Academia Xideral Java Monterrey

Daniel Ivan Anaya Alvarez

SQL Activity

In this activity we are going to get the ropes to use SQL on java with the method FPA, this activity requires to use the jakarta driver that takes jdbc and simplifies it, makes it simplier to code. In this activity I created a new user named “actividadsql” I also made a new data base for this named the same as the user. Now as the content used for this data base was simply using a simple table and add age into it, I needed to adapt the java script to match the name of the data base, user, password and also creating the new pointers to the new column of age. At the end I modified the constructor and the print in order to see it. I just did some basic modifications, I retained the name student because if I changed that it would make a chaos on all that code.

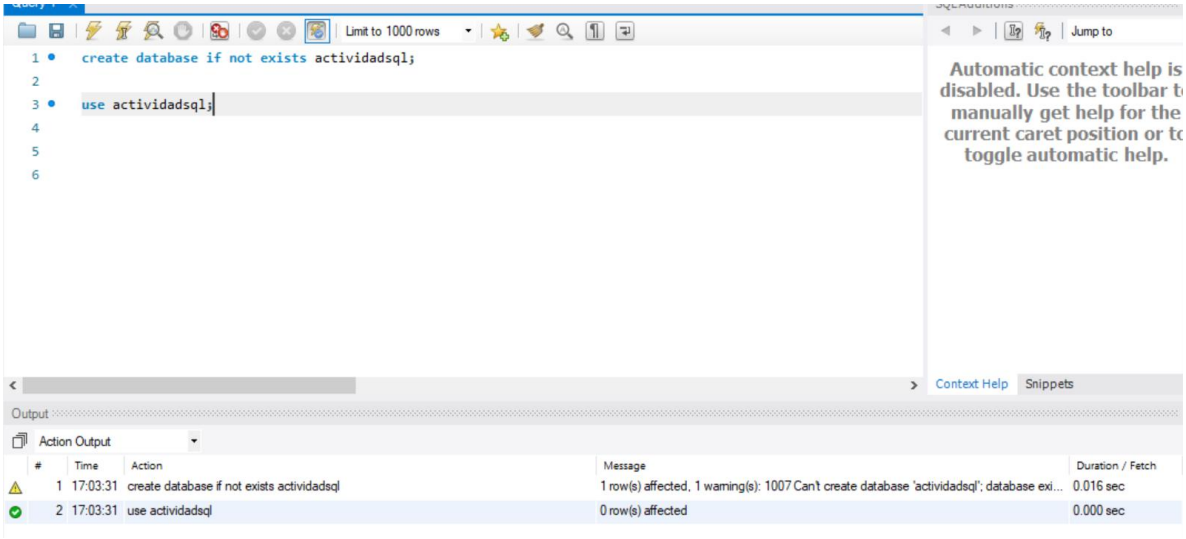
Appendix:

Creating an user:

```
-- Drop user first if they exist

DROP USER if exists 'actividadsql'@'%' ;


-- Now create user with prop privileges

CREATE USER 'actividadsql'@'%' IDENTIFIED BY 'xideral';


GRANT ALL PRIVILEGES ON * . * TO 'actividadsql'@'%';
```

Creating a data base:

```
create database if not exists actividadsql;


use actividadsql;
```



Creating a table and adding age to it

```
USE actividadsql;
drop table if exists activitysql;

CREATE TABLE `activitysql` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `last_name` varchar(64) DEFAULT NULL,
  `first_name` varchar(64) DEFAULT NULL,
  `email` varchar(64) DEFAULT NULL,
  `department` varchar(64) DEFAULT NULL,
  `salary` DECIMAL(10,2) DEFAULT NULL,
  `age` INTEGER(100) DEFAULT NULL,

  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;

INSERT INTO `activitysql` (`id`,`last_name`,`first_name`,`email`, `department`, `salary`,`age`) VALUES (1,'Doe','John','john.doe@foo.com', 'HR', 55000.00, 33);
```

INSERT INTO `activitysql` (`id`,`last_name`,`first_name`,`email`, `department`, `salary`, `age`) VALUES (2,'Public','Mary','mary.public@foo.com', 'Engineering', 75000.00, 34);

INSERT INTO `activitysql` (`id`,`last_name`,`first_name`,`email`, `department`, `salary`, `age`) VALUES (3,'Queue','Susan','susan.queue@foo.com', 'Legal', 130000.00, 24);

INSERT INTO `activitysql` (`id`,`last_name`,`first_name`,`email`, `department`, `salary`, `age`) VALUES (4,'Williams','David','david.williams@foo.com', 'HR', 120000.00, 18);

INSERT INTO `activitysql` (`id`,`last_name`,`first_name`,`email`, `department`, `salary`, `age`) VALUES (5,'Johnson','Lisa','lisa.johnson@foo.com', 'Engineering', 50000.00, 43);

INSERT INTO `activitysql` (`id`,`last_name`,`first_name`,`email`, `department`, `salary`, `age`) VALUES (6,'Smith','Paul','paul.smith@foo.com', 'Legal', 100000.00, 52);

Application.properties:

```
spring.datasource.url=jdbc:mysql://localhost:3306/actividadsql
spring.datasource.username=actividadsql
spring.datasource.password=xideral

# Turn off the Spring Boot banner
 spring.main.banner-mode=off

# Reduce logging level. Set logging level to warn
logging.level.root=warn
```

Student.java

```
package com.luv2code.cruddemo.entity;

import jakarta.persistence.*;

@Entity
@Table(name="activitysql")
public class Student {

    // define fields
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    private int id;
```

```java
@Column(name="first_name")
private String firstName;

@Column(name="last_name")
private String lastName;

@Column(name="email")
private String email;

@Column(name="age")
private int age;
// define constructors
public Student() {

}

public Student(String firstName, String lastName, String email, int age) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.email = email;
    this.age = age;
    }

// define getters/setters

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
```

```
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    // define toString() method

    @Override
    public String toString() {
        return "Trabajdor {" +
                "id=" + id +
                ", firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", email='" + email + '\'' +
                ", Age=" + age +
                '}';
    }

}
```

Student.DAO.java

```
package com.luv2code.cruddemo.dao;

import com.luv2code.cruddemo.entity.Student;

import java.util.List;

public interface StudentDAO {

    void save(Student theStudent);

    Student findById(Integer id);

    List<Student> findAll();

    List<Student> findByLastName(String theLastName);

    void update(Student theStudent);

    void delete(Integer id);

    int deleteAll();
}
```

StudentDAOImpl.java

```
package com.luv2code.cruddemo.dao;
```

```java
import com.luv2code.cruddemo.entity.Student;
import jakarta.persistence.EntityManager;
import jakarta.persistence.TypedQuery;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Repository
public class StudentDAOImpl implements StudentDAO {

    // define field for entity manager
    private EntityManager entityManager;

    // inject entity manager using constructor injection
    @Autowired
    public StudentDAOImpl(EntityManager entityManager) {
        this.entityManager = entityManager;
    }

    // implement save method
    @Override
    @Transactional
    public void save(Student theStudent) {
       System.out.println("Save Estudiante");
        entityManager.persist(theStudent); //<==JPA
    }

    @Override
    public Student findById(Integer id) {
        return entityManager.find(Student.class, id);//<==JPA
    }

    @Override
    public List<Student> findAll() {
        // create query
        TypedQuery<Student> theQuery = entityManager.createQuery("FROM Student",
Student.class);

        // return query results
        return theQuery.getResultList();//<==JPA
    }

    @Override
    public List<Student> findByLastName(String theLastName) {
        // create query
        TypedQuery<Student> theQuery = entityManager.createQuery(
                                        "FROM Student WHERE lastName=:theData",
Student.class);

        // set query parameters
        theQuery.setParameter("theData", theLastName);
```

```
        // return query results
        return theQuery.getResultList();
    }

    @Override
    @Transactional
    public void update(Student theStudent) {
        entityManager.merge(theStudent); //<==JPA
    }

    @Override
    @Transactional
    public void delete(Integer id) {

        // retrieve the student
        Student theStudent = entityManager.find(Student.class, id);

        // delete the student
        entityManager.remove(theStudent);//<==JPA
    }

    @Override
    @Transactional
    public int deleteAll() {

        int numRowsDeleted = entityManager.createQuery("DELETE FROM
Student").executeUpdate(); //<==JPA

        return numRowsDeleted;
    }
}
```

Cruddemoapplication.java

```
package com.luv2code.cruddemo;

import com.luv2code.cruddemo.dao.StudentDAO;
import com.luv2code.cruddemo.entity.Student;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

import java.util.List;

@SpringBootApplication
```

```java
public class CruddemoApplication {

	public static void main(String[] args) {
		SpringApplication.run(CruddemoApplication.class, args);
	}

	@Bean
	public CommandLineRunner commandLineRunner(StudentDAO studentDAO) {

		return runner -> {
			createStudent(studentDAO);

			createMultipleStudents(studentDAO);

			readStudent(studentDAO);

			queryForStudents(studentDAO);

			queryForStudentsByLastName(studentDAO);

//			updateStudent(studentDAO);

			deleteStudent(studentDAO);

			//deleteAllStudents(studentDAO);
		};
	}

	private void deleteAllStudents(StudentDAO studentDAO) {

		System.out.println("Deleting all students");
		int numRowsDeleted = studentDAO.deleteAll();
		System.out.println("Deleted row count: " + numRowsDeleted);
	}

	private void deleteStudent(StudentDAO studentDAO) {

		int studentId = 15;
		System.out.println("Deleting student id: " + studentId);
		studentDAO.delete(studentId);
	}

//	private void updateStudent(StudentDAO studentDAO) {
//
//		// retrieve student based on the id: primary key
//		int studentId = 12;
//		System.out.println("Getting student with id: " + studentId);
//		Student myStudent = studentDAO.findById(studentId);
//
//		// change first name to "John"
//		System.out.println("Updating student ...");
//		myStudent.setFirstName("John");
//
//		// update the student
//		studentDAO.update(myStudent);
```

```
//
//			// display the updated student
//			System.out.println("Updated student: " + myStudent);
//		}

		private void queryForStudentsByLastName(StudentDAO studentDAO) {

			// get a list of students
			List<Student> theStudents = studentDAO.findByLastName("Doe");

			// display list of students
			for (Student tempStudent : theStudents) {
				System.out.println(tempStudent);
			}
		}

		private void queryForStudents(StudentDAO studentDAO) {

			// get a list of students
			List<Student> theStudents = studentDAO.findAll();

			// display list of students
			for (Student tempStudent : theStudents) {
				System.out.println(tempStudent);
			}
		}

		private void readStudent(StudentDAO studentDAO) {

			// create  a student object
			System.out.println("Creating new student object ...");
			Student tempStudent = new Student("Epeneto", "Duck",
"daffy@luv2code.com", 24);

			// save the student
			System.out.println("Saving the student ...");
			studentDAO.save(tempStudent);

			// display id of the saved student
			int theId = tempStudent.getId();
			System.out.println("Saved student. Generated id: " + theId);

			// retrieve student based on the id: primary key
			System.out.println("Retrieving student with id: " + theId);
			Student myStudent = studentDAO.findById(theId);

			// display student
			System.out.println("Found the student: " + myStudent);
		}

		private void createMultipleStudents(StudentDAO studentDAO) {

			// create multiple students
			System.out.println("Creating 3 student objects ...");
```

```
            Student tempStudent1 = new Student("John", "Doe",
"john@luv2code.com", 30);
            Student tempStudent2 = new Student("Mary", "Public",
"mary@luv2code.com",22);
            Student tempStudent3 = new Student("Bonita", "Applebum",
"bonita@luv2code.com",34);

            // save the student objects
            System.out.println("Saving the students ...");
            studentDAO.save(tempStudent1);
            studentDAO.save(tempStudent2);
            studentDAO.save(tempStudent3);
      }

      private void createStudent(StudentDAO studentDAO) {

            // create the student object
            System.out.println("Creating new student object ...");
            Student tempStudent = new Student("Pedro", "Doe",
"pedro@luv2code.com", 39);

            // save the student object
            System.out.println("Saving the student ...");
            studentDAO.save(tempStudent);

            // display id of the saved student
            System.out.println("Saved student. Generated id: " +
tempStudent.getId());
      }
}
```

Updated table after junning the Java Script

id	last_name	first_name	email	department	salary	age
1	Doe	John	john.doe@foo.com	HR	55000.00	33
2	Public	Mary	mary.public@foo.com	Engineering	75000.00	34
3	Queue	Susan	susan.queue@foo.com	Legal	130000.00	24
4	Williams	David	david.williams@foo.com	HR	120000.00	18
5	Johnson	Lisa	lisa.johnson@foo.com	Engineering	50000.00	43
6	Smith	Paul	paul.smith@foo.com	Legal	100000.00	52
7	Doe	Pedro	pedro@luv2code.com	NULL	NULL	39
8	Doe	John	john@luv2code.com	NULL	NULL	30
9	Public	Mary	mary@luv2code.com	NULL	NULL	22
10	Applebum	Bonita	bonita@luv2code.com	NULL	NULL	34
11	Duck	Epeneto	daffy@luv2code.com	NULL	NULL	24
12	Doe	Pedro	pedro@luv2code.com	NULL	NULL	39
13	Doe	John	john@luv2code.com	NULL	NULL	30
14	Public	Mary	mary@luv2code.com	NULL	NULL	22
16	Duck	Epeneto	daffy@luv2code.com	NULL	NULL	24

Java Print

```
Save Estudiante
Creating new student object ...
Saving the student ...
Save Estudiante
Saved student. Generated id: 16
Retrieving student with id: 16
Found the student: Trabajdor {id=16, firstName='Epeneto', lastName='Duck', email='daffy@luv2code.com', Age=24}
Trabajdor {id=1, firstName='John', lastName='Doe', email='john.doe@foo.com', Age=33}
Trabajdor {id=2, firstName='Mary', lastName='Public', email='mary.public@foo.com', Age=34}
Trabajdor {id=3, firstName='Susan', lastName='Queue', email='susan.queue@foo.com', Age=24}
Trabajdor {id=4, firstName='David', lastName='Williams', email='david.williams@foo.com', Age=18}
Trabajdor {id=5, firstName='Lisa', lastName='Johnson', email='lisa.johnson@foo.com', Age=43}
Trabajdor {id=6, firstName='Paul', lastName='Smith', email='paul.smith@foo.com', Age=52}
Trabajdor {id=7, firstName='Pedro', lastName='Doe', email='pedro@luv2code.com', Age=39}
Trabajdor {id=8, firstName='John', lastName='Doe', email='john@luv2code.com', Age=30}
Trabajdor {id=9, firstName='Mary', lastName='Public', email='mary@luv2code.com', Age=22}
Trabajdor {id=10, firstName='Bonita', lastName='Applebum', email='bonita@luv2code.com', Age=34}
Trabajdor {id=11, firstName='Epeneto', lastName='Duck', email='daffy@luv2code.com', Age=24}
Trabajdor {id=12, firstName='Pedro', lastName='Doe', email='pedro@luv2code.com', Age=39}
Trabajdor {id=13, firstName='John', lastName='Doe', email='john@luv2code.com', Age=30}
Trabajdor {id=14, firstName='Mary', lastName='Public', email='mary@luv2code.com', Age=22}
Trabajdor {id=15, firstName='Bonita', lastName='Applebum', email='bonita@luv2code.com', Age=34}
Trabajdor {id=16, firstName='Epeneto', lastName='Duck', email='daffy@luv2code.com', Age=24}
Trabajdor {id=1, firstName='John', lastName='Doe', email='john.doe@foo.com', Age=33}
Trabajdor {id=7, firstName='Pedro', lastName='Doe', email='pedro@luv2code.com', Age=39}
Trabajdor {id=8, firstName='John', lastName='Doe', email='john@luv2code.com', Age=30}
Trabajdor {id=12, firstName='Pedro', lastName='Doe', email='pedro@luv2code.com', Age=39}
Trabajdor {id=13, firstName='John', lastName='Doe', email='john@luv2code.com', Age=30}
Deleting student id: 15
```