

MVC 패턴 이해와 실습

2.1 뷰 템플릿과 MVC 패턴

1) 뷰 템플릿(View Template)

- 화면을 담당하는 기술
- 웹 페이지(View)를 하나의 틀(Template)로 만들고 여기에 변수를 삽입해 서로 다른 페이지로 보여줌
- 머스테치(Mustache) : 뷰 템플릿을 만드는 도구

2) MVC 패턴

- 컨트롤러(Controller) : 클라이언트의 요청에 따라 서버에서 이를 처리하는 역할
- 모델(Model) : 데이터를 관리하는 역할
- MVC 패턴(Model-View-Controller Pattern) : 웹 페이지를 화면에 보여주고(View), 클라이언트의 요청을 받아 처리하고(Controller), 데이터를 관리하는(Model) 역할을 나누는 기법

2.2 MVC 패턴을 활용해 뷰 템플릿 페이지 만들기

1) 뷰 템플릿 페이지 만들기

- src > main > resource > templates 디렉터리에 생성
- 뷰 템플릿 엔진인 mustache를 사용
 - 그외 템플릿 엔진으로 Thymeleaf, JSP 등이 있음

2) 컨트롤러 만들고 실행하기

- src > main > java 디렉터리의 com.example.firstproject에 패키지로 생성
- @Controller 어노테이션 : Controller 클래스 패키지 (org.springframework.stereotype.Controller)가 자동으로 импорт
 - *어노테이션(annotation) : 소스 코드에 추가해 사용하는 메타 데이터의 일종
 - 앞에 @ 기호를 붙여 사용
 - 메타 데이터 : 프로그램에서 처리해야 할 데이터가 아니라 컴파일 및 실행 과정에서 코드를 어떻게 처리해야 할지 알려 주는 추가 정보
- 메서드로 페이지를 반환하려면 `return "파일명";` 으로 작성하면 서버가 알아서 '파일명.mustache' 파일을 찾아 웹 브라우저로 전송
- 페이지를 반환해 달라는 URL 요청을 접수
 - 메소드 앞에 @GetMapping() 추가
 - 자동으로 org.springframework.web.bind.annotation.GetMapping 패키지가 임포트됨

- 괄호 안에 URL 주소를 넣음
 - ex) "/hi"

🐛 한글 깨짐 현상(???, ?????!) 발생 시 해결

- src > main > resources > application.properties 파일에 코드 추가
 - server.servlet.encoding.force=true

3) 모델 추가하기

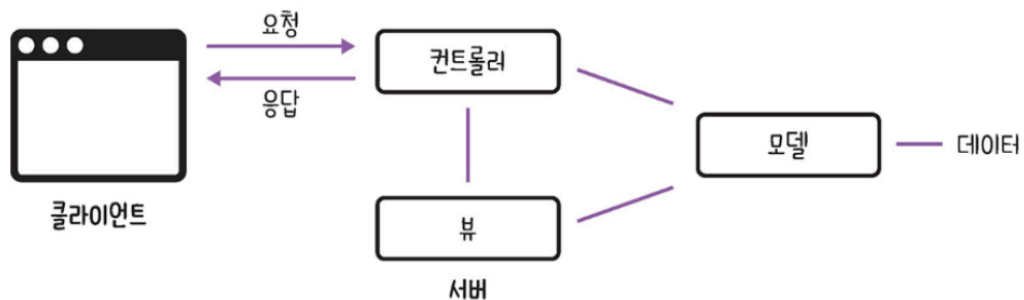
- 머스태치 문법을 사용해 뷰 템플릿 페이지에 변수 삽입
 - {{변수명}}
 - 변수명을 사용하면 변수값에 따라 결과가 다르게 출력됨
 - 변수명에 해당하는 변수를 찾을 수 없어 에러가 생김 → 모델 사용
 - 메서드에 Model 타입의 model 매개변수를 추가 ⇒ Model 클래스 패키지가 자동으로 임포트됨
- addAttribute() 메서드 : 모델에서 변수를 등록
 - model.addAttribute("변수명", 변수값)
 - ex) model.addAttribute("username", "헤림");



MVC 패턴 요약

뷰 페이지 만들기 : greetings.mustache
 ⇒ 컨트롤러 만들기: FirstControlller.java
 ⇒ 컨트롤러에서 뷰 페이지 반환하기: return "greetings";
 ⇒ 뷰 페이지에 변수 삽입하기 : {{username}}
 ⇒ 컨트롤러에 모델 추가하기 : niceToMeetYou(Model model)
 ⇒ 모델에서 변수 등록하기 : model.addAttribute("username", "헤림");

2.3 MVC의 역할과 실행 흐름 이해하기



페이지 실행에서 컨트롤러의 동작


- 컨트롤러 선언

- URL 요청 접수
- 메서드 수행
- 모델 객체 가져오기
- 모델 변수 등록
- 뷰 템플릿 페이지 반환


2.4 뷰 템플릿 페이지에 레이아웃 적용하기

레이아웃(layout) : 화면에 요소를 배치하는 일

- 헤더-푸터 레이아웃(header-footer layout)
 - 상단의 헤더(header) 영역에는 사이트 안내를 위한 내비게이션을 넣고, 하단의 푸터(footer)영역에는 사이트 정보, 두 영역 사이에는 사용자가 볼 핵심 내용인 콘텐츠(content)를 배치

 부트스트랩(Bootstrap) : 웹 페이지를 쉽게 만들 수 있도록 작성해 놓은 코드 모음

- 각종 레이아웃, 버튼, 입력창 등 디자인을 미리 구현해 놓은 것

 뷰 템플릿 파일을 불러 올 때는 `{{>파일명}}`으로 작성