<?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 https://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>2.6.0-SNAPSHOT</version>

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

</parent>

<groupId>com.revature.springrest</groupId>

<artifactId>............................... </artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>springrestdemo1</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>11</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>spring-milestones</id>

<name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>

<snapshots>

<enabled>false</enabled>

</snapshots>

</repository>

<repository>

<id>spring-snapshots</id>

<name>Spring Snapshots</name>

<url>https://repo.spring.io/snapshot</url>

<releases>

<enabled>false</enabled>

</releases>

</repository>

</repositories>

<pluginRepositories>

<pluginRepository>

<id>spring-milestones</id>

<name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>

<snapshots>

<enabled>false</enabled>

</snapshots>

</pluginRepository>

<pluginRepository>

<id>spring-snapshots</id>

<name>Spring Snapshots</name>

<url>https://repo.spring.io/snapshot</url>

<releases>

<enabled>false</enabled>

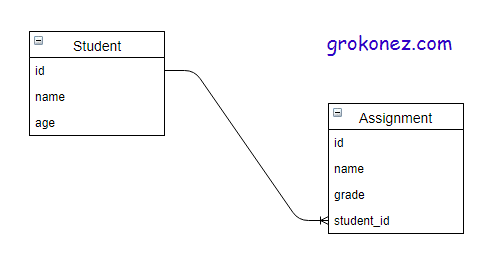
</releases>

</pluginRepository>

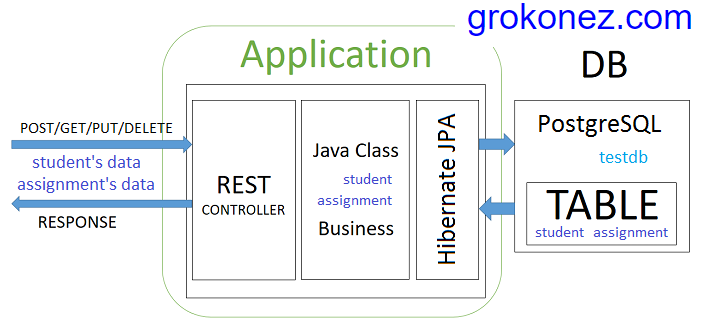
</pluginRepositories>

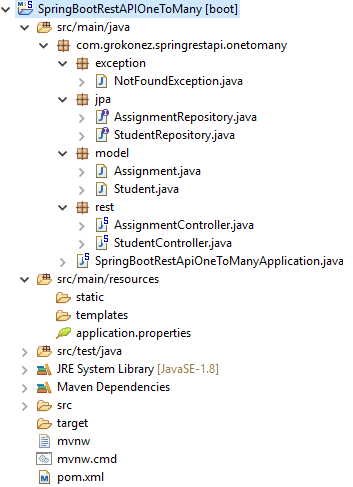
</project>

We have 2 models Student & Assignment with One-to-Many association relationship:



We create a SpringBoot project as below:





Hibernate JPA configuration for 2 models Student & Assignment:

– Student model ->

@Entity

@Table(name = "students")

@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})

public class Student implements Serializable{

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name")

private String name;

@Column(name = "age")

private int age;

@OneToMany(mappedBy = "student", cascade = CascadeType.ALL, fetch = FetchType.LAZY)

private Set assignments;

public Student() {}

public Student(String name, int age) {

this.name = name;

this.age = age;

}

// Getter & Setter methods

// ...

– Assignment model ->

@Entity

@Table(name = "assignments")

@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})

public class Assignment implements Serializable{

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name")

private String name;

@Column(name = "grade")

private int grade;

@ManyToOne(fetch = FetchType.LAZY, optional = false)

@JoinColumn(name = "student\_id", nullable = false)

@JsonIgnore

private Student student;

public Assignment() {}

public Assignment(String name, int grade) {

this.name = name;

this.grade = grade;

}

// Getter & Setter methods

// ...

We exposes RestAPIs for Post/Get/Put/Delete Students & Assignments:

– Student ->

* @GetMapping("/api/students"): get all Students
* @GetMapping("/api/students/{id}"): get a Student by ID
* @PostMapping("/api/students"): post a Student
* @PutMapping("/api/students/{id}"): update a Student
* @DeleteMapping("/api/students/{id}"): delete a Student

– Assignments ->

* @GetMapping("/students/{studentId}/assignments"): get a Assignment by Student’s ID
* @PostMapping("/students/{studentId}/assignments"): add an Assignment
* @PutMapping("/students/{studentId}/assignments/{assignmentId}"): update an Assignment
* @DeleteMapping("/students/{studentId}/assignments/{assignmentId}"): delete an Assignment by ID

Now we’ll create a project from scratch ->

**Practice**

**Create SpringBoot project**

We use SpringToolSuite to create a Java 8 SpringBoot project with below 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>org.postgresql</groupid>

<artifactid>postgresql</artifactid>

<scope>runtime</scope>

</dependency>

**OneToMany Models**

– Student model ->

package com.ozenero.springrestapi.onetomany.model;

import java.io.Serializable;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToMany;

import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Entity

@Table(name = "students")

@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})

public class Student implements Serializable{

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name")

private String name;

@Column(name = "age")

private int age;

@OneToMany(mappedBy = "student", cascade = CascadeType.ALL, fetch = FetchType.LAZY)

private Set<assignment> assignments;

public Student() {}

public Student(String name, int age) {

this.name = name;

this.age = age;

}

public void setId(Long id) {

this.id = id;

}

public Long getId() {

return this.id;

}

public void setName(String name) {

this.name = name;

}

public String getName() {

return this.name;

}

public void setAge(int age) {

this.age = age;

}

public int getAge() {

return this.age;

}

public void setAssignments(Set<assignment> assignments) {

this.assignments = assignments;

}

public Set<assignment> getAssignments(){

return this.assignments;

}

}

</assignment></assignment></assignment>

– Assignment model ->

package com.ozenero.springrestapi.onetomany.model;

import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Entity

@Table(name = "assignments")

@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})

public class Assignment implements Serializable{

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name")

private String name;

@Column(name = "grade")

private int grade;

@ManyToOne(fetch = FetchType.LAZY, optional = false)

@JoinColumn(name = "student\_id", nullable = false)

@JsonIgnore

private Student student;

public Assignment() {}

public Assignment(String name, int grade) {

this.name = name;

this.grade = grade;

}

public void setId(Long id) {

this.id = id;

}

public Long getId() {

return this.id;

}

public void setName(String name) {

this.name = name;

}

public String getName() {

return this.name;

}

public void setGrade(int grade) {

this.grade = grade;

}

public int getGrade() {

return this.grade;

}

public void setStudent(Student student) {

this.student = student;

}

public Student getStudent() {

return this.student;

}

}

**JPA Repositories**

– StudentRepository ->

package com.ozenero.springrestapi.onetomany.jpa;

import org.springframework.data.jpa.repository.JpaRepository;

import com.ozenero.springrestapi.onetomany.model.Student;

public interface StudentRepository extends JpaRepository<student, long=""> {

}

</student,>

– AssignmentRepository ->

package com.ozenero.springrestapi.onetomany.jpa;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import com.ozenero.springrestapi.onetomany.model.Assignment;

public interface AssignmentRepository extends JpaRepository<assignment, long=""> {

List<assignment> findByStudentId(Long studentId);

}

</assignment></assignment,>

Add datasource configurations in application.properties file ->

spring.datasource.url=jdbc:postgresql://localhost/testdb

spring.datasource.username=postgres

spring.datasource.password=123

spring.jpa.generate-ddl=true

#spring.jackson.serialization.fail-on-empty-beans=false

**Expose Rest APIs**

– Create NotFoundException ->

package com.ozenero.springrestapi.onetomany.exception;

import org.springframework.http.HttpStatus;

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

@ResponseStatus(code = HttpStatus.NOT\_FOUND, reason = "NotFoundException not found")

public class NotFoundException extends RuntimeException {

private static final long serialVersionUID = 1L;

public NotFoundException(String message) {

super(message);

}

public NotFoundException(String message, Throwable cause) {

super(message, cause);

}

}

– StudentController APIs ->

package com.ozenero.springrestapi.onetomany.rest;

import java.util.List;

import java.util.Optional;

import javax.validation.Valid;

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

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.PutMapping;

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

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

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

import com.ozenero.springrestapi.onetomany.exception.NotFoundException;

import com.ozenero.springrestapi.onetomany.jpa.StudentRepository;

import com.ozenero.springrestapi.onetomany.model.Student;

@RestController

@RequestMapping("/api")

public class StudentController {

@Autowired

private StudentRepository studentRepository;

@GetMapping("/students")

public List getAllStudents() {

return studentRepository.findAll();

}

@GetMapping("/students/{id}")

public Student getStudentByID(@PathVariable Long id) {

Optional optStudent = studentRepository.findById(id);

if(optStudent.isPresent()) {

return optStudent.get();

}else {

throw new NotFoundException("Student not found with id " + id);

}

}

@PostMapping("/students")

public Student createStudent(@Valid @RequestBody Student student) {

return studentRepository.save(student);

}

@PutMapping("/students/{id}")

public Student updateStudent(@PathVariable Long id,

@Valid @RequestBody Student studentUpdated) {

return studentRepository.findById(id)

.map(student -> {

student.setName(studentUpdated.getName());

student.setAge(studentUpdated.getAge());

return studentRepository.save(student);

}).orElseThrow(() -> new NotFoundException("Student not found with id " + id));

}

@DeleteMapping("/students/{id}")

public String deleteStudent(@PathVariable Long id) {

return studentRepository.findById(id)

.map(student -> {

studentRepository.delete(student);

return "Delete Successfully!";

}).orElseThrow(() -> new NotFoundException("Student not found with id " + id));

}

}

– AssignmentController APIs ->

package com.ozenero.springrestapi.onetomany.rest;

import java.util.List;

import javax.validation.Valid;

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

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.PutMapping;

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

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

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

import com.ozenero.springrestapi.onetomany.exception.NotFoundException;

import com.ozenero.springrestapi.onetomany.jpa.AssignmentRepository;

import com.ozenero.springrestapi.onetomany.jpa.StudentRepository;

import com.ozenero.springrestapi.onetomany.model.Assignment;

@RestController

@RequestMapping("/api")

public class AssignmentController {

@Autowired

private AssignmentRepository assignmentRepository;

@Autowired

private StudentRepository studentRepository;

@GetMapping("/students/{studentId}/assignments")

public List getContactByStudentId(@PathVariable Long studentId) {

if(!studentRepository.existsById(studentId)) {

throw new NotFoundException("Student not found!");

}

return assignmentRepository.findByStudentId(studentId);

}

@PostMapping("/students/{studentId}/assignments")

public Assignment addAssignment(@PathVariable Long studentId,

@Valid @RequestBody Assignment assignment) {

return studentRepository.findById(studentId)

.map(student -> {

assignment.setStudent(student);

return assignmentRepository.save(assignment);

}).orElseThrow(() -> new NotFoundException("Student not found!"));

}

@PutMapping("/students/{studentId}/assignments/{assignmentId}")

public Assignment updateAssignment(@PathVariable Long studentId,

@PathVariable Long assignmentId,

@Valid @RequestBody Assignment assignmentUpdated) {

if(!studentRepository.existsById(studentId)) {

throw new NotFoundException("Student not found!");

}

return assignmentRepository.findById(assignmentId)

.map(assignment -> {

assignment.setName(assignmentUpdated.getName());

assignment.setGrade(assignmentUpdated.getGrade());

return assignmentRepository.save(assignment);

}).orElseThrow(() -> new NotFoundException("Assignment not found!"));

}

@DeleteMapping("/students/{studentId}/assignments/{assignmentId}")

public String deleteAssignment(@PathVariable Long studentId,

@PathVariable Long assignmentId) {

if(!studentRepository.existsById(studentId)) {

throw new NotFoundException("Student not found!");

}

return assignmentRepository.findById(assignmentId)

.map(assignment -> {

assignmentRepository.delete(assignment);

return "Deleted Successfully!";

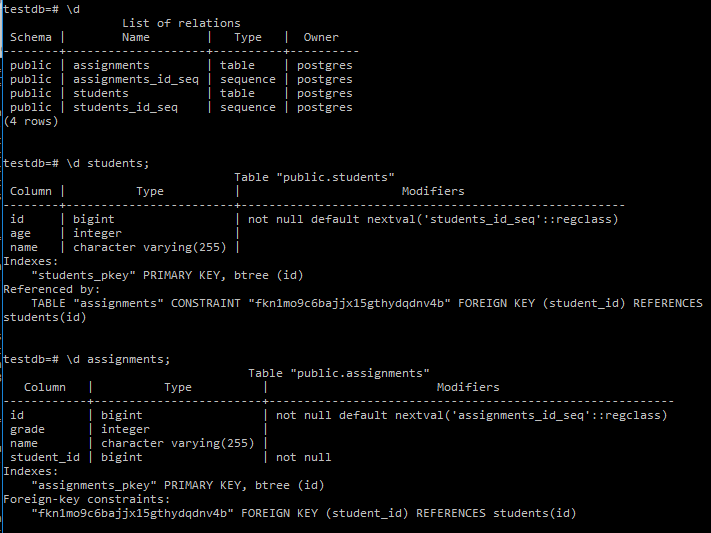
}).orElseThrow(() -> new NotFoundException("Contact not found!"));

}

}

**Run & Check Results**

– Run the SpringBoot project with commandline mvn spring-boot:run.  
2 tables is created in PostgreSQL ->

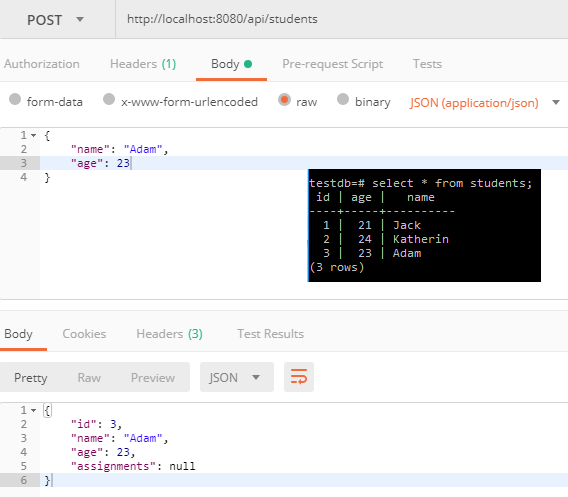


– Add Students post -> <http://localhost:8080/api/students>

{ “name”: “ram”,

“age”:20

}

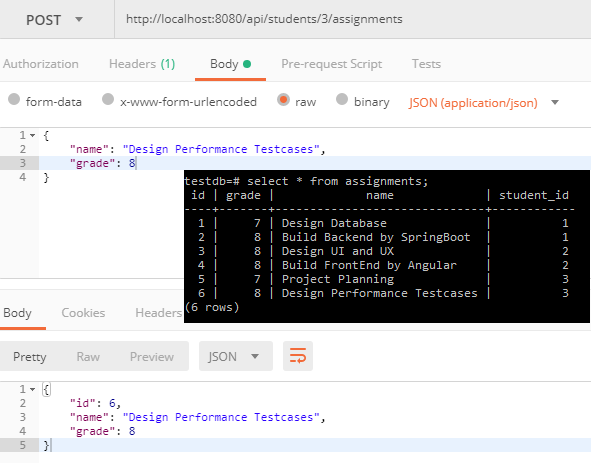


– Add Assignments -> <http://localohost:8080/api/students/1/assignments>

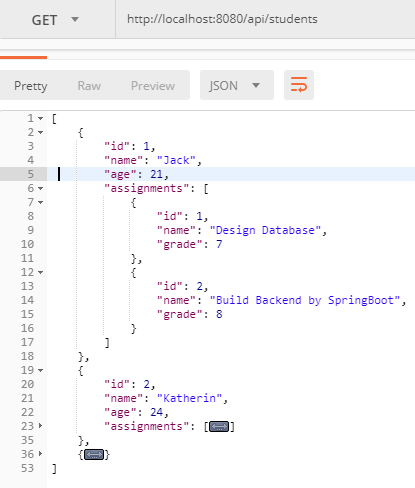
{ “name” :”ram”,

“grade”:

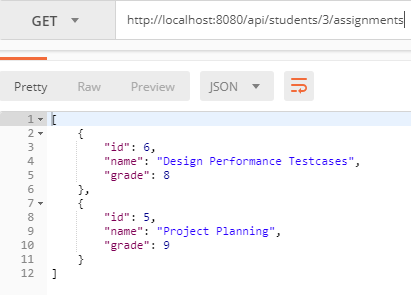
}



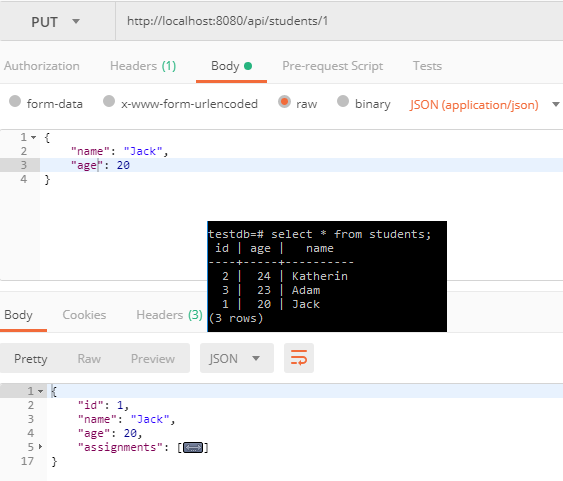
– Get All Students ->



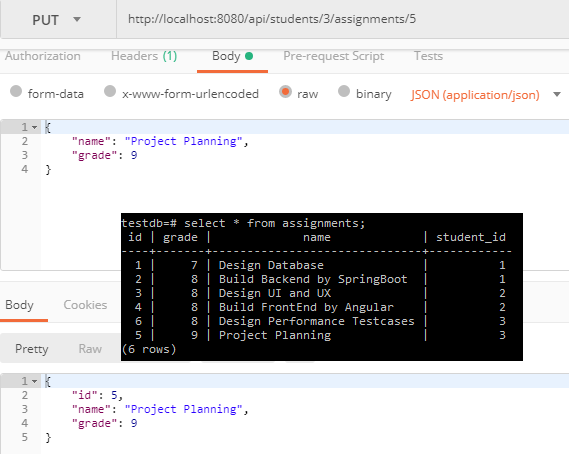
– Get All Assignments of a Student->



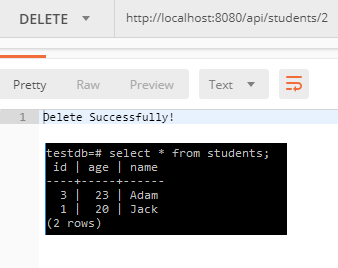
– Update Student ->



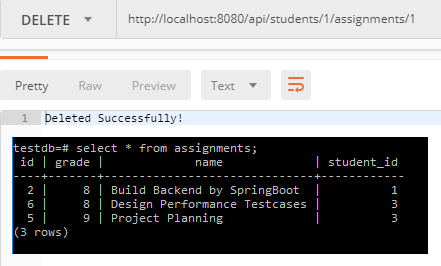
– Update Assignment ->



– Delete a Student -> image



– Delete an Assignment ->



**Note**: PostgreSQL commandline ->

PostgreSQL\9.6\bin>psql.exe --username="postgres" -W: connect to PostgreSQL ->

C:\Program Files\PostgreSQL\9.6\bin>psql.exe --username="postgres" -W

Password for user postgres:

psql (9.6.9)

WARNING: Console code page (437) differs from Windows code page (1252)

8-bit characters might not work correctly. See psql reference

page "Notes for Windows users" for details.

Type "help" for help.

– \l: List of databases

– \c testdb: connect to “testdb”

– \d: List of relations ->

testdb=# \d

List of relations

Schema | Name | Type | Owner

--------+--------------------+----------+----------

public | assignments | table | postgres

public | assignments\_id\_seq | sequence | postgres

public | students | table | postgres

public | students\_id\_seq | sequence | postgres

(4 rows)

– \d assignments: Get Schema of a table assignments ->

testdb=# \d assignments;

Table "public.assignments"

Column | Type | Modifiers

------------+------------------------+----------------------------------------------------------

id | bigint | not null default nextval('assignments\_id\_seq'::regclass)

grade | integer |

name | character varying(255) |

student\_id | bigint | not null

Indexes:

"assignments\_pkey" PRIMARY KEY, btree (id)

Foreign-key constraints:

"fkn1mo9c6bajjx15gthydqdnv4b" FOREIGN KEY (student\_id) REFERENCES students(id)