프로젝트 기반의 Java Web 애플리케이션 개발

보수교육 - 2023.09.23 ~ 2023.09.24

강의 일정

1일차

시간	단원명	세부학습 내용
10:00~11:00	스프링 프레임워크 시작하기	• 프론트엔드/백엔드 소개 • 스프링 프레임워크 소개 • 프로젝트 개발환경 구축
11:00~12:00	스프링 프레임워크 프로젝트 분석하기	• 스프링 프레임워크 프로젝트 생성 • 프로젝트 구조 분석 및 주요 파일 설명 • 패키지 생성 및 클래스 파일 생성
12:00~13:00	request와 response 처리하기	• client, server 개념 • request 및 response 처리하기 • 요청 방식에 따른 Controller 메서드 구현 • MVC 패턴을 활용한 request, response 처리 • contoller, service, dao패키지의 클래스에서 필요한 어노테이션 작성

강의 일정

1일차

시간	단원명	세부학습 내용
14:00~15:00	form 다루기	• html에서 form태그를 활용해 데이터 넘기기 • dto 클래스의 필드 이름과 넘길 데이터의 name 값 설정하기 • form 태그를 사용해 입력한 정보를 MVC 패턴을 활용하여 백엔드 데이터 처리
15:00~16:00	데이터베이스 연동하기	• database 및 table 생성 • mybatis 활용하기 • database에서 테이블 생성 • 입력한 정보 처리를 위한 mybatis 구현
16:00~17:00	데이터베이스 저장 처리	• mybatis에서 저장 처리를 위한 insert 쿼리문 실행 • 저장된 데이터를 데이터베이스에서 확인 • 정보에 대한 정보를 insert 쿼리문에 맞게 작성 • 테이블에서 저장이 잘됐는지 확인

강의 일정

2일차

시간	단원명	세부학습 내용
10:00~11:00	데이터베이스 조회 기능 구현	• mybatis에서 조회하기 위한 select 쿼리문 작성 • 저장된 데이터를 목록으로 구현 • 목록에서 링크를 통해 상세정보 보기 구현
11:00~12:00	데이터베이스 수정 기능 구현	• mybatis에서 수정을 위한 update 쿼리문 작성 • 수정된 내용을 view에서 확인하기 • 수정된 정보를 상세보기를 통해 확인
12:00~13:00	데이터베이스 삭제 기능 구현	• mybatis에서 삭제를 위한 delete 쿼리문 작성 • 목록에서 해당 데이터가 삭제됐는지 확인

강의 일정

2일차

시간	단원명	세부학습 내용
14:00~15:00	페이징 처리 기능 구현	• 페이징 처리 기능 소개 • 페이징 처리 프로세스
15:00~16:00	스프링 영화관 프로젝트 I	• 영화 목록, 등록 페이지 생성 • 영화정보 관리를 위한 테이터베이스
16:00~17:00	스프링 영화관 프로젝트 II	• 제공된 resource를 프로젝트 적용하기 • 영화 정보 상세보기, 수정 페이지 생성

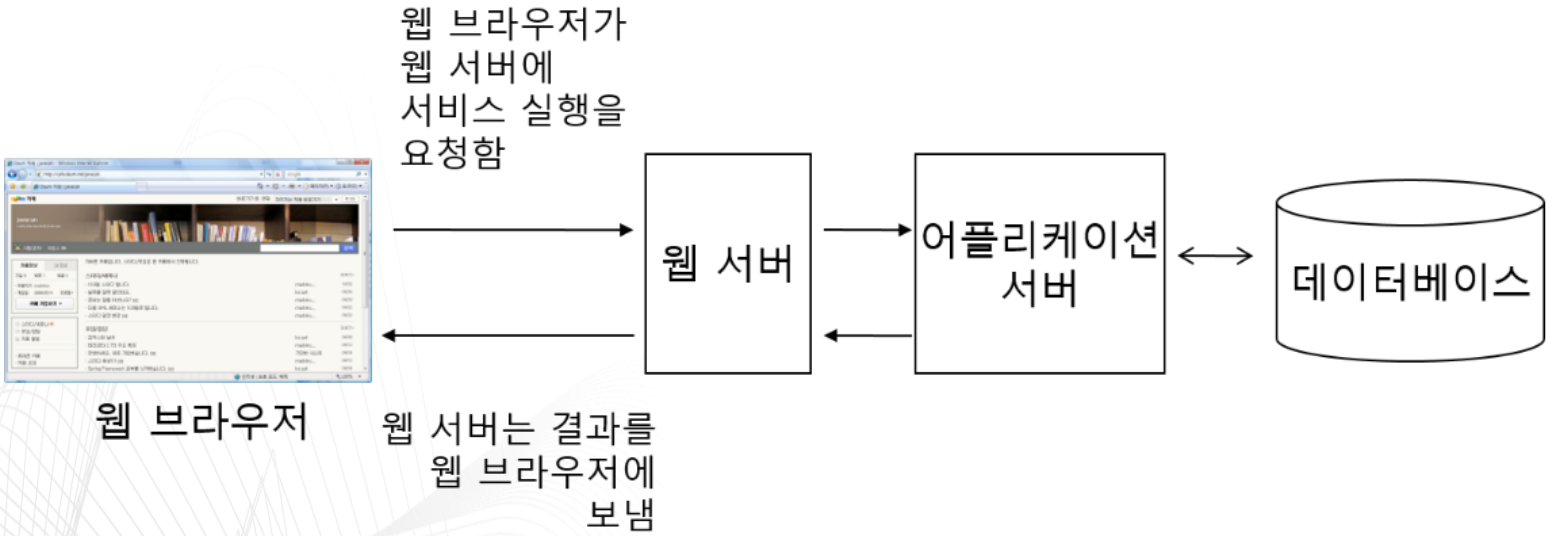
1교시

스프링 프레임워크 시작하기

개요

- Spring Boot와 Thymeleaf, MyBatis를 활용한 Web 애플리케이션 개발

- 기본적인 웹 운영 형태



사용자는 웹 브라우저를 통해 서버에 요청(Request)을 전송하면 서버는 요청한 내용(자원, 데이터 등)을 처리하여 응답(Response)을 전송하는 구조.

웹 애플리케이션의 구성

웹 애플리케이션 = 프론트엔드 + 백엔드

- 프론트엔드(Front-end)
  - 웹 사용자가 브라우저 프로그램(MS 엣지, 구글 크롬, 사파리 등)을 통해 보는 화면을 개발하는 분야.

  - HTML
    - HTML은 하이퍼텍스트와 마크업 언어로 구성. 하이퍼텍스트는 페이지들 사이의 링크, 마크업 언어는 웹페이지의 구조를 정의. 웹 페이지의 뼈대.
  - CSS
    - CSS는 종속된 스타일시트(Cascading Style Sheets). 웹페이지에 다양한 스타일을 적용할 수 있게 해줌으로써 애플리케이션 페이지를 표시하는 프로세스를 단순하게 만들어주는 디자인 언어. 웹 페이지의 모습.
  - Javascript
    - 사용자들과 상호작용하는 기능을 제공. (최근 JS는 백엔드 개발에도 활용되고 있음)

웹 애플리케이션의 구성

웹 애플리케이션 = 프론트엔드 + 백엔드

- 백엔드(Back-end)
  - 서버 측 개발 분야로, 사용자가 요청한 기능(서비스 로직)을 내부적으로 수행하며, 데이터를 저장하고 관리하는 프로그램을 개발하는 분야.

  - Java
    - 가장 인기 있는 프로그래밍 언어 중 하나이자 객체지향 프로그래밍 언어(본 과정 주요 언어)
  - Javascript
    - 백엔드와 프론트엔드 모두에서 사용.(Node.js)
  - Python
    - 다양한 개발에 활용되며, 특히 딥러닝, 데이터 사이언스, 인공지능 분야에서 많이 사용

Spring Framework

- 로드 존슨(Rod Johnson)이 2002년에 출판한 저서 Expert One-on-One J2EE Design and Development에서 선보인 예제 소스 코드에서 시작하여 현재까지 발전된 자바 기반의 웹 프레임워크.
- 한국 전자정부 표준 프레임워크의 기반 기술이며, 한국 정보화 진흥원에서 공공기관의 웹 서비스 제공 시에 스프링 프레임워크를 권장하고 있음.

Spring Framework 주요 특징

POJO(Plain Old Java Object) 방식

- ‘오래된 방식의 간단한 자바 오브젝트’라는 뜻. Java EE(EJB) 등의 중량 프레임워크들을 사용하게 되면서 해당 프레임워크에 종속된 "무거운" 객체를 만들게 된 것에 반발해서 사용되게 된 용어. 특정 자바 모델이나 기능, 프레임워크 등을 따르지 않은 자바 오브젝트를 지칭.

AOP(Aspect Oriented Programming)

- ‘관점 지향 프로그래밍’. 로깅, 트랜잭션, 보안 등 여러 부분에서 공통적으로 사용되는 코드(기능)를 분리하여 관리하는 프로그래밍 방식.

DI(Dependency Injection)

- ‘의존성 주입’. 객체 간의 의존 관계를 소스 코드 내부에서 처리하지 않고, 외부 설정으로 정의되는 방식. 소소의 재사용성과 객체 간의 결합도를 낮출 수 있음. 스프링 프레임워크가 DI를 처리.

Spring Framework 주요 특징

IoC(Inversion of Control)

- '제어의 역전'. 개발자가 작성한 코드가 외부 라이브러리를 사용하는 방식(전통적인 방식)이 아니라 외부 라이브러리(프레임워크)가 개발자의 코드를 필요에 따라 사용하는 방식.

Lifecycle 관리

- 자바 객체의 생성, 소멸을 프레임워크가 관리.

Spring Boot

Spring Framework를 사용하여 더 빠르고 쉽게 웹 애플리케이션과 마이크로서비스를 개발하도록 돕는 도구.

핵심 기능

- 사전 설정된 종속성 항목으로 애플리케이션을 초기화. 내장형 자동 구성 기능과 설정에 따라 Spring Framework와 3rd-party 패키지를 자동으로 구성.
- 개발자의 초기 선택에 따라 설치할 패키지(라이브러리)와 사용할 기본 값을 지정. 개발자는 단순한 웹 폼인 Spring Boot Initializer로 프로젝트를 시작.
- Tomcat과 같은 웹 서버를 앱에 포함하여 외부 웹 서버에 의존하지 않고 자체적으로 실행되는 독립형 애플리케이션을 개발할 수 있음.

개발 환경 설정

개발 도구

- JDK 17 이상
  - openjdk (https://github.com/ojdkbuild/ojdkbuild)

- 통합개발환경(IDE)
  - JetBrain IntelliJ IDEA Ultimate 버전(30일 무료 평가판 사용, 로그인 필요)
  - (https://www.jetbrains.com/ko-kr/idea/download/?section=windows)

- Database(DBMS)
  - H2 Database

- 수업 관련 자료 공유(Github)
  - https://github.com/tiblo/spring_edu

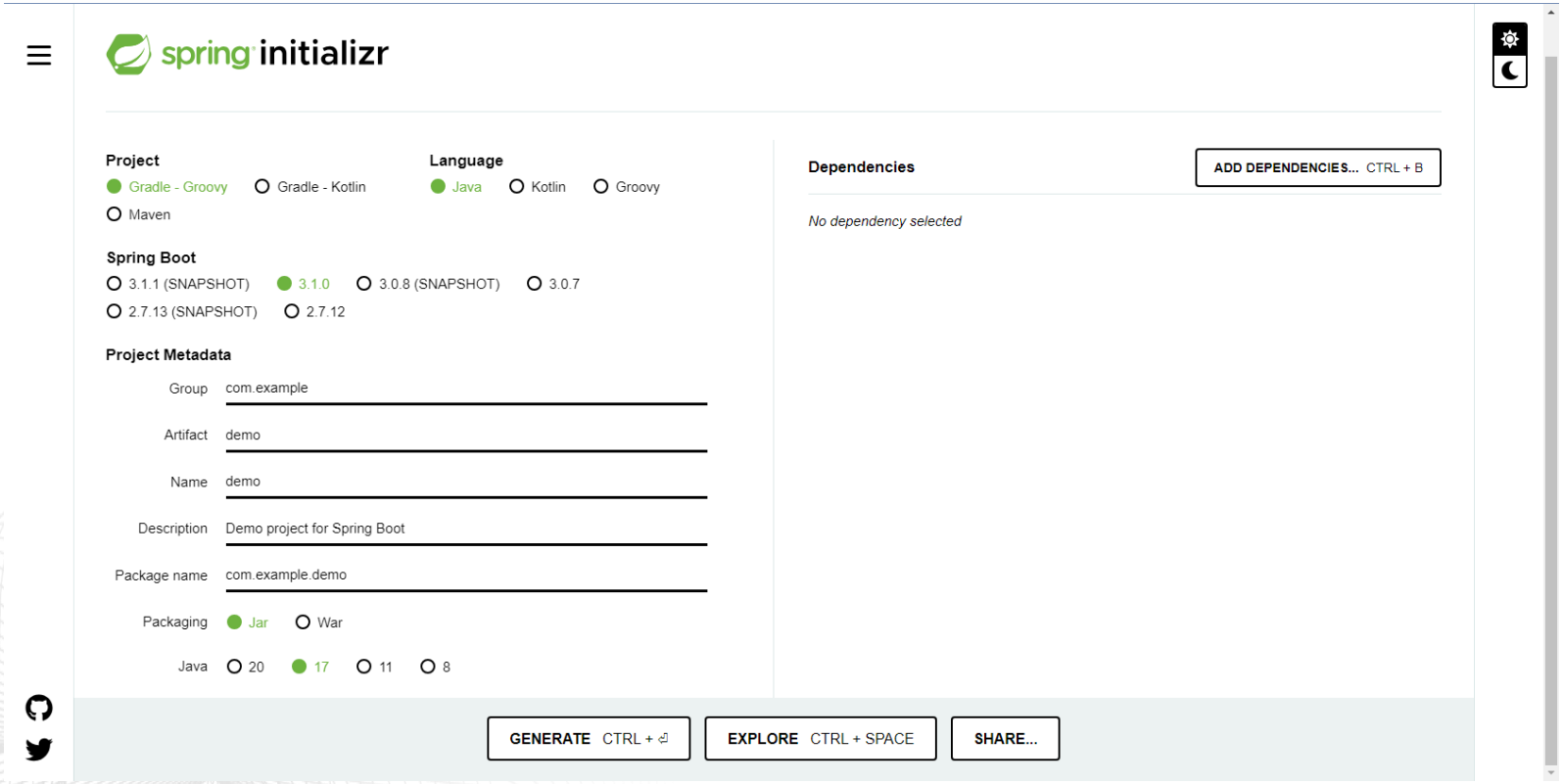
2교시

스프링 프로젝트 분석하기

Spring Boot 프로젝트 생성

방식 1.

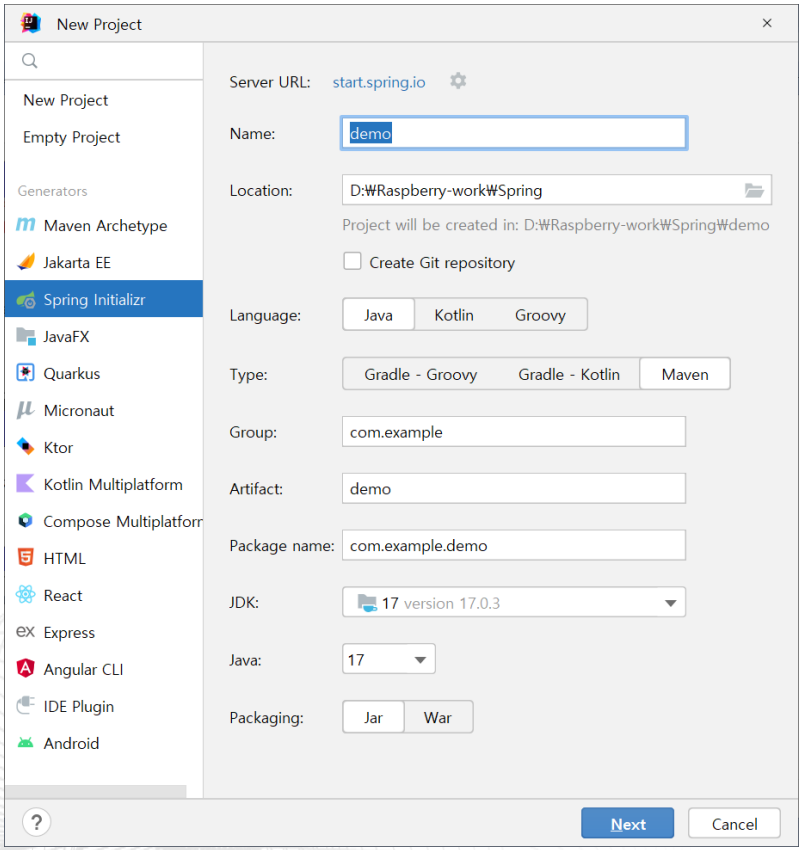
- https://start.spring.io/ 사이트를 통해 프로젝트를 생성하고, 다운로드 하여 개발환경에서 불러오는 방법



Spring Boot 프로젝트 생성

방식 2.

- IDEA에서 프로젝트 생성 - 1



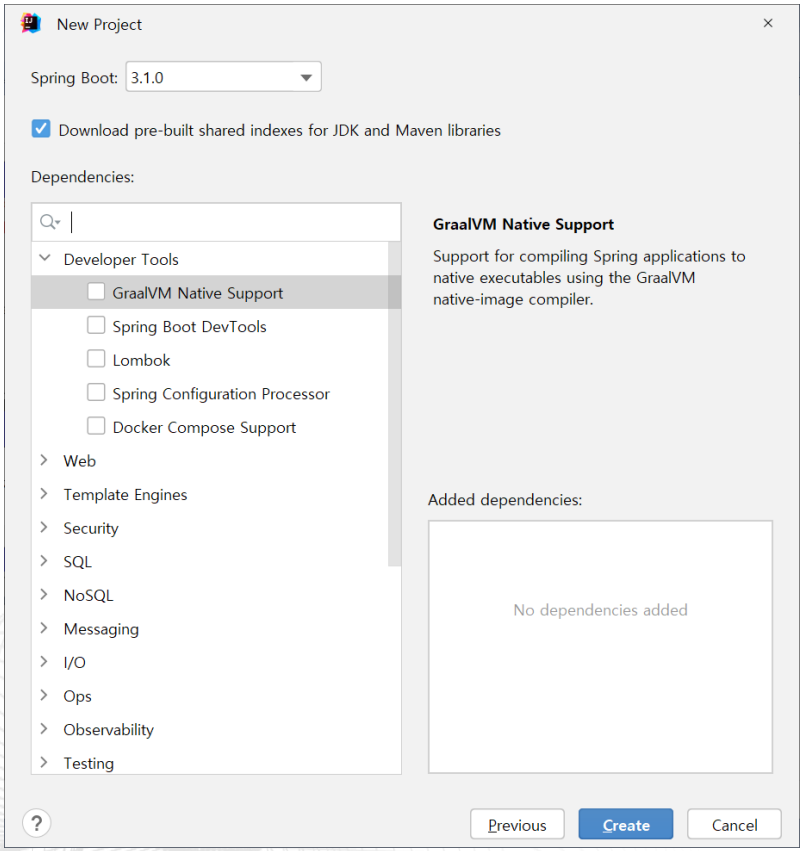
Name : start01
Location : 작업 폴더
Language : Java
Type : Maven
Group : com.icia
Artifact : start01
Package name : com.icia.start01
JDK : 17
Java : 17
Packaging : Jar

빨간 글자만 작성 후 Next.

Spring Boot 프로젝트 생성

방식 2.

- IDEA에서 프로젝트 생성 - 2



Dependency(초기 라이브러리 설정)

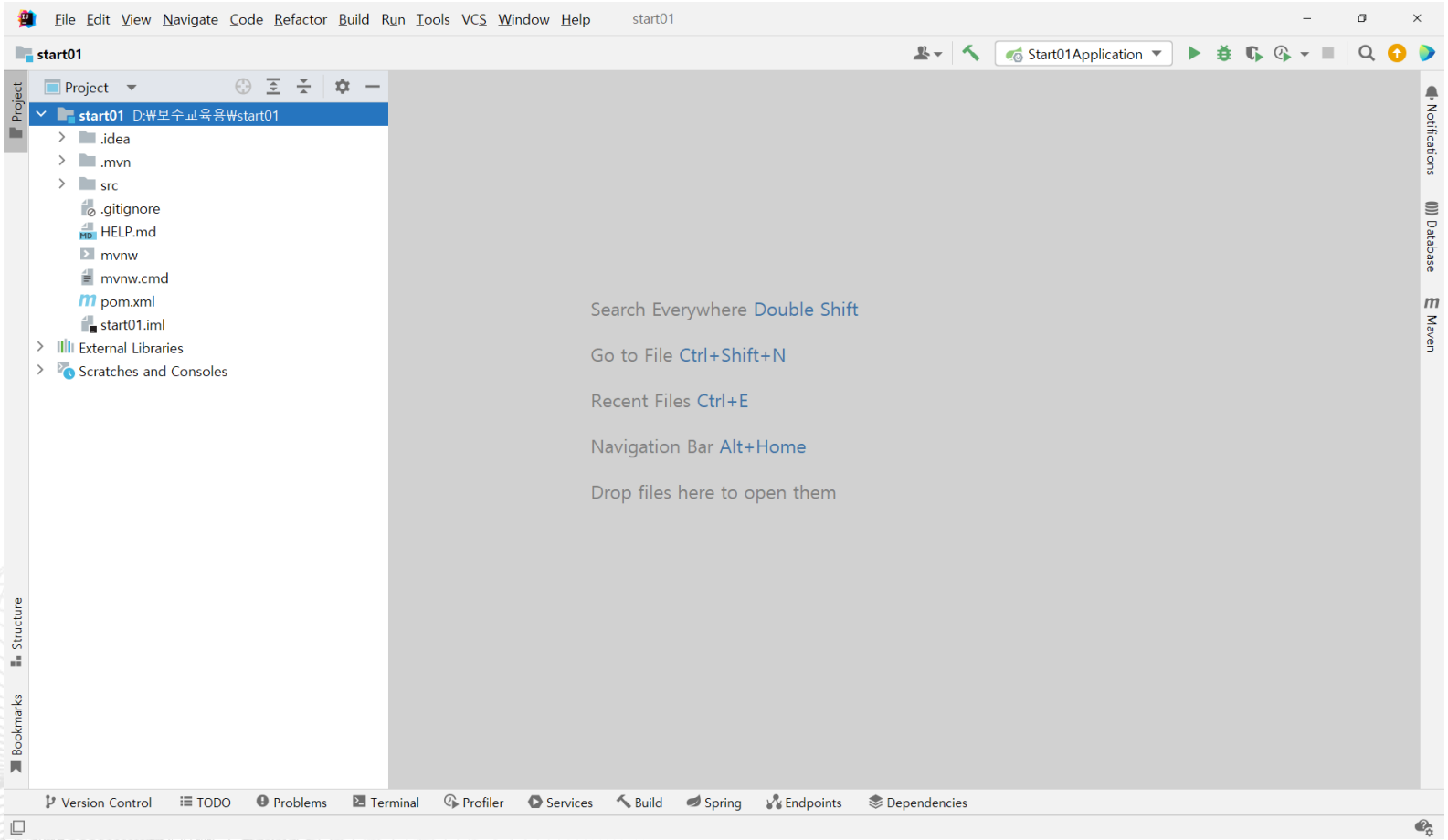
- Developer Tools
  - Spring Boot DevTools
  - Lombok
- Web
  - Spring Web
- Template Engines
  - Thymeleaf

선택 후 Create.

Spring Boot 프로젝트 생성

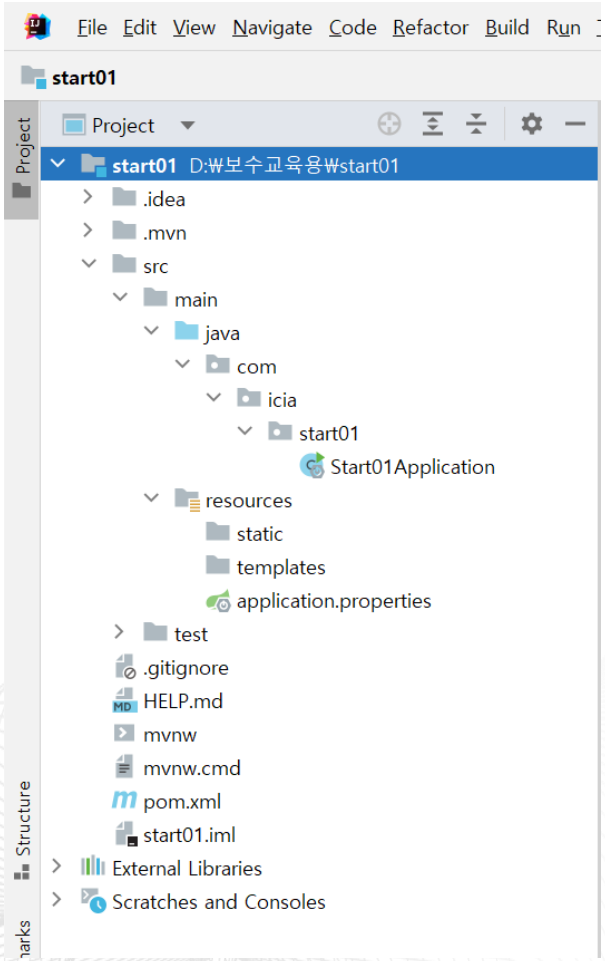
방식 2.

- IDEA에서 프로젝트 생성 - 3



초기 화면 처리 및 실행

작업 폴더 - src/main



작업 폴더 – src/main
java 폴더 : java 소스 코드 작성.
resources 폴더 : 이미지, 스타일시트, 자바스크립트, HTML 페이지 작성.

작성 내용
1. ‘start01’에 Package 생성
   - controller
2. controller 패키지에 Java Class 생성
   - HomeController
3. templates 폴더에 HTML file을 생성
   - home.html
4. Start01Application 실행

초기 화면 처리 및 실행

HomeController.Java

```
...
@Controller
public class HomeController {
    @GetMapping("/")
    public String home(){
        return "home";
    }
}
```

초기 화면 처리 및 실행

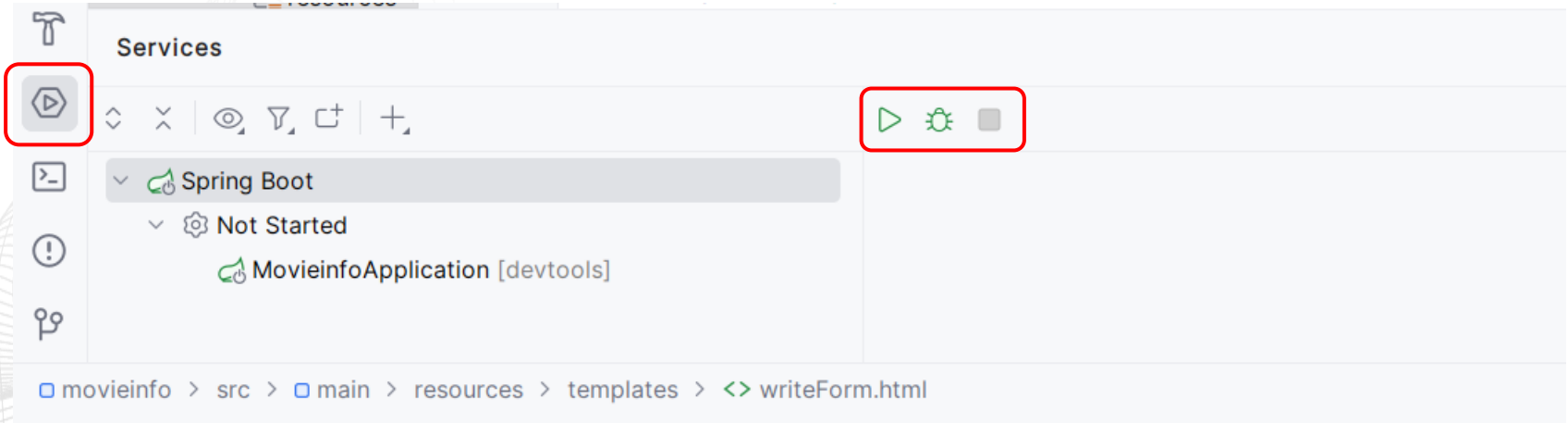
home.html

```
...
<body>
    <h1>첫 페이지</h1>
    <p>처음으로 보이는 페이지</p>
</body>
...
```

초기 화면 처리 및 실행

실행 환경 설정

- 좌측하단의 'Services' 아이콘을 클릭.
- '+' (Add service) > Run Configuration Type > Spring Boot 선택.
- '▶'를 눌러 실행.
  - Lombok requires enabled annotation processing 창이 뜨면 Enable annotation processing 클릭
- 'Start01Application :8080/'에서 마지막 '8080/' 부분을 클릭.
  - 브라우저로 실행



Resources 폴더의 구성

static 폴더

- 배경이미지, 스타일시트, 자바스크립트 등 웹의 정적 자원(Static Resources)을 저장하는 폴더
- static 폴더 밑에 각 자원을 위한 폴더를 생성하여 각각 저장

templates 폴더

- HTML 템플릿을 저장하는 폴더
- 접속자의 화면에 보여질 HTML 문서를 작성
- Thymeleaf 템플릿을 사용하여 페이지를 작성

application.properties

- 정적 자원의 위치 지정, DB 설정, 파일 업로드, 에러 페이지 등 웹 프로그램 실행 시 필요한 설정 내용을 작성.

3교시

REQUEST와 RESPONSE 처리하기

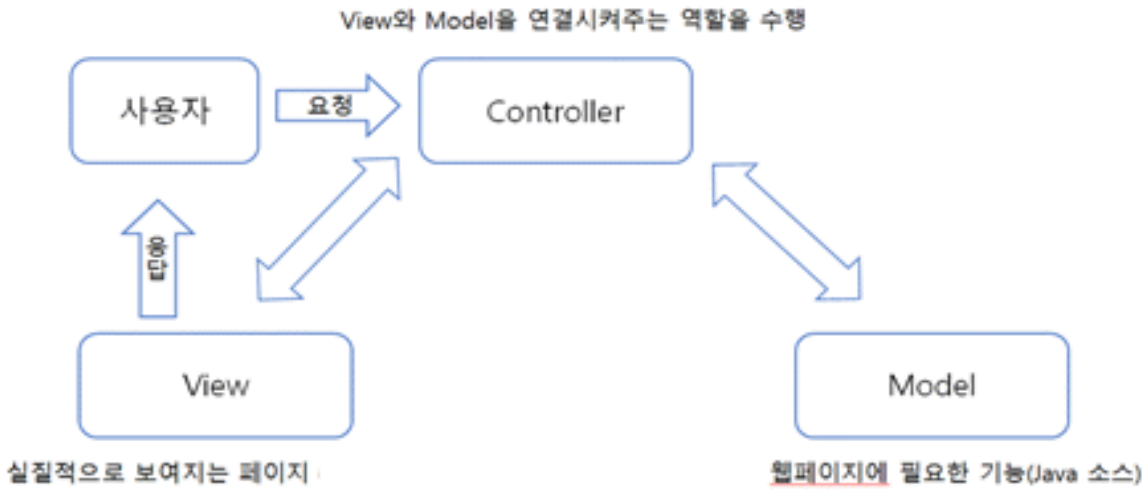
MVC 패턴

웹 프로그래밍에서 사용하는 디자인 패턴

- 모델(Model) - Service
  - 비즈니스 영역의 로직을 처리
- 뷰(View)
  - 비즈니스 영역에 대한 프레젠테이션 뷰(사용자가 보게 될 결과 화면)를 담당
- 컨트롤러(Controller)
  - 사용자의 입력 처리와 흐름제어를 담당. 사용자의 요청에 대해서 알맞은 모델을 사용하고 사용자에게 보여줄 뷰를 선택

MVC 패턴

MVC 패턴의 3가지 구성요소의 연결



- 사용자는 원하는 기능을 처리하기 위한 모든 요청을 컨트롤러에 전송.
- 컨트롤러는 비즈니스와 관련된 기능을 제공하는 모델을 이용해서 사용자의 요청을 처리.
- 비즈니스 로직을 수행한 후 컨트롤러는 사용자에게 보여줄 뷰를 선택.
- 선택된 뷰는 사용자에게 알맞은 결과 화면을 출력.

MVC 패턴 관련 패키지 구성

프로젝트 패키지

- controller
  - 전체 웹 프로그램을 제어하는 컨트롤러 클래스 작성
- service
  - 비즈니스 관점에서 각 기능을 처리하는 서비스 클래스 작성
  - MVC 패턴에서의 Model 역할을 담당
- dto(Data Transfer Object)
  - 데이터 전송에 활용할 데이터 저장/전송 클래스 작성
- dao(Data Access Object)
  - DB 처리를 위한 인터페이스 작성
- util
  - 기타 보조적인 기능을 제공할 클래스 작성

Spring Framework에서 Model은 Spring과 Controller 간의 데이터 전송에 사용하는 객체

요청(Request)과 응답(Response)

Client/Server 전송 모델

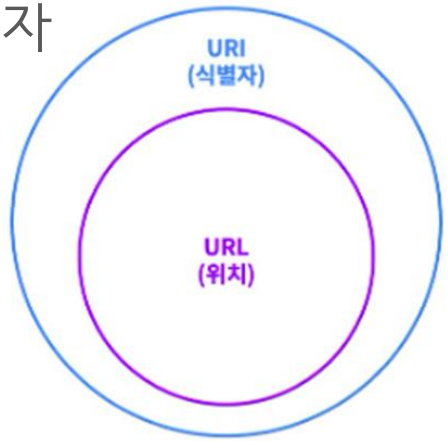
- 서비스 요청자인 client와 서비스 자원의 제공자인 서버 간의 작업을 분리한 분산 애플리케이션 구조
- Client
  - 서비스를 사용하는 사용자 또는 사용자의 단말(컴퓨터)
  - 서비스에 대한 요청(Request)을 생성하여 Server에 전달
- Server
  - 서비스를 제공하는 컴퓨터
  - 서비스에 대한 응답(Response)을 생성하여 Client에 전달

- 웹 애플리케이션은 Client/Server 구조로 동작함
- 객체지향 언어 기반의 Request는 객체로 생성되어 Controller에 전달됨

Request의 처리

URI와 URL

- 웹 사이트 주소뿐만 아니라 컴퓨터 네트워크 상의 자원(Resource)을 모두 나타내는 표기법
- 자원이란?
  - 웹 상에서 제공되는 모든 것 - 글자, 이미지, 영상, 음성, 기타 파일 등
- URI(Uniform Resource Identifier) - 통합 자원 식별자
- URL(Uniform Resource Locator) - 통합 자원 위치
  - 엄밀하게 따지면 다르지만 혼용하여 많이 사용됨.



Context Path
Web Application Server(Tomcat)에서 웹 상의 어플리케이션들을 구분하기 위한 Path.

Request mapping

- 사용자의 요청을 URI로 구분하여 처리
- 요청 URI와 웹 프로그램의 기능을 제공하는 Controller 메소드를 연결하는 작업

Response의 처리

일반적으로 사용자의 요청은 HTML 문서 형태로 응답

Response의 유형

- Request 처리 결과를 담은 HTML 문서
- 개별적인 웹 자원(글자, 파일 등)

View란 디자인된 HTML문서에 사용자가 요청한 자원을 담은 Response

- 정적 자원(Static Resource)
  - HTML 문서에 담긴 고정화된 데이터로 항상 동일한 내용을 제공
- 동적 자원(Dynamic Resource)
  - 사용자의 요청에 따라 변경되는 데이터(실시간 뉴스 등)

Request와 Response 처리

home.html 페이지에서 다른 페이지로 이동 처리

- templates 폴더
  - home.html 페이지에 링크(이동) 요소 작성
  - intro.html 페이지 작성

- HomeController.java
  - URI 매핑 메소드 작성

Server에서 Client로 데이터 전송하기

- Model과 ModelAndView 객체 활용
- Thymeleaf에서 데이터 출력하기
  - th:text의 활용

4교시

FORM 다루기

Thymeleaf 활용

Thymeleaf 네임스페이스 명시

- <html lang="ko" xmlns:th="http://www.thymeleaf.org">

기본 문법

- html 태그 안에 th 문법을 추가

  <태그 th:[속성]="데이터 및 표현식">

  - 표현식
    - ${식별자} : 일반적인 글자 데이터 출력
    - @{link} : a 태그의 href 속성이나 form 태그의 action 속성에서 uri 처리
    - *{field} : server에서 dto 객체 형태로 전송할 때, dto 객체의 필드를 출력
      - 먼저 상위 요소에서 th:object로 객체를 지정
  - 자바스크립트에서의 출력
    - [[${식별자}]]

Thymeleaf 활용

주요 속성

- 데이터 출력
  - th:text - 서버에서 전달 받은 값을 해당 요소의 innerText로 출력
  - th:utext - 서버에서 전달 받은 값을 해당 요소의 innerHTML로 출력
- 링크 처리
  - th:href - 링크 연결 속성. a 태그에서 사용
  - th:action - form 태그의 action 속성 대신 사용
- 객체 처리
  - th:object - dto와 같은 객체로 데이터를 전송하는 경우 사용
    - 하위 요소에서 *{field}로 각 데이터를 가져올 수 있음
- 스크립트 처리
  - th:inline - 스크립트 언어 지정
    - <script th:inline="javascript">

Thymeleaf 활용

제어용 태그

- th:block
  - 제어문 및 객체 설정에 사용

제어 속성

- 조건 제어 속성
  - th:if - if문에 해당하는 속성(else는 없음)
  - th:unless - if문의 반대에 해당하는 속성. 조건식을 th:if와 똑같이 작성하면 else문의 역할을 수행
  - th:switch, th:case - switch, case에 해당하는 속성
- 반복 제어 속성
  - th:each - for문에 해당하는 속성

Thymeleaf 활용

공통 영역 처리

- th:fragment
  - header나 nav, footer 같은 페이지 공통 부분의 처리를 위한 속성
  - 하나의 html 파일에 여러 분할 영역을 작성
  - 공통 부분이 필요한 페이지에서 활용

- Fragment 활용 속성
  - th:insert - 태그 안으로 fragment 요소를 삽입

```
<header th:insert="~{fragments::header('테스트용 헤더')}"></header>
```

```
▼<header> == $0
    <h2>테스트용 헤더</h2>
  </header>
```

  - th:replace – 태그를 fragment 요소로 교체

```
<header th:replace="~{fragments::header('테스트용 헤더')}"></header>
```

```
<h2>테스트용 헤더</h2>
```

사용자 데이터 전송

form 태그를 활용한 데이터 전송

- 전송 방식
  - GET 방식 - 데이터가 URI에 노출되는 방식(a 태그의 전송 방식)
    - 노출되어도 무방하며, 짧은 소수의 데이터 전송에 적합
  - POST 방식
    - 노출되어서는 안되거나, 긴 데이터 전송에 적합

- 방식 지정
  - form 태그의 method 속성으로 설정
    - <form th:action="@{uri}" method="get">

- controller 클래스에서 uri 매핑 시 구별하여 처리
  - get 방식 - @GetMapping("uri")
  - post 방식 - @PostMapping("uri")

Thymeleaf 설정

초기 Dependency 설정

- Developer Tools
  - Spring Boot DevTools
- Template Engines
  - Thymeleaf

Thymeleaf 설정

기타 설정

- 프로젝트 수정 후 서버 재시작을 자동으로 처리하는 설정

- application.properties 설정
  - spring.thymeleaf.cache=false
  - spring.devtools.livereload.enabled=true
  - spring.devtools.restart.enabled=true

- File > Settings
  - Build, Execution, Deployment > Compiler
    - Build project automatically 체크
  - Advanced Settings
    - Allow auto-make to start even if developed application is currently running 체크

5 - 9교시

데이터베이스 연동하기

Database 처리

H2 Database

- 자바로 구현된 오픈소스 데이터베이스
- 별도의 설치 없이 바로 사용 가능(개발 테스트용으로 많이 사용)

H2 Database 활용을 위한 설정

- 초기 Dependency 추가
  - H2 Database

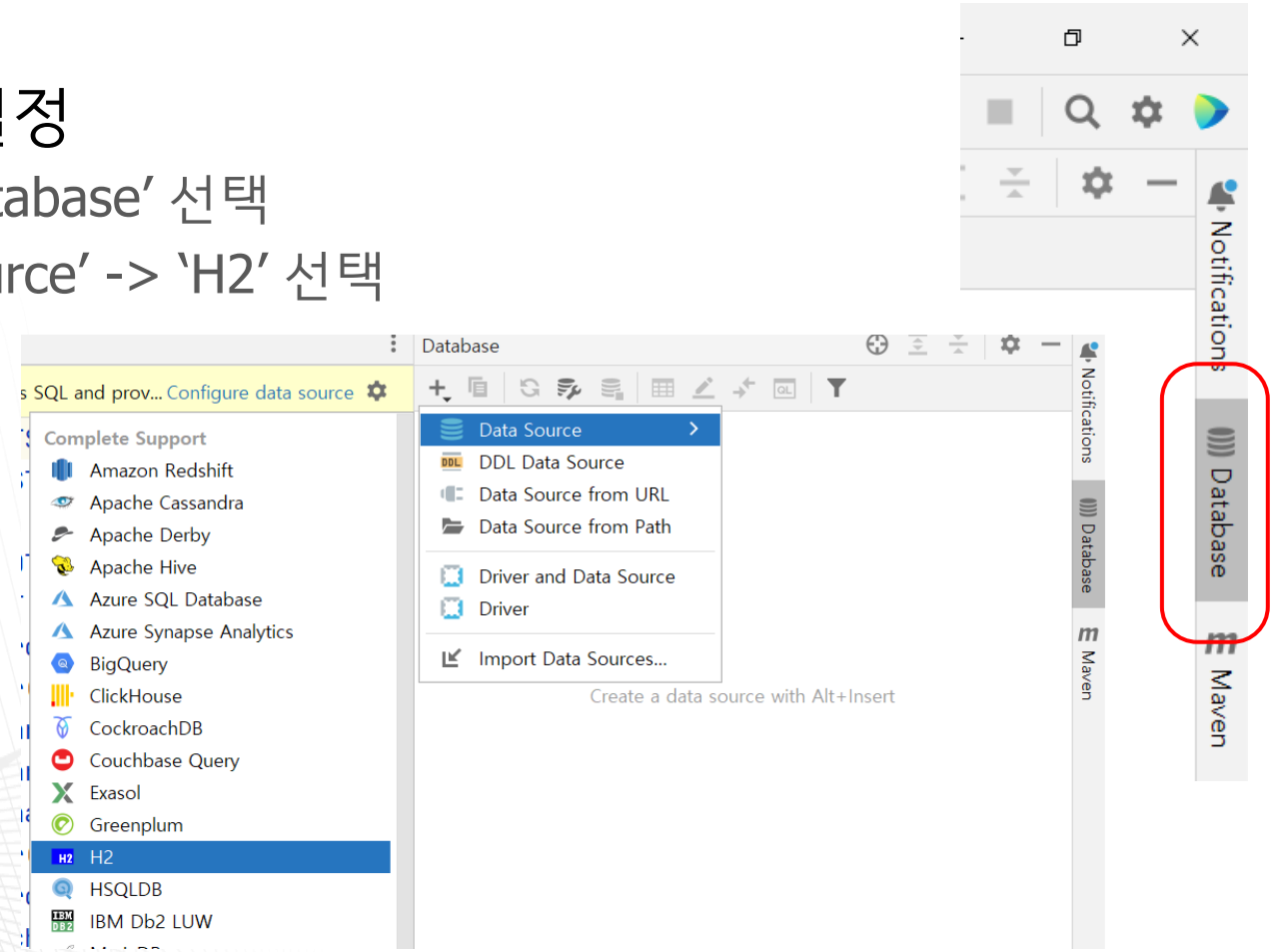
Database 처리

DDL 관련 작업

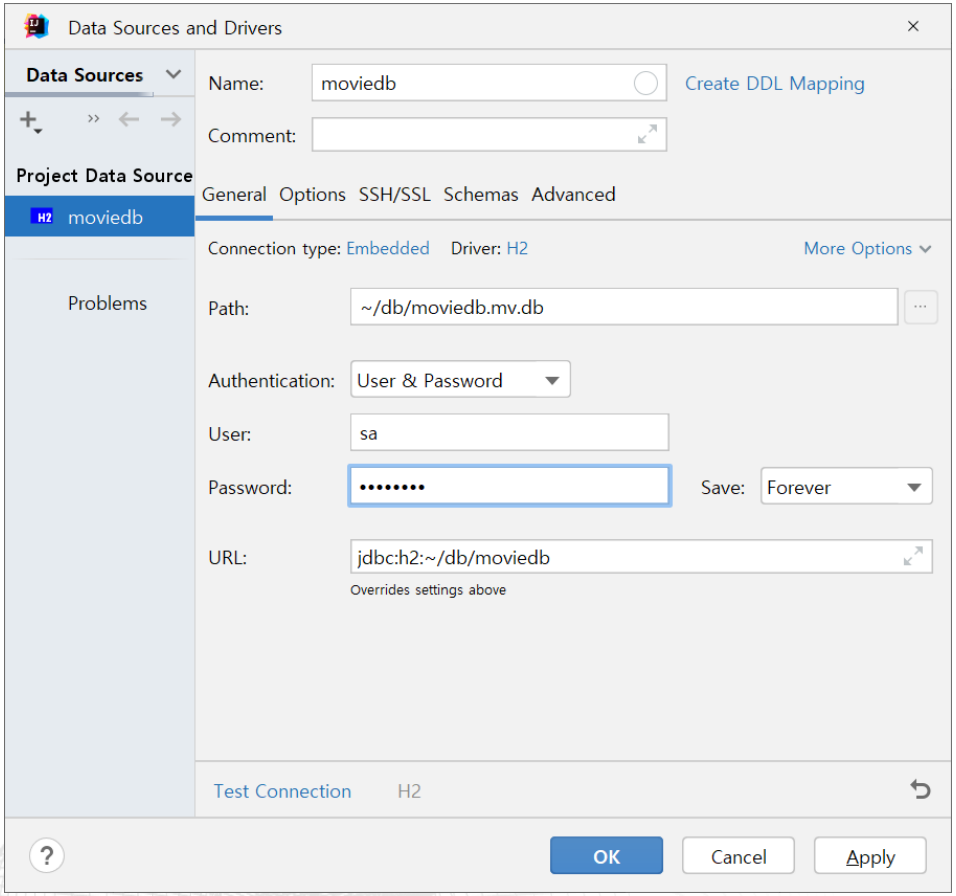
- resources 폴더에 테이블 및 뷰 생성 관련 DDL 파일 작성

Data Source 설정

- Intellij 우측 'Database' 선택
- '+' -> 'Data Source' -> 'H2' 선택



Database 처리



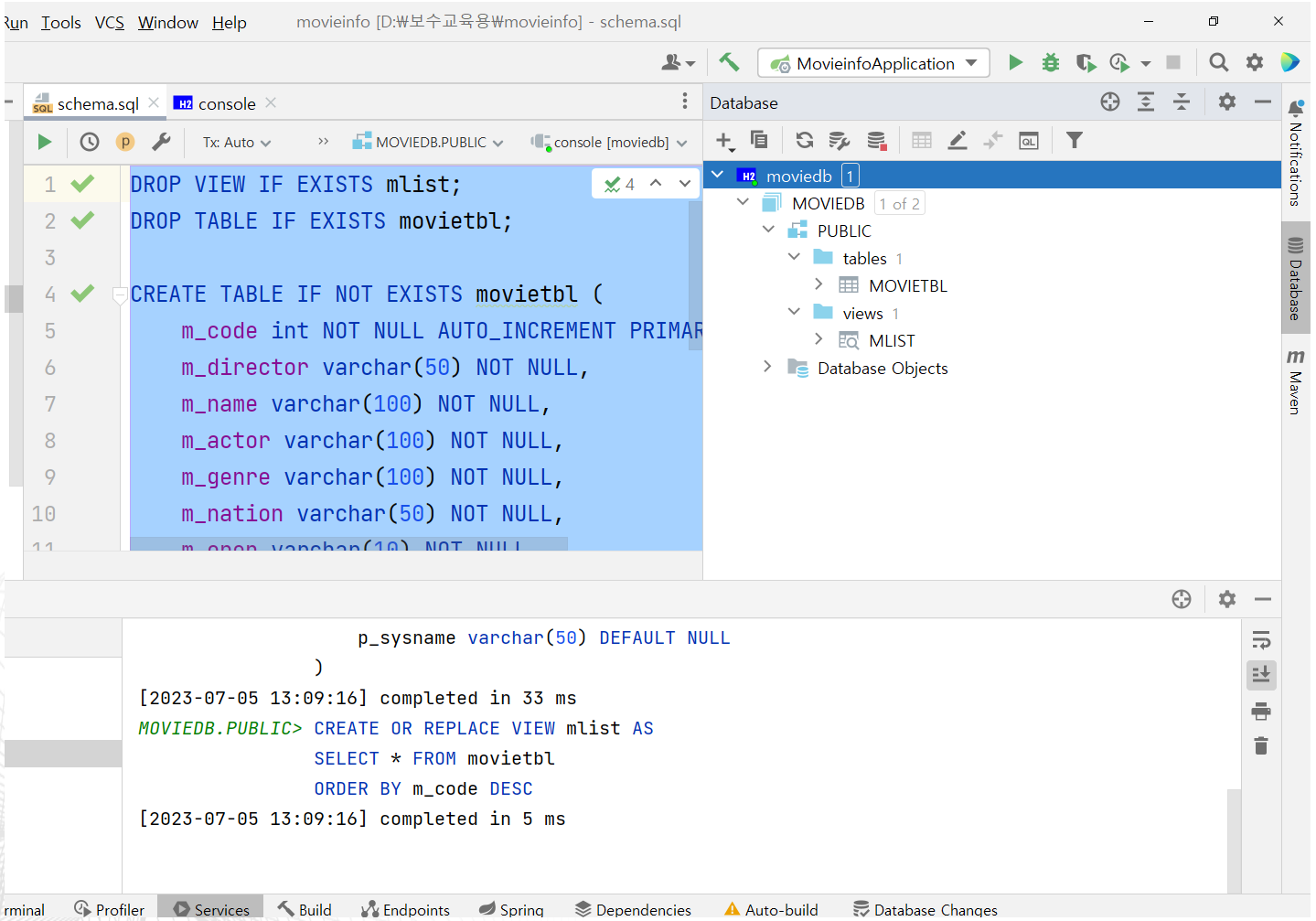
Data Source 설정

- Name : moviedb
- Path : ~/db/moviedb.mv.db
- User : sa
- Password : 12341234
- URL : jdbc:h2:~/db/moviedb

- URL을 변경하면 Name과 Path 가 함께 변경됨
- Test Connection을 수행하면 'C:/users/사용자명/'에 H2 DB 파일이 생성됨
- URL은 application.properties 에서 사용

Database 처리

SQL 스크립트를 실행하여 테이블 및 뷰 생성



MyBatis Framework

관계형 데이터베이스 프로그래밍을 좀 더 쉽게 할 수 있게 도와 주는 SQL Mapper 프레임워크

- SQL 문장과 소스코드를 분리하여 서비스 변화에 대한 유연성을 높이고, 소스코드의 가독성을 높여 개발 및 유지보수의 편의성을 제공

MyBatis 활용을 위한 설정

- 초기 Dependency 선택
  - SQL
    - MyBatis Framework 선택

MyBatis Framework

application.properties 설정

- DB 관련 설정
  - spring.datasource.driver-class-name=org.h2.Driver
  - spring.datasource.url=jdbc:h2:~/db/moviedb;MODE=MySQL
  - spring.datasource.username=sa
  - spring.datasource.password=12341234
- MyBatis 관련 설정
  - mybatis.type-aliases-package=com.icia.movieinfo.dto <- DTO 패키지
  - mybatis.mapper-locations=classpath:mappers/*.xml

MyBatis Framework

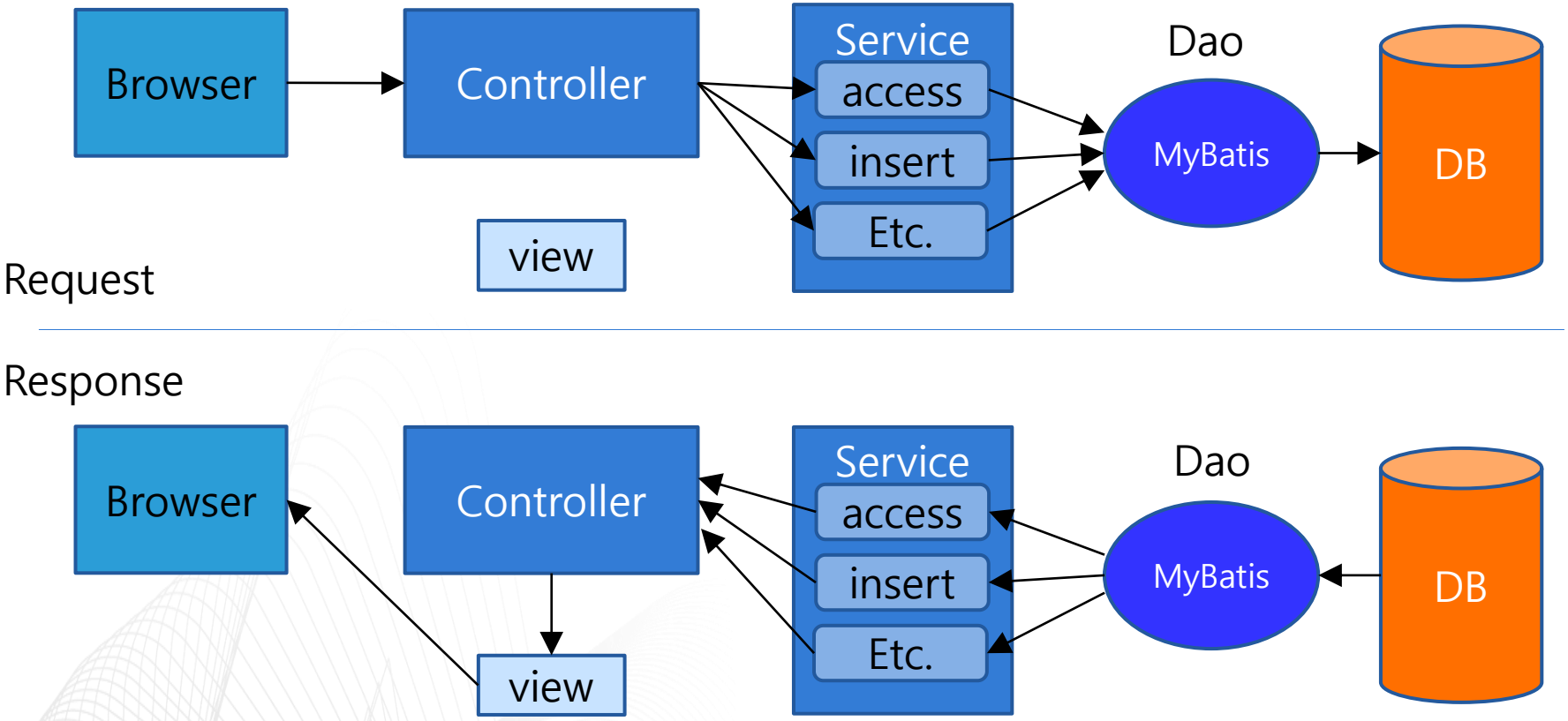
MyBatis Mapper

- 소스코드와 연동할 SQL 쿼리문을 작성하는 xml 문서
- resources > mappers 폴더를 생성하여 저장
  - MyBatisX 플러그인을 사용하면 좀더 편하게 mapper 문서를 작성할 수 있음.
  - File > Setting
    - Plugins에서 mybatis 검색 후 설치

DB access를 위한 코드 작업

- dao 패키지에 해당 코드 작업을 위한 interface 생성
  - MyBatis 프레임워크가 mapper의 SQL과 interface 코드를 기반으로 하여 자동으로 처리
  - interface에 서비스 기능을 수행할 수 있는 CRUD 메소드 선언
  - mapper에 CRUD를 위한 SQL query문 작성

Spring Data 처리 프로세스



10교시

페이징 처리 기능 구현 및 기타

Paging(Pagination)

한 HTML 문서에 보여져야할 내용이 많을 경우, 여러 페이지로 나눠서 보여주는 방식

- 구글 예)

- 전체 데이터를 한 화면에 보일 데이터의 수로 분할하여 각 화면 당 번호를 붙이고, 각 번호에 링크를 연결하는 방식.
  - <a> 태그를 사용하여 링크로 연결.
- 페이징 객체는 분할한 데이터에 링크를 연결하는 html 코드를 생성하는 객체.

메시지 처리 - RedirectAttribute

Redirect Vs. Forward

- 웹 애플리케이션에서 페이지를 전환하는 방식
- 페이지를 전환하는 주체에 따른 분류
  - Redirect - client
    - Server가 client가 요청한 uri의 응답으로 다른 uri에 접속하라고 명령, client는 명령에 따라 페이지를 전환(Response로 처리)
  - Forward - server
    - Server 내부에서 처리의 결과에 따라 다른 uri로 페이지를 전환(Request로 처리)
- Redirect 방식에서는 View에 메시지를 담을 수 없음
  - model에 담을 경우 1회성 메시지가 페이지에 계속 유지됨

RedirectAttribute는 redirect 방식에서 1회성의 임시 데이터(메시지) 전송을 위해 사용

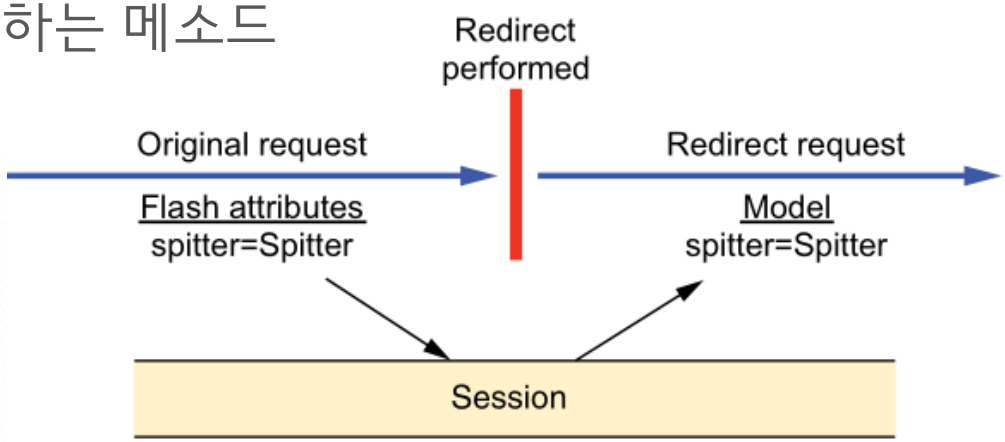
메시지 처리 - RedirectAttribute

동작 방식

- Redirect가 발생하기 전에 모든 1회성 메시지를 session에 복사
- Redirection 이후에는 저장된 메시지를 session에서 model로 이동시킴.
- 헤더에 파라미터를 붙이지 않기 때문에 URL에 노출되지 않음.

addFlashAttribute()

- RedirectAttributes가 제공하는 메소드.
- Redirect 직전 플래시에 저장하는 메소드
- Redirect 이후에는 소멸



File upload : Multi-part

Multi-part란 한 웹 문서 안에 문자와 2진 데이터(이미지, 동영상, 오디오 및 다양한 형식의 파일)가 같이 포함되어 있는 형태.

application.properties 설정(파일 허용 용량 설정)

- spring.servlet.multipart.max-file-size=5MB -> 개별 파일 크기
- spring.servlet.multipart.max-request-size=25MB -> 전체 요청 크기

form 태그

- enctype="multipart/form-data" 속성을 작성(must!)
- input 태그 type 속성을 'file'로 작성(파일 선택 창 기본 제공)

Error 페이지

처리가 필요한 HTTP 상태 코드

- 200 : 정상 코드(처리할 필요 없음. 제대로 잘 처리되었음을 나타냄)
- 404 : 사용자가 요청한 페이지 없음
  - 서버 리뉴얼 작업 등의 이유로 페이지파일이 소실되거나 uri 매핑이 어긋남
  - 사용자의 오타 등
- 500 : 서버 오류
  - 웹 프로그램 소스 코드 실행 시 예외사항이 발생되는 경우

Error 페이지

application.properties 설정

- server.error.whitelabel.enabled=false
- server.error.path=/error/
- server.error.include-message=always
- server.error.include-exception=true
- server.error.include-stacktrace=always

templates 폴더 아래에 error 폴더 생성

- 404 에러 대응 : 404.html 작성
- 500 에러 대응 : 500.html 작성

주요 Annotation

Annotation

- 특별한 목적의 주석
  - 코드 사이에 주석처럼 쓰여서 특별한 의미, 기능을 수행하도록 하는 기법

Spring Framework

- @SpringBootApplication
  - Spring boot 프로젝트의 auto-configuration을 담당
- @Controller
  - 작성한 클래스를 Spring의 컨트롤러 bean으로 등록
- @Service
  - 작성한 클래스를 Spring의 서비스 bean으로 등록
- @Autowired
  - IoC를 위한 어노테이션. 필요 시 Spring Framework가 해당 인스턴스를 주입
- @GetMapping/@PostMapping

주요 Annotation

Lombok

- @Getter/@Setter
  - Dto 클래스의 Getter/Setter 메소드 자동 생성
- @SLF4J
  - Lombok에서 제공하는 로그 기능을 사용할 수 있도록 제공
- @AllArgsConstructor
  - 모든 멤버변수의 초기값을 받아서 처리하는 생성자 자동 생성

MyBatis

- @Mapper
  - Dao 클래스와 CRUD Mapping용 XML 문서를 연결
- CRUD용 annotation(mapper xml을 사용하지 않을 경우)
  - @Select, @Insert, @Delete, @Update 등
    - 예) @Select("SELECT * FROM table")

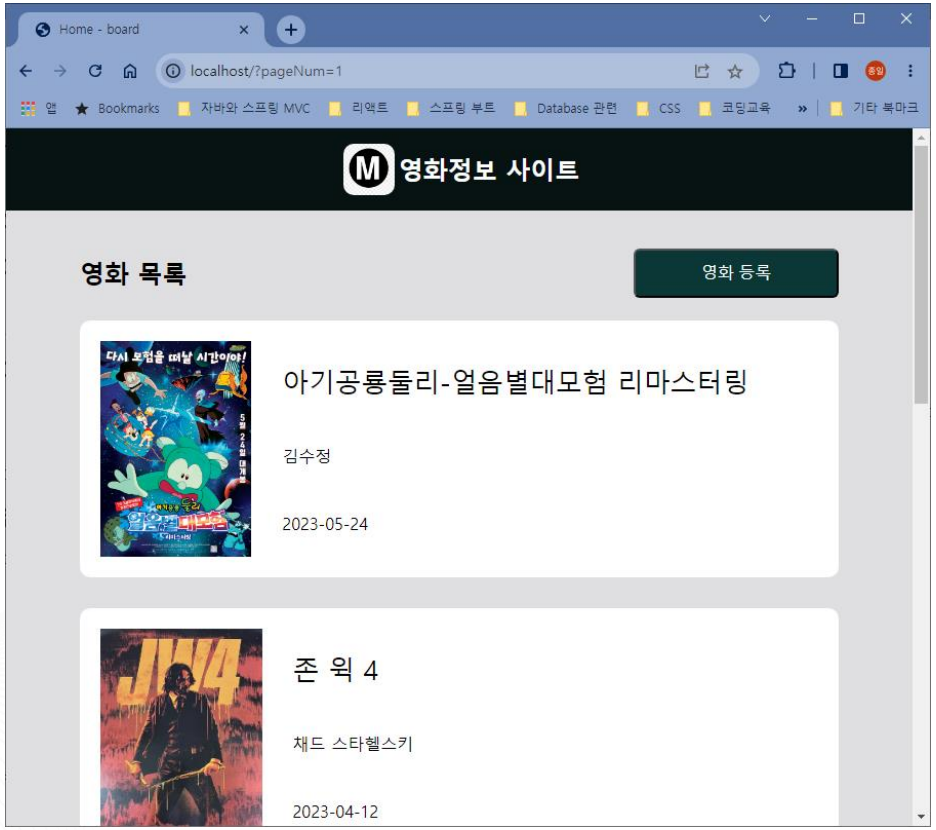
11 - 12 교시

실습 프로젝트

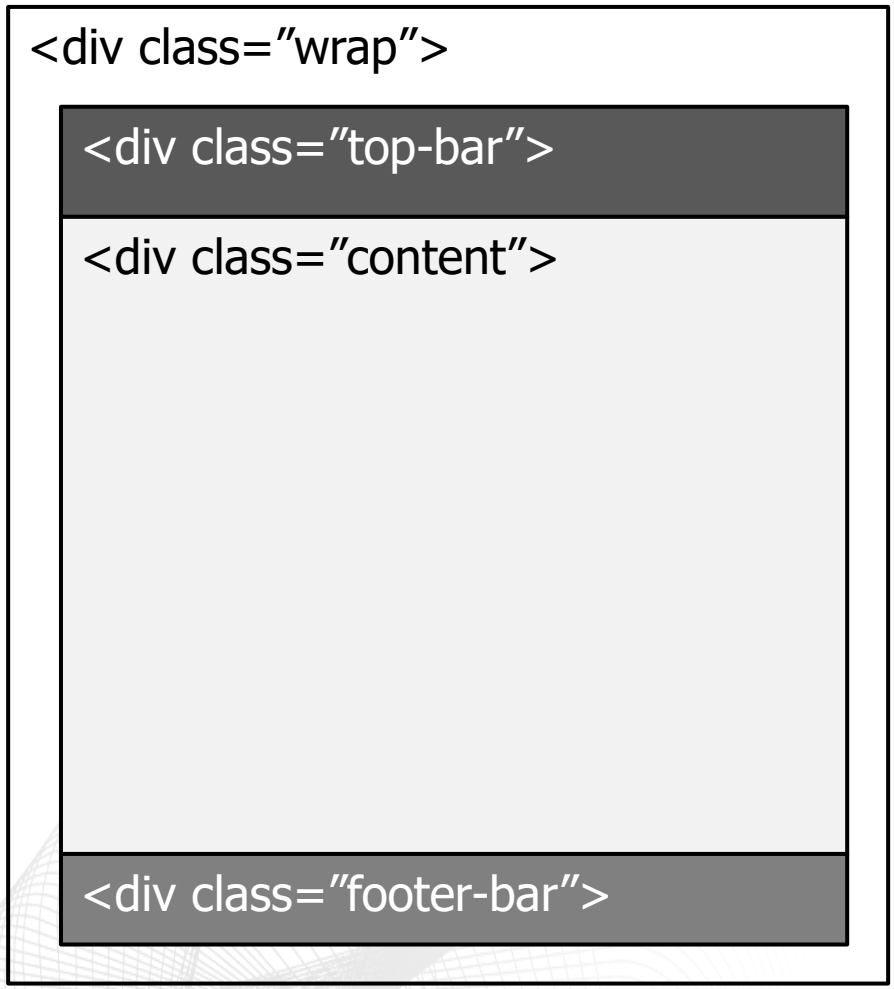
Project : 영화 정보 사이트

MVC 패턴을 활용한 Spring Boot 프로젝트

- 제공 기능
  - 영화 정보 등록
  - 영화 정보 조회
  - 영화 정보 수정
  - 영화 정보 삭제



CSS 적용을 위한 HTML 구성



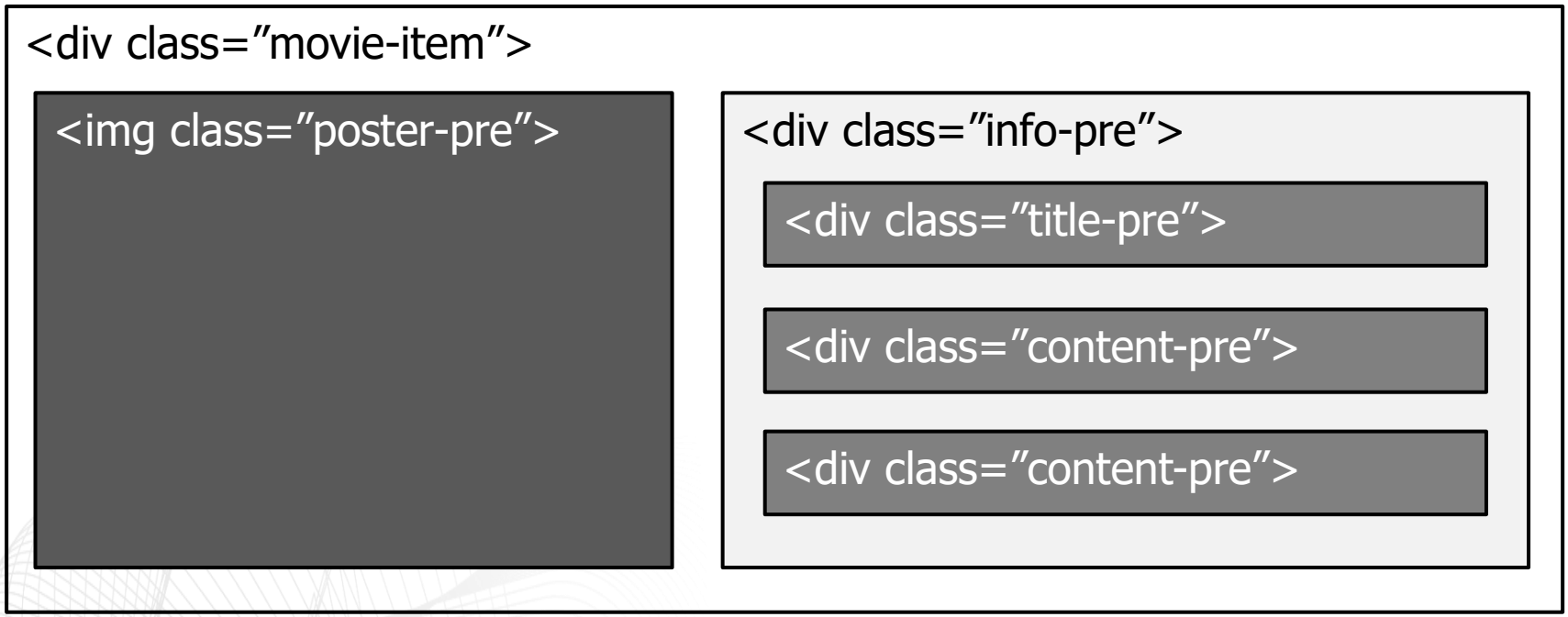
페이지 공통 구성

- 모든 페이지의 구성은 동일
- header.html과 footer.html은 같은 페이지를 insert 함.

CSS 적용을 위한 HTML 구성

영화 목록

- 모든 영화 정보의 구성은 동일



CSS 적용을 위한 HTML 구성

상세 보기