**SPRING BOOT AND HIBERNATE INTEGRATION**

**Versions**

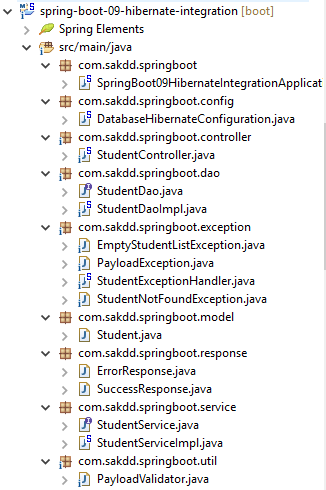
Maven 3.5.4

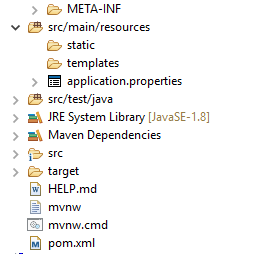
Hibernate 5.0.12

Spring Boot 1.5.19

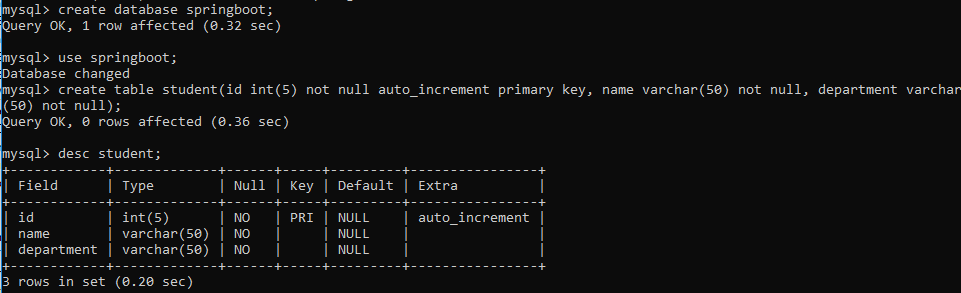
Java 1.8

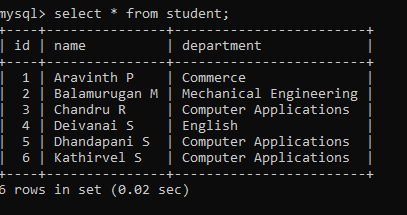
**Structure**





**DB and Table**





**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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>

<parent>

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

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

<version>1.5.19.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<groupId>com.sakdd.springboot</groupId>

<artifactId>spring-boot-09-hibernate-integration</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-boot-09-hibernate-integration</name>

<description>Demo Application Spring Boot and Hibernate Integration</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**MAIN APPLICATION**

**SpringBoot09HibernateIntegrationApplication.java**

package com.sakdd.springboot;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Import;

import com.sakdd.springboot.config.DatabaseHibernateConfiguration;

@Import(DatabaseHibernateConfiguration.class)

@SpringBootApplication

public class SpringBoot09HibernateIntegrationApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBoot09HibernateIntegrationApplication.class, args);

}

}

**CONFIGURATION**

**DatabaseHibernateConfiguration.java**

package com.sakdd.springboot.config;

import java.util.Properties;

import javax.sql.DataSource;

import org.hibernate.SessionFactory;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.PropertySource;

import org.springframework.core.env.Environment;

import org.springframework.jdbc.datasource.DriverManagerDataSource;

import org.springframework.orm.hibernate5.HibernateTransactionManager;

import org.springframework.orm.hibernate5.LocalSessionFactoryBean;

import org.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration

@EnableTransactionManagement

@PropertySource(value = { "classpath:application.properties" })

public class DatabaseHibernateConfiguration

{

@Autowired

private Environment env;

@Bean

public DataSource dataSource()

{

DriverManagerDataSource dataSource=new DriverManagerDataSource();

dataSource.setDriverClassName(env.getProperty("spring.datasource.driver-class-name"));

dataSource.setUrl(env.getProperty("spring.datasource.url"));

dataSource.setUsername(env.getProperty("spring.datasource.username"));

dataSource.setPassword(env.getProperty("spring.datasource.password"));

return dataSource;

}

public Properties getHibernateProperties()

{

Properties hibernateProperties=new Properties();

hibernateProperties.setProperty("hibernate.dialect", env.getProperty("hibernate.dialect"));

hibernateProperties.setProperty("hibernate.show\_sql", env.getProperty("hibernate.show\_sql"));

return hibernateProperties;

}

@Bean

public LocalSessionFactoryBean sessionFactory()

{

LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();

sessionFactory.setDataSource(dataSource());

sessionFactory.setPackagesToScan(env.getProperty("hibernate.packagesToScan"));

sessionFactory.setHibernateProperties(getHibernateProperties());

return sessionFactory;

}

@Bean

@Autowired

public HibernateTransactionManager transactionManager(SessionFactory sessionFactory)

{

HibernateTransactionManager transactionManager=new HibernateTransactionManager();

transactionManager.setSessionFactory(sessionFactory);

return transactionManager;

}

}

**MODEL**

**Student.java**

package com.sakdd.springboot.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name="student")

public class Student

{

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

@Column(name="id")

private int id;

@Column(name="name")

private String name;

@Column(name="department")

private String department;

public int getId()

{

return id;

}

public void setId(int id)

{

this.id = id;

}

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

public String getDepartment()

{

return department;

}

public void setDepartment(String department)

{

this.department = department;

}

public Student()

{

super();

}

public Student(int id, String name, String department)

{

super();

this.id = id;

this.name = name;

this.department = department;

}

}

**RESPONSE**

**SuccessResponse.java**

package com.sakdd.springboot.response;

public class SuccessResponse

{

private int successCode;

private String successMessage;

public int getSuccessCode()

{

return successCode;

}

public void setSuccessCode(int successCode)

{

this.successCode = successCode;

}

public String getSuccessMessage()

{

return successMessage;

}

public void setSuccessMessage(String successMessage)

{

this.successMessage = successMessage;

}

public SuccessResponse()

{

super();

}

public SuccessResponse(int successCode, String successMessage)

{

super();

this.successCode = successCode;

this.successMessage = successMessage;

}

}

**ErrorResponse.java**

package com.sakdd.springboot.response;

public class ErrorResponse

{

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;

}

public ErrorResponse()

{

super();

}

public ErrorResponse(int errorCode, String errorMessage)

{

super();

this.errorCode = errorCode;

this.errorMessage = errorMessage;

}

}

**DAO**

**StudentDao.java**

package com.sakdd.springboot.dao;

import java.util.List;

import com.sakdd.springboot.model.Student;

public interface StudentDao

{

public List<Student> getStudents();

public Student getStudentById(int id);

public Student saveStudent(Student student);

public Student updateStudent(Student student);

public String deleteStudentById(int id);

}

**StudentDaoImpl.java**

package com.sakdd.springboot.dao;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

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

import org.springframework.stereotype.Repository;

import com.sakdd.springboot.model.Student;

@Repository

public class StudentDaoImpl implements StudentDao

{

@Autowired

private SessionFactory sessionFactory;

@Override

public List<Student> getStudents()

{

Session session = sessionFactory.getCurrentSession();

@SuppressWarnings("unchecked")

List<Student> students = session.createCriteria(Student.class).list();

return students;

}

@Override

public Student getStudentById(int id)

{

Session session = sessionFactory.getCurrentSession();

Student student = session.get(Student.class, id);

return student;

}

@Override

public Student saveStudent(Student student)

{

Session session = sessionFactory.getCurrentSession();

session.save(student);

return student;

}

@Override

public Student updateStudent(Student student)

{

Session session = sessionFactory.getCurrentSession();

Student foundStudent = session.get(Student.class, student.getId());

foundStudent.setId(student.getId());

foundStudent.setName(student.getName());

foundStudent.setDepartment(student.getDepartment());

session.update(foundStudent);

return student;

}

@Override

public String deleteStudentById(int id)

{

Session session = sessionFactory.getCurrentSession();

Student foundStudent = session.get(Student.class, id);

session.delete(foundStudent);

return "Deleted Successfully";

}

}

**SERVICE**

**StudentService.java**

package com.sakdd.springboot.service;

import java.util.List;

import com.sakdd.springboot.model.Student;

public interface StudentService

{

public List<Student> getStudents();

public Student getStudentById(int id);

public Student saveStudent(Student student);

public Student updateStudent(Student student);

public String deleteStudentById(int id);

}

**StudentServiceImpl.java**

package com.sakdd.springboot.service;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.sakdd.springboot.dao.StudentDao;

import com.sakdd.springboot.model.Student;

@Service

public class StudentServiceImpl implements StudentService

{

@Autowired

private StudentDao studentDao;

@Override

@Transactional

public List<Student> getStudents()

{

return studentDao.getStudents();

}

@Override

@Transactional

public Student getStudentById(int id)

{

return studentDao.getStudentById(id);

}

@Override

@Transactional

public Student saveStudent(Student student)

{

return studentDao.saveStudent(student);

}

@Override

@Transactional

public Student updateStudent(Student student)

{

return studentDao.updateStudent(student);

}

@Override

@Transactional

public String deleteStudentById(int id)

{

return studentDao.deleteStudentById(id);

}

}

**CONTROLLER**

**StudentController.java**

package com.sakdd.springboot.controller;

import java.util.List;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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.bind.annotation.RestController;

import com.sakdd.springboot.exception.EmptyStudentListException;

import com.sakdd.springboot.exception.PayloadException;

import com.sakdd.springboot.exception.StudentNotFoundException;

import com.sakdd.springboot.model.Student;

import com.sakdd.springboot.service.StudentService;

import com.sakdd.springboot.util.PayloadValidator;

@RestController

@RequestMapping("studentRestApi")

public class StudentController

{

@Autowired

private StudentService studentService;

@RequestMapping(value="/student", method=RequestMethod.GET)

public ResponseEntity<List<Student>> getStudents()

{

List<Student> students = studentService.getStudents();

if(!students.isEmpty())

{

return new ResponseEntity<List<Student>>(students,HttpStatus.OK);

}

else

{

throw new EmptyStudentListException("Students does not exist in the database");

}

}

@RequestMapping(value="/student/{id}", method=RequestMethod.GET)

public ResponseEntity<Student> getStudentById(@PathVariable("id") int id)

{

Student foundStudent = studentService.getStudentById(id);

if(foundStudent!=null)

{

return new ResponseEntity<Student>(foundStudent,HttpStatus.OK);

}

else

{

throw new StudentNotFoundException("Student does not exist");

}

}

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

public ResponseEntity<Student> saveStudent(@RequestBody Student payload)

{

boolean payloadValiationStatus = PayloadValidator.validatePayload(payload);

if(payloadValiationStatus)

{

Student savedStudent = studentService.saveStudent(payload);

return new ResponseEntity<Student>(savedStudent,HttpStatus.OK);

}

else

{

throw new PayloadException("Malformed Payload, Student id must not be defined");

}

}

@RequestMapping(value="/student", method=RequestMethod.PUT)

public ResponseEntity<Student> updateStudent(@RequestBody Student payload)

{

Student foundStudent = studentService.getStudentById(payload.getId());

if(foundStudent!=null)

{

Student updatedStudent = studentService.updateStudent(payload);

return new ResponseEntity<Student>(updatedStudent,HttpStatus.OK);

}

else

{

throw new StudentNotFoundException("Student to update does not not exist");

}

}

@RequestMapping(value="/student/{id}", method=RequestMethod.DELETE)

public ResponseEntity<String> deleteStudentById(@PathVariable("id") int id)

{

Student foundStudent = studentService.getStudentById(id);

if(foundStudent!=null)

{

String deleteStatus = studentService.deleteStudentById(id);

return new ResponseEntity<String>(deleteStatus,HttpStatus.OK);

}

else

{

throw new StudentNotFoundException("Student to delete does not not exist");

}

}

}

**EXCEPTION AND HANDLER**

**EmptyStudentListException.java**

package com.sakdd.springboot.exception;

public class EmptyStudentListException extends RuntimeException

{

private static final long serialVersionUID = 1L;

private String exceptionMessage;

public String getExceptionMessage()

{

return exceptionMessage;

}

public void setExceptionMessage(String exceptionMessage)

{

this.exceptionMessage = exceptionMessage;

}

public EmptyStudentListException(String exceptionMessage)

{

super(exceptionMessage);

this.exceptionMessage=exceptionMessage;

}

}

**PayloadException.java**

package com.sakdd.springboot.exception;

public class PayloadException extends RuntimeException

{

private static final long serialVersionUID = 1L;

private String exceptionMessage;

public String getExceptionMessage()

{

return exceptionMessage;

}

public void setExceptionMessage(String exceptionMessage)

{

this.exceptionMessage = exceptionMessage;

}

public PayloadException(String exceptionMessage)

{

super(exceptionMessage);

this.exceptionMessage=exceptionMessage;

}

}

**StudentNotFoundException.java**

package com.sakdd.springboot.exception;

public class StudentNotFoundException extends RuntimeException

{

private static final long serialVersionUID = 1L;

private String exceptionMessage;

public String getExceptionMessage()

{

return exceptionMessage;

}

public void setExceptionMessage(String exceptionMessage)

{

this.exceptionMessage = exceptionMessage;

}

public StudentNotFoundException(String exceptionMessage)

{

super(exceptionMessage);

this.exceptionMessage=exceptionMessage;

}

}

**StudentExceptionHanlder.java**

package com.sakdd.springboot.exception;

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 com.sakdd.springboot.response.ErrorResponse;

@ControllerAdvice

public class StudentExceptionHandler

{

@ExceptionHandler(StudentNotFoundException.class)

public ResponseEntity<ErrorResponse> studentNotFoundExcpetionHandler(Exception ex)

{

ErrorResponse errorResponse =new ErrorResponse();

errorResponse.setErrorCode(HttpStatus.NOT\_FOUND.value());

errorResponse.setErrorMessage(ex.getMessage());

return new ResponseEntity<ErrorResponse>(errorResponse,HttpStatus.NOT\_FOUND);

}

@ExceptionHandler(PayloadException.class)

public ResponseEntity<ErrorResponse> payloadExcpetionHandler(Exception ex)

{

ErrorResponse errorResponse =new ErrorResponse();

errorResponse.setErrorCode(HttpStatus.BAD\_REQUEST.value());

errorResponse.setErrorMessage(ex.getMessage());

return new ResponseEntity<ErrorResponse>(errorResponse,HttpStatus.BAD\_REQUEST);

}

@ExceptionHandler(EmptyStudentListException.class)

public ResponseEntity<ErrorResponse> emptyStudentListExceptionHandler(Exception ex)

{

ErrorResponse errorResponse =new ErrorResponse();

errorResponse.setErrorCode(HttpStatus.NOT\_FOUND.value());

errorResponse.setErrorMessage(ex.getMessage());

return new ResponseEntity<ErrorResponse>(errorResponse,HttpStatus.NOT\_FOUND);

}

@ExceptionHandler(Exception.class)

public ResponseEntity<ErrorResponse> gloablExcpetionHandler(Exception ex)

{

ErrorResponse errorResponse=new ErrorResponse();

errorResponse.setErrorCode(HttpStatus.BAD\_REQUEST.value());

errorResponse.setErrorMessage("The request could not be understood by the server due to malformed syntax.");

return new ResponseEntity<ErrorResponse>(errorResponse,HttpStatus.BAD\_REQUEST);

}

}

**APPLICATION PROPERTIES**

**application.properties**

hibernate.dialect=org.hibernate.dialect.MySQLDialect

hibernate.show\_sql=true

hibernate.packagesToScan=com.sakdd.springboot.model

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/springboot

spring.datasource.username=root

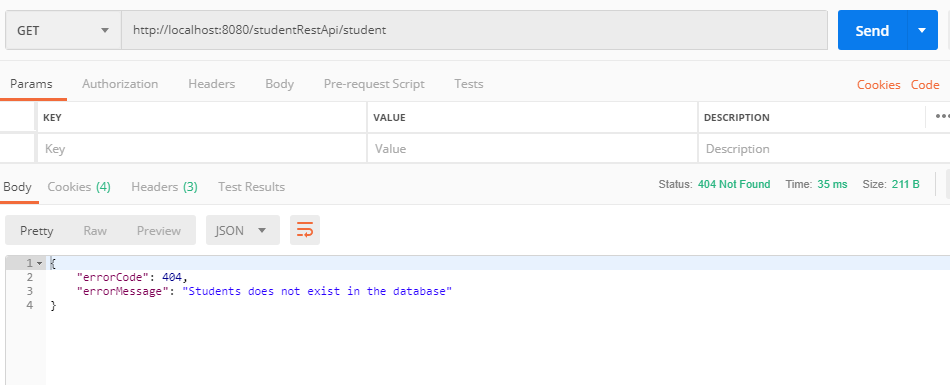
spring.datasource.password=root

**Output Screenshots**

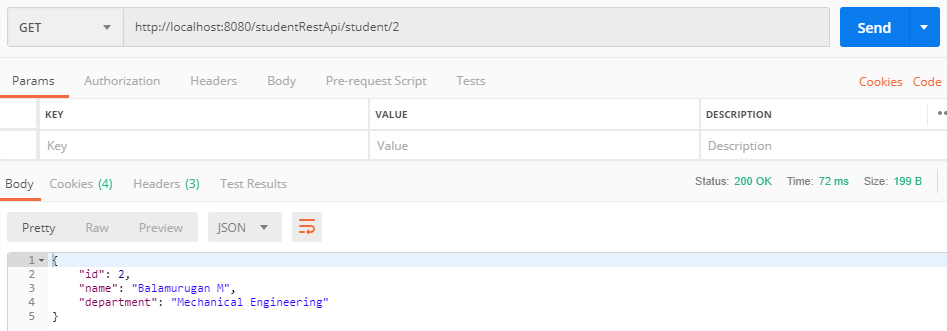
**Fetching all the students**



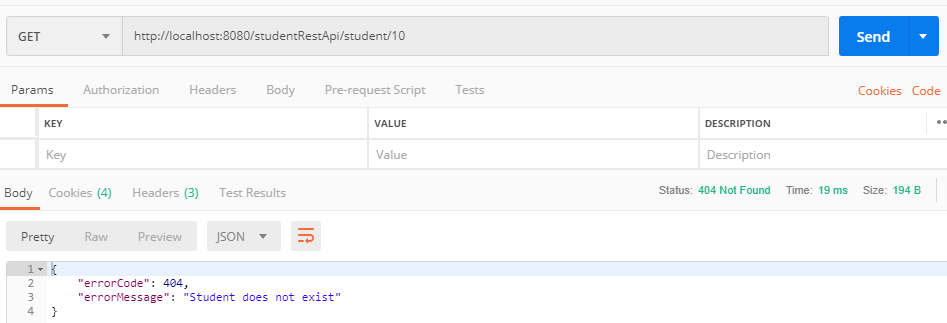
**Throwing error if the student list is empty**



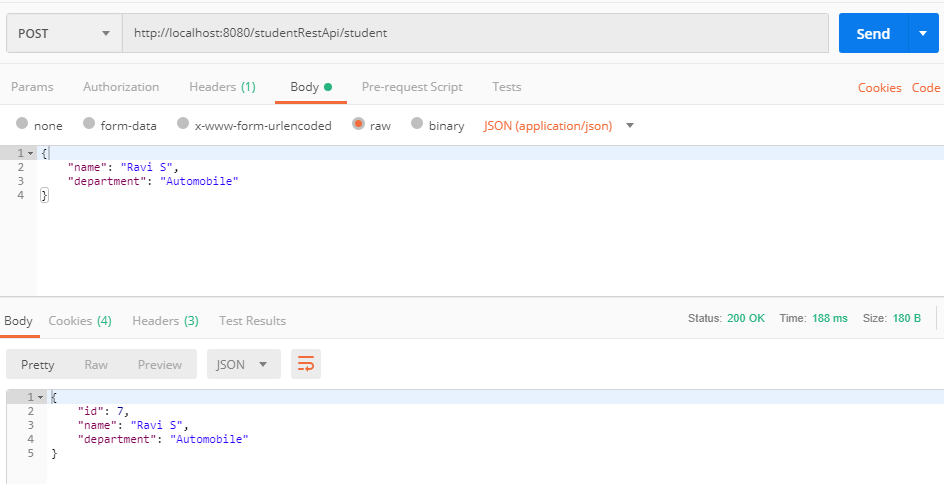
**Fetching single student**



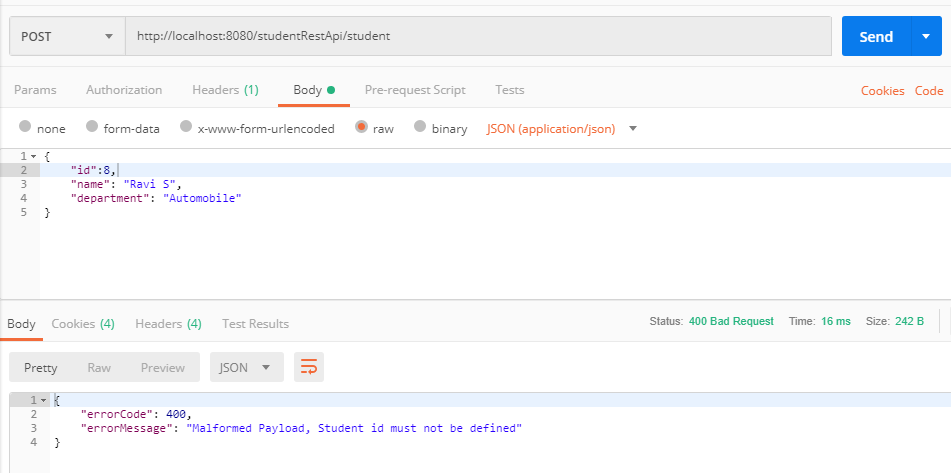
**Throwing error if student not found**



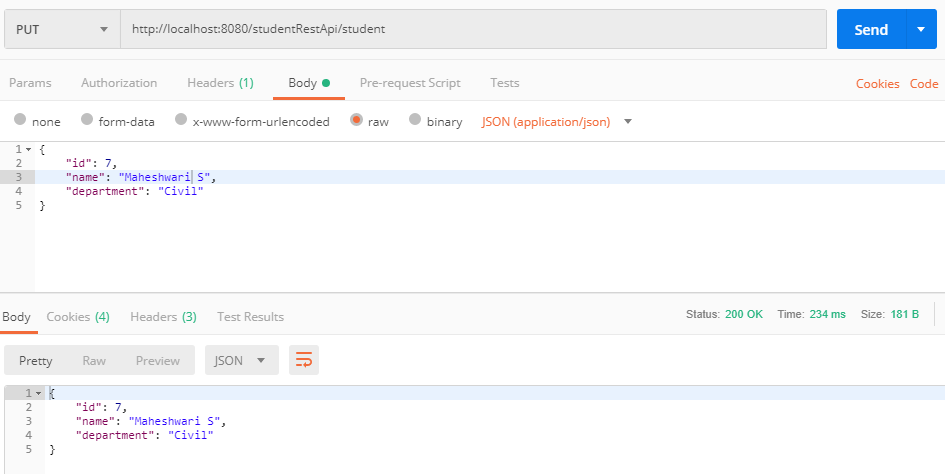
**Saving a student**



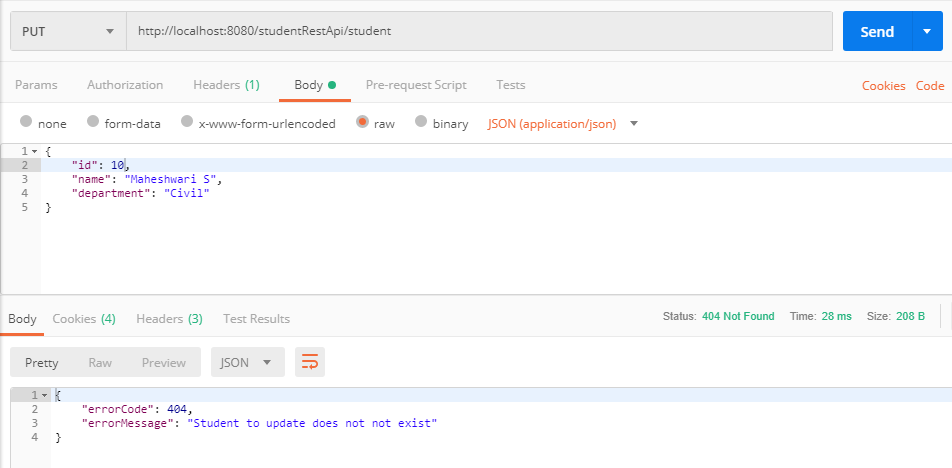
**Throwing error if id added in the payload**



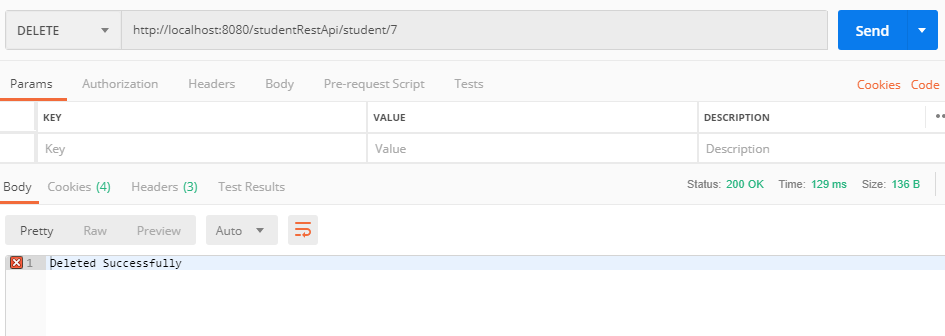
**Updating a student**



**Throwing error if student not found**



**Deleting a student**



**Throwing error if student not found**

