submit 과정:

웹에서 입력 폼에 데이터를 입력하고, submit 버튼을 누른다 >

input 태그에 입력하고, submit 버튼을 누른다 >

서버의 URL 이 요청된다. (http request)

http://localhost:8088/student/edit?id=3 POST 요청

입력 폼에 입력된 데이터도 이 요청에 같이 담겨 전송된다. (request parameter) >

mvc 엔진이 그 요청을 받아서, 요청된 URL과 일치하는 액션 메소드를 찾는다 >

위 액션 메소드의 파라미터가 객체이기 때문에 (Student student),

spring web mvc 엔진이 아래의 일들이 자동으로 처리

입력폼에 입력된 데이터를 spring이 자동 검사해 주는 기능이 spring form validation.

|  |  |
| --- | --- |
| @NotEmpty | 값이 입력되었는지 검사한다 |
| @NotBlank | 공백만 입력된 것은 아닌지 검사한다 |
| @Size(min=2, max=20) | 입력된 문자열의 최소 크기 최대 크기를 검사한다 |
| @Pattern(regexp="남|여") | 입력된 문자열이 정규식에 일치하는지 검사한다 |
| @Email | 이메일 주소 문자열인지 검사한다 |
| @Min(1) | 입력된 값이 최소 1 이상인지 검사한다 |
| @Max(10) | 입력된 값이 최대 10 이하인지 검사한다 |

Student 객체에 채워진 데이터에 문제가 없는지 검사한다.

@NotEmpty, @Size 등 어노테이션 규칙을 검사한다.

Student 객체에 채워진 데이터가, 이 어노테이션의 규칙에 부합하는지를,

spring form validation이 자동 검사한다.

에러가 발견되면 BindingResult 객체에 등록된다

액션 메소드가 호출된다

spring form validatoin 처리 과정

Student 객체에 채워진 데이터에 문제가 없는지 검사한다.

@NotEmpty, @Size 등 어노테이션 규칙을 검사한다.

Student 객체에 채워진 데이터가, 이 어노테이션의 규칙에 부합하는지를,

spring form validation이 자동 검사한다.

에러가 발견되면 BindingResult 객체에 등록

DB에서 조회한 데이터를 채우기 위한 용도 -> DTO 클래스

입력폼에 입력된 데이터를 채우기 위한 용도 -> 모델 클래스

**public StudentEdit findOne(int id)**

이 메소드는 학생 수정 화면의 입력 폼에 채울 학생 객체를 조회해서 리턴한다.

학생 수정 화면에서는 StudentEdit 모델 객체를 사용해야 한다.

그런데 StudentMapper 클래스의 findOne 메소드가 리턴한 것은 Student DTO 객체이다.

그래서 toEditModel 메소드를 호출하여, Student DTO 객체를 StudentEdit 모델 객체로 변환해서 리턴

return modelMapper.map(studentEdit, Student.class);

studentEdit 객체의 속성값을 복사해서, Student 객체를 생성하여 리턴.

삭제 버튼을 클릭하면, 저장 버튼을 클릭했을 때와 똑같이 form이 submit 되고,

현재 URL(student/edit)을 POST 요청하게 된다.

저장 버튼을 클릭해서 form이 submit 되는 경우에는 서버에 전달된 request parameter에

name="cmd" value="save" 값이 들어있다.

삭제 버튼을 클릭해서 form이 submit 되는 경우에는 서버에 전달된 request parameter에

name="cmd" value="delete" 값이 들어있다.

student/edit URL이 POST 요청된 경우에 위 두 액션 메소드들 중 하나가 호출된다.

request parameter에 name="cmd" value="save" 값이 들어있으면 edit 액션 메소드가 호출되고,

request parameter에 name="cmd" value="delete" 값이 들어있으면 delete 액션 메소드가 호출된다.

삭제 요청의 경우에는 입력된 데이터를 검사할 필요 없다.

즉 StudentEdit 모델 객체에 들어있는 데이터를 검사할 필요 없다.

그래서 StudentEdit 모델 객체 앞에 @Valid 어노테이션이 없다.

그렇지만 삭제가 실패했을 때 에러 메시지를 뷰에 전달해야 한다.

그래서 에러 메시지를 전달하기 위해 BindingResult 객체는 여전히 필요하다.

액션 메소드의 파라미터가 객체일 경우에, 다음과 같은 일이 자동으로 일어난다.

- 그 객체에 request parameter 데이터가 자동으로 채워진다.

이때 그 객체의 set 메소드 이름과 request parameter 데이터 이름이 일치하는 것만 채워진다.

입력 폼에 입력된 데이터도 request parameter 데이터로 서버에 전달되고

URL의 query string도 request parameter 데이터로 서버에 전달된다.

pg=%d&sz=%d" 형태의 URL query string을 출력하기 위한 메소드이다.

뷰에서 이 메소드의 리턴값을 출력하려면 **${ pagination.queryString }해야함.**

Application.properties 파일의 다음 설정에 의해서, StudentMapper에서 실행되는 SQL 명령들이

console 창에 출력된다.

logging.level.net.skhu.mapper.StudentMapper=DEBUG

노란색: 방금 실행된 SQL 명령

연두색: 실행된 SQL 명령의 ? 부분에 채워진 mybatis 파라미터 값

파란색: SQL 명령이 조회한 레코드 수