

# 2장 뷰 템플릿과 MVC 패턴

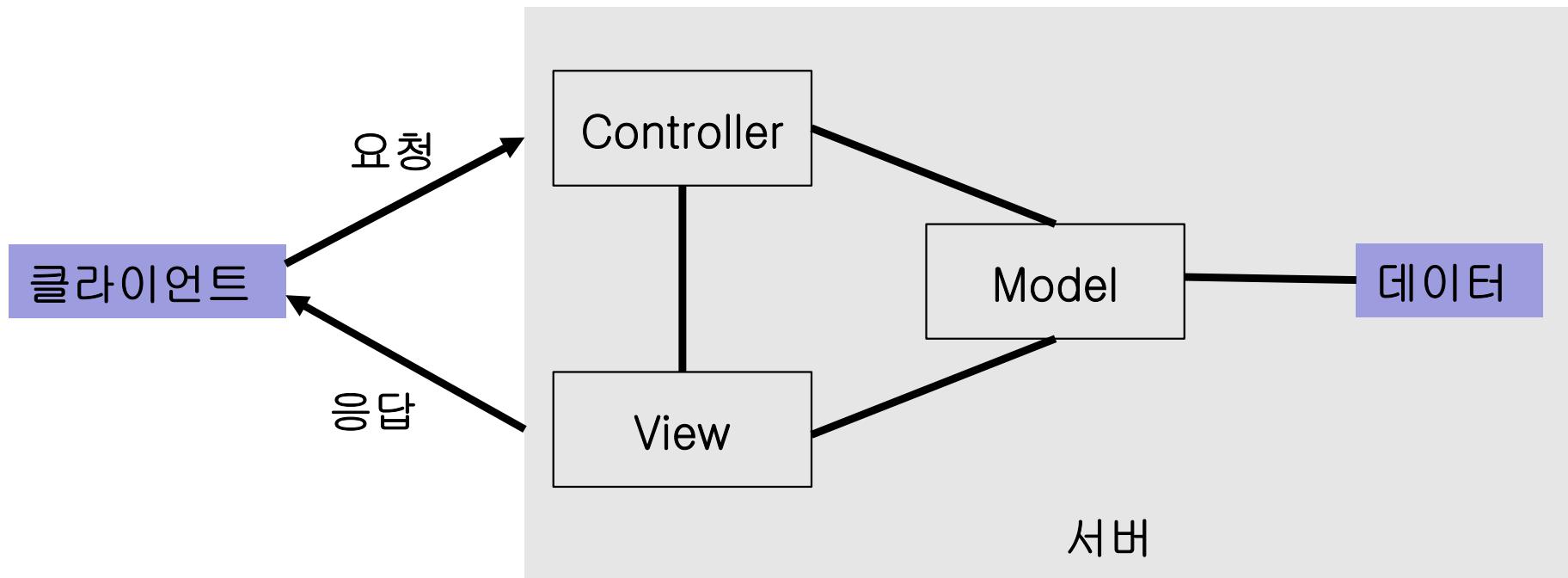
출처: 코딩 자율학습 스프링부트3 자바 백엔드 개발 입문, 홍팍, 길벗, 2023

# 뷰 템플릿(View Template)

- 화면을 담당하는 기술
- 웹 페이지(view)를 하나의 텀(Template)로 만들고, 여기에 변수를 삽입하여 서로 다른 페이지로 보여 줄 수 있게 하는 기술
- 주요 뷰 템플릿 엔진
  - 타임리프(Thymeleaf)
  - 머스테치(Mustache)
  - 아파치 프리마커(FreeMarker)
  - JSP
  - ...
- 머스테치(Mustache)
  - 클라이언트/서버 템플릿을 하나의 문법으로 모두 사용 가능
  - 단순한 문법
  - 다양한 언어 지원
  - 로직 코드 제한으로 뷰(화면)과 서버의 역할을 명확히 분리

# MVC(Model-View-Controller) 패턴

- 웹 애플리케이션을 화면에 보여주고(View), 클라이언트의 요청을 받아 처리하고(Controller), 데이터를 관리(Model)하는 세가지 역할로 나눠서 구성
  - 모델(Model): 데이터를 관리하는 역할
  - 뷰(View) : 웹 페이지를 화면에 보여 주는 역할
  - 컨트롤러(Controller): 클라이언트의 요청을 받아 처리하고, 이를 처리하는 역할



# MVC(Model-View-Controller) 패턴

- MVC 패턴의 장점

- 역할 분리로 유지보수가 쉬움
  - ✓ 모델, 뷰, 컨트롤러의 한 부분을 수정해도 다른 부분에 영향이 최소화
- 협업에 유리
  - ✓ 개발자는 모델과 컨트롤러, 디자이너는 뷰로 나눠 협업
- 재사용성
  - ✓ 같은 모델을 여러 뷰에서 재사용
- 확장성 향상
  - ✓ 새로운 기능 추가 시 기존 구조를 크게 변경하지 않아도 됨
- 테스트 용이성
  - ✓ 로직이 분리되어 있어 단위 테스트가 용이
  - ✓ 컨트롤러와 모델을 독립적으로 테스트

# localhost:8080/hi 요청 처리 구현 과정

- 뷰 템플릿 만들기
  - template 디렉토리에 greeting.mustache 만들기
- 컨트롤러 만들기
  - com.example.firstproject.controller 패키지의 FirstController.java에 greeting() 메소드 만들기
- 모델 사용하기
  - 컨트롤러의 호출 메소드의 매개변수로 Model 객체 받아오기
  - Model 객체에 뷰 템플릿에 전달할 변수 등록하기

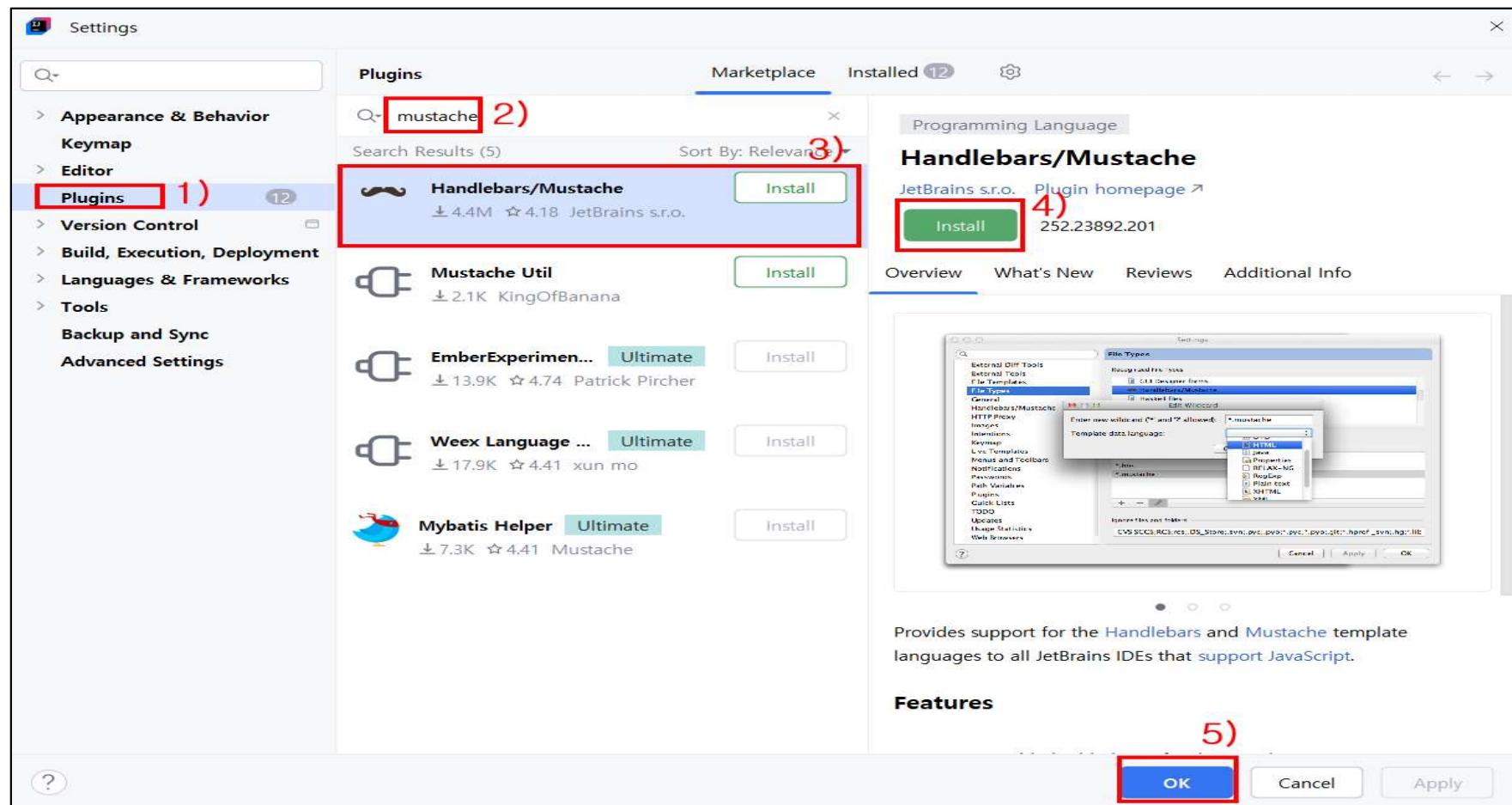
# Mustache로 뷰 템플릿 페이지 만들기

- 뷰 템플릿 파일을 저장하는 디렉토리
  - [src - main - resources - templates] 디렉토리
- 파일 확장자: .mustache

# Mustache로 뷰 템플릿 페이지 만들기

- IntelliJ IDEA에서 머스테치 플러그인 설치

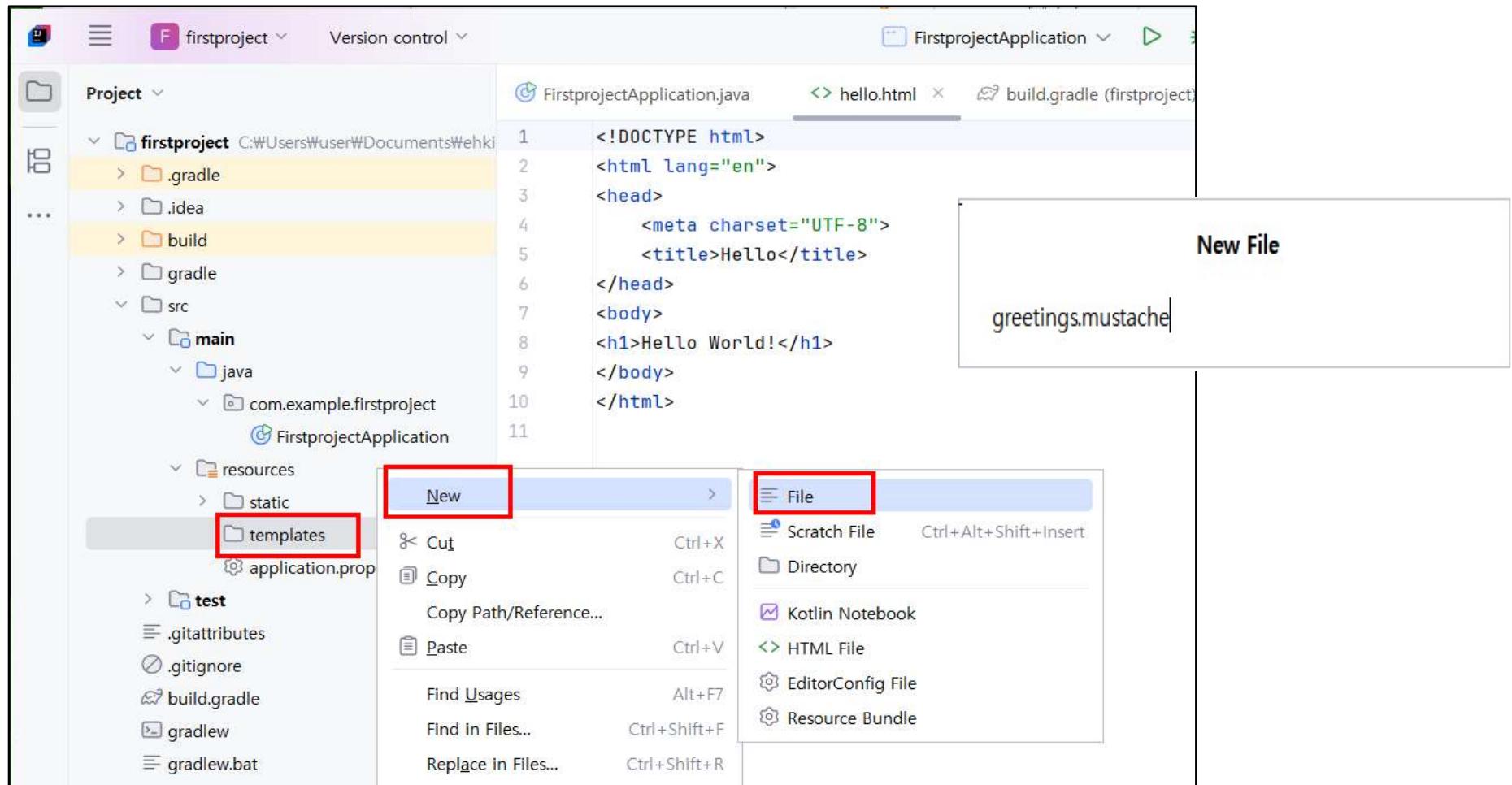
- 메뉴 [ File - Settings ]
- 왼쪽 목록에서 [plugins]
- [Marketplace] 탭에서 mustache 검색 - [Handlebar/Mustache] 선택
- [Install]



# Mustache로 뷰 템플릿 페이지 만들기

- **greetings.mustache 파일 만들기**

- [src - main - resources - templates] 디렉토리에서 마우스 오른쪽 버튼 클릭
- 컨텍스트 메뉴에서 [New - File] 클릭
- New File 창에 ‘greetings.mustache’ 입력 후, 엔터키



# Mustache로 뷰 템플릿 페이지 만들기

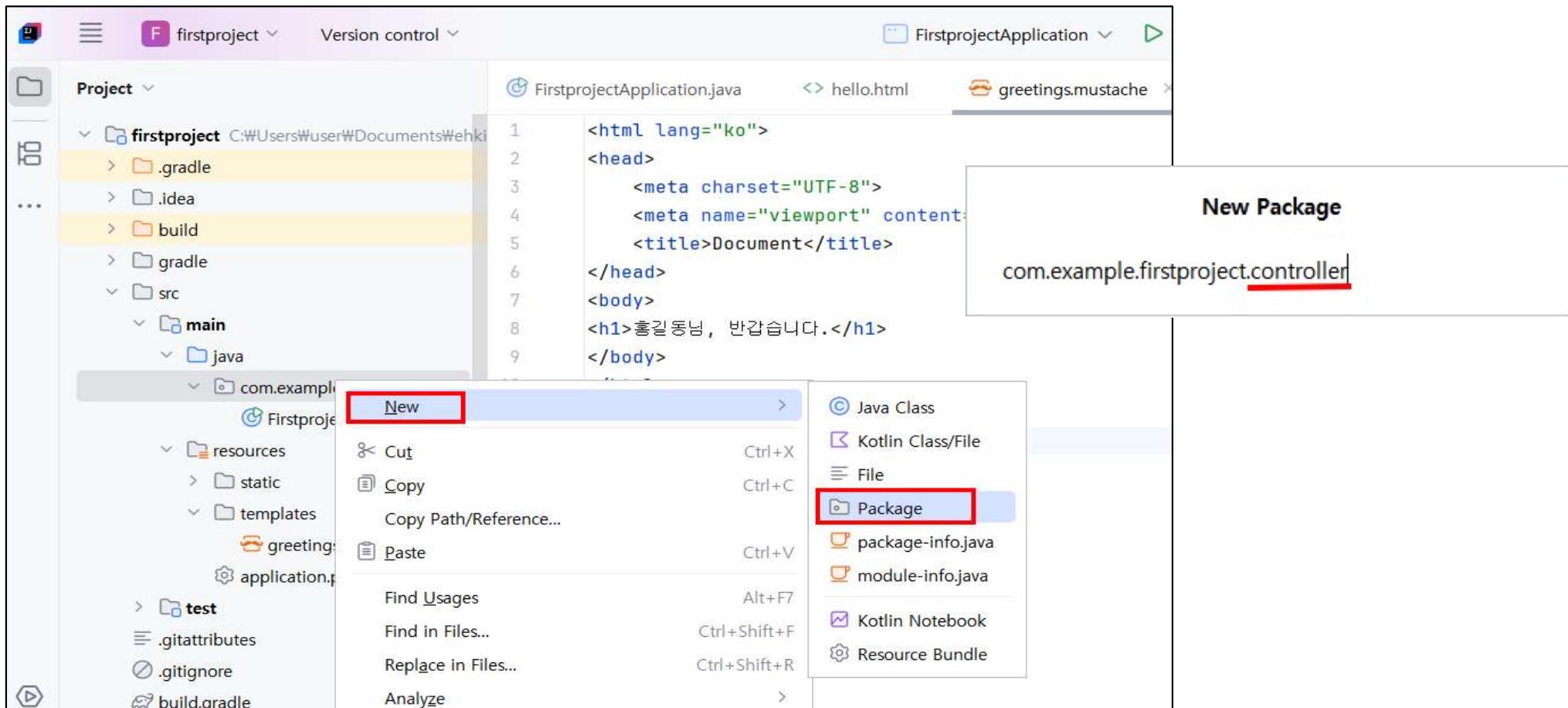
- greetings.mustache 파일 편집
  - 첫 줄 왼쪽에 ‘doc’ 입력한 후, tab 키 → 기본 HTML 코드 자동 완성
  - 아래와 같이 내용 추가

```
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Document</title>
</head>
<body>
    <h1>홍길동님, 반갑습니다.</h1>
</body>
</html>
```

# 컨트롤러 만들고 실행하기

- controller 패키지 만들기

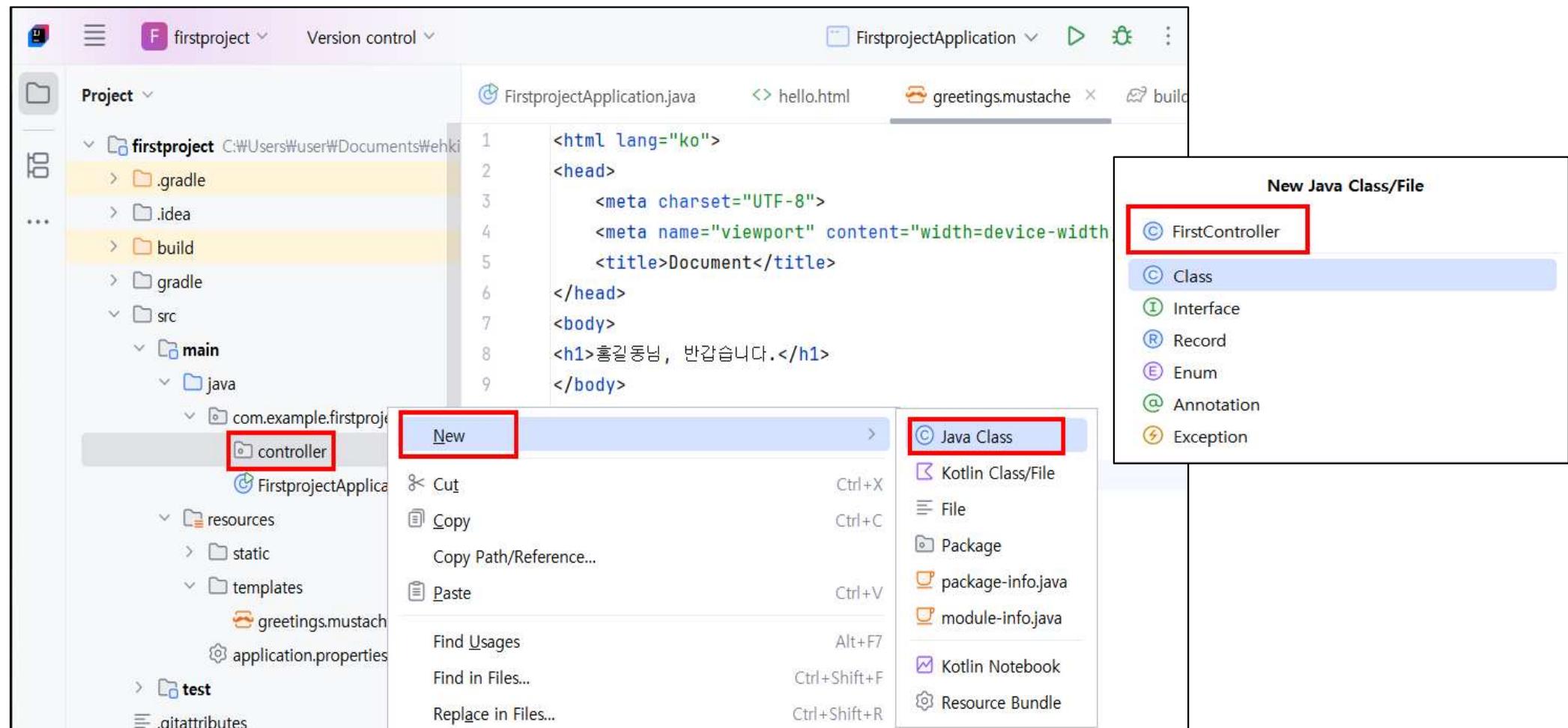
- [src - main - java] 디렉토리 아래에 있는 기본 설정 패키지 (com.example.firstproject)에서 마우스 오른쪽 버튼 클릭
- 컨텍스트 메뉴에서 [New - Package] 클릭
  - ✓ 패키지 이름으로 ‘controller’ 입력



# 컨트롤러 만들고 실행하기

- 컨트롤러 클래스 만들기

- Controller 패키지에서 마우스 오른쪽 버튼 클릭
- [New - Java Class] 클릭, 파일명: ‘FirstController’ 입력



# 컨트롤러 만들고 실행하기

- **@Controller**

- FirstController 클래스가 컨트롤러임을 선언하는 애노테이션(annotation)
- org.springframework.stereotype.Controller를 임포트

The screenshot shows the IntelliJ IDEA interface with the following details:

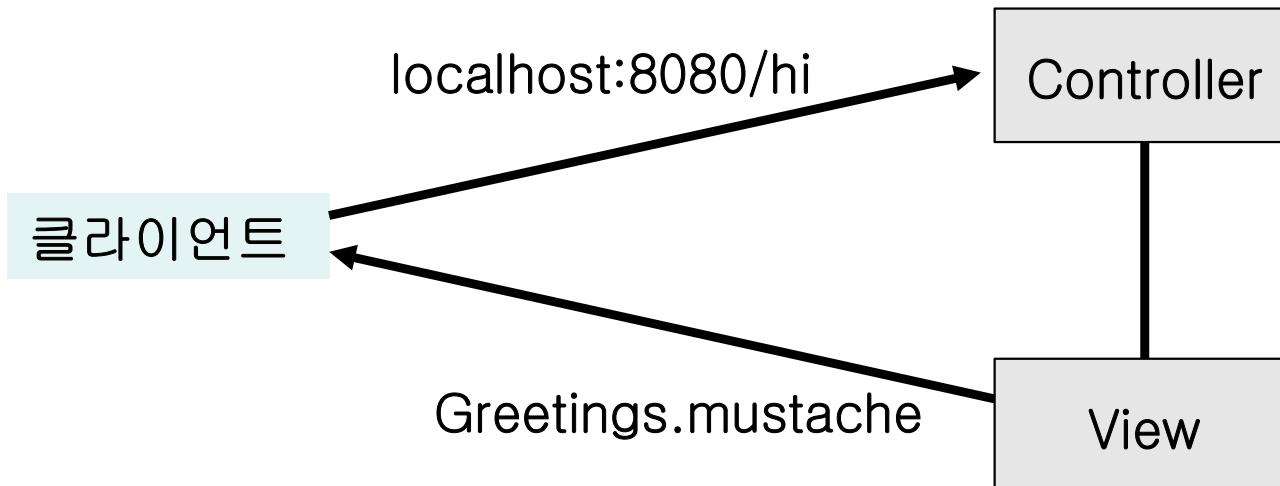
- Project View:** Shows the project structure under "firstproject". Key files include .gradle, .idea, build, gradle, src/main/java/com.example.firstproject/controller/FirstController.java, resources/templates/greetings.mustache, and application.properties.
- Code Editor:** The file `FirstController.java` is open. The code is as follows:

```
1 package com.example.firstproject.controller;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.GetMapping;
5
6 @Controller no usages
7 public class FirstController {
8
9     @GetMapping("/hi") no usages
10    public String niceMeetYou(){
11        return "greetings";
12    }
13 }
14 }
```

Annotations and imports are highlighted with red and purple underlines. The method `niceMeetYou()` and its return value "greetings" are highlighted with a green box.

# 컨트롤러 만들고 실행하기

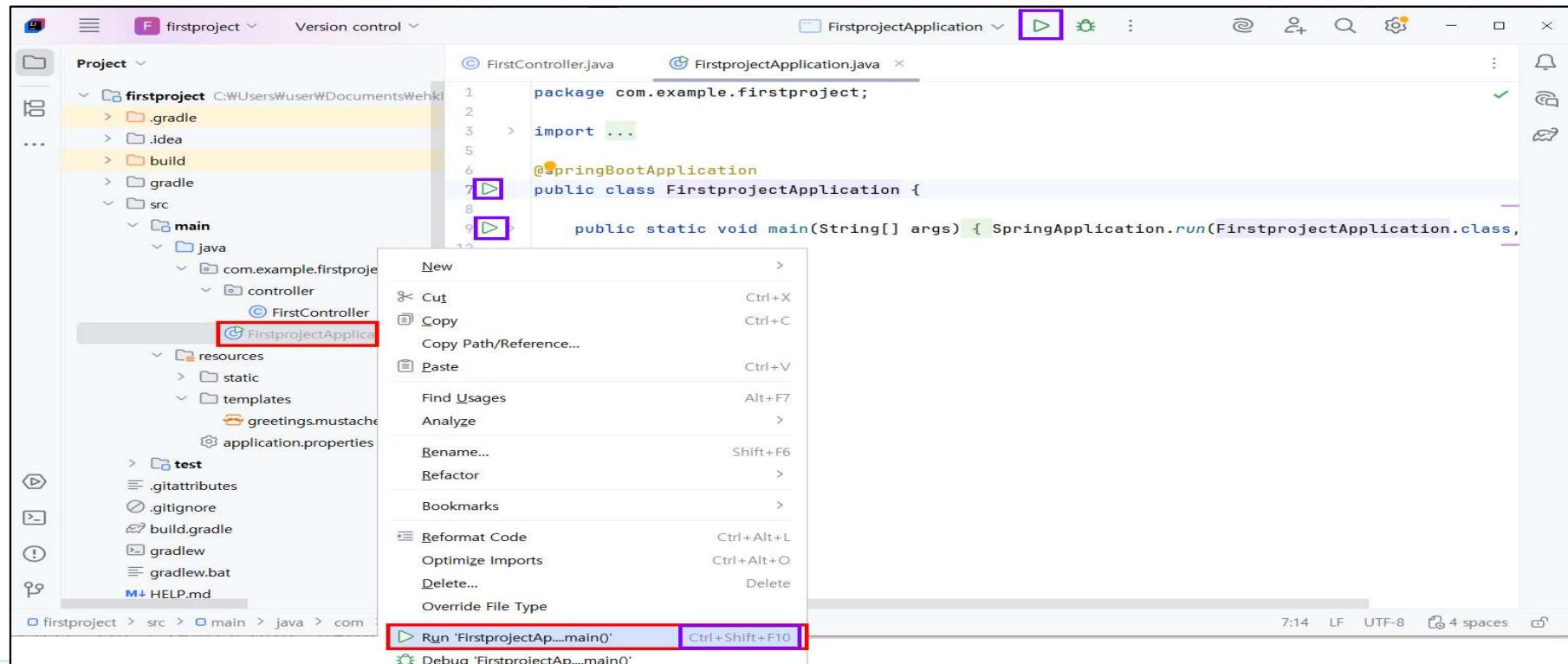
- **@GetMapping (“/hi”)**
  - 웹브라우저로 <http://localhost:8080/hi> 를 접속하면, 컨트롤러의 niceToMeetYou() 메소드를 호출
- **public String niceToMeetYou()**
  - 리턴값으로 greetings.mustache를 반환하려면 파일 이름인 ‘greetings’만 반환
  - 서버가 template 디렉토리에서 greetings.mustache 파일을 찾아 웹 브라우저로 전송



# 서버 실행 후, localhost:8080/hi 접속

- 서버 실행: FirstprojectApplication 실행

- [src - main - java - com.example.firstproject.controller - FirstController]에서 마우스 오른쪽 클릭
- 컨텍스트 메뉴에서 [Run 'FirstprojectAp…main()'] 클릭
  - ✓ 서버를 실행시키는 다른 방법
    - 단축기 사용: [Ctrl + Shift + f10]
    - FirstController.java 에디터 창에서 7번, 9번 라인의 ▶ 클릭
    - IntelliJ 타이틀 바의 ▶ 클릭



# 서버 실행 후, localhost:8080/hi 접속

서버 재시작

서버 종료

- 서버 실행 확인

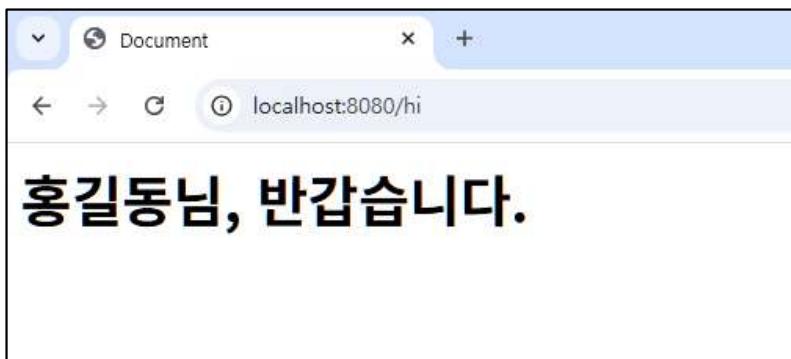
The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure with modules like .gradle, .idea, build, gradle, and src containing main, java, com.example.firstproject, controller, FirstController, and FirstprojectApplication.
- Code Editor:** Displays `FirstprojectApplication.java` with the following code:

```
package com.example.firstproject;  
import ...  
  
@SpringBootApplication  
public class FirstprojectApplication {  
  
    public static void main(String[] args) { SpringApplication.run(FirstprojectApplication.class, args); }  
}
```
- Run Tab:** Shows the run configuration for "firstproject [com.example.firstproject.FirstprojectApplication.main...]" with a green play button and the output window below.
- Output Window:** Displays the application logs:

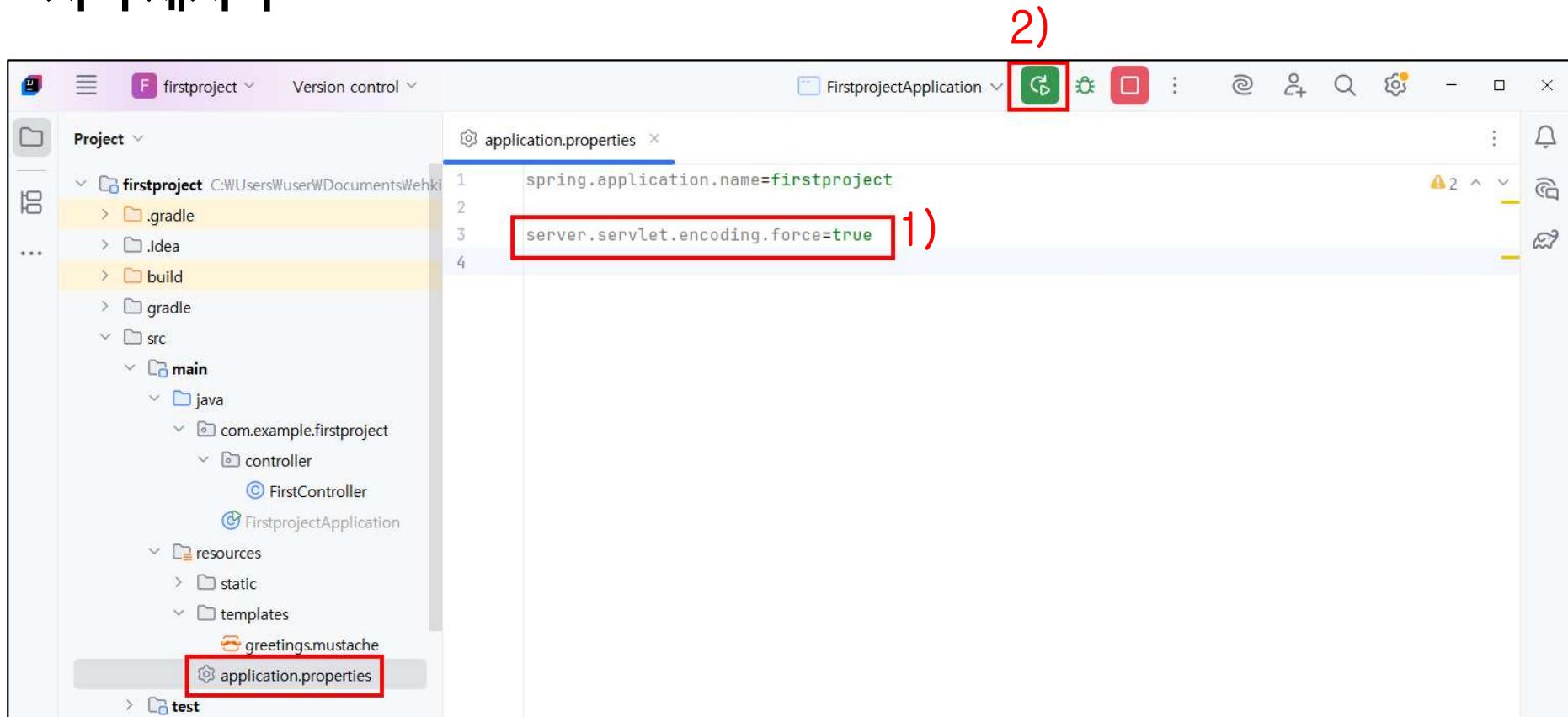
```
providerInitiator : HHH000021: Bytecode provider name : bytebuddy  
PlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.JtaPlatform]  
EntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'  
JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may not return until the end of the current HTTP request.  
TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''  
Application : Started FirstprojectApplication in 6.934 seconds (process running for 7.592)
```

- localhost:8080/hi 접속



# 실행 결과 한글이 깨질 때

- 실행결과 한글 깨짐 현상 발생(???, ????) 할 때
- [src - main - resource - application.properties] 파일에 server.servlet.encoding.force=true 추가
- 서버 재시작



# 모델 사용하기

- 머스테치 문법을 사용해 템플릿에 변수를 삽입

- 형식: `{{변수명}}`

`greetings.mustache`

```
<body>
<h1>{{username}}님, 반갑습니다.</h1>
</body>
```

# 모델 사용하기

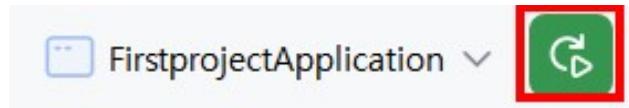
- 모델은 컨트롤러의 메서드에서 매개변수로 받아 옴
- 모델에서 뷰에 전달할 변수를 등록할 때는 `addAttribute()` 사용
  - 형식: `model.addAttribute("변수명", 변수값);`

FirstController.java

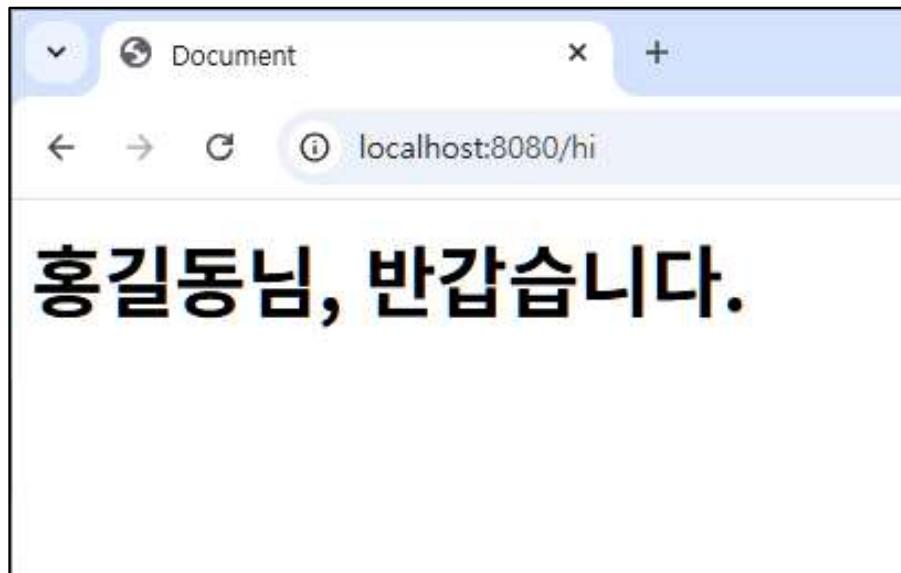
```
import org.springframework.ui.Model;  
...  
@GetMapping("/hi")  
public String niceMeetYou(Model model){ // model 객체 받아오기  
    model.addAttribute("username", "홍길동");  
    return "greetings";  
}  
...
```

# 서버 재실행 후, localhost:8080/hi 접속

- 타이틀 바에 있는 재실행(rerun) 버튼 클릭



- localhost:8080/hi 접속



# localhost:8080/bye 요청 처리 구현

- 구현 내용
  - 컨트롤러 구현
    - ✓ FirstController.java에 seeYouNext() 구현
  - 뷰 템플릿구현
    - ✓ templates 디렉토리 내에 goodbye.mustache 파일 구현
  - Model
    - ✓ 컨트롤러의 seeYouNext(Model model), 매개변수로 model 객체를 받아 옴
    - ✓ 뷰에 넘겨줄 변수를 model.addAttribute()를 사용하여 등록

# localhost:8080/bye 요청 처리 구현

- 컨트롤러 구현: FirstController.java에 추가

- URL 요청 접수: @GetMapping("/bye") 애노테이션 추가
- /bye 요청을 처리할 seeYouNext() 메서드 구현
- 메서드의 반환값으로 뷰 템플릿 페이지의 이름: 'goodbye'

```
@GetMapping("/bye")
public String seeYouNext(){
    return "goodbye";
}
```

- 뷰 템플릿 구현 : goodbye.mustache

- templates 디렉토리에서 마우스 오른쪽 버튼 클릭, [New - File]
- 파일명, 'goodbye.mustache' 입력
- Goodbye.mustache 파일 에디터의 첫 줄, 왼쪽에 doc 입력 후, 엔터키
- 아래 내용 추가

```
<body>
    <h1>{{nickname}}님, 다음에 또 만나요!</h1>
</body>
```

# localhost:8080/bye 요청 처리 구현

- model

- 모델객체 받아오기

- ✓ FirstController.java에서 setYouNext(Model model)로 수정

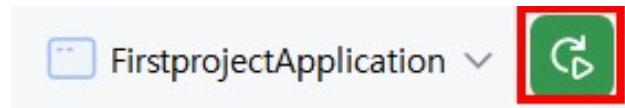
- 모델객체에서 뷰 템플릿 goodbye.mustache에 전달할 변수 등록
    - ✓ model.addAttribute("nickname", "홍번개");

FirstController.java

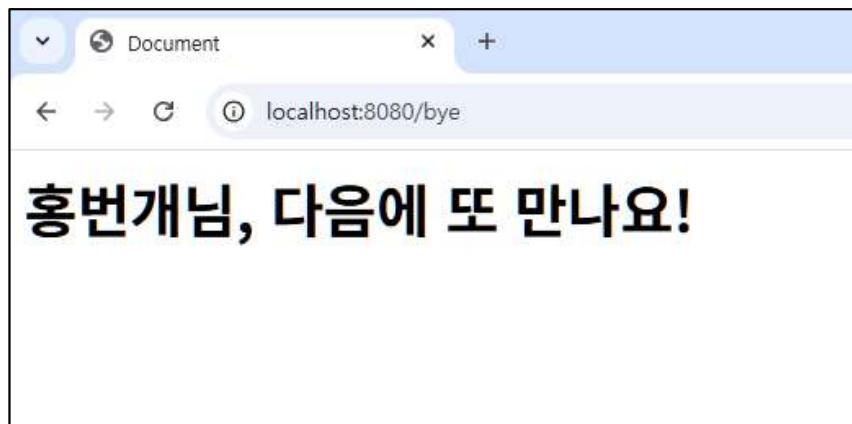
```
@GetMapping("/bye")
public String seeYouNext(Model model){
    model.addAttribute("nickname", "홍번개");
    return "goodbye";
}
```

# 서버 재실행 후, localhost:8080/bye 접속

- 타이틀 바에 있는 서버 재실행 버튼 클릭

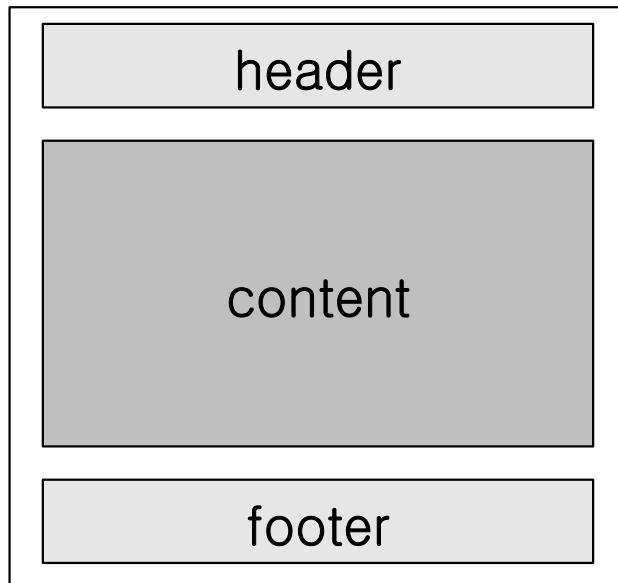


- localhost:8080/bye 접속



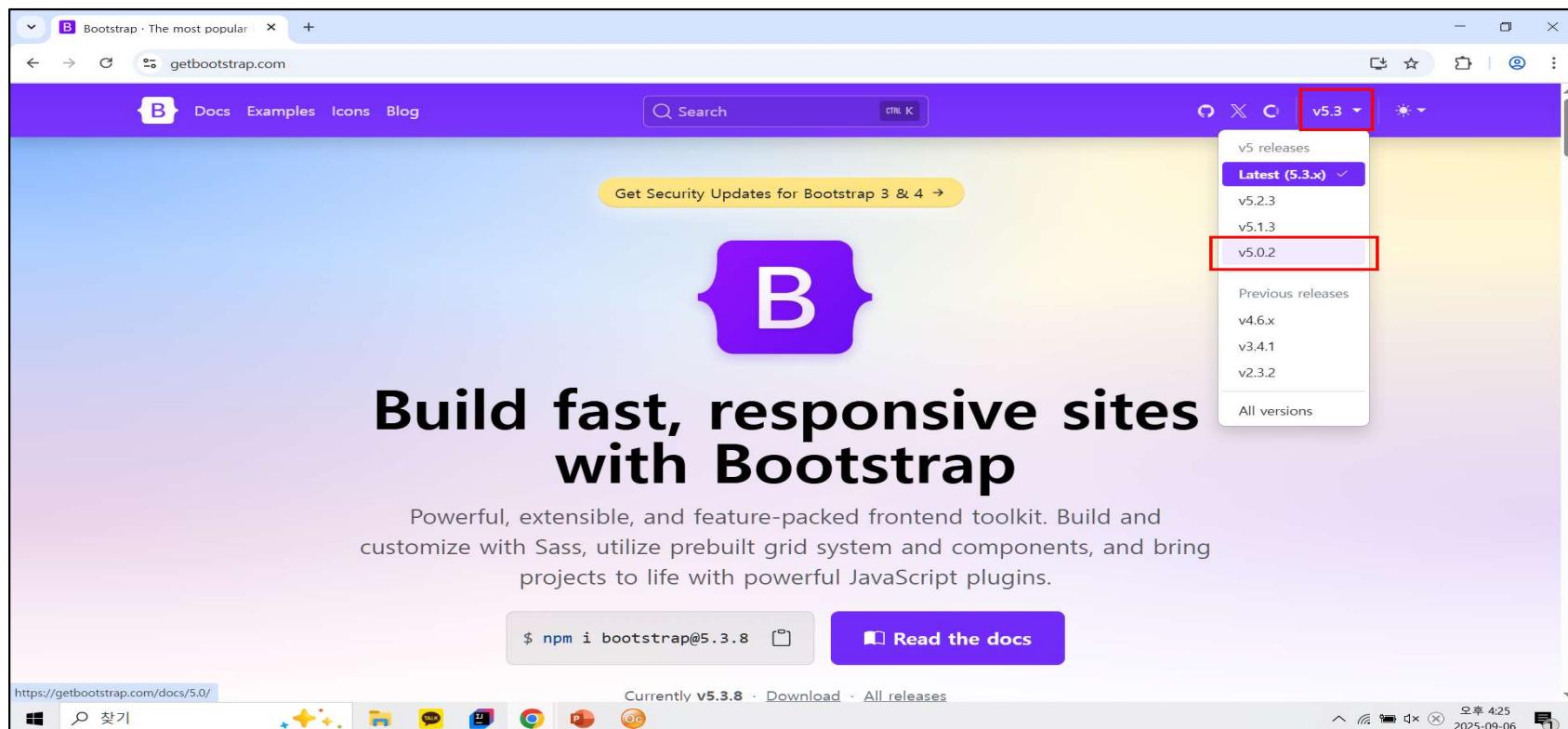
# 뷰 템플릿 페이지에 레이아웃 적용하기

- **레이아웃(layout)**
  - 화면에 요소를 배치하는 일
- 웹 페이지는 같은 요소로도 어떻게 배치하느냐에 따라서 페이지의 느낌이 달라짐
- **헤더-푸터 레이아웃(header-footer layout)**
  - 기본이 되는 레이아웃
  - 헤더 영역: 사이트를 안내를 위한 내비게이션 배치
  - 푸터 영역: 사이트 정보 배치
  - 컨텐츠 영역: 헤더와 푸터 사이에 위치, 사용자가 볼 핵심 내용 배치



# /hi 페이지에 헤더-푸터 레이아웃 적용하기

- **부트스트랩**
  - 웹 페이지를 쉽게 만들 수 있도록 작성해 놓은 코드 모음
  - 각종 레이아웃, 버튼, 입력창 등 디자인을 미리 구현해 놓음
  - 사용자는 제공되는 코드( html, css, 자바스크립트 코드)를 가져와 사용
- **부트스트랩(bootstrap) 사용하여 쉽고 빠르게 페이지를 작성**
- <https://getbootstrap.com> 접속, v5.0.2 선택



# /hi 페이지에 헤더-푸터 레이아웃 적용하기

- Bootstrap v5.0.0 페이지에서 아래로 스크롤하면, 스타터 템플릿(Starter template)이 보임. 오른쪽 위에 있는 [copy] 클릭

The screenshot shows a web browser window displaying the Bootstrap v5.0.0 documentation at [getbootstrap.com/docs/5.0/getting-started/introduction/](https://getbootstrap.com/docs/5.0/getting-started/introduction/). The left sidebar has sections like 'Getting started' (Introduction, Download, Contents, Browsers & devices, JavaScript, Build tools, Webpack, Parcel, Accessibility, RFS, RTL), 'Customize', and 'Layout'. The main content area features a heading 'Starter template' with a subtext about setting up pages with the latest standards. Below is a code snippet of an HTML document structure, and a 'Copy' button is highlighted with a red box. The right sidebar lists 'On this page' links including Quick start, CSS, JS, Bundle, Separate, Modules, Components, Starter template, Important globals, HTML5 doctype, Responsive meta tag, Box-sizing, Reboot, and Community.

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
    <h1>Hello, world!
  </body>

```

# /hi 페이지에 헤더-푸터 레이아웃 적용하기

- 스타터 템플릿을 greetings.mustache 템플릿의 기본 뼈대로 사용
  - greetings.mustache 파일의 모든 내용을 지우고, 복사한 starter template 내용을 붙여넣기 함
  - Option 주석은 지움

```
greetings.mustache ×

<!doctype html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet">

    <title>Hello, world!</title>
</head>
<body>
    <h1>Hello, world!</h1>

    <!-- Optional JavaScript; choose one of the two! --> 삭제
        <!-- Option 1: Bootstrap Bundle with Popper -->
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>

        <!-- Option 2: Separate Popper and Bootstrap JS --> 삭제
        <!--
        <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js" integrity="sha384-IQsoaAHFcsXkynj8e5b+D0L4zJugr7q9vZd4HrJ1ce4Fq33wvZd4HrJ1ce4Fq33wv" crossorigin="anonymous"></script>
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js" integrity="sha384-BSw1+4U9ePd28anT35cEz3Uxqet1nIwHcZq3OcfiDfC" crossorigin="anonymous"></script>
        -->
</body>
</html>
```

# /hi 페이지에 헤더-푸터 레이아웃 적용하기

- **greetings.mustache**를 3개의 레이아웃으로 나눔
  - 주석을 사용해서, navigation, content, site info 부분으로 나눔
  - <h1>hello, world</h1>은 삭제
  - Content 영역에 아래 내용 추가
    - ✓ <h1>{{username}} 님, 반갑습니다!</h1>

```
<body>
  <!-- navigation -->

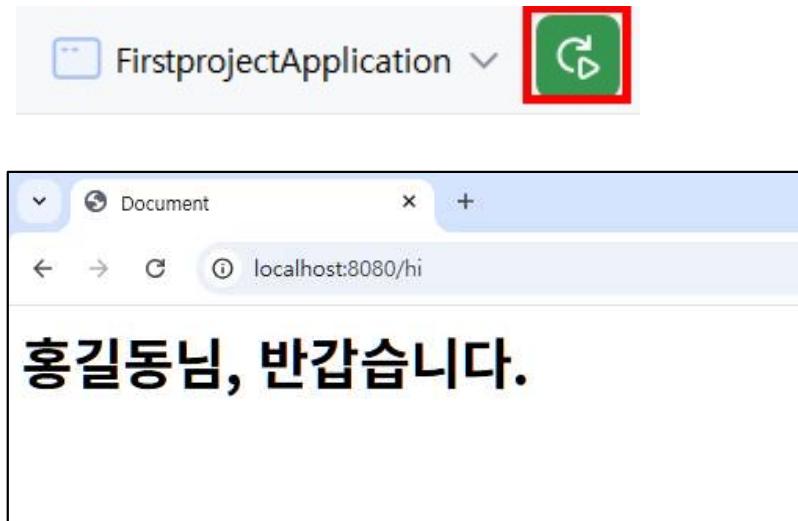
  <!-- content -->
  <h1>{{username}} 님, 반갑습니다.</h1>
  <!-- content -->

  <!-- site info -->

  <script src="https://cdn.jsdelivr.net/npm/bot-logger@1.0.0/dist/bot-logger.js"></script>
</body>
```

# /hi 페이지에 헤더-푸터 레이아웃 적용하기

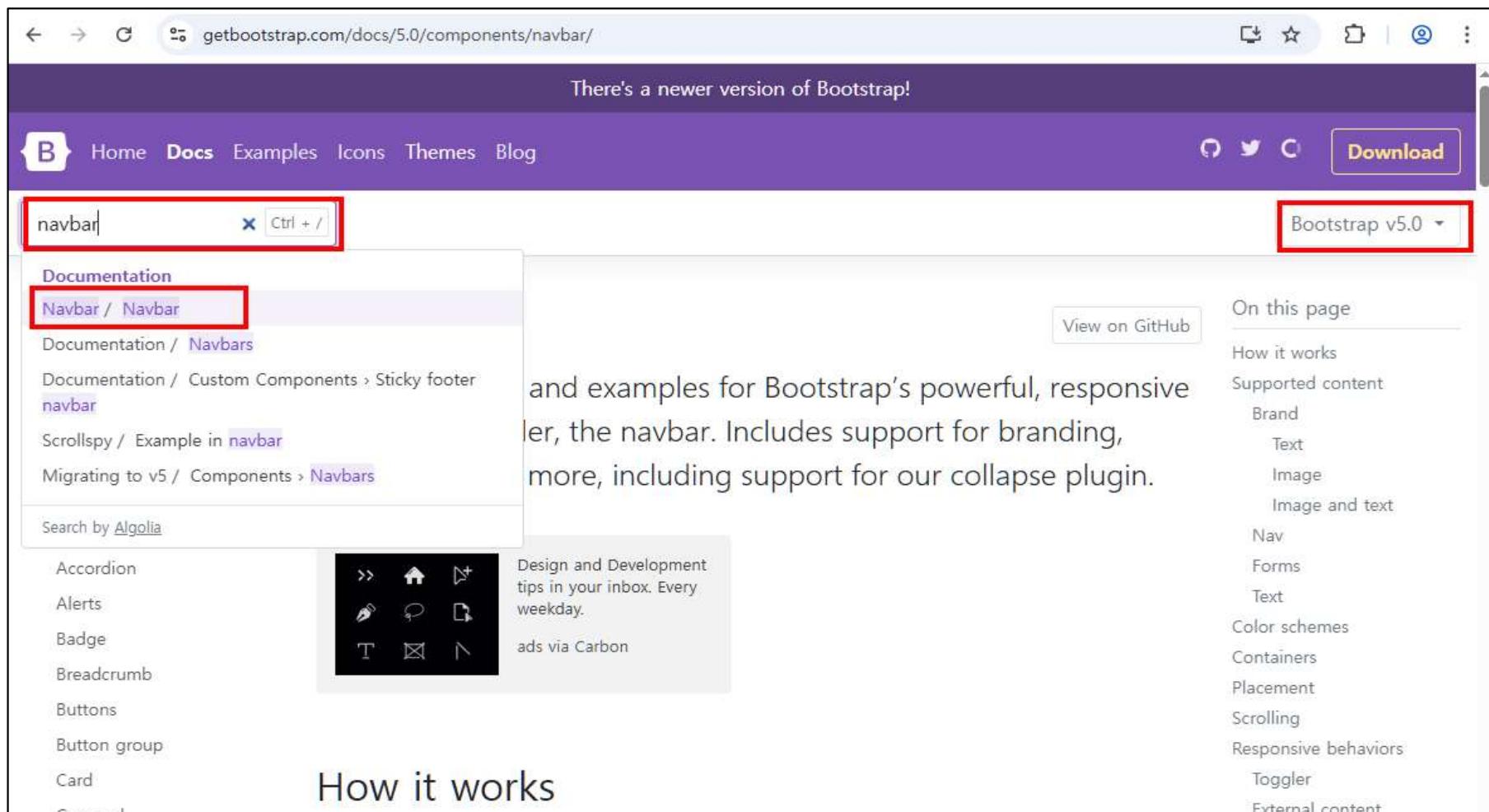
- 서버를 재시작하고, localhost:8080/hi 접속하여 결과 확인



# /hi 페이지에 헤더-푸터 레이아웃 적용하기

- 헤더 영역에 내비게이션 바 추가

- 부트스트랩 홈페이지(<https://getbootstrap.com>)에서, v5.0.2 선택
- 검색창에 ‘navbar’ 입력 후, 목록에서 Navbar/Navbar 선택



# /hi 페이지에 헤더-푸터 레이아웃 적용하기

- NavBar 페이지에서 마우스 스크롤 다운하면, 내비게이션 바의 예가 보임
- 반응형 네비게이션 바이므로, 브라우저 너비를 조정하면 모양이 바뀜
- [copy] 버튼 클릭

The screenshot shows the Bootstrap documentation for the Navbar component. The URL is [getbootstrap.com/docs/5.0/components/navbar/](https://getbootstrap.com/docs/5.0/components/navbar/). The page title is "Navbar". On the left, there's a sidebar with links like "Getting started", "Customize", "Layout", "Content", "Forms", and "Components". The "Components" section is expanded, showing "Accordion", "Alerts", "Badge", "Breadcrumb", "Buttons", and "Button group". A red arrow points from the "Components" link in the sidebar down to the "Components" heading on the main content page. The main content area has a purple header with "Navbar" and "Docs Examples Icons Themes Blog". Below the header, there's a search bar and a "Download" button. The main text discusses the Navbar's features and includes a snippet of code. At the bottom, there's a sidebar with links like "View on GitHub", "On this page", "How it works", "Supported content", "Brand", "Text", "Image", "Image and text", "Nav", "Forms", "Text", "Color schemes", "Containers", "Placement", "Scrolling", and "Responsive behaviors". A small advertisement for "Auth0" is visible at the bottom.

The screenshot shows the Bootstrap documentation for the Navbar component. The URL is [getbootstrap.com/docs/5.0/components/navbar/](https://getbootstrap.com/docs/5.0/components/navbar/). The page title is "Navbar". The top navigation bar has tabs for "Navbar", "Home", "Link", "Dropdown", "Disabled", and two search fields. A red box highlights the "Navbar" tab. Below the navigation, there's a code snippet for the Navbar component. A blue "Copy" button is located in the top right corner of the code block.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

The screenshot shows the Bootstrap documentation for the Navbar component. The URL is [getbootstrap.com/docs/5.0/components/navbar/](https://getbootstrap.com/docs/5.0/components/navbar/). The page title is "Navbar". The top navigation bar has tabs for "Navbar", "Home", "Link", "Dropdown", "Disabled", and two search fields. A red box highlights the "Navbar" tab. Below the navigation, there's a code snippet for the Navbar component. A blue "Copy" button is located in the top right corner of the code block.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

# /hi 페이지에 헤더-푸터 레이아웃 적용하기

- greetings.mustache 파일에 <!-- navigation --> 아래에 붙여넣기

```
<!--navigation -->  
  
<nav class="navbar navbar-expand-lg navbar-light bg-light">  
(중략)  
<nav>  
  
<!-- content -->
```

- 서버를 재시작하고, localhost:8080/hi 접속하여 결과 확인



# /hi 페이지에 헤더-푸터 레이아웃 적용하기

- Site info 추가하기

```
<!-- site info -->


<hr>
<p>© CloudStudying | <a href="#">Privacy</a> | <a href="#">Terms</a></p>
</div>

<script src="(생략)" integrity="(생략)" crossorigin="anonymous"></script>

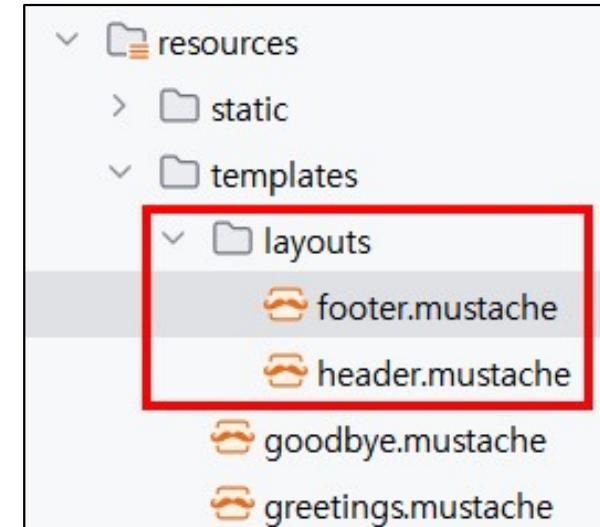

```

- 서버를 재시작하고, localhost:8080/hi 접속하여 결과 확인



# 헤더와 푸터를 템플릿으로 만들기

- 웹페이지마다 중복되는 코드들을 하나의 틀로 만들어 변수화
- /hi, /bye 페이지에 중복되는 navigation, site info 코드들을 템플릿으로 만들어 변수화
- [templates] 디렉토리에서 마우스 오른쪽 버튼 클릭,  
[New - Directory] 클릭
  - 새 디렉토리명, 'layouts' 입력 후, 엔터키
- [templates - layouts] 디렉토리에서 마우스 오른쪽 버튼 클릭,  
[New - File] 클릭
  - 새 파일명, 'header.mustache' 입력 후, 엔터키
  - 새 파일명, 'footer.mustache' 입력 후, 엔터키



# 헤더와 푸터를 템플릿으로 만들기

- greeting.mustache 파일의 첫 째줄, <!doctype html> ~ </nav>까지 선택후, 잘라내기 (ctrl + x)
- header.mustache 파일에 붙여넣기 (ctrl + v)

header.mustache

```
<!doctype html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet">

    <title>Hello, world!</title>
</head>
<body>
    <!--navigation -->
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        (중략)
    </nav>
```

# 헤더와 푸터를 템플릿으로 만들기

- 뷰 템플릿 파일 불러오기
  - 형식: {{>}파일경로/파일명}}
  - ✓ 파일경로는 현재의 mustache 파일의 위치를 기준으로 함
  - ✓ 파일명에는 확장자 (.mustache) 를 포함
- greeting.mustache 파일의 잘라내기 한 부분에,  
header.mustache 템플릿 파일 불러오기

greeting.mustache

```
{{>layouts/header}}
```

```
<!-- content -->
<h1>{{username}}님, 반갑습니다!</h1>

<!-- site info -->
<div class="md-5 container-fluid">
    <hr>
    <p>© CloudStudying | <a href="#">Privacy</a> | <a href="#">Terms</a></p>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.js">
</body>
</html>
```

# 헤더와 푸터를 템플릿으로 만들기

- greeting.mustache에서 <!-- site info --> ~ </html>까지 잘라내기
- footer.mustache 파일에 붙여넣기

footer.mustache

```
<!-- site info -->
<div class="md-5 container-fluid">
  <hr>
  <p>© CloudStudying | <a href="#">Privacy</a> | <a href="#">Terms</a></p>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.b
</body>
</html>
```

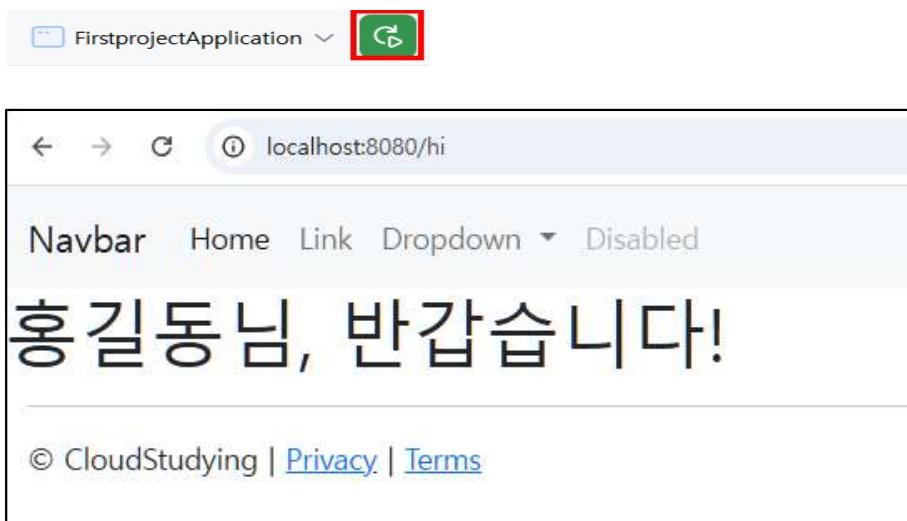
# 헤더와 푸터를 템플릿으로 만들기

- greeting.mustache 파일의 잘라내기 한 부분에, footer.mustache 템플릿 파일 불러오기

greeting.mustache

```
 {{>layouts/header}}  
  
<!-- content --&gt;<br/><h1>{{username}}님, 반갑습니다!</h1>  
  
 {{>layouts/footer}}
```

- 서버를 재시작하고, localhost:8080/hi 접속하여 결과 확인



# /bye 페이지에 헤더-푸터 템플릿 적용하기

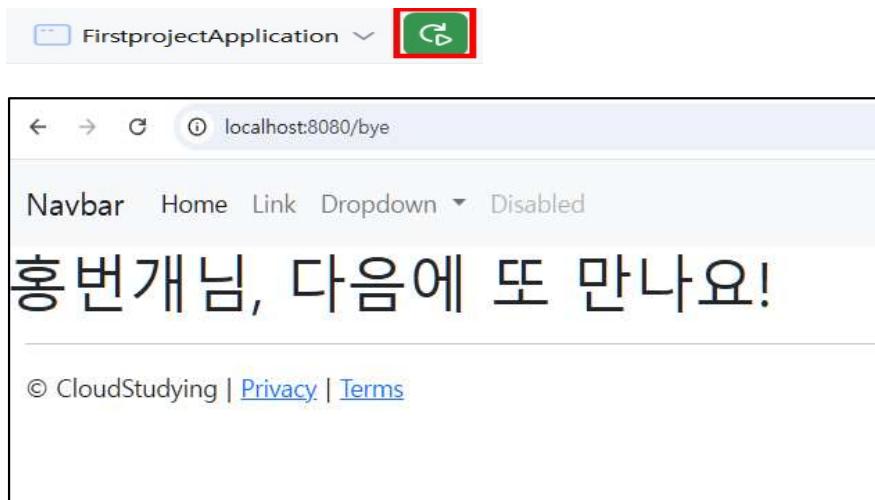
- goodbye.mustache 파일에서 header.mustache와 footer.mustache 템플릿을 불러오기

```
{{>layouts/header}}
```

```
<!-- content -->
<h1>{{nickname}}님, 다음에 또 만나요!</h1>
```

```
{{>layouts/footer}}
```

- 서버를 재시작하고, localhost:8080/hi 접속하여 결과 확인



# 콘텐츠 영역에 스타일 넣기

- greeting.mustache 와 goodbye.mustache 파일의 콘텐츠 부분을 <div>~</div> 태그로 감싸고, 부트스트랩에서 제공하는 class 속성을 추가

greeting.mustache

```
{>layouts/header}

<!-- content -->


# {{username}}님, 반갑습니다!



{>layouts/footer}
```

goodbye.mustache

```
{>layouts/header}

<!-- content -->


# {{nickname}}님, 다음에 또 만나요!



{>layouts/footer}
```

- 서버를 재시작하고, localhost:8080/hi 접속하여 결과 확인

