레코드 한 개 조회하기

2019-11-11

이승진

목차

[1. SQL 2](#_Toc24479663)

[2. StudentDAO 구현 2](#_Toc24479664)

[1) StudentDAO.java #1 2](#_Toc24479665)

[2) StudentDAO1.java 5](#_Toc24479666)

[3) StudentDAO2.java 7](#_Toc24479667)

[3. student1.jsp 구현 9](#_Toc24479668)

[4. query string과 request parameter 11](#_Toc24479669)

[1) query string 11](#_Toc24479670)

[2) request parameter 11](#_Toc24479671)

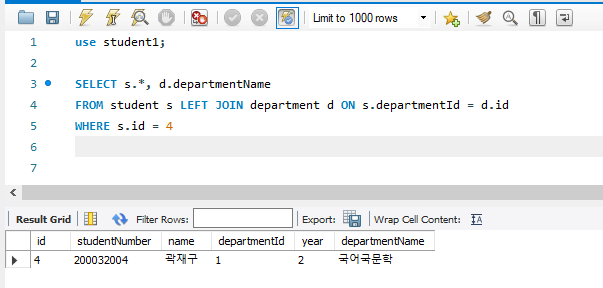
[5. student2.jsp 구현 12](#_Toc24479672)

[1) 실행 에러 13](#_Toc24479673)

[2) 정상적인 실행 14](#_Toc24479674)

[6. 실습 15](#_Toc24479675)

# SQL



|  |
| --- |
| use student1;  SELECT s.\*, d.departmentName  FROM student s LEFT JOIN department d ON s.departmentId = d.id  WHERE s.id = 4 |

# StudentDAO 구현

## StudentDAO.java #1

src/lecture1/jdbc1/StudentDAO.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80 | package lecture1.jdbc1;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.util.ArrayList;  import java.util.List;  import lecture1.DB;  public class StudentDAO {  public static List<Student> findAll() throws Exception {  String sql = "SELECT s.\*, d.departmentName " +  "FROM student s LEFT JOIN department d ON s.departmentId = d.id";  try (Connection connection = DB.getConnection("student1");  PreparedStatement statement = connection.prepareStatement(sql);  ResultSet resultSet = statement.executeQuery()) {  ArrayList<Student> list = new ArrayList<Student>();  while (resultSet.next()) {  Student student = new Student();  student.setId(resultSet.getInt("id"));  student.setStudentNumber(resultSet.getString("studentNumber"));  student.setName(resultSet.getString("name"));  student.setDepartmentId(resultSet.getInt("departmentId"));  student.setYear(resultSet.getInt("year"));  student.setDepartmentName(resultSet.getString("departmentName"));  list.add(student);  }  return list;  }  }  public static List<Student> findByName(String name) throws Exception {  String sql = "SELECT s.\*, d.departmentName " +  "FROM student s LEFT JOIN department d ON s.departmentId = d.id " +  "WHERE s.name LIKE ?";  try (Connection connection = DB.getConnection("student1");  PreparedStatement statement = connection.prepareStatement(sql)) {  statement.setString(1, name + "%");  try (ResultSet resultSet = statement.executeQuery()) {  ArrayList<Student> list = new ArrayList<Student>();  while (resultSet.next()) {  Student student = new Student();  student.setId(resultSet.getInt("id"));  student.setStudentNumber(resultSet.getString("studentNumber"));  student.setName(resultSet.getString("name"));  student.setDepartmentId(resultSet.getInt("departmentId"));  student.setYear(resultSet.getInt("year"));  student.setDepartmentName(resultSet.getString("departmentName"));  list.add(student);  }  return list;  }  }  }  public static Student findById(int id) throws Exception {  String sql = "SELECT s.\*, d.departmentName " +  "FROM student s LEFT JOIN department d ON s.departmentId = d.id " +  "WHERE s.id = ?";  try (Connection connection = DB.getConnection("student1");  PreparedStatement statement = connection.prepareStatement(sql)) {  statement.setInt(1, id);  try (ResultSet resultSet = statement.executeQuery()) {  if (resultSet.next()) {  Student student = new Student();  student.setId(resultSet.getInt("id"));  student.setStudentNumber(resultSet.getString("studentNumber"));  student.setName(resultSet.getString("name"));  student.setDepartmentId(resultSet.getInt("departmentId"));  student.setYear(resultSet.getInt("year"));  student.setDepartmentName(resultSet.getString("departmentName"));  return student;  } else  return null;  }  }  }  } |

### 조회 메소드의 리턴 타입

findAll 메소드와 findByName 메소드는 Student 객체 목록을 조회한다.

그래서 이 메소드는 Student 객체 목록(ArrayList<Student>)을 리턴한다.

findById 메소드는 Student 객체 한 개를 조회한다.

(조회조건인 id 필드가 기본키(primary key) 이기 때문이다.)

그래서 이 메소드는 Student 객체를 한 개만 리턴한다.

### 조회 결과 레코드 여러개 꺼내기

findAll 메소드와 findByName 메소드는 Student 객체 목록을 조회한다.

그래서 조회 결과(ResultSet)에서 학생 레코드를 여러 개 반복해서 꺼내야 한다.

while (resultSet.next()) {

.... 조회 결과 레코드 꺼내기 ...

}

### 조회 결과 레코드 한 개 꺼내기

findById 메소드는 Student 객체 한 개를 조회한다.

그래서 조회 결과(ResultSet)에서 학생 레코드를 한 개만 꺼낸다.

if (resultSet.next()) {

.... 조회 결과 레코드 꺼내기 ...

}

### SQL 명령 조회 결과 레코드가 없을 때

DB 테이블에 레코드가 전혀 없거나, 조회 조건에 부합하는 레코드가 없으면,

조회 결과 ResultSet 객체에 담긴 데이터는 없다.

이 경우에, findAll 메소드와 findByName 메소드의

while (resultSet.next()) 조건식은 처음부터 false 이다.

그래서 결국 이 두 메소드는,

내부에 아무것도 들어있지 않은, 빈 ArrayList<Student> 객체를 리턴하게 된다.

조회 결과 ResultSet 객체에 담긴 데이테가 없을 때,

findById 메소드의

if (resultSet.next()) 조건식은 false 이다.

그래서 결국 findById 메소드는 null 값을 리턴하게 된다. (줄76)

### 코드 중복

findAll 메소드, findByName 메소드, findByID 메소드에 공통 부분이 있다.

여러 곳에 동일한 소스 코드가 반복된다면, 그 코드를 메소드로 추출하는 것이 좋다.

중복되는 코드

|  |
| --- |
| Student student = new Student();  student.setId(resultSet.getInt("id"));  student.setStudentNumber(resultSet.getString("studentNumber"));  student.setName(resultSet.getString("name"));  student.setDepartmentId(resultSet.getInt("departmentId"));  student.setYear(resultSet.getInt("year"));  student.setDepartmentName(resultSet.getString("departmentName")); |

## StudentDAO1.java

src/lecture1/jdbc1/StudentDAO1.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74 | package lecture1.jdbc1;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.ArrayList;  import java.util.List;  import lecture1.DB;  public class StudentDAO1 {  private static Student getStudentFrom(ResultSet resultSet) throws SQLException {  Student student = new Student();  student.setId(resultSet.getInt("id"));  student.setStudentNumber(resultSet.getString("studentNumber"));  student.setName(resultSet.getString("name"));  student.setDepartmentId(resultSet.getInt("departmentId"));  student.setYear(resultSet.getInt("year"));  student.setDepartmentName(resultSet.getString("departmentName"));  return student;  }  public static List<Student> findAll() throws Exception {  String sql = "SELECT s.\*, d.departmentName " +  "FROM student s LEFT JOIN department d ON s.departmentId = d.id";  try (Connection connection = DB.getConnection("student1");  PreparedStatement statement = connection.prepareStatement(sql);  ResultSet resultSet = statement.executeQuery()) {  ArrayList<Student> list = new ArrayList<Student>();  while (resultSet.next()) {  Student student = getStudentFrom(resultSet);  list.add(student);  }  return list;  }  }  public static List<Student> findByName(String name) throws Exception {  String sql = "SELECT s.\*, d.departmentName " +  "FROM student s LEFT JOIN department d ON s.departmentId = d.id " +  "WHERE s.name LIKE ?";  try (Connection connection = DB.getConnection("student1");  PreparedStatement statement = connection.prepareStatement(sql)) {  statement.setString(1, name + "%");  try (ResultSet resultSet = statement.executeQuery()) {  ArrayList<Student> list = new ArrayList<Student>();  while (resultSet.next()) {  Student student = getStudentFrom(resultSet);  list.add(student);  }  return list;  }  }  }  public static Student findById(int id) throws Exception {  String sql = "SELECT s.\*, d.departmentName " +  "FROM student s LEFT JOIN department d ON s.departmentId = d.id " +  "WHERE s.id = ?";  try (Connection connection = DB.getConnection("student1");  PreparedStatement statement = connection.prepareStatement(sql)) {  statement.setInt(1, id);  try (ResultSet resultSet = statement.executeQuery()) {  if (resultSet.next()) {  Student student = getStudentFrom(resultSet);  return student;  } else  return null;  }  }  }  } |

findAll 메소드, findByName 메소드, findByID 메소드에 공통 부분을

getStudentFrom 메소드로 추출하였다.

### 좀더 개선

|  |
| --- |
| Student student = getStudentFrom(resultSet);  list.add(student); |

위 코드를 아래와 같이 줄일 수 있다.

|  |
| --- |
| list.add(getStudentFrom(resultSet)); |

## StudentDAO2.java

src/lecture1/jdbc1/StudentDAO2.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69 | package lecture1.jdbc1;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.ArrayList;  import java.util.List;  import lecture1.DB;  public class StudentDAO2 {  private static Student getStudentFrom(ResultSet resultSet) throws SQLException {  Student student = new Student();  student.setId(resultSet.getInt("id"));  student.setStudentNumber(resultSet.getString("studentNumber"));  student.setName(resultSet.getString("name"));  student.setDepartmentId(resultSet.getInt("departmentId"));  student.setYear(resultSet.getInt("year"));  student.setDepartmentName(resultSet.getString("departmentName"));  return student;  }  public static List<Student> findAll() throws Exception {  String sql = "SELECT s.\*, d.departmentName " +  "FROM student s LEFT JOIN department d ON s.departmentId = d.id";  try (Connection connection = DB.getConnection("student1");  PreparedStatement statement = connection.prepareStatement(sql);  ResultSet resultSet = statement.executeQuery()) {  ArrayList<Student> list = new ArrayList<Student>();  while (resultSet.next())  list.add(getStudentFrom(resultSet));  return list;  }  }  public static List<Student> findByName(String name) throws Exception {  String sql = "SELECT s.\*, d.departmentName " +  "FROM student s LEFT JOIN department d ON s.departmentId = d.id " +  "WHERE s.name LIKE ?";  try (Connection connection = DB.getConnection("student1");  PreparedStatement statement = connection.prepareStatement(sql)) {  statement.setString(1, name + "%");  try (ResultSet resultSet = statement.executeQuery()) {  ArrayList<Student> list = new ArrayList<Student>();  while (resultSet.next())  list.add(getStudentFrom(resultSet));  return list;  }  }  }  public static Student findById(int id) throws Exception {  String sql = "SELECT s.\*, d.departmentName " +  "FROM student s LEFT JOIN department d ON s.departmentId = d.id " +  "WHERE s.id = ?";  try (Connection connection = DB.getConnection("student1");  PreparedStatement statement = connection.prepareStatement(sql)) {  statement.setInt(1, id);  try (ResultSet resultSet = statement.executeQuery()) {  if (resultSet.next())  return getStudentFrom(resultSet);  else  return null;  }  }  }  } |

# student1.jsp 구현

WebContent/jdbc1/student1.jsp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ page import="java.util.List, lecture1.jdbc1.\*" %>  <%  int id = 4;  Student student = StudentDAO2.findById(id);  %>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <meta name="viewport" content="width=device-width, initial-scale=1">  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>  <style>  body { font-family: 굴림체; }  table.table { width: 500px; }  table td:nth-child(1) { background-color: #eee; }  </style>  </head>  <body>  <div class="container">  <h1>학생</h1>  <table class="table table-bordered table-condensed">  <tr>  <td>ID</td>  <td><%= student.getId() %></td>  </tr>  <tr>  <td>이름</td>  <td><%= student.getName() %></td>  </tr>  <tr>  <td>학과ID</td>  <td><%= student.getDepartmentId() %></td>  </tr>  <tr>  <td>학과명</td>  <td><%= student.getDepartmentName() %></td>  </tr>  <tr>  <td>학년</td>  <td><%= student.getYear() %></td>  </tr>  </table>  </div>  </body>  </html> |

|  |
| --- |
| <%  int id = 4;  Student student = StudentDAO2.findById(id);  %> |

id 필드 값이 4 인, student 테이블에서 조회한다.

|  |
| --- |
| <%= student.getId() %>  <%= student.getName() %>  <%= student.getDepartmentId() %>  <%= student.getDepartmentName() %>  <%= student.getYear() %> |

조회한 student 값을 출력한다.

# query string과 request parameter

## query string

서버에 요청되는 URL이 아래와 같을 경우에

|  |
| --- |
| http://localhost:8080/lecture1/jdbc1/student2.jsp |

위 URL에서 query string 부분은 없다.

서버에 요청되는 URL이 아래와 같을 경우에

|  |
| --- |
| http://localhost:8080/lecture1/jdbc1/student2.jsp?id=4 |

위 URL에서 query string 부분은 ?id=4 이다.

여기서 id 부분이 데이터의 이름이고, 4 부분이 데이터 값이다.

## request parameter

query string으로 전달된 데이터를, 서버에서 꺼낼 때는, request parameter라고 부른다.

즉 URL에서 query string 부분은, 서버에 request parameter 형태로 전달된다.

### request paramter 꺼내기

|  |
| --- |
| String s = request.getParameter("id"); |

위 코드에서 "id" 부분은, request parameter 데이터의 이름이다.

이 이름은 query string 이름과 일치해야 한다.

그리고 request parameter 데이터의 타입은 언제나 String 이다.

### int 타입으로 변환하기

|  |
| --- |
| String s = request.getParameter("id");  int id = Integer.parseInt(s); |

문자열 s를 int 타입으로 변환하는 코드는 위와 같다.

서버에 요청된 URL이 아래와 같다면,

http://localhost:8080/lecture1/jdbc1/student2.jsp?id=4

위 코드에서 s 변수에 대입되는 값은 "4" 이고,

id 변수에 대입괴는 값은 4 이다.

서버에 요청된 URL이 아래와 같다면,

http://localhost:8080/lecture1/jdbc1/student2.jsp

위 코드에서 s 변수에 대입되는 값은 null 이고, (request parameter가 없으므로)

Integer.parseInt(s) 메소드 호출에서 에러가 발생한다. (NumberFormatException 에러)

# student2.jsp 구현

WebContent/jdbc1/student2.jsp

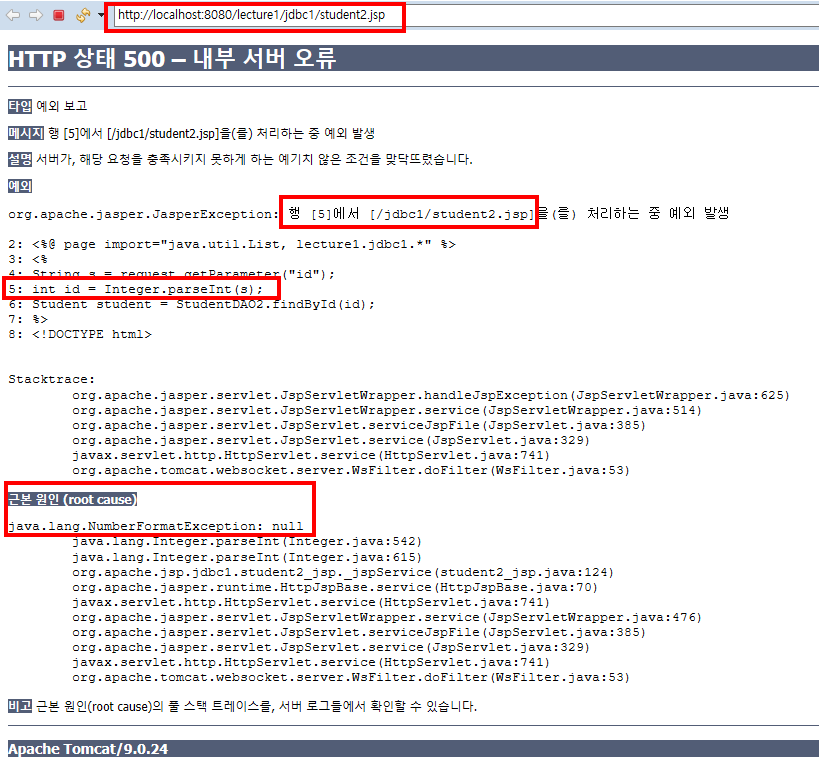
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ page import="java.util.List, lecture1.jdbc1.\*" %>  <%  String s = request.getParameter("id");  int id = Integer.parseInt(s);  Student student = StudentDAO2.findById(id);  %>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <meta name="viewport" content="width=device-width, initial-scale=1">  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>  <style>  body { font-family: 굴림체; }  table.table { width: 500px; }  table td:nth-child(1) { background-color: #eee; }  </style>  </head>  <body>  <div class="container">  <h1>학생</h1>  <table class="table table-bordered table-condensed">  <tr>  <td>ID</td>  <td><%= student.getId() %></td>  </tr>  <tr>  <td>이름</td>  <td><%= student.getName() %></td>  </tr>  <tr>  <td>학과ID</td>  <td><%= student.getDepartmentId() %></td>  </tr>  <tr>  <td>학과명</td>  <td><%= student.getDepartmentName() %></td>  </tr>  <tr>  <td>학년</td>  <td><%= student.getYear() %></td>  </tr>  </table>  </div>  </body>  </html> |

## 실행 에러

student2.jsp 편집창에서 Ctrl+F11 단축키를 눌렀을 때,

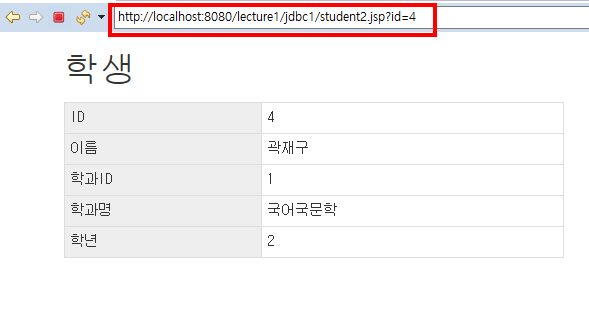
서버에 요청되는 URL은 다음과 같다.

그리과 아래와 같은 에러가 발생한다.



### 위 에러의 원인은 무엇인가?

## 정상적인 실행



URL을 위와 같이 수정하고 실행하면, 정상적으로 실행된다.

그 이유는 무엇인가?

# 실습

user 테이블에서 id 필드 값으로 레코드를 한 개 조회해서 출력하는 페이지를 구현하시오.