웹프래임워크
Week14_이론과제

20155137
안원영

컨트롤러 & db & mapper

```java
@RequestMapping(value = "/week13/std_list.do", method = RequestMethod.GET)
public String getStudent(Model model,
        @RequestParam(value="seq", required=false, defaultValue="0") int seq) {
    //seq가 0이 아니면 하나의student정보만 가져옴

    if(seq == 0) {// 전체 학생 조회
        List<Student> std = stddb.select();
        model.addAttribute("std_cmd", std);//std_cmd라는 곳에 std 정보를 저장
        return "week13/list";//리스트 형태로 보여주는 view 호출

    }else {
        Student std = stddb.select(seq);
        model.addAttribute("msg",std);
        return "week13/result";// 개별 학생으로 나오는 view 호출

    }

}

@RequestMapping(value = "/week13/std_delete.do", method = RequestMethod.GET)
public String delete(@RequestParam(value="seq", required=true) int seq){
    stddb.delete(seq);//stddb클래스에 있는 delete메소드 호출
    return "redirect:/week13/std_list.do";
    // redirect : 이것을 사용하면 해당 url을 다시 리로드 시켜준다.

}
```

```java
//학생 조회 리스트 - 전체
public List<Student> select(){//db에 있는 자료들을 list형식으로 가져온다.
    String sql = "select * from student";
    List<Student> std = jdbcTemplate.query(sql, new StdMapper());

    return std;
}

//학생 조회 리스트 - 개별
public Student select(int seq) {// 매개변수로 넘어온 seq에 해당하는 student 정보만 db에서 가져온다.
    String sql = "select * from student where seq=?";
    Student std = jdbcTemplate.queryForObject(sql, new Object[] {seq}, new StdMapper());
    //seq에 해당하는 정보들은 object형식이므로 배열로 가져온다.
    return std;

}

public void update(Student std) {//삭제 후 업데이트되야하기 때문에
    String sql = "update student set id=?, name=?, age=? where seq=?";
    jdbcTemplate.update(sql, std.getId(), std.getName(), std.getAge(),std.getSeq());

}

public int delete(int seq) {//seq에 해당하는 데이터를 지우는 곳
    String sql = "delete from student where seq=?";
    return jdbcTemplate.update(sql, seq);

}
```

```java
// DB에서 사용되는 데이터 형 타입과 자바의 데이터형 타입을 일치시키기 위한 클래스
// 자바와 DB의 데이터 타입을 맞추는 과정이 필요할때 사용되는 클래스로, RowMapper를 implements 사용해서 만든다.
public class StdMapper implements RowMapper<Student>{

    @Override
    public Student mapRow(ResultSet rs, int rowNum) throws SQLException {
        //RowMapper 라는 클래스의 mapRow를 오버로딩 해서 사용
        // ResultSet은 요청에 대한 결과값이 있음
        // rowNum은 레코드(행)의 갯수가 저장된다.

        Student std = new Student();
        std.setSeq(rs.getInt("seq"));
        std.setAge(rs.getInt("age"));
        std.setId(rs.getString("id"));
        std.setName(rs.getString("name"));

        return std;
    }

}
```

Result.jsp & list.jsp & result

```
</head>
<body>

<h1>입력한 내용 출력 </h1>
<table>

    <tr><td>아이디</td><td>${msg.id} </td></tr><!-- commandName으로 지정된 msg에 입력 데이터가 저장되어있음
    <tr><td>이름</td><td>${msg.name} </td></tr>
    <tr><td>나이</td><td>${msg.age} </td></tr>

</table>
<a href="std_delete.do?seq=${msg.seq}">${msg.name} 데이터를 삭제하시겠습니까? (click)</a>

</body>
</html>
```

```
<html>
<head>
    <title> Student리스트 정보 보여주는 곳 </title>
</head>
<body>

<h1>리스트 화면</h1>
<table border=1>
    <tr>
    <th>순서</th>
    <th>아이디</th>
    <th>성명</th>
    <th>나이</th>
    </tr>

    <!-- 리스트 형태로 받아오기 때문에 반복문 수행 -->
    <c:forEach var="std_n" items="${std_cmd}"><!-- std_cmd라는 객체에 모든 list 
    <!-- 하나의 레코드를 std_n이 가지게 된다.  -->
    <tr>
    <td><a href="std_list.do?seq=${std_n.seq}"> ${std_n.seq}</a></td>
    <!-- 링크를 누르면 가상 화면으로 해당 seq에 해당하는 계정 정보를 보여줌  -->
    <td>${std_n.id} </td>
    <td>${std_n.name} </td>
    <td>${std_n.age} </td>
    </tr>
    </c:forEach>

</table>

</body>
</html>
```

회원 정보 등록 화면

아이디 Test111
이름 Test111_name
나이 20
전송

입력한 내용 출력

아이디 Test111
이름 Test111_name
나이 20
Test111_name 데이터를 삭제하시겠습니까? (click)

리스트 화면

순서	아이디	성명	나이
3	aaa	ahn	12
4	ahnwy27	안원영	25
5	test	김만두	20
8	Test111	Test111_name	20

리스트 화면

순서	아이디	성명	나이
3	aaa	ahn	12
4	ahnwy27	안원영	25
5	test	김만두	20