

04) CRUD

04) CRUD

1. 목표
2. 준비 사항
3. 요구 사항
 - 3.1 데이터베이스
 - 3.2 페이지
4. 결과 예시

1. 목표

- 데이터를 생성, 조회, 삭제, 수정할 수 있는 Web Application 제작
 - Python Web Framework를 통한 데이터 조작
 - Object Relational Mapping에 대한 이해
 - Template Variable을 활용한 Template 제작
 - 영화 추천 사이트의 영화 정보 데이터 관리
-

2. 준비 사항

1. (필수) Python Web Framework 선택
 - Flask
 2. (필수) Python Web Framework 사용을 위한 환경 설정
 - Local, C9 등 개발 가능 환경
-

3. 요구 사항

3.1 데이터베이스

- ORM을 통해서 작성 될 클래스의 이름은 `Movie`, 테이블 명은 `movies` 입니다.
- 다음과 같은 정보를 저장합니다.
 - 모든 필드 값에는 빈 값이 들어갈 수 없습니다. `title` 을 제외한 다른 필드는 중복이 허용됩니다.

필드명	자료형	설명
title	String	영화명
title_en	String	영화명(영문)
audience	Interger	누적 관객수
open_date	String(기본) / DateTime(선택)	개봉일
genre	String	장르
watch_grade	String	관람등급
score	Float	평점
poster_url	TEXT	포스터 이미지 URL
description	TEXT	영화소개

3.2 페이지

1. 영화 목록 [index]

1. (필수) 해당 페이지에 접근하는 URL은 `/movies/` 입니다.
2. (필수) 데이터베이스에 존재하는 모든 영화의 목록이 표시 되며, 각 영화의 `title`, `score` 가 표시됩니다.
3. (필수) `title` 을 클릭 시, 해당 영화 정보 조회 페이지로 이동합니다.
4. (필수) 영화 목록 최상단에 새 영화 등록 링크가 있으며, 클릭 시 영화 정보 생성 Form 페이지로 이동합니다.
5. (선택) 추가적인 영화 정보가 표시됩니다.
6. (선택) 사용되는 html 및 bootstrap 의 제한은 없습니다.
7. (선택) 베이스 템플릿인 `base.html` 을 작성합니다.
8. (선택) favicon 을 삽입합니다.

2. 영화 정보 생성 Form [new]

1. (필수) 해당 페이지에 접근하는 URL은 `/movies/new/` 입니다.
2. (필수) 영화 정보를 작성할 수 있는 Form이 표시 되며, 다음과 같은 input들을 가지고 있습니다.

필드명	HTML tag	input type
title	input	text
title_en	input	text
audience	input	number
open_date	input	date
genre	input	text
watch_grade	input	text
score	input	number
poster_url	input	text
description	textarea	없음

3. **(필수)** Form에 작성된 정보는 Submit 버튼 클릭 시 **영화 정보 생성** 페이지로 생성 요청(request)과 함께 전송됩니다.
4. (선택) 요청을 보내는 방식(method)은 GET, POST 중 어느 것을 사용하여도 무관합니다.
5. (선택) **score** 는 사용자로부터 최소 0점에서 최대 5점까지 0.5 점 간격으로 받을 수 있도록 합니다.

3. 영화 정보 생성 [create]

1. **(필수)** 해당 페이지에 접근하는 URL은 **/movies/create/** 입니다.
2. **(필수)** 이전 페이지로부터 전송 받은 데이터를 데이터베이스에 저장합니다.
3. **(필수 - 택1)**
 - '영화 정보 생성이 완료되었습니다.' 라는 플래시 메시지가 표시되며, **영화 목록** 으로 이동하는 링크가 함께 표시됩니다.
 - 해당 페이지에서 저장한 영화 정보를 조회하는 **영화 정보 조회** 페이지로 Redirect 합니다.

4. 영화 정보 조회 [show]

1. **(필수)** 해당 페이지에 접근하는 URL은 **/movies/1/**, **/movies/2/** 등이며, 동적으로 할당되는 부분이 존재합니다. 동적으로 할당되는 부분에는 데이터베이스에 저장된 영화 정보의 Primary Key가 들어갑니다.
2. **(필수)** 해당 Primary Key를 가진 영화의 모든 정보가 표시됩니다.
3. **(필수)** 영화 정보의 최하단에는 **목록**, **수정**, **삭제** 링크가 있으며, 클릭 시 각각 **영화 목록**, 해당 **영화 정보 수정 Form**, 해당 **영화 정보 삭제** 페이지로 이동합니다.

5. 영화 정보 수정 Form [edit]

1. **(필수)** 해당 페이지에 접근하는 URL은 **/movies/1/edit/**, **/movies/2/edit/** 등이며, 동적으로 할당되는 부분이 존재합니다. 동적으로 할당되는 부분에는 데이터베이스에 저장된 영화 정보의 Primary

Key가 들어갑니다.

2. **(필수)** 해당 Primary Key를 가진 영화 정보를 수정할 수 있는 Form이 표시되며, 이전 정보가 입력된 채로 표시됩니다.
3. **(필수)** Form에 작성된 정보는 Submit 버튼 클릭 시 **영화 정보 수정** 페이지로 수정 요청(request)과 함께 전송됩니다.
4. (선택) 요청을 보내는 방식(method)은 GET, POST 중 어느 것을 사용하여도 무관합니다.

6. 영화 정보 수정 [update]

1. **(필수)** 해당 페이지에 접근하는 URL은 `/movies/1/update/`, `/movies/2/update/` 등이며, 동적으로 할당되는 부분이 존재합니다. 동적으로 할당되는 부분에는 데이터베이스에 저장된 영화 정보의 Primary Key가 들어갑니다.
2. **(필수)** 해당 Primary Key를 가진 영화 정보를 이전 페이지로부터 전송 받은 데이터로 변경하여 데이터베이스에 저장합니다.
3. **(필수 - 택1)**
 - '영화 정보 수정이 완료되었습니다.' 플래시 메시지가 표시되며, **영화 목록**으로 이동하는 링크가 표시됩니다.
 - 해당 페이지에서 수정한 영화 정보를 조회하는 **영화 정보 조회** 페이지로 Redirect 합니다.

7. 영화 정보 삭제 [delete]

1. **(필수)** 해당 페이지에 접근하는 URL은 `/movies/1/delete/`, `/movies/2/delete/` 등이며, 동적으로 할당되는 부분이 존재합니다. 동적으로 할당되는 부분에는 데이터베이스에 저장된 영화 정보의 Primary Key가 들어갑니다.
2. **(필수)** 해당 Primary Key를 가진 영화 정보를 데이터베이스에서 삭제합니다.
3. **(필수 - 택1)**
 - '영화 정보 삭제가 완료되었습니다.' 플래시 메시지가 표시되며, **영화 목록**으로 이동하는 링크가 표시됩니다.
 - **영화 정보 목록** 페이지로 Redirect 합니다.

4. 결과 예시

Python Web Framework 를 활용해 작성한 모든 파일을 `04_crud` 디렉토리에 위치하도록 합니다.

결과물은 반드시 `README.md` 에 활용하였던 내용을 정리 해주세요.

```
04_crud/
  README.md
  app.py
  models.py
  db_flask.sqlite3
  migrations/
  ...
  templates/
```

```
index.html
edit.html
new.html
show.html
base.html (선택)
static/ (선택)
...
```