JDBC DB 검색기능구현 프로젝트

JDBC DB Implementation of search function project

공간정보 아카데미 12기
팀장 이정민

공간정보 아카데미 12기
조원 정준안

공간정보 아카데미 12기
조원 권수현

개요 구조 기술 팀원 분담 느낀점 Q&A

프로젝트 개요 ; Project outline ___

구조; Structure__

기술 ; Description__

팀원 분담; Division of team members__

전체 코드 ; Full code ___

Q&A ; Q&A ___

개요 구조 기술 팀원 분담 느낀점 Q&A

Project outline ___

프로젝트 명

JDBC DB 검색기능구현 프로젝트

개발 일정

25.07.18~25.07.21

기능

직원 이름으로 직원 정보를 검색

입사 년도로 검색

부서 번호로 검색

직무로 검색

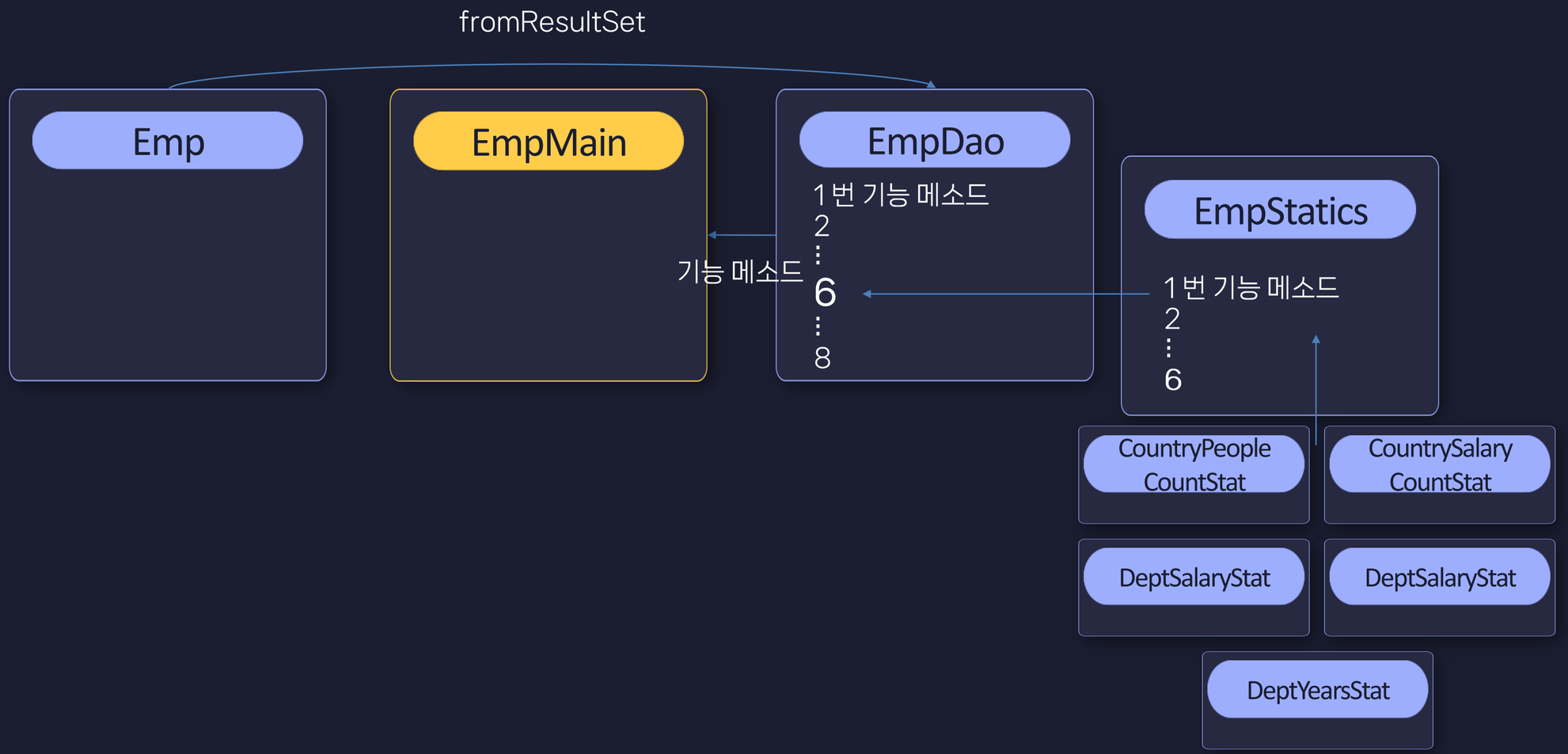
도시 이름으로 검색

통계 자료를 출력

부서장 성으로 부서원 검색

나라 이름으로 그 나라에 근무하는 직원 조회

Structure___


fromResultSet
Emp
EmpMain
EmpDao
1번 기능 메소드
2
6
8
기능 메소드
EmpStatics
1번 기능 메소드
2
6
CountryPeople CountStat
CountrySalary CountStat
DeptSalaryStat
DeptSalaryStat
DeptYearsStat

CountryPeopleCountStat___

```java
package project1_200;
public class CountryPeopleCountStat {
    public String name;
    public int peopleCount;
    @Override
    public String toString() {
        return "국가이름 = " + name + ", 인원수 = " + peopleCount + "명";
    }

    public static CountryPeopleCountStat fromResultSet(java.sql.ResultSet rs)
    throws java.sql.SQLException {
        CountryPeopleCountStat stat = new CountryPeopleCountStat();
        stat.name = rs.getString("country_id");
        stat.peopleCount = rs.getInt("count");
        return stat;
    }
}
```

CountrySalaryStat__

```java
package project1_200;
public class CountrySalaryStat {
    public String name;
    public double avgSalary;
    @Override
    public String toString() {
        return "국가이름 = " + name + ", 평균연봉 = " + avgSalary;
    }

    public static CountrySalaryStat fromResultSet(java.sql.ResultSet rs)
    throws java.sql.SQLException {
        CountrySalaryStat stat = new CountrySalaryStat();
        stat.name = rs.getString("country_id");
        stat.avgSalary = rs.getDouble("avg_salary");
        return stat;
    }
}
```

DeptSalaryMinMaxStat___

```java
package project1_200;
public class DeptSalaryMinMaxStat {
    public int id;
    public String name;
    public int maxSalary;
    public String maxEmpName;
    public int minSalary;
    public String minEmpName;

    @Override
    public String toString() {
        return "부서ID: " + id + ", 부서명: " + name + ", 최고연봉: " + maxSalary +
        "(" + maxEmpName + ")" + ", 최저연봉: " + minSalary + "(" + minEmpName + ")";
    }

    public static DeptSalaryMinMaxStat fromResultSet(java.sql.ResultSet rs)
    throws java.sql.SQLException {
        DeptSalaryMinMaxStat stat = new DeptSalaryMinMaxStat();
        stat.id = rs.getInt("department_id");
        stat.name = rs.getString("department_name");
        stat.maxSalary = rs.getInt("max_salary");
        stat.maxEmpName = rs.getString("max_emp_name");
        stat.minSalary = rs.getInt("min_salary");
        stat.minEmpName = rs.getString("min_emp_name");
        return stat;
    }
}
```

DeptSalaryStat__

```java
package project1_200;
public class DeptSalaryStat {
    public int id;
    public String name;
    public double avgSalary;

    @Override
    public String toString() {
        return "Department ID: " + id + ", Department Name: " + name +
        ", Average Salary: " + String.format("%.2f", avgSalary);
    }

    public static DeptSalaryStat fromResultSet(java.sql.ResultSet rs)
    throws java.sql.SQLException {
        DeptSalaryStat stat = new DeptSalaryStat();
        stat.id = rs.getInt("department_id");
        stat.name = rs.getString("department_name");
        stat.avgSalary = rs.getDouble("avg_salary");
        return stat;
    }
}
```

DeptYearsStat___

```java
package project1_200;
public class DeptYearsStat {
    public int id;
    public String name;
    public double avgYears;

    @Override
    public String toString() {
        return "Department ID: " + id + ", Department Name: " + name +
        ", Average Years: " + String.format("%.2f", avgYears);
    }

    public static DeptYearsStat fromResultSet(java.sql.ResultSet rs)
    throws java.sql.SQLException {
        DeptYearsStat stat = new DeptYearsStat();
        stat.id = rs.getInt("department_id");
        stat.name = rs.getString("department_name");
        stat.avgYears = rs.getDouble("avg_years");
        return stat;
    }
}
```

EmpStatistics___

```java
public class EmpStatistics {
    public static void showStatisticsMenu() {
        Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.println("==== 통계 자료 ====");
            System.out.println("1. 부서별 평균 연봉");
            System.out.println("2. 부서별 평균 근속연수");
            System.out.println("3. 국가별 평균 연봉");
            System.out.println("4. 국가별 직원수");
            System.out.println("5. 부서별 최고/최저 연봉 직원 정보");
            System.out.println("6. 근속연수 Top 5 직원 랭킹");
            System.out.println("0. 이전 메뉴로");
            System.out.print("메뉴 선택: ");
            try {
                int menu = sc.nextInt();
                switch (menu) {
                    case 1:
                        printDeptAvgSalary();
                        break;
                    case 2:
                        printDeptAvgYears();
                        break;
                    case 3:
                        printCountryAvgSalary();
                        break;
```

```java
                    case 4:
                        printCountryPeopleCount();
                        break;
                    case 5:
                        printDeptSalaryExtremes();
                        break;
                    case 6:
                        printTop5LongestEmployees();
                        break;
                    case 0:
                        return;
                    default:
                        System.out.println("잘못된 입력입니다.");
                }
            } catch (java.util.InputMismatchException e) {
                System.out.println("정수를 입력하세요.");
                sc.nextLine();
            }
        }
    }
```

메소드

- 부서별 평균 연봉 출력 함수 : printDeptAvgSalary()
- 부서별 평균 근속 연수 출력 함수 : printDeptAvgYears()
- 국가별 평균 연봉 출력 함수 : printCountryAvgSalary()
- 국가별 직원수 출력 함수 : printCountryPeopelCount()
- 부서별 최고/최저 연봉 직원 출력 함수 : printDeptSalaryExtremes()
- 근속연수 Top 5 직원 랭킹 출력 함수 : printTop5LongestEmployees()

EmpStatistics___

```java
public class EmpStatistics {
    public static void showStatisticsMenu() {
        Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.println("==== 통계 자료 ====");
            System.out.println("1. 부서별 평균 연봉");
            System.out.println("2. 부서별 평균 근속연수");
            System.out.println("3. 국가별 평균 연봉");
            System.out.println("4. 국가별 직원수");
            System.out.println("5. 부서별 최고/최저 연봉 직원 정보");
            System.out.println("6. 근속연수 Top 5 직원 랭킹");
            System.out.println("0. 이전 메뉴로");
            System.out.print("메뉴 선택: ");
            try {
                int menu = sc.nextInt();
                switch (menu) {
                    case 1:
                        printDeptAvgSalary();
                        break;
                    case 2:
                        printDeptAvgYears();
                        break;
                    case 3:
                        printCountryAvgSalary();
                        break;
```

- 메소드화 하여 간결하게 보이게 하기

```java
                    case 4:
                        printCountryPeopleCount();
                        break;
                    case 5:
                        printDeptSalaryExtremes();
                        break;
                    case 6:
                        printTop5LongestEmployees();
                        break;
                    case 0:
                        return;
                    default:
                        System.out.println("잘못된 입력입니다.");
                }
            } catch (java.util.InputMismatchException e) {
                System.out.println("정수를 입력하세요.");
                sc.nextLine();
            }
        }
    }
```

- 문자열로 잘못 입력할 경우 방지

EmpStatistics___

```java
// 부서별 최고/최저 연봉 직원 정보
    private static void printDeptSalaryExtremes() {
        try {
            List<DeptSalaryMinMaxStat> stats = EmpDAO.getDeptSalaryExtremeStats();
            System.out.println("[부서별 최고/최저 연봉 직원 정보]");
            for (DeptSalaryMinMaxStat stat : stats) {
                System.out.println(stat);
            }
        } catch (Exception e) {
            System.out.println("오류 발생: " + e.getMessage());
        }
    }
```

- 배열의 정보
  하나씩 출력하기

Emp___

```java
package project1_200;
import java.sql.ResultSet;
import java.sql.SQLException;

public class Emp {

    public int id;
    public String name;
    public String email;
    public String phone;
    public String hireDate;
    public int salary;
    public String deptName;
    public double commissionPct;
    public int years;

    @Override
    public String toString() {
        return "Emp [사번: " + id + ", 이름: " + name + ", 이메일: " + email + ", 전화번호: "
                + phone + ", 입사일: " + hireDate + ", 연봉: " + salary + ", 부서명: " + deptName + "]";
    }

    public static Emp fromResultSet(ResultSet rs) throws SQLException {
        Emp emp = new Emp();
        emp.id = rs.getInt(1);
        emp.name = rs.getString(2);
        emp.email = rs.getString(3);
        emp.phone = rs.getString(4);
        emp.hireDate = rs.getString(5);
        emp.salary = rs.getInt(6);
        emp.deptName = rs.getString(7);
        emp.commissionPct = rs.getDouble(8); // 추가
        return emp;
    }
    // 근속연수 Top 5 직원 랭킹 조회 시 사용(years 변수 포함됨됨)
    public static Emp fromResultSet(ResultSet rs, boolean Years) throws SQLException {
        Emp emp = fromResultSet(rs);
        if (Years) {
            emp.years = rs.getInt(9);
        }
        return emp;
    }
}
```

EmpDao___

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;
import java.io.FileReader;

public class EmpDAO { // DAO
    private static Properties props = new Properties();
    static {
        try {
            props.load(new FileReader("db-info.properties"));
        } catch (Exception e) {
            throw new RuntimeException("db-info.properties 파일을 읽는 중 오류 발생", e);
        }
    }
  // 1
  public static List<Emp> getEmpListByDeptNo(int deptNo) throws Exception {
    List<Emp> result = new ArrayList<Emp>();
    Class.forName(props.getProperty("driverClassName"));
    String sql = "select e.employee_id, CONCAT(e.first_name, ' ', e.last_name) as name,
    e.email, e.phone_number, e.hire_date, e.salary, d.department_name, e.commission_pct "
        + "from employees e join departments d " + "on e.department_id = d.department_id
        + "where d.department_id = ?";
    try (Connection con = DriverManager.getConnection(props.getProperty("url"),
    props.getProperty("userName"), props.getProperty("password"));
        PreparedStatement stmt = con.prepareStatement(sql)) {
        while (rs.next()) {
          result.add(Emp.fromResultSet(rs));
        }
      }
    }
    return result;
  }
  }
```

java.sql 패키지에 있는 Connection이라는 클래스를 불러온다.

Properties 클래스에서 객체를 저장하는 props라는 변수 정의

"db-info.properties" text file에서 FileReader 클래스를 활용해 정보를 불러온다.

만약에 오류가 발생하면 파일을 읽는 중 오류 발생이라는 메시지를 보낸다.

부서번호로 직원정보를 불러오는 sql문을 작성한다.

main함수에서 실행한 함수searchByDeptNo()를 실행하면 EmpDAO에서 getEmpListByDeptNo() 함수가 실행된다.

con객체로부터 작성한 sql문을 준비합니다. PreparedStatement의 stmt객체 생성

Class라는 클래스의 forName이라는 메서드가 실행되어 정보가 로딩된 props에서 "driverClassName"의 value값을 가져온다.

Emp클래스의 fromResultSet함수를 호출하여 result리스트에 값을 넣는다.

EmpMain___

```java
public class EmpMain {
  public static void main(String[] args) throws Exception {

    Scanner sc = new Scanner(System.in);

    while (true) {
            System.out.println("==== 직원 관리 시스템 ====");
            System.out.println("1. 부서 번호로 검색");
            System.out.println("2. 직원 이름으로 검색");
            System.out.println("3. 입사 년도로 검색");
            System.out.println("4. 직무로 검색");
            System.out.println("5. 도시 이름으로 검색");
            System.out.println("6. 통계자료");
            System.out.println("7. 부서장 성으로 부서원 검색");
            System.out.println("8. 나라 이름으로 근무하는 직원 검색");
            System.out.println("0. 종료");
            System.out.print("메뉴 선택: ");
            String menuInput = sc.nextLine();
            int menu;
            try {
                menu = Integer.parseInt(menuInput);
            } catch (NumberFormatException e) {
                System.out.println("숫자를 입력하세요.");
                continue;
```

입력값을 String으로 받아주고
이를 Integer.parseInt()
함수를 통해 정수로 바꾸는
단계를 추가하여 정수가 아닌
입력 값에 대해 오류 유도

사용 변수 및 메서드 명

- searchByDeptNo()
- searchByEmpName()
- searchByHireYear()
- searchByJobId()
- searchByCityName()
- searchByManagerLastName()
- searchBycountryName()
- showStatisticsMenu()

EmpMain___

```
    }
    switch (menu) {
      case 1:
        searchByDeptNo(sc);
        break;
      case 2:
        searchByEmpName(sc);
        break;
      case 3:
        searchByHireYear(sc);
        break;
      case 4:
        searchByJobId(sc);
        break;
      case 5:
        searchByCityName(sc);
        break;
      case 6:
        EmpStatistics.showStatisticsMenu();
        break;
```

```
      case 7:
        searchByManagerLastName(sc);
        break;
      case 8:
        searchByCountryName(sc);
        break;
      case 0:
        System.out.println("프로그램을 종료합니다.");
        return;
      default:
       System.out.println("잘못된 입력입니다.");
    }
    }
```

정해진 case 번호 외에 정수 값을 받을 시 default를 통해 오류를 처리합니다.

EmpMain___

```
private static void searchByDeptNo(Scanner sc) {
        while (true) {
            System.out.println("부서 번호를 입력하세요");
            try {
                String input = sc.nextLine();
                if (input.trim().isEmpty()) {
                    System.out.println("공백을 입력하여 오류가 발생했습니다. 부서 번호
                    continue;
                }
                int deptNo = Integer.parseInt(input);
                List<Emp> result = EmpDAO.getEmpListByDeptNo(deptNo);
                if (result.isEmpty()) {
                    System.out.println("해당하는 부서가 없습니다.");
                } else {
                    for (Emp emp : result) {
                        System.out.println(emp);
                    }
                }
                break;
            } catch (NumberFormatException e) {
                System.out.println("정수가 아닌 값을 입력하여 오류가 발생했습니다. 부
            } catch (Exception e) {
                System.out.println("알 수 없는 오류가 발생했습니다: " + e.getMessage(
            }
        }
    }
```

각각의 메서드에서 발생하는 오류는 try – catch 문을 통해 예외 처리하였으며 이때 while 문을 사용하여 오류가 발생하여 어플리케이션이 비정상적으로 종료되는 것을 해결했습니다.

부서 번호 입력 시 공백을 입력하게 되면 오류가 발생하여 이를 isEmpty() 메서드를 사용하여 처리하였습니다. 이때 isEmpty() 경우 ""는 true를 반환하나 " " 은 false 값을 반환하므로 trim()메서드를 사용하여 앞뒤의 공백을 먼저 제거해줬습니다.

정수가 아닌 String, float 등의 값을 입력하면 예외 처리하여 재입력하도록 유도합니다.

Division of team members ___

팀장

이정민

ㄴ 부서장 성으로 직원 정보 검색기능 구현

ㄴ 통계 기능 구현(부서별 평균, 연봉, 부서별 평균 근속연수, 부서별 최고/최저 연봉 직원, 근속 연수 Top 5 직원 랭킹

ㄴ 코드 총괄(메서드화, 클래스화, 오류코드 수정)

팀원

권수현

ㄴ 직무,도시 이름, 나라 이름으로 직원 정보 검색 기능 구현

ㄴ main 오류 코드 수정 (적절한 값을 입력하지 않았을 때 나타나는 오류 수정)

ㄴ PPT 제작

팀원

정준안

ㄴ 직원 이름, 입사 년도, 부서 번호로 직원 정보 검색 기능 구현

ㄴ 통계 기능 구현(국가별 연봉, 국가별 직원수)

ㄴ db-info.properties의 정보 이용하여 연결 구현

Lessons learned ___

팀장

이정민

이번 프로젝트에서 팀장을 맡아 과제를 진행하면서, 팀원들과의 협업이 얼마나 중요한지 다시 한 번 느꼈습니다. 모두가 원하는 방향을 찾고, 팀원 각자의 의견을 조율하는 과정은 앞으로 있을 모든 프로젝트의 기본임을 잊지않고 가져갈 것입니다. 혼자 코딩할 때와 달리 함께 고민하고 해결책을 찾아가는 과정이 정말 즐겁고 뜻깊었으며, 매우 짧은 시간이었지만 모두가 크게 성장할 수 있는 의미있는 시간이 되지 않았나 생각이 듭니다.

팀원

권수현

처음에 구조를 잡고 기능을 구현하려고 했지만 기능에 대한 이해가 적었기 때문에 시행착오를 겪었다. 결국, 코드를 작성한 뒤 구조를 구체화 시키는 방식으로 전환했다. 또한 똑같은 기능의 코드를 짜도 방식이 다르다는 점을 느끼고 많은 사람들의 코드를 봐야겠다고 느꼈습니다.

팀원

정준안

이번 프로젝트에서 제가 느낀 점은 앞으로 프로그래밍하면서 매서드화와 클래스화를 위한 체계적인 설계를 잘해야 한다는 점인 것 같습니다..
코드의 길이를 줄이는 설계를 하는 과정에서 많은 어려움을 느꼈지만, 동료의 도움으로 잘 극복할 수 있었던 것 같고, 부족한 저를 끌고 간 팀원에 감사함을 표합니다.
마지막으로 프로젝트를 이대로 끝내지 않고 혼자 A - Z까지 다시 설계하는 시간을 가지며, 한 단계 성장할 수 있는 발판을 마련해야겠습니다.

Q&A ___

자유롭게 질문해주세요!