**JDBC #6**

2014-12-15

이승진

**학습목표**

JDBC5 폴더의 userList.jsp, userEdit.jsp, userAdd.jsp를 가져다가,   
student2.mdf 데이터베이스의 Student 테이블로 수정하여 구현

# 프로젝트 생성

이클립스 메뉴: File - New - Dynamic Web Project

Project Name: student2

## sqljdbc4.jar 파일 복사

윈도우 탐색기에서 sqljdbc4.jar 파일을 선택하고 Ctrl+C 키를 누름.

이클립스 화면 왼쪽에서 student2 프로젝트 아래의

WecContent/WEB-INF/lib 폴더를 선택하고 Ctrl+V 키를 누름

## studentSelectPage1 저장프로시져 생성

SQL Management Studio에서 student2 데이터베이스를 우클릭하고

메뉴에서 New Query

|  |  |
| --- | --- |
| 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 | CREATE PROCEDURE [dbo].[studentSelectPage1]  @currentPage INT, /\* 현재 페이지 번호 \*/  @pageSize INT /\* 한 페이지의 레코드 수 \*/  AS  BEGIN  SET NOCOUNT ON;  DECLARE @topRecordNo INT, @topRecordID VARCHAR(50)  /\* 화면에 표시할 첫 레코드가 몇번째 레코드인가 계산 \*/  SET @topRecordNo = (@currentPage - 1 ) \* @pageSize + 1  /\* 화면에 표시할 첫 레코드까지만 조회하도록 제한 \*/  SET ROWCOUNT @topRecordNo  /\* 화면에 표시할 첫 레코드의 [ID] 필드값 얻기 \*/  SELECT @topRecordID = [sid]  FROM [Student]  ORDER BY [sid]  /\* 한 페이지만 조회하도록 제한 \*/  SET ROWCOUNT @pageSize  /\* 화면에 표시할 레코드들만 조회 \*/  SELECT \*  FROM [Student]  WHERE [sid] >= @topRecordID  ORDER BY [sid]  END |

Query 창에서 위의 SQL 문장을 입력하고 F5 키를 눌러서 실행한다

테이블명이 [User]에서 [Student]로 수정되었다.

기본키 필드명이 [ID]에서 [sid]로 수정되었다.

기본키 필드 타입이 int에서 문자열 타입으로 바뀌었으므로

기본키를 저장할 지역변수 @topRecordID의 타입이 int에서 VARCHAR(50)으로 수정되었다.

저장프로시져 명은 studentSelectPage1 이다.

# 자바 클래스 생성

## student2 패키지 생성

자바 클래스들을 넣어 놓을 자바 패키지(Java Package)를 먼저 만들자

이클립스 화면 왼쪽에서 student2 프로젝트를 우클릭하고, 메뉴에서 New - Package

Package Name: student2

## DB.java 생성

student2.mdf 데이터베이스에 연결하는데 사용할 DB 클래스를 만들자

이클립스 화면 왼쪽에서 student2 프로젝트를 우클릭하고, 메뉴에서 New - Class

Package: student2

Class Name: DB

### src/student2/DB.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | package student2;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.SQLException;  public class DB {  private static final String JDBC\_DRIVER\_NAME = "com.microsoft.sqlserver.jdbc.SQLServerDriver";  private static final String DB\_URL = "jdbc:sqlserver://127.0.0.1;databaseName=student2";  private static final String USER\_ID="sa";  private static final String USER\_PASSWORD="test123";  public static Connection getConnection() throws ClassNotFoundException, SQLException {  Class.forName(JDBC\_DRIVER\_NAME);  Connection connection = DriverManager.getConnection(DB\_URL, USER\_ID, USER\_PASSWORD);  return connection;  }    } |

노란색으로 칠해진 부분이 수정되었다.

## RequestParameter.java 생성

request 객체에서 데이터를 꺼낼 때 사용할 RequestParameter 클래스를 만들자

이클립스 화면 왼쪽에서 student2 프로젝트를 우클릭하고, 메뉴에서 New - Class

Package: student2

Class Name: RequestParameter

### src/student2/RequestParameter.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 | package student2;  import java.io.UnsupportedEncodingException;  import javax.servlet.http.HttpServletRequest;  public class RequestParameter {    private HttpServletRequest request;    public RequestParameter(HttpServletRequest request) throws UnsupportedEncodingException {  this.request = request;  request.setCharacterEncoding("UTF-8");  }    public String getString(String name, String defaultValue) {  String s = request.getParameter(name);  return s == null ? defaultValue : s.trim();  }    public int getInt(String name, int defaultValue) {  String s = request.getParameter(name);  if (s == null) return defaultValue;  try {  return Integer.parseInt(s);  } catch (NumberFormatException e) {  }  return defaultValue;  }    public int[] getIntValues(String name) {  String[] slist = request.getParameterValues(name);  int[] ilist = new int[slist.length];  for (int i=0; i < slist.length; ++i)  ilist[i] = Integer.parseInt(slist[i]);  return ilist;  }  public String[] getStringValues(String name) {  String[] a = request.getParameterValues(name);  if (a == null) return new String[0];  return a;  }    public boolean getBoolean(String name, boolean defaultValue) {  String s = request.getParameter(name);  return (s == null) ? defaultValue : Boolean.parseBoolean(s);  }  } |

jsp 페이지에서 데이터를 꺼낼 때 request.getParameter("*데이터이름*") 메소드를 사용한다.

혹시 전달된 데이터가 없을 경우에 이 메소드는 null을 리턴한다.

전달된 데이터는 언제나 String 타입의 문자열로 리턴된다.

만약 전달된 데이터가 int, boolean 타입이라면 Integer.parseInt(...), Boolean.parseBoolean(...) 메소드를 사용하여 바꾸어 주어야 한다.

이런 작업을 jsp 페이지에 쉽게 구현할 수 있도록 RequestParameter 클래스의 메소드로 구현하였다.

(줄15) getString 메소드

첫째 파라미터는 꺼낼 데이터 이름이다.

이 값으로 request.getParameter("데이터이름") 메소드를 호출하여 데이터를 꺼낸다.

둘째 파라미터는 전달된 데이터가 없을 때 사용할 디폴트 값이다.

전달된 데이터가 없다면 getString 메소드는 둘째 파라미터의 값을 그대로 리턴한다.

(줄20) getInt 메소드

첫째 파라미터는 꺼낼 데이터 이름이다.

이 값으로 request.getParameter("데이터이름") 메소드를 호출하여 데이터를 꺼낸다.

꺼낸 데이터를 int 타입으로 변환하여 리턴한다.

둘째 파라미터는 전달된 데이터가 없을 때 사용할 디폴트 값이다.

전달된 데이터가 없다면 getInt 메소드는 둘째 파라미터의 값을 그대로 리턴한다.

(줄30) getIntValues 메소드

첫째 파라미터는 꺼낼 데이터 이름이다.

이 이름으로 전달된 값이 여러개의 int일 때 이 메소드를 사용한다.

전달된 값을 int[] 배열에 채워 리턴한다.

(줄38) getStringValues 메소드

첫째 파라미터는 꺼낼 데이터 이름이다.

이 이름으로 전달된 값이 여러개의 문자열일 때 이 메소드를 사용한다.

전달된 값을 String[] 배열에 채워 리턴한다.

(줄44) getBoolean 메소드

첫째 파라미터는 꺼낼 데이터 이름이다.

이 값으로 request.getParameter("데이터이름") 메소드를 호출하여 데이터를 꺼낸다.

꺼낸 데이터를 boolean 타입으로 변환하여 리턴한다.

둘째 파라미터는 전달된 데이터가 없을 때 사용할 디폴트 값이다.

전달된 데이터가 없다면 getBoolean 메소드는 둘째 파라미터의 값을 그대로 리턴한다.

## Student.java 생성

DB의 Student 테이블에서 조회한 값을 멤버 변수로 저장하기 위한 Student 클래스를 만들자

### src/student2/Student.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 | package student2;  public class Student {  String sid;  String studentName;  String department;  int year;  String telephone;  public String getSid() {  return sid;  }  public void setSid(String sid) {  this.sid = sid;  }  public String getStudentName() {  return studentName;  }  public void setStudentName(String studentName) {  this.studentName = studentName;  }  public String getDepartment() {  return department;  }  public void setDepartment(String department) {  this.department = department;  }  public int getYear() {  return year;  }  public void setYear(int year) {  this.year = year;  }  public String getTelephone() {  return telephone;  }  public void setTelephone(String telephone) {  this.telephone = telephone;  }  } |

## StudentDAO1.java 생성

DB의 Student 테이블에 대한 SQL 명령을 구현한 StudentDAO1 클래스를 만들자

### src/student2/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  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140 | package student2;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.ArrayList;  public class StudentDAO1 {    private static Student makeStudent(ResultSet resultSet) throws SQLException {  Student student = new Student();  student.setSid(resultSet.getString("sid").trim());  student.setStudentName(resultSet.getString("studentName"));  student.setDepartment(resultSet.getString("department").trim());  student.setYear(resultSet.getInt("year"));  student.setTelephone(resultSet.getString("telephone"));  return student;  }  public static Student selectBySid(String sid) throws Exception {  Student user = null;  Connection connection = null;  PreparedStatement statement = null;  ResultSet resultSet = null;  try {  connection = DB.getConnection();  String sql = "SELECT \* FROM [Student] WHERE [sid] = ?";  statement = connection.prepareStatement(sql);  statement.setString(1, sid);  resultSet = statement.executeQuery();  if (resultSet.next()) user = makeStudent(resultSet);  } finally {  if (resultSet != null) resultSet.close();  if (statement != null) statement.close();  if (connection != null) connection.close();  }  return user;  }    public static ArrayList<Student> selectPage(int currentPage, int pageSize) throws Exception {  Connection connection = null;  PreparedStatement statement = null;  ResultSet resultSet = null;  try {  connection = DB.getConnection();  String sql = "EXEC studentSelectPage1 ?, ?";  statement = connection.prepareStatement(sql);  statement.setInt(1, currentPage);  statement.setInt(2, pageSize);  resultSet = statement.executeQuery();  ArrayList<Student> list = new ArrayList<Student>();  while (resultSet.next())  list.add(makeStudent(resultSet));  return list;  } finally {  if (resultSet != null) resultSet.close();  if (statement != null) statement.close();  if (connection != null) connection.close();  }  }    public static int getRecordCount() throws Exception {  int count = 0;  Connection connection = null;  PreparedStatement statement = null;  ResultSet resultSet = null;  try {  connection = DB.getConnection();  String sql = "SELECT COUNT(\*) FROM [Student]";  statement = connection.prepareStatement(sql);  resultSet = statement.executeQuery();  if (resultSet.next())  count = resultSet.getInt(1);  } finally {  if (resultSet != null) resultSet.close();  if (statement != null) statement.close();  if (connection != null) connection.close();  }  return count;  }    public static void insert(String sid, String studentName, String department, int year, String telephone) throws Exception {  Connection connection = null;  PreparedStatement statement = null;  try {  connection = DB.getConnection();  String sql = "INSERT [Student] ([sid], [studentName], [department], [year], [telephone]) VALUES (?, ?, ?, ?, ?)";  statement = connection.prepareStatement(sql);  statement.setString(1, sid);  statement.setString(2, studentName);  statement.setString(3, department);  statement.setInt(4, year);  statement.setString(5, telephone);  statement.executeUpdate();  } finally {  if (statement != null) statement.close();  if (connection != null) connection.close();  }  }    public static void updateInfo(String sid, String studentName, String department, int year, String telephone) throws Exception {  Connection connection = null;  PreparedStatement statement = null;  try {  connection = DB.getConnection();  String sql = "UPDATE [Student] SET [studentName] = ?, [department] = ?, [year] = ?, [telephone] = ? WHERE [sid] = ?";  statement = connection.prepareStatement(sql);  statement.setString(1, studentName);  statement.setString(2, department);  statement.setInt(3, year);  statement.setString(4, telephone);  statement.setString(5, sid);  statement.executeUpdate();  } finally {  if (statement != null) statement.close();  if (connection != null) connection.close();  }  }  public static void delete(String sid) throws Exception {  Connection connection = null;  PreparedStatement statement = null;  try {  connection = DB.getConnection();  String sql = "DELETE FROM [Student] WHERE [sid] = ?";  statement = connection.prepareStatement(sql);  statement.setString(1, sid);  statement.executeUpdate();  } finally {  if (statement != null) statement.close();  if (connection != null) connection.close();  }  }    } |

테이블명에 해당하는 User를 Student로 수정하였다.

필드명도 수정하였다.

(줄13, 줄15)

Student 테이블의 sid 필드와 department 필드는 CHAR(5) 타입이다.

SQL Server의 CHAR(...) 타입은 고정 길이 문자열이다.

SQL Server의 **고정 길이** 문자열인 CHAR(...) 타입이나 NCHAR(...) 타입의 필드값을 Java에서 읽어오면,

문자열 뒤에 공백 문자가 붙어있는 문제점이 있다.

이 공백 문자를 제거하기 위해서 줄13, 줄5에서 **trim()** 메소드를 사용하였다.

자바 String 클래스의 trim 메소드는 문자열 뒤의 공백을 제거하여 리턴한다.

SQL Server에서 VARCHAR(...) 타입이나 NVARCHAR(...) 타입은 **가변 길이 문자열이다**. 이들 타입의 필드값을 Java에서 읽어올 때는 **공백 문제가 발생하지 않는다**.

따라서 영문 문자열이라면 VARCHAR(...) 타입을, 한글 문자열이라면 NVARCHAR(...) 타입을 사용하여 테이블을 설계하면 편하다.

(줄89)에의 INSERT 명령에서 기본키인 [sid] 필드값도 포함되었다.

User 테이블의 ID 필드는 DB에서 자동 부여되는 일련번호이므로 INSERT 할 때 제외되지만,

Student 테이블의 sid 필드는 자동 부여되는 값이 아니므로 INSERT 할 때 값이 지정되어야 한다.

# JSP 페이지 생성

## studentList.jsp

### WebContent/01/studentList.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  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  81  82  83  84  85  86  87  88  89  90  91  92  93 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ page import="java.util.\*, student2.\*" %>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <title>lecture1</title>  <link href="http://maxcdn.bootstrapcdn.com/bootstrap/2.3.2/css/bootstrap.min.css" rel="stylesheet" media="screen">  <style>  thead tr { background: #eee; }  tbody tr:hover { background-color: #ffa; cursor: pointer; }  </style>  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>  <script>  $(function() {  $("tbody tr").click(function() {  location.href = "studentEdit.jsp?sid=" + $(this).attr("data-id") + "&" + location.search.replace("?","");  });  $("div.pagination a").click(function() {  $("input[name=pg]").val($(this).attr("data-page"));  $("form").submit();  });  });  </script>  </head>  <%  String s = request.getParameter("pg");  int currentPage = (s == null) ? 1 : Integer.parseInt(s);  int pageSize = 10;  ArrayList<Student> list = StudentDAO1.selectPage(currentPage, pageSize);  int recordCount = StudentDAO1.getRecordCount();  int pageCount = (recordCount + pageSize - 1) / pageSize;  int basePage = ((currentPage - 1) / 10) \* 10;  %>  <body>  <div class="container">  <h1>학생 목록</h1>  <hr />  <div class="pull-right">  <a href="studentAdd.jsp" class="btn btn-small btn-primary">학생 추가</a>  </div>  <form>  <input type="hidden" name="pg" value="1" />  <table class="table table-bordered">  <thead>  <tr>  <th>학번</th>  <th>이름</th>  <th>학과</th>  <th>학년</th>  <th>전화</th>  </tr>  </thead>  <tbody>  <% for (Student student : list) { %>  <tr data-id="<%= student.getSid() %>">  <td><%= student.getStudentName() %></td>  <td><%= student.getDepartment() %></td>  <td><%= student.getYear() %></td>  <td><%= student.getTelephone() %></td>  </tr>  <% } %>  </tbody>  </table>    <div class="pagination pagination-small pagination-centered">  <ul>  <% if (basePage > 0) { %>  <li><a data-page="<%= basePage %>">Prev</a></li>  <% } %>  <% for (int i=1; i <= 10 && basePage + i <= pageCount; ++i) { %>  <li class='<%= basePage+i==currentPage ? "active" : "" %>'>  <a data-page="<%= basePage+i %>"><%= basePage + i %></a>  </li>  <% } %>  <% if (basePage + 11 <= pageCount) { %>  <li><a data-page="<%= basePage+11 %>">Next</a></li>  <% } %>  </ul>  </div>  </form>  </div>  </body>  </html> |

학생 목록 페이지를 구현한 studentList.jsp 파일이다.

(줄18) 상세 페이지의 파일명이 studentEdit.jsp 로 변경되었다.

(줄18) 상세 페이지 studentEdit.jsp URL이 요청될 때, query string으로 기본키(primary key)가 전달되어야 한다.

User 테이블의 기본키는 id 필드여서 상세 페이지 요청 URL이 **userEdit.jsp?id=22** 형태이었고,

Student 테이블의 기본키는 sid 필드이므로 상세 페이지 요청 URL은 **studentEdit.jsp?sid=s022** 형태이다.

(줄32,33) 클래스명만 변경됨

User -> Student

UserDAO4 -> StudentDAO1

(줄62~67) 클래스명과 get 메소드명이 변경됨.

(줄18, 줄63) data-id

이 두 줄에서 사용된 data-id 부분이 서로 일치해야 한다.

즉 두 줄이 전부 data-id 이거나, 둘 다 data-sid 이면 된다.

## studentEdit.jsp

### WebContent/01/studentEdit.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  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  81  82  83  84 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ page import="student2.\*, java.util.\*" %>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <title>lecture1</title>  <link href="http://maxcdn.bootstrapcdn.com/bootstrap/2.3.2/css/bootstrap.min.css" rel="stylesheet" media="screen">  <style>  form > div { margin-top: 20px; }  </style>  </head>  <%  String infoMsg = null;  RequestParameter param = new RequestParameter(request);  String sid = param.getString("sid", "0");  Student student = StudentDAO1.selectBySid(sid);  String studentName = param.getString("studentName", student.getStudentName());  String department = param.getString("department", student.getDepartment());  int year = param.getInt("year", student.getYear());  String telephone = param.getString("telephone", student.getTelephone());  String cmd = param.getString("cmd", "");  if (request.getMethod().equals("POST")) {  if ("저장".equals(cmd)) {  StudentDAO1.updateInfo(sid, studentName, department, year, telephone);  infoMsg = "저장되었습니다";  } else if ("삭제".equals(cmd)) {  StudentDAO1.delete(sid);  infoMsg = "삭제되었습니다";  }  }  String queryString = request.getQueryString().replaceAll("sid=[a-zA-Z0-9]+&?","");  String listUrl = "studentList.jsp?" + queryString;  %>  <body>  <div class="container">  <h1>학생 정보 수정</h1>  <hr />    <form method="post">  <div>학번</div>  <input type="text" name="sid" value="<%= sid %>" />    <div>이름</div>  <input type="text" name="studentName" value="<%= studentName %>" />    <div>학과</div>  <input type="text" name="department" value="<%= department %>" />    <div>학년</div>  <input type="text" name="year" value="<%= year %>" />  <div>전화</div>  <input type="text" name="telephone" value="<%= telephone %>" />      <div>  <button type="submit" class="btn" name="cmd" value="저장">  <i class="icon-hdd"></i> 저장  </button>  <button type="submit" class="btn" name="cmd" value="삭제" onclick="return confirm('삭제하시겠습니까?');">  <i class="icon-remove"></i> 삭제  </button>  <a href="<%= listUrl %>" class="btn">  <i class="icon-list"></i> 목록으로  </a>  </div>  </form>    <% if (infoMsg != null) { %>  <div class="alert alert-info"><%= infoMsg %></div>  <% } %>    </div>  </body>  </html> |

학생 상세 페이지를 구현한 studentEdit.jsp 파일이다.

(줄18) 하늘색으로 표시된 "sid"

학생 목록페이지에서 학생 상세 페이지로 넘어올 때 요청되는 URL은 **studentEdit.jsp?sid=s022** 형태이다.

이 URL의 query string으로 전달된 기본키(primary key)가 sid=s022 부분이므로

request 객체에서 이 기본키 값을 꺼낼 때의 이름이 "sid" 이어야 한다.

"sid" 이름으로 꺼낸 값을 지역변수 sid에 대입한다.

(줄18) 지역변수 sid에 대입된 값으로 DB의 Student 테이블에서 조회한다.

(줄21~25) 하늘색으로 표시된 studentName, department, year, telephone, cmd 부분.

request 객체에서 꺼낼 데이터의 이름들이다.

목록페이지에서 상세 페이지 studentEdit.jsp로 넘어올 때에는 request 객체에 이 이름의 데이터들이 들어있지 않다.

상세 페이지의 입력 폼에 데이터를 입력하고 저장 버튼이나 삭제 버튼을 클릭하여

상세 페이지가 다시 요청(submit)될 때 request 객체에는 이 이름의 데이터들이 들어있다.

(줄52~61) 하늘색으로 표시된 studentName, department, year, telephone, cmd 부분.

이 페이지의 입력폼에 입력할 데이터의 이름을 지정한다.

줄52~61의 하늘색으로 표시된 이름들과, 줄21~25의 하늘색으로 표시된 이름들이 일치해야한다.

(줄21~24) RequestParameter 클래스의 getString 메소드나 getInt 메소드 호출.

이 메소드의 첫째 파라미터는 request 객체에서 꺼낼 데이터의 이름이다.

이 메소드는 그 꺼낸 데이터를 리턴한다.

꺼낸 데이터가 null이면, 이 메소드는 둘째 파라미터 값을 그대로 리턴한다.

(줄21~24) 녹색으로 표시된 studentName, department, year, telephone 부분.

녹색으로 표시된 studentName, department, year, telephone들은 Java 지역 변수이다.

이 지역 변수의 값이 줄52~61의 녹색으로 표시된 JSP 표현식 태그에서 출력되고 있다.

(줄37) String queryString = request.getQueryString().replaceAll("sid=**[a-zA-Z0-9]+**&?","");

예를 들어 이 상세 페이지를 요청한 URL이 **studentEdit.jsp?sid=s033&pg=3** 이라고 할 때,

request.getQueryString() 메소드는 이 URL에서 query string 부분(**sid=s033&pg=3**)을 리턴한다.

자바 String 클래스의 replaceAll 메소드는 문자열에서 일부분을 찾아서 다른 문자열로 치환한다.

텍스트 에디터의 찾기&바꾸기(find&replace) 기능과 유사한 일을 한다.

**원본문자열**.replaceAll(**정규식패턴**, **바꿀문자열** )

**원본문자열**에서 **정규식패턴**을 찾아서(find) **바꿀문자열**로 치환(replace)하여 만들어진 새 문자열을 리턴한다.

sid=[a-zA-Z0-9]+&?

다음과 같은 문자열들에 해당하는 정규식 패턴이다.

"sid=a1&"

"sid=B1"

"sid=cd23&"

"sid=s022"

"sid=gH536&"

"sid=Jkm536"

## 정규식 (regular expression)

정규식은 문자열에서 일부분을 찾을 때 사용하는 패턴 식이다.

예를들어 문자열에서 "the"를 찾으면 정확히 "the" 문자열만 찾아지고 "The"은 대소문자가 다르므로 찾아지지 않는다. 대소문자 상관없이 찾을 때 패턴 식을 사용해야 한다.

정규식의 구성요소는 다음과 같다.

|  |  |
| --- | --- |
| . | 아무 문자나 한 문자 |
| [ ] | 괄호안의 문자 중 아무거나 한 문자 |
| [^ ] | 괄호안의 문자를 제외하고 아무거나 한 문자 |
| [ - ] | 일련의 문자중 아무거나 한 문자 |
| + | 바로 앞의 문자가 1번 이상 반복됨 |
| \* | 바로 앞의 문자가 0번 이상 반복됨 |
| **?** | **바로 앞의 문자가 0번 or 1번** |
| ^ | 라인의 선두 |
| $ | 라인의 끝 |

**정규식 예:** id=[0-9]+

**설명: id= 부분은 정확히 일치해야 한다. [0-9]+ 부분은 0에서 9사이 문자들이 아무거나 여러개 올 수 있다.**

**일치하는 문자열의 예:**

id=3

id=12

id=980

**정규식 예:** id=[0-9]+&

**설명:**  id= 부분은 정확히 일치해야 한다. [0-9]+ 부분은 0에서 9사이 문자들이 아무거나 여러개 올 수 있다.

& 부분도 정확히 일치해야 한다.

**일치하는 문자열의 예:**

id=3&

id=12&

id=980&

**정규식 예:** id=[0-9]+&?

**설명:**  id= 부분은 정확히 일치해야 한다. [0-9]+ 부분은 0에서 9사이 문자들이 아무거나 여러개 올 수 있다.

& 부분도 정확히 일치해야 한다. 그런데 & 문자가 0번이나 1번 나와야 한다. 즉 & 문자가 있어도 되고

없어도 된다.

**일치하는 문자열 의 예:**

id=3&

id=3

id=12&

id=12

id=980&

id=980

**정규식 예:** sid=[a-z0-9]+&?

**설명:**  sid= 부분은 정확히 일치해야 한다.

[a-z0-9]+ 부분은 a에서 z사이 문자나 0에서 9사이 문자들이 아무거나 여러개 올 수 있다.

& 부분도 정확히 일치해야 한다. 그런데 **& 문자가 있어도 되고 없어도 된다.**

**일치하는 문자열의 예:**

sid=23

sid=a3

sid=a3&

sid=1b2f

sid=1b2f&

sid=s98cd4

sid=s98cd4&

**정규식 예:** sid=[a-zA-Z0-9]+&?

**설명:**  sid= 부분은 정확히 일치해야 한다.

**[a-zA-Z0-9]+ 부분은 a에서 z사이 문자나 A에서 Z사이 문자나 0에서 9사이 문자들이**

**아무거나 여러개 올 수 있다.**

& 부분도 정확히 일치해야 한다. 그런데 & 문자가 있어도 되고 없어도 된다.

**일치하는 문자열의 예:**

sid=23

sid=a3

sid=A3&

sid=1b2F

sid=1b2F&

sid=s98cD4

sid=s98cD4&

User 테이블의 기본키 id 필드값은 정수이므로 URL의 query string 정규식 패턴은 **id=[0-9]+** 이다.

**Student 테이블의 기본키 sid 필드값은 알파벳과 숫자가 포함된 문자열이므로 URL의 query string 정규식 패턴은 sid=[a-zA-Z0-9]+ 이다.**

## studentAdd.jsp

### WebContent/01/studentAdd.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  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  81  82  83  84  85  86 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ page import="student2.\*, java.util.\*" %>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <title>lecture1</title>  <link href="http://maxcdn.bootstrapcdn.com/bootstrap/2.3.2/css/bootstrap.min.css" rel="stylesheet" media="screen">  <style>  form > div:not(.box) { margin-top: 20px; }  div.box label { display: inline-block; margin-right: 20px; }  body table.table { width: 400px; }  body table.table tr:first-child { background-color: #eee; }  </style>  </head>  <%  String errMsg = null,infoMsg = null;  RequestParameter param = new RequestParameter(request);  String sid = param.getString("sid", "");  String studentName = param.getString("studentName", "");  String department = param.getString("department", "");  int year = param.getInt("year", 0);  String telephone = param.getString("telephone", "");  String cmd = param.getString("cmd", "");  if (request.getMethod().equals("POST")) {  if ("저장".equals(cmd)) {  if (sid.length() >= 3) {  if (StudentDAO1.selectBySid(sid) == null) {  if (studentName.length() >= 2) {  StudentDAO1.insert(sid, studentName, department, year, telephone);  infoMsg = "저장되었습니다";  }  else errMsg = "이름이 너무 짧습니다";  }  else errMsg = "학번이 중복됩니다";  }  else errMsg = "학번이 너무 짧습니다";  }  }  %>  <body>  <div class="container">  <h1>학생 추가</h1>  <hr />    <form method="post">  <div>학번</div>  <input type="text" name="sid" value="<%= sid %>" />    <div>이름</div>  <input type="text" name="studentName" value="<%= studentName %>" />    <div>학과</div>  <input type="text" name="department" value="<%= department %>" />    <div>학년</div>  <input type="text" name="year" value="<%= year %>" />    <div>전화</div>  <input type="text" name="telephone" value="<%= telephone %>" />    <div>  <button type="submit" class="btn" name="cmd" value="저장">  <i class="icon-hdd"></i> 저장  </button>  <a href="studentList.jsp" class="btn">  <i class="icon-list"></i> 목록으로  </a>  </div>  </form>    <% if (errMsg != null) { %>  <div class="alert alert-error"><%= errMsg %></div>  <% } %>  <% if (infoMsg != null) { %>  <div class="alert alert-info"><%= infoMsg %></div>  <% } %>    </div>  </body>  </html> |

Student 테이블의 sid 필드는 자동 부여되는 값이 아니므로 INSERT 할 때 값이 지정되어야 한다.

그래서 학생 추가 페이지의 입력폼에 sid 값도 입력을 받아서, StudentDAO1 클래스의 insert 메소드를 호출할 때 파라미터로 전달한다.