@RequestBody User user){**

**User savedUser = userservice.save(user);**

**//appending the uri to the path.---> This is very important**

**URI uri = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(savedUser.getId()).toUri();**

**return ResponseEntity.created(uri).build();**

**}**

**@DeleteMapping("/all/{id}")**

**public User deleteUser(@PathVariable int id){**

**return userservice.deleteUser(id);**

**}**

**}**

**Exception Handling**

**Say if the suer is not found and we try to get an user that is** [**http://localhost:8080/all/1000**](http://localhost:8080/all/1000)**it gives an message of 200OK which is not correct.**

**We create the following in the controller**

**@GetMapping("/all/{id}")**

**public** User getUser(@PathVariable **int** id){

System.***out***.println(userservice.findUser(id));

User user = userservice.findUser(id);

**if**(user == **null**)

//Create the followign usernotfound exception

**throw new UserNotFoundException("id-"+id);**

**return** user;

**}**

**The user not found exception is as follows.**

**WE create the Exception not found in the exception package.**

**We extend the Custom Exception with the runtime exception.**

**We create the ResponseStatus with HTTPStatus not found.**

**We create the constructor based on the super constructor.**

**package org.vilas.userservice.exception;**

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

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

@ResponseStatus(HttpStatus.***NOT\_FOUND***)

**public** **class** UserNotFoundException **extends** RuntimeException {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** UserNotFoundException(String message) {

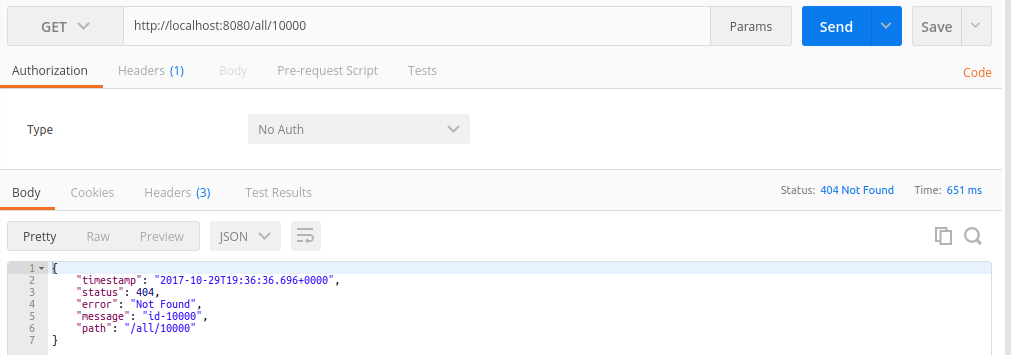
**super**(message);

// **TODO** Auto-generated constructor stub

}

}

When firing the <http://localhost:8080/all/50000>we get the following in the postman.



**Create the Generic Exception.**

**In the organization we create a standard strucutre.**

**Create a new Exception Class**

**ExceptionResponse**

**package org.vilas.userservice.exception;**

**import** java.util.Date;

**public** **class** ExceptionResponse {

**private** Date timeStamp;

**private** String message;

**private** String details;

**public** ExceptionResponse(Date timeStamp, String message, String details) {

**super**();

**this**.timeStamp = timeStamp;

**this**.message = message;

**this**.details = details;

}

**public** Date getTimeStamp() {

**return** timeStamp;

}

**public** **void** setTimeStamp(Date timeStamp) {

**this**.timeStamp = timeStamp;

}

**public** String getMessage() {

**return** message;

}

**public** **void** setMessage(String message) {

**this**.message = message;

}

**public** String getDetails() {

**return** details;

}

**public** **void** setDetails(String details) {

**this**.details = details;

}

}

**We create an CustomizedResponseEntityExceptionHandler which extends ResponseEntityExceptionHandler**

**and this customresponseentityexception handler should be handled by all controllers.**

**//Handles the** {@link ExceptionHandler @ExceptionHandler}, {@link InitBinder @InitBinder}--Date, or

**// \* {@link ModelAttribute @ModelAttribute} shared across all the controllers.**

**package org.vilas.userservice.exception;**

**import** java.util.Date;

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

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

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

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

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

**import** org.springframework.web.context.request.WebRequest;

**import** org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

@RestController

@ControllerAdvice // this will be applicable to all the controllers/

**public** **class** CustomizedResponseEntityExceptionHandler **extends**

ResponseEntityExceptionHandler{

@ExceptionHandler(Exception.**class**)

**public** **final** ResponseEntity<Object> handleAllExceptions(Exception ex, WebRequest request) {

ExceptionResponse expres = **new** ExceptionResponse(**new** Date(), ex.getMessage(), request.getDescription(**false**));

**return** **new** ResponseEntity(expres, HttpStatus.***INTERNAL\_SERVER\_ERROR***);

}

@ExceptionHandler(UserNotFoundException.**class**)

**public** **final** ResponseEntity<Object> userNotFoundException(Exception ex, WebRequest request) {

ExceptionResponse expres = **new** ExceptionResponse(**new** Date(), ex.getMessage(), request.getDescription(**false**));

**return** **new** ResponseEntity(expres, HttpStatus.***NOT\_FOUND***);

}

}

**Implementing Validations**