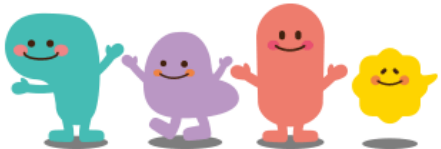


# 2주차

## 자바웹프로그래밍(2)

강사 : 최도현

안녕하세요!



# 오늘의 할일 - 1

- 웹 트렌드 분석

백엔드 프로그래밍

스프링 부트의 등장

전자 정부 표준 프레임워크



# 백엔드 프로그래밍

- 백엔드 프로그래밍
  - 서버 기반 정보 요청/응답
  - 웹 애플리케이션 구현
    - 크로스 플랫폼 유무 중요
- Lib의 한계 → 프레임워크 필요성
  - 높은 생산성과 유지 보수 필요
    - 독립 개발 → 공통 표준 모듈
  - 소규모 → 큰 규모 프로젝트
    - 점점 복잡하고 설정이 어려움

사실, 웹 문서, 교재 등 표현이 조금 다름

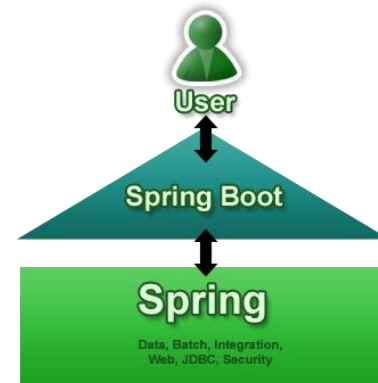
프론트엔드	백엔드
<b>라이브러리</b> jQuery, D3.js, Bootstrap 등	<b>서버언어</b> Python, Java, PHP 등
<b>프레임워크</b> React, Angular, Vue 등	<b>프레임워크</b> Node.js, Spring, Django 등

특징	프레임워크	라이브러리
유저코드의 작성	프레임워크 클래스를 서브클래싱 해서 작성	독립적으로 작성
호출흐름	프레임워크코드가 유저코드를 호출	유저코드가 라이브러리를 호출
실행흐름	프레임워크가 제어	유저코드가 제어
객체의 연동	구조프레임워크가 정의	독자적으로 정의

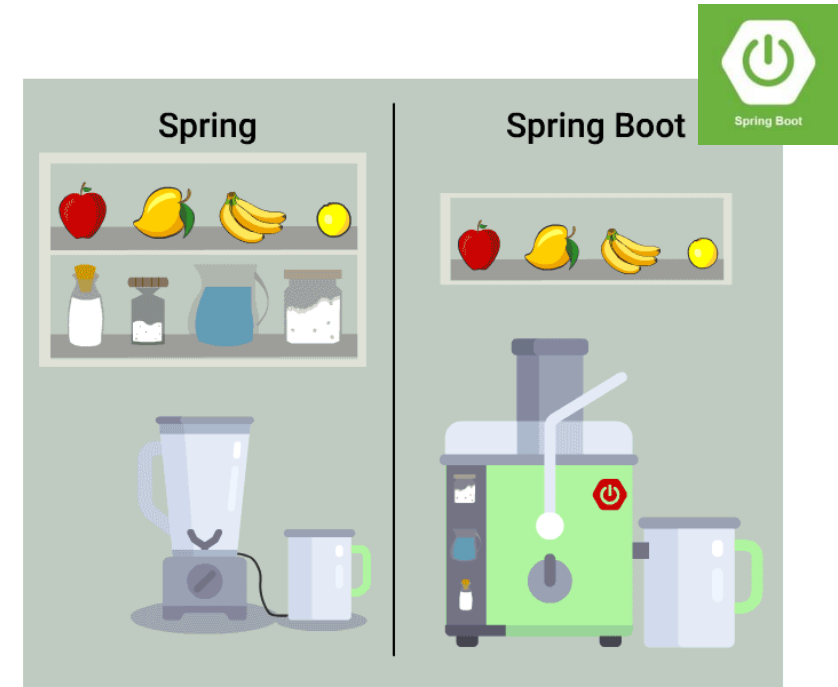
〈표 1〉 프레임워크와 라이브러리의 비교

# 스프링 부트의 등장

- 스프링 프레임 워크(2004년)
  - 오픈 소스 기반 대표 웹 프레임워크
  - 문제점 : 진입 장벽이 매우 높음
- 스프링 부트의 등장!
  - 작은 규모 및 단순 서블릿 관리
    - 자바 언어, JVM 위에서 실행됨
  - 주요 편의성/자동화 기능 강화
    - 자동환경설정
    - 자동 빌드 및 리로드 등
    - 강제 뷰 분리 및 인스턴스 공유
- 자바 풀스택 개발자 로드맵에서 인기
  - 자바 언어 필수, 비교적 쉬움



스프링 프레임워크 → 스프링 부트 등장

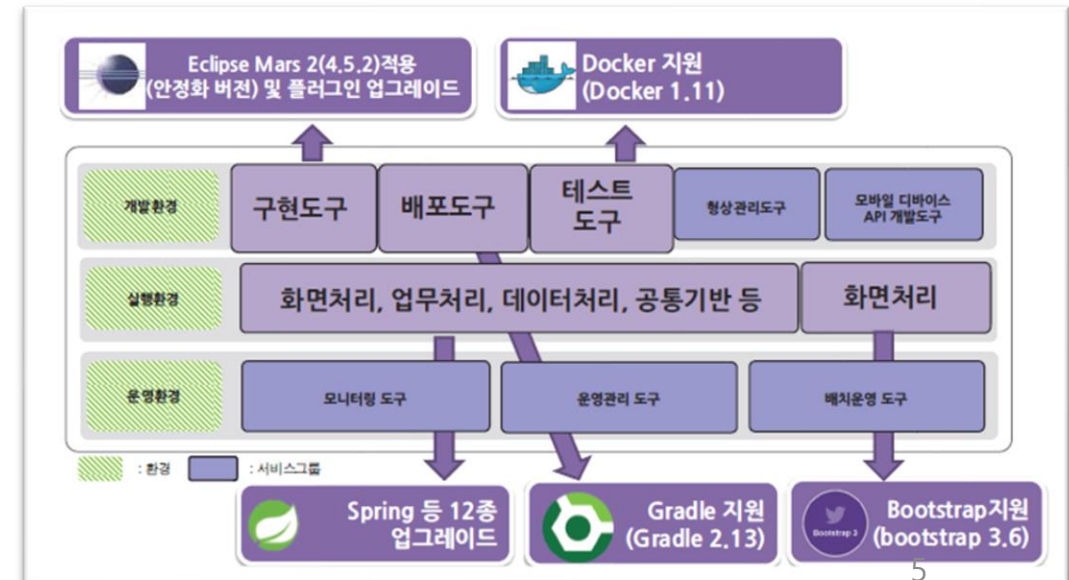


# 전자 정부 표준 프레임워크

- 스프링 기반 eGov 공개(2010년 이후)
  - 주요 개발 가이드 제공
    - <https://www.egovframe.go.kr/>
    - 최근 4.2.x 버전
  - 국내 주요 공공기관 웹 프로젝트
    - 자바, JSP, jQuery, eGov 기술 요구
    - 최근 스프링 부트 지원 시작
- 주요 개발 환경 요구사항
  - 코어 : 스프링 프레임워크 5.3
    - 아파치 탐켓 8 버전
  - 서블릿 3.1 이상
    - 자바 8 이상(최신 버전 권장)
- 우리는? 통합 IDE는 VS CODE로!



대부분 스프링 환경에 의존



# 오늘의 할일 - 2

- 개발 환경 설정

VS CODE - 스프링 부트 준비

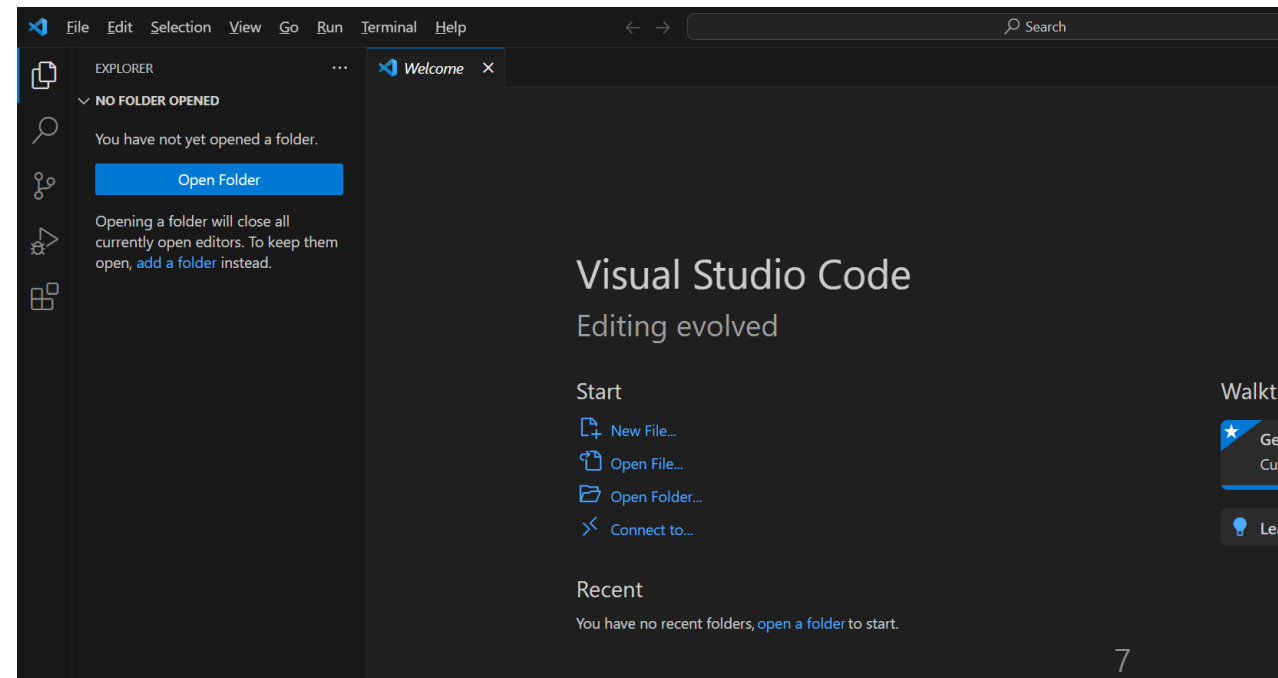
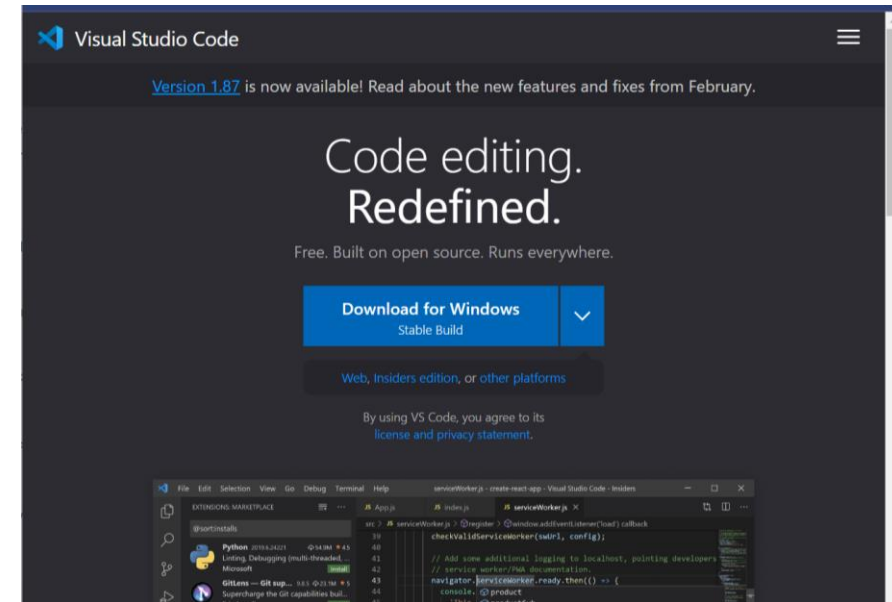
기본 화면 실행하기

깃 허브 연동



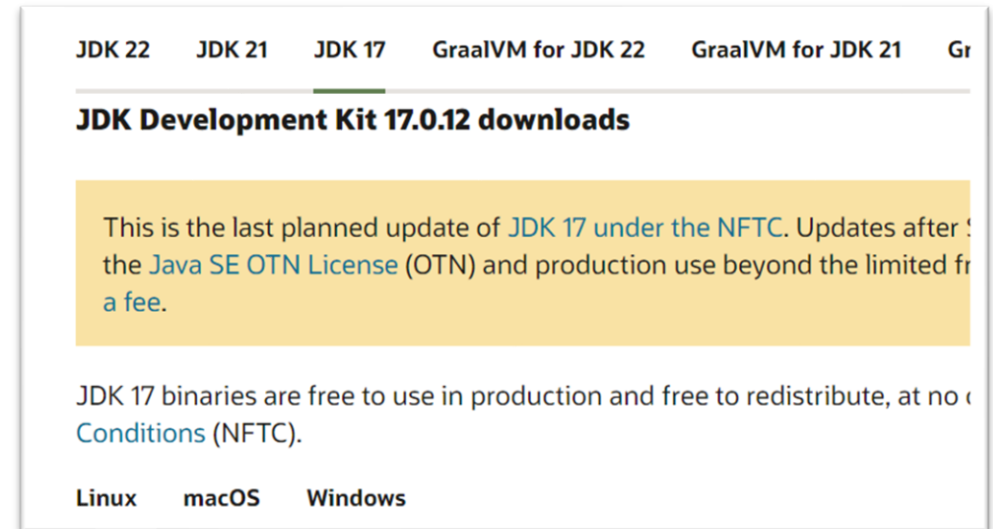
# VS CODE 다운 및 설치

- 현재 PC 및 노트북 확인
  - VS CODE 설치 및 실행
    - <https://code.visualstudio.com/>
- 기본 메뉴 UI
  - 탐색기
    - 프로젝트 폴더
  - 검색
  - 소스 제어
    - 깃 허브 제어
  - 실행 및 디버그
    - Live server를 통해 실행
  - 확장



# 스프링 부트 준비

- 버전선택 및 다운로드
  - 기존 HTML5 등 표준 언어
    - 언어 버전에 큰 영향이 없음
  - 스프링 개발에 앞서
    - 업데이트 지원 주기 약 3년
    - 호환성 조합 꼭 필요
- 주요 버전
  - 스프링 부트 : 최소 3.2.X 버전 이상
  - JAVA : 최소 17 또는 21 권장
  - (선행)자바 언어 다운로드 후 설치
    - JAVA : [웹 사이트 링크 클릭](#)





# 스프링 부트 준비

- VS CODE 확장 모듈

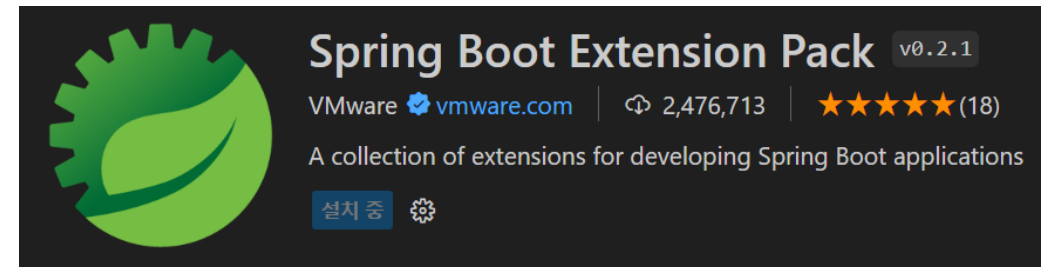
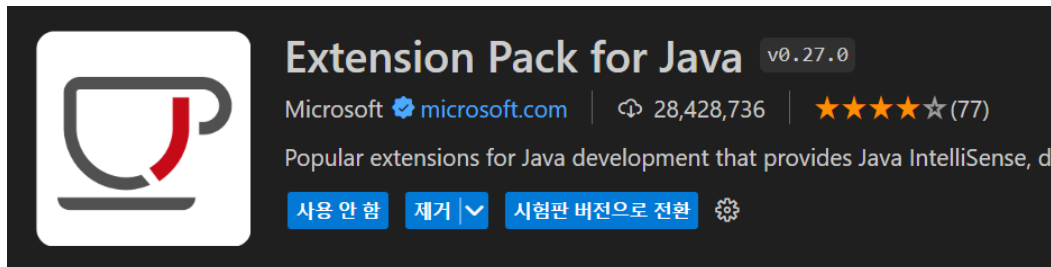
- 직접 검색 후 설치

- Extension Pack for Java

- 자동 추가 : Visual Studio IntelliCode, Language Support for Java, Debugger for Java, Maven for Java, Java Test Runner, Project Manager for Java

- Spring Boot Extension Pack

- 자동 추가 : Spring Boot Tools, Spring Initializr Java Support, Spring Boot Dashboard, Cloudfoundry Manifest YML, Concourse CI Pipeline Editor



# 스프링 부트 준비

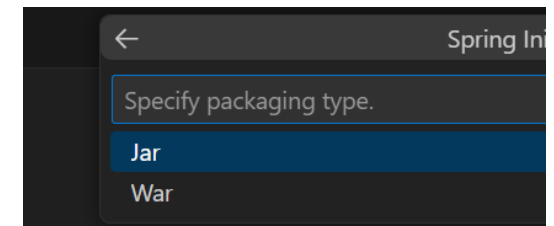
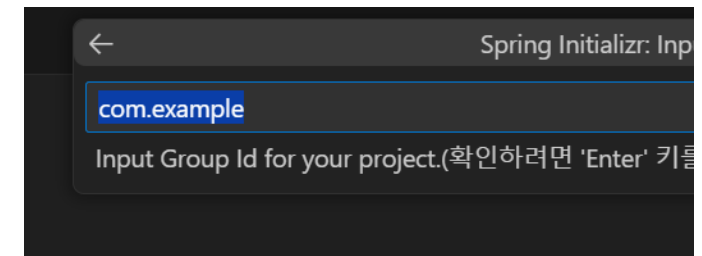
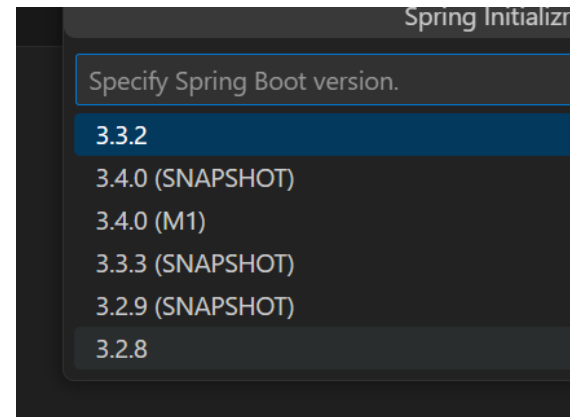
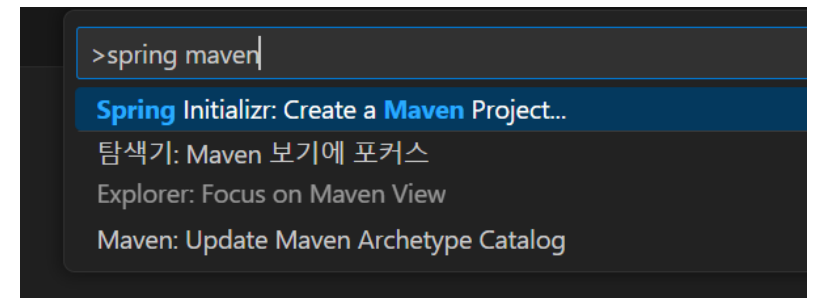
- JDK 환경 설정
  - VS CODE → 파일 → 기본설정
    - 설정 → 설정 검색
      - 검색 창(java home) 입력
- **setting.json에서 편집 클릭**
  - Jdk 자바 홈 경로 추가
    - 그림과 같이 2가지 모두 추가
- 주의 : 버전 경로 확인



```
{  
  "liveServer.settings.donotShowInfoMsg": true,  
  "git.autofetch": true,  
  "easycode.model": "gpt-4-1106-preview",  
  "files.autoSave": "afterDelay",  
  "java.jdt.ls.java.home": "C:\\Program Files\\Java\\jdk-17",  
  "spring-boot.ls.java.home": "C:\\Program Files\\Java\\jdk-17"  
}
```

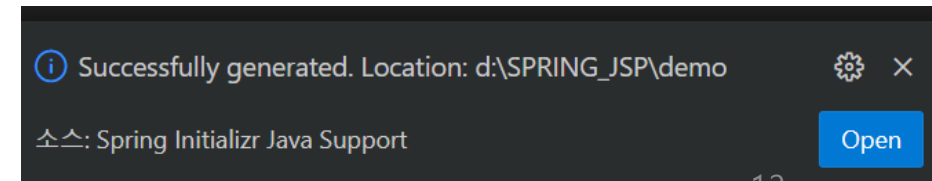
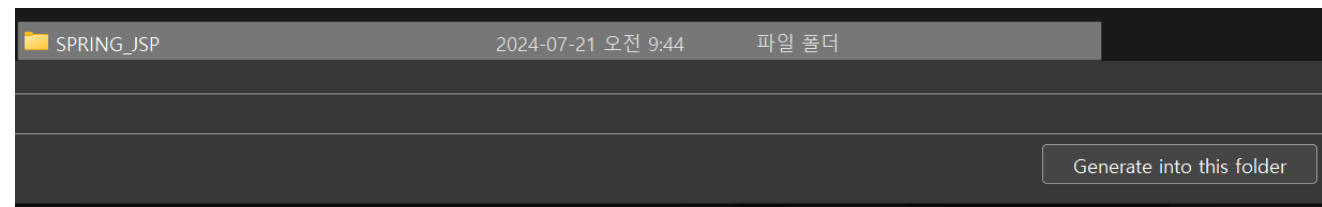
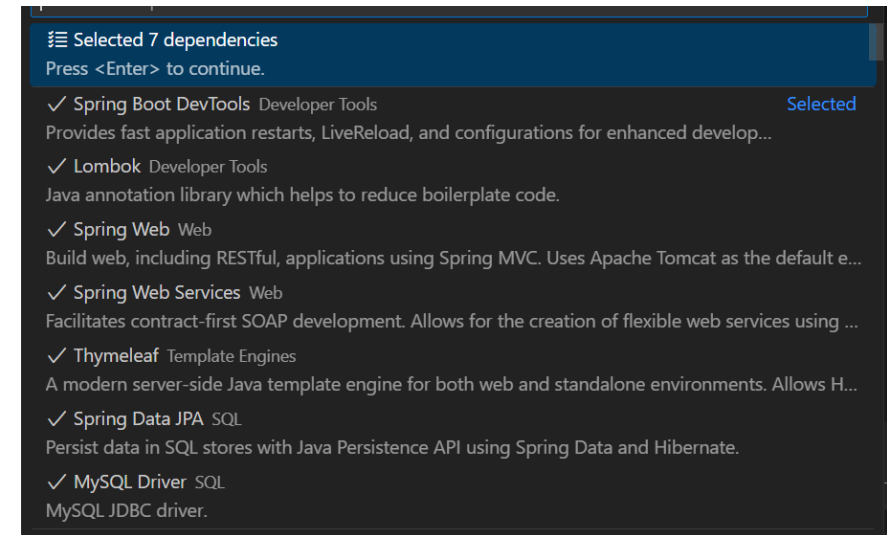
# 스프링 부트 준비

- 새 프로젝트 만들기
  - 팔레트 : ctrl + shift + p (상단 바)
    - spring maven 입력
  - 추가 정보
    - 버전 3.3.x
    - 프로젝트 그룹 입력(엔터)
    - 패키징 Jar
    - 자바 언어, 자바 버전 17
  - 참고 : 프로젝트 생성 관련
    - [웹 사이트 제공](#)



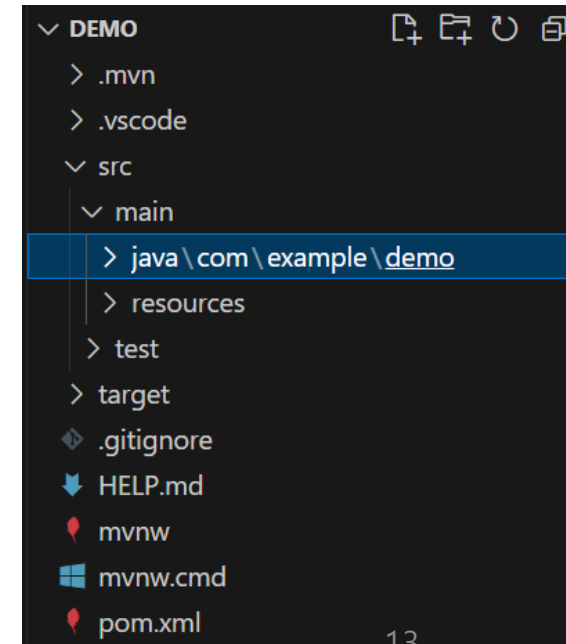
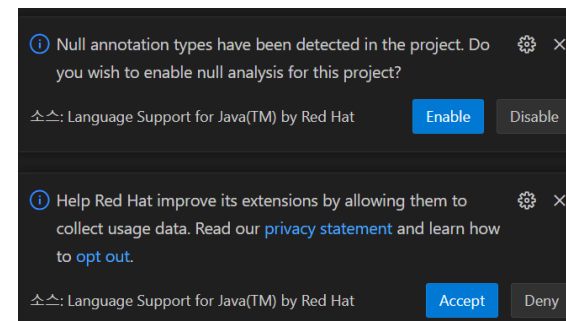
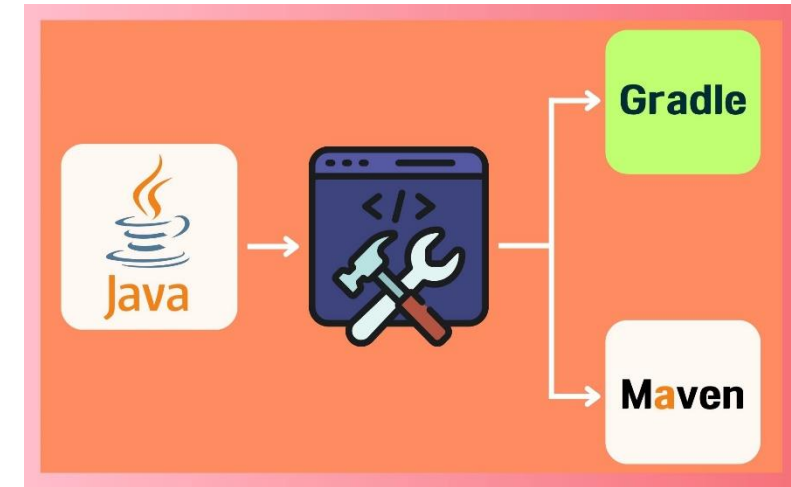
# 스프링 부트 준비

- 새 프로젝트 만들기
  - 추가 S/W 모듈 선택(의존성)
    - 7가지 선택 : 그림 참고
  - 엔터 클릭, 폴더 : SPRING\_BOOT\_학번
    - 새로운 폴더를 생성한 후 지정
- 하단 프로젝트 Open
  - 잠시 대기 하면 로딩됨
  - 폴더 신뢰 OK.



# 스프링 부트 준비

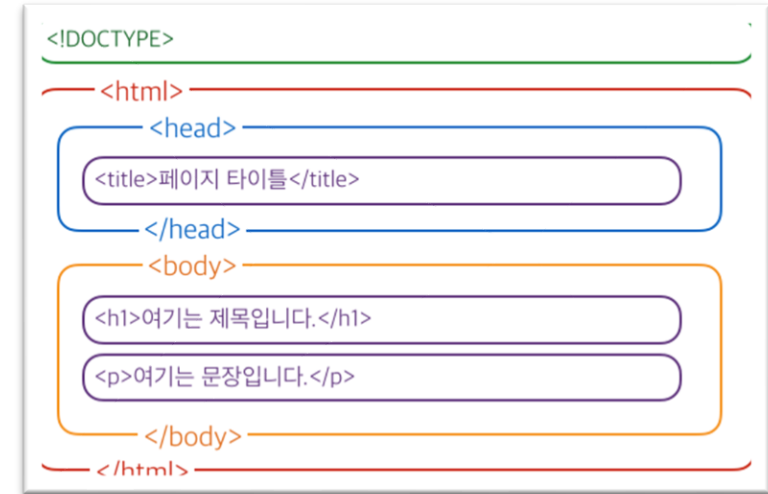
- 프로젝트 폴더 확인
  - 하단 설정 활성화(모두 수락)
- 프로젝트 폴더 구조(maven)
  - 핵심 폴더
    - `src` → `main` → `java` (루트 패키지 경로)
      - Resource (주요 리소스)
        - Static : css, 이미지, js 등
    - `test` → 테스트 코드(실행하면 생성됨)
    - `pom.xml` (빌드 및 종속성 설정)
  - 다양한 폴더 및 설정 파일 존재
    - 수업 진도 이후 천천히 살펴본다.



# 기본 화면 실행하기

- 루트 폴더 하위의 resource 폴더
  - templates 하위에 index.html 파일을 생성한다.
- 다음과 같이 작성하고 저장한다.
  - 기존 HTML5를 그대로 사용할 수 있다.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>index 메인페이지</title>
</head>
<body>
  <h1>안녕하세요!</h1>
</body>
</html>
```



## HTML5

- HTML의 기본 기능을 넘어 다양한 콘텐츠와 어플리케이션을 브라우저에 표현하고 실행하기 위해 만들어지고 있는 언어
- 확장 가능한 형태의 HTML이라는 의미

웹 브라우저의 호환성과 구조적이고 의미있는 Markup 기능 및 편리한 웹 폼 기능을 제공

CSS + HTML + JavaScript

디자인 표현 및  
관리적인 요소

기본골격

프로그래밍적  
요소

# 자바 클래스 파일

- Demo 폴더 하위 .java 소스파일
  - DemoApplication.java을 연다.
  - 기본 폴더 구조 = 패키지

```
package com.example.demo; // 현재 폴더 위치
```

```
import org.springframework.boot.SpringApplication; // 스프링 핵심 클래스
```

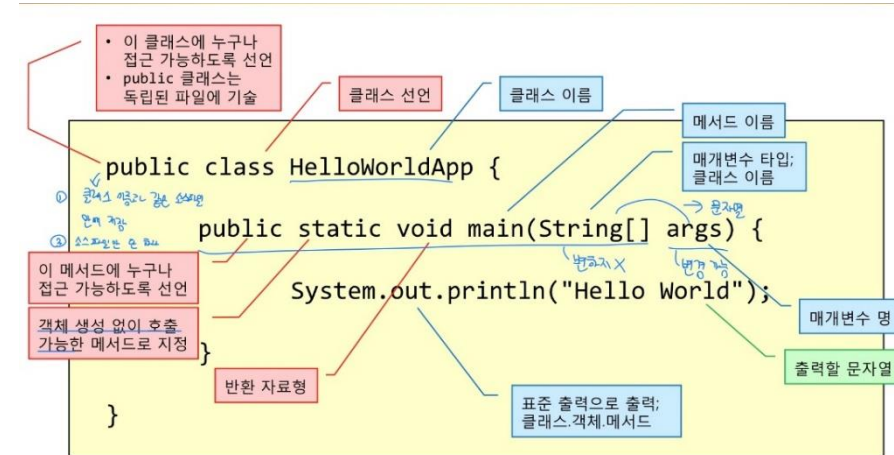
```
import org.springframework.boot.autoconfigure.SpringBootApplication; // 자동 설정 기능 활성화
```

```
@SpringBootApplication // 애노테이션(스프링 부트 APP 명시, 하위 다양한 설정을 자동 등록)
```

```
public class DemoApplication { // 클래스 이름
```

```
    public static void main(String[] args) { // 메인 메서드(프로그램 시작점)  
        SpringApplication.run(DemoApplication.class, args); // run 메서드로 실행  
    }
```

- 어노테이션을 통해 자동 설정
  - 코드 상단에 특정 역할을 명시

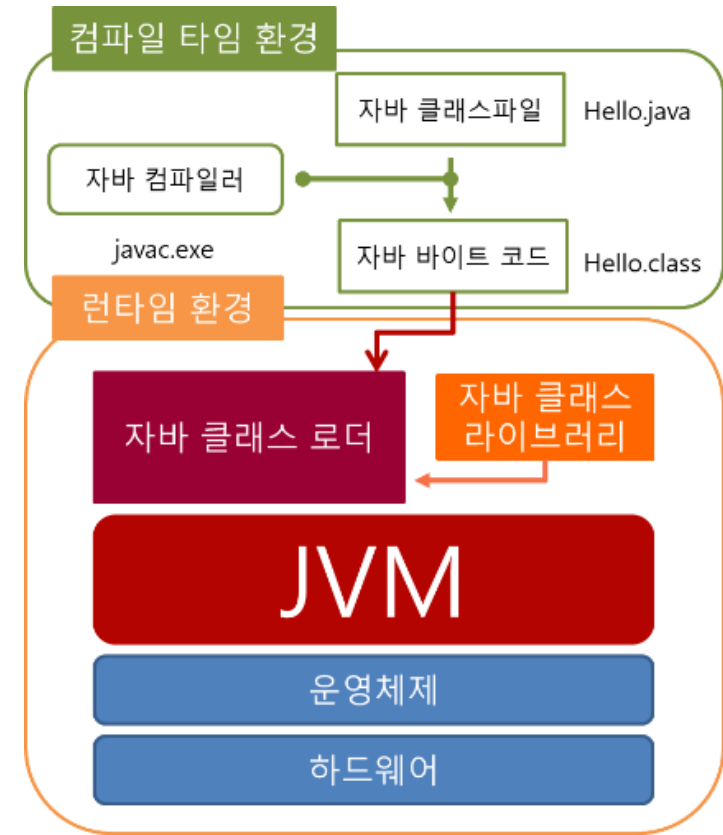


## Spring Boot Annotations

Annotation	의미
@SpringBootApplication	Spring boot application 으로 설정
@Controller	View를 제공하는 controller로 설정
@RestController	REST API 를 제공하는 controller로 설정
@RequestMapping	URL 주소를 맵핑
@GetMapping	Http GetMethod URL 주소 맵핑
@PostMapping	Http PostMethod URL 주소 맵핑
@PutMapping	Http PutMethod URL 주소 맵핑
@DeleteMapping	Http DeleteMethod URL 주소 맵핑
@RequestParam	URL Query Parameter 맵핑
@RequestBody	Http Body를 Parsing 맵핑
@Valid	POJO Java class의 검증

# 자바 클래스 파일

- 저는 자바 언어가 처음인데요?
  - 자바 언어의 JVM(자바 가상 머신)
    - 변환된 바이트 코드를 실행
    - 플랫폼 독립성 제공
  - C#과 같이 GC 존재
    - 메모리 관리(자동)

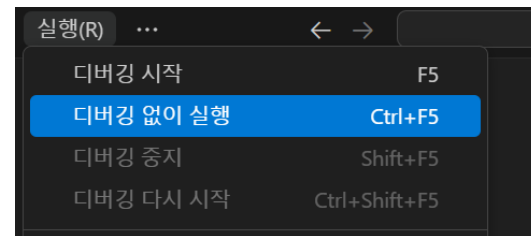


특징	설명	장점	단점
플랫폼 독립성	운영체제나 하드웨어에 독립적으로 작동	어떤 환경에서도 동일하게 실행 가능	초기 컴파일 과정 필요
바이트코드 실행	자바 소스 코드를 바이트코드로 컴파일하여 실행	플랫폼 독립성 확보	인터프리터 또는 JIT 컴파일 과정 필요
가비지 컬렉션	개발자가 메모리 관리를 직접 수행하지 않아도 됨	메모리 누수 방지	초기 성능 저하 가능성
보안	샌드박스 방식으로 실행	시스템 보안 강화	복잡한 구현 필요
성능	JIT 컴파일러를 사용하여 성능 향상	빠른 실행 속도	초기 컴파일 과정 필요

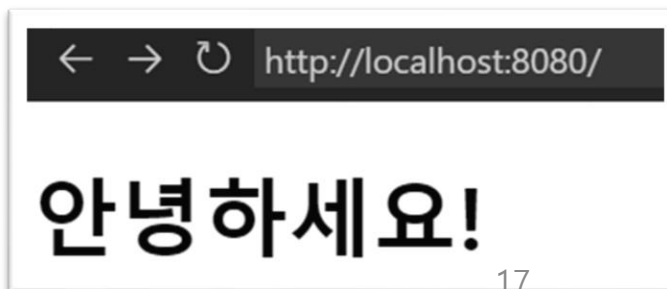
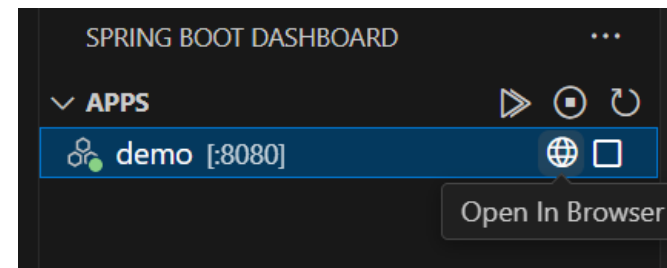


# 기본 화면 실행하기

- 실행하기
  - DemoApplication.java 파일을 열자.
  - 메인 함수 확인
- 디버그 없이 실행(ctrl + f5)
  - 스프링 부트 서비스가 시작된다.
  - 아파치 탐켓, 8080포트로 실행됨
- 좌측 대시 보드
  - 스프링 전용 대시 보드 확인
  - 인터넷 아이콘을 클릭해보자.
- 직접 웹 브라우저 접속
  - 포트 에러나는 경우
  - localhost:8080 실행

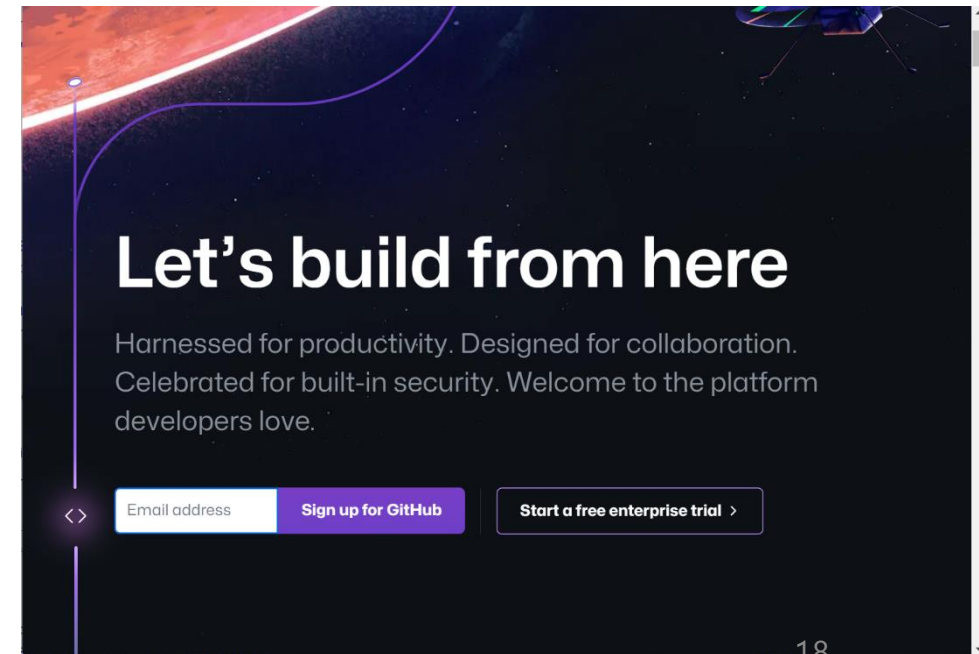


```
Tomcat initialized with port 8080 (http)
Starting service [Tomcat]
Starting Servlet engine: [Apache Tomcat/10.1.26]
Initializing Spring embedded WebApplicationContext
Root WebApplicationContext: initialization completed in 1333 ms
LiveReload server is running on port 35729
Exposing 1 endpoint(s) beneath base path '/actuator'
Tomcat started on port 8080 (http) with context path ''
Started DemoApplication in 2.655 seconds (process running for 3.216)
Initializing Spring DispatcherServlet 'dispatcherServlet'
Initializing Servlet 'dispatcherServlet'
Completed initialization in 1 ms
```



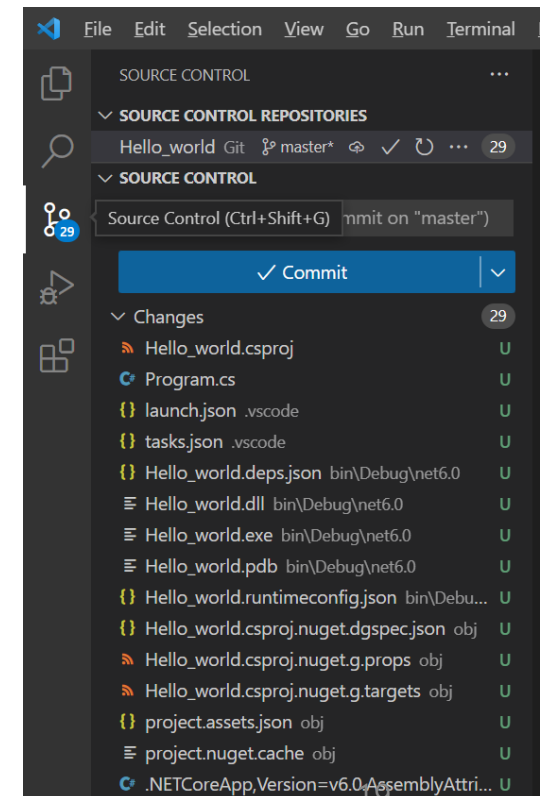
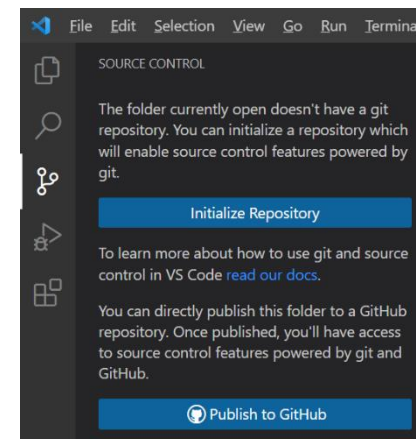
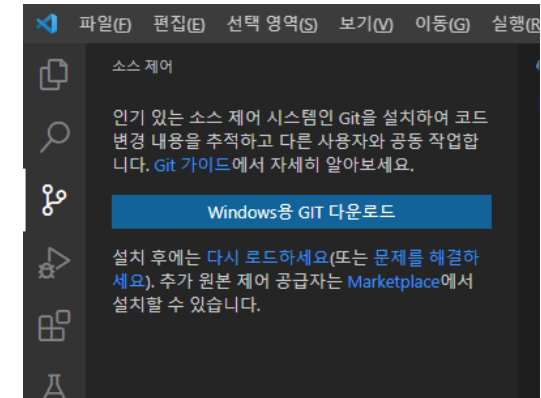
# GITHUB 연동하기

- GITHUB는?
  - 프로젝트 폴더 → 서버 업로드 관리(원격)
    - 현재 프로젝트 폴더 : 로컬
  - 장점? IDE 개발 연동, 소스코드 버전 관리
    - 실무에서 대부분 활용 : 팀 프로젝트
    - **학기 내 : 졸업 작품까지 GO~**
- 깃 허브 접속 및 로그인
  - <https://github.com/>, (가입한다.)
    - 인증 및 절차 진행...
  - 로그인 해보자.



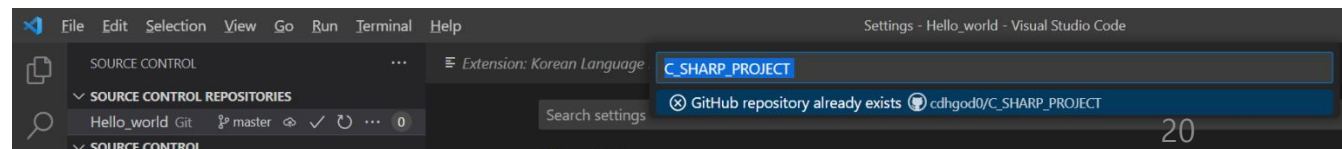
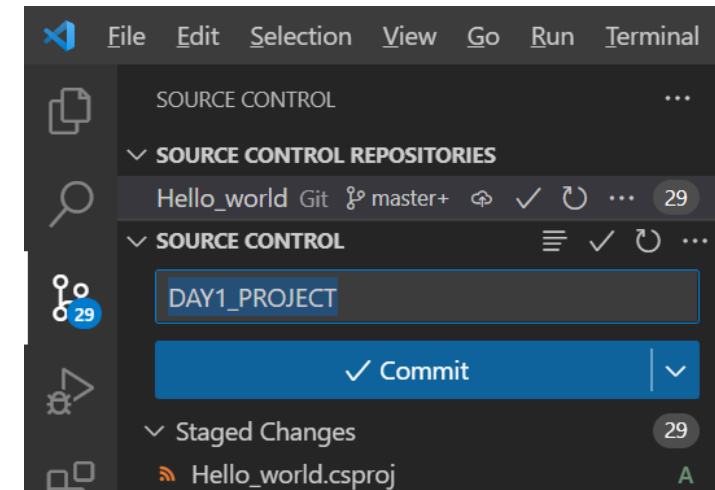
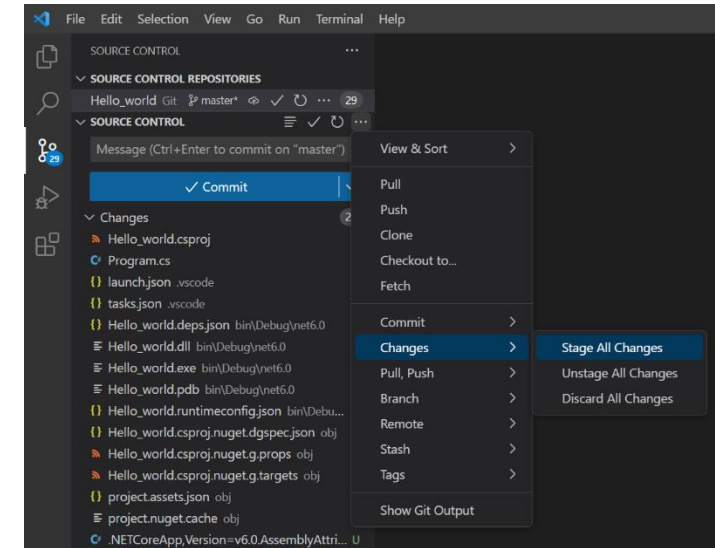
# GITHUB 연동하기

- CODE 좌측 메뉴 → 소스 제어(Source Control)
  - 윈도우용 GIT app 다운로드 및 설치(위 git 계정 연동)
    - <https://git-scm.com/download/win>
      - 64-bit Git for Windows Setup (대부분 os)
  - 실행 후 설치를 완료
    - 상관없이 끝까지 Next~~ Finish
- 깃 설치 완료 후
  - CODE에서 다시 로드하세요, 클릭!
- 리포지토리 초기화 클릭(첫 연동)
  - 모든 소스코드가 추가됨



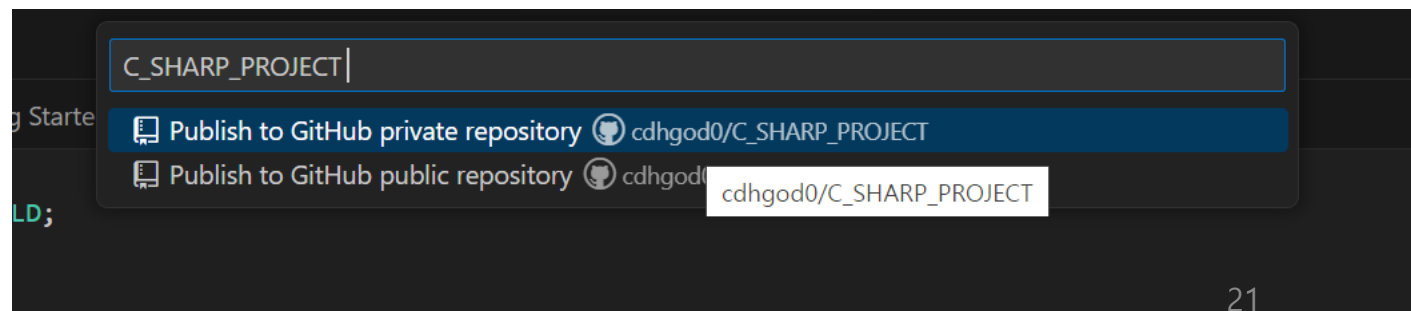
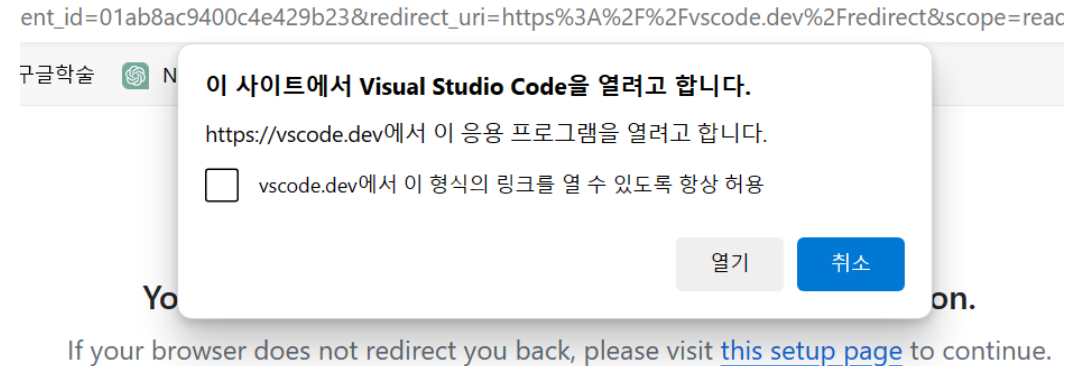
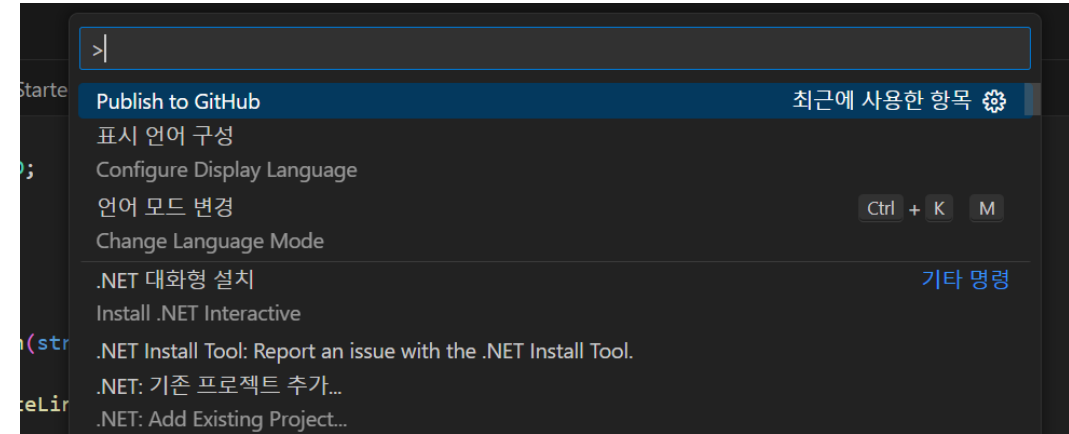
# GITHUB 연동하기

- (직접 x) CODE에서 새 저장소로 업로드 하기
  - 소스 제어(Source Control) → 마우스 클릭
    - 스테이징을 모두 적용(소스 코드 변화 내역을 모두 적용)
  - 소스 제어 하단 → 텍스트 입력 창(필수)
    - 2주차 개발환경 설정 입력 후 --> Commit 클릭
- Publish\_브랜치(공개)
  - SPRING\_학번 입력해보자. (자동 생성)
  - 깃 허브 웹 사이트 확인



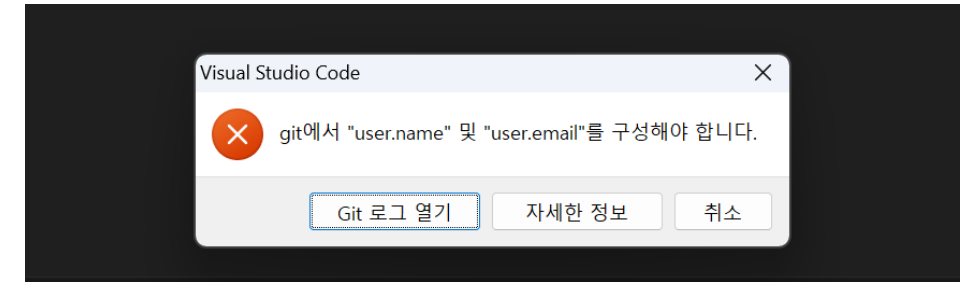
# GITHUB 연동하기

- 계정 연동이 자동으로 안된 경우(초기)
  - 전체 커맨트 팔레트 확인
    - 단축키 : CTRL + SHIPFT + P
  - Publish to Github 입력
    - 웹 브라우저 창이 자동으로 열림
    - 본인 계정 로그인(로그인 되어 있는 경우)
    - CODE에서 열기 허용
  - 저장소 이름 입력
    - SPRING\_학번



# GITHUB 연동하기

- 계정 관련 이메일, 이름 등록 문제(초기)
  - 기본 이름, 이메일 입력 요구
- 자세한 정보 버튼을 클릭
  - 그림과 같이 터미널에 git 명령어를 입력한다.
- 게시 Branch 클릭
- 인증 한번 더 GO!
- 이후 인증 없이 업로드 가능

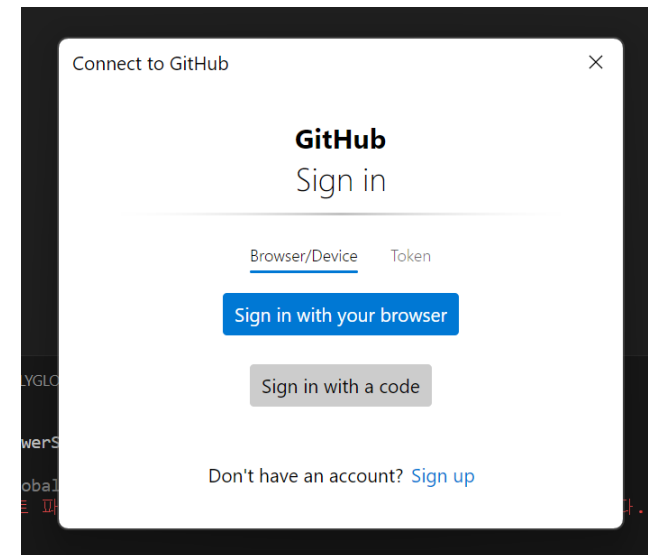
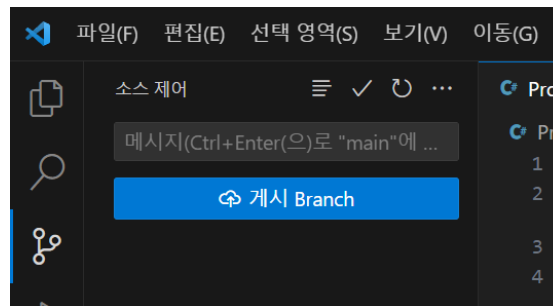


```
PS D:\C#TEST\HELLOWORLD>
* 복원된 기록

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

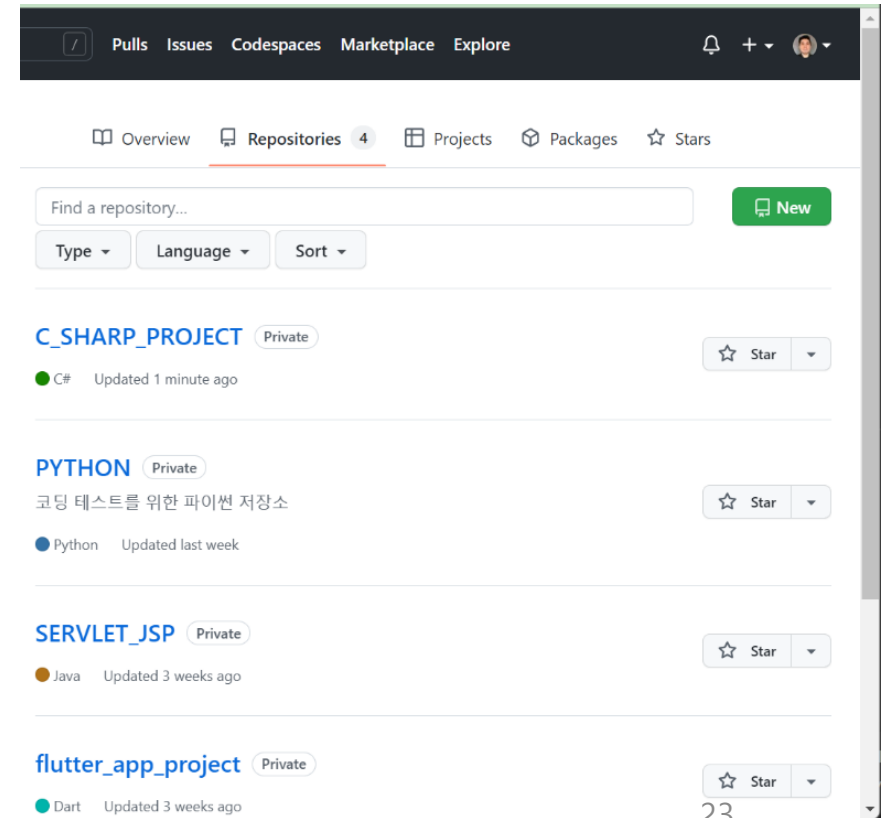
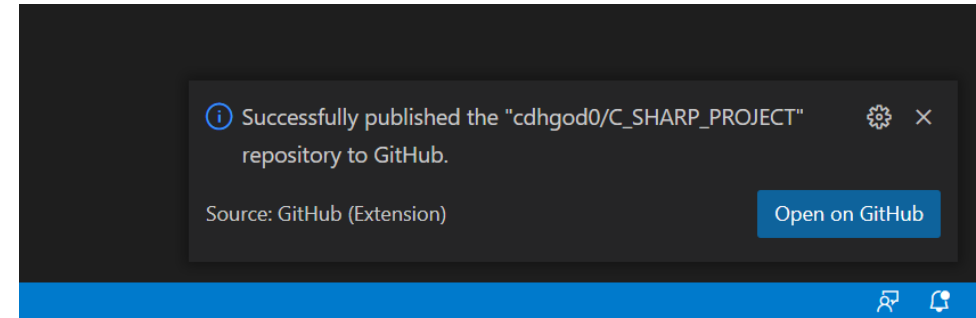
새로운 기능 및 개선 사항에 대 한 최신 PowerShell을 설치 하세요! https://aka.ms/PSWindows

PS D:\C#TEST\HELLOWORLD>
PS D:\C#TEST\HELLOWORLD>
PS D:\C#TEST\HELLOWORLD> git config --global user.name "admin"
PS D:\C#TEST\HELLOWORLD> git config --global user.email cdhgod@nate.com
PS D:\C#TEST\HELLOWORLD>
```



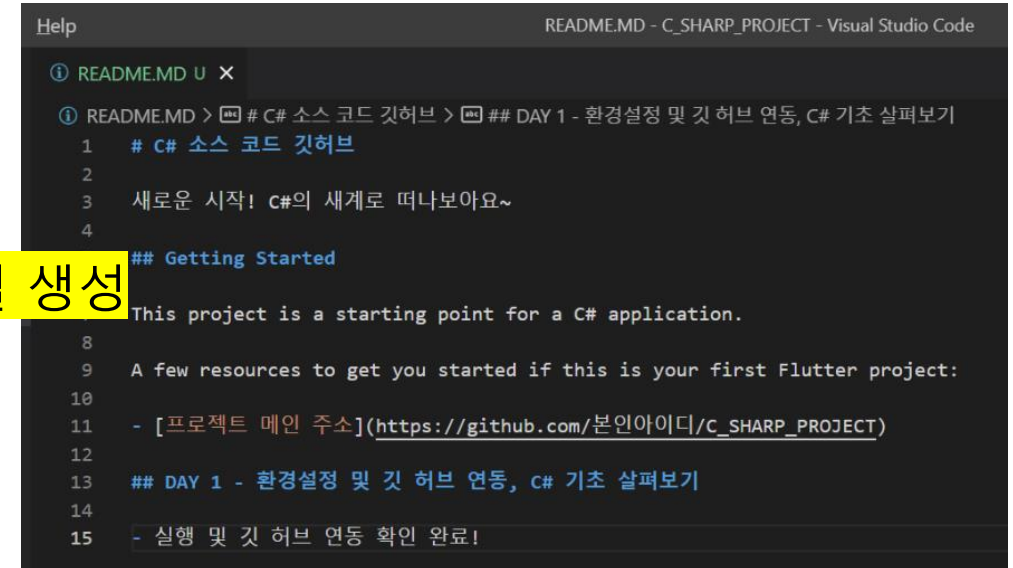
# GITHUB 연동하기

- 개인 저장소 확인
  - 깃 허브 열기 확인 & 직접 접속
    - 참고 : <https://github.com/>
- 리포지토리 확인
  - 클릭 → 내부 프로젝트 파일 확인
  - 폴더 및 파일 확인



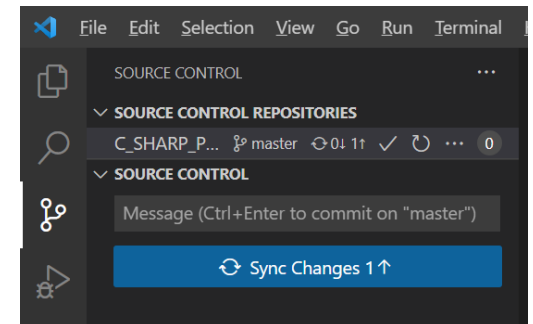
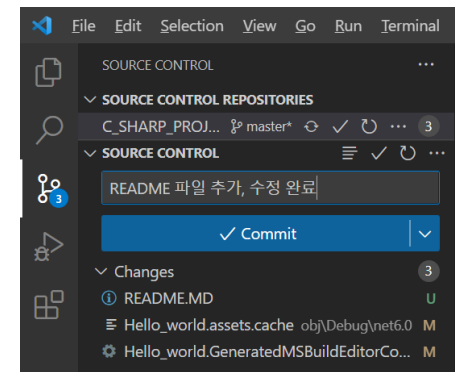
# GITHUB 연동하기

- 개인 업로드 실습
  - README.MD 파일 추가하기
  - 프로젝트 폴더 → 마우스 오른쪽 클릭 → 직접 파일 생성
    - README.MD 옆 내용과 같이 입력하고 저장
- 수정한 파일 다시 업로드
  - 소스 제어(Source Control) → 마우스 클릭
    - 수정된 파일이 몇 개 추가 되어 있다!
    - 스테이징을 모두 적용(소스 코드 변화 내역을 모두 적용)
  - 소스 제어 하단 → 텍스트 입력 창(필수)
    - README.MD 파일 추가, 수정 완료 입력 --> Commit 클릭
  - 업로드 하기
    - 푸시(PUSH) 또는 동기화(Sync Changes) 클릭



Help README.MD - C\_SHARP\_PROJECT - Visual Studio Code

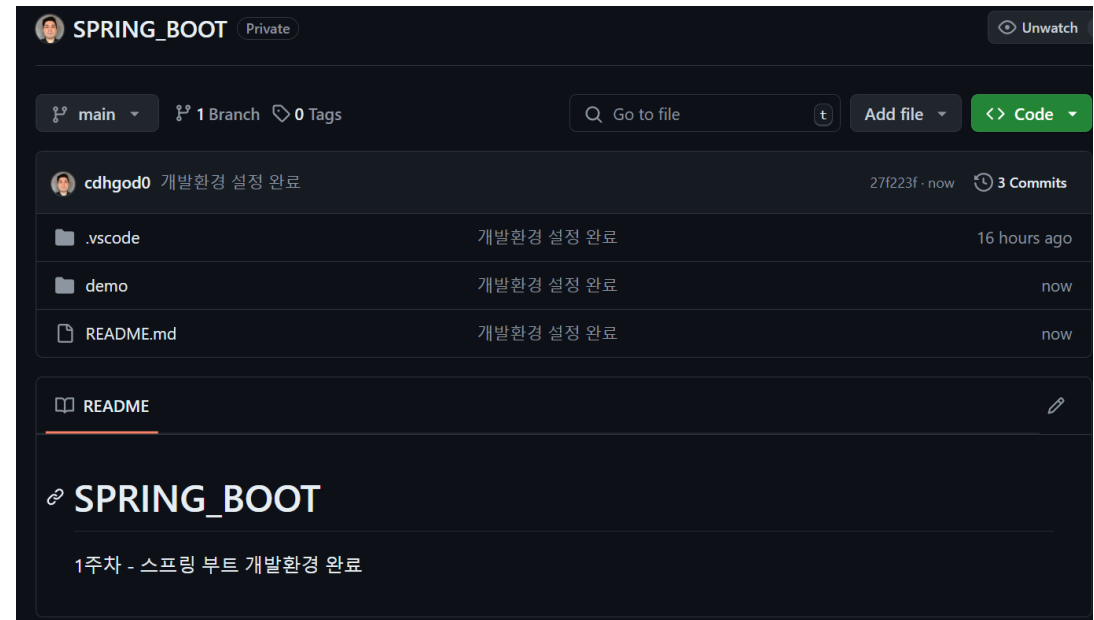
```
① README.MD U X
① README.MD > # C# 소스 코드 깃허브 > ## DAY 1 - 환경설정 및 깃 허브 연동, C# 기초 살펴보기
1 # C# 소스 코드 깃허브
2
3 새로운 시작! C#의 세계로 떠나보아요~
4
## Getting Started
This project is a starting point for a C# application.
8
9 A few resources to get you started if this is your first Flutter project:
10
11 - [프로젝트 메인 주소](https://github.com/본인아이디/C_SHARP_PROJECT)
12
13 ## DAY 1 - 환경설정 및 깃 허브 연동, C# 기초 살펴보기
14
15 - 실행 및 깃 허브 연동 확인 완료!
```





# GITHUB 연동하기

- 업로드 완료 후 저장소 확인
  - 깃 허브 열기 확인 & 직접 접속
    - 주소 : <https://github.com/>
- 깃 허브 소스 코드 백업(중요!)
  - 음, 아직 사용이 익숙하지 않아요
    - 소스코드 문제, 업로드 에러 등등
  - But, 일단 소스코드를 백업하고 싶다!
    - 깃 허브 화면 <> Code 클릭 → Download ZIP
  - 프로젝트 통째로 .ZIP파일로 압축 다운



# 오늘의 할일 - 3

- 동작 과정 및 메인 페이지

스프링 부트 동작 과정

thymeleaf

URL 맵핑과 컨트롤러

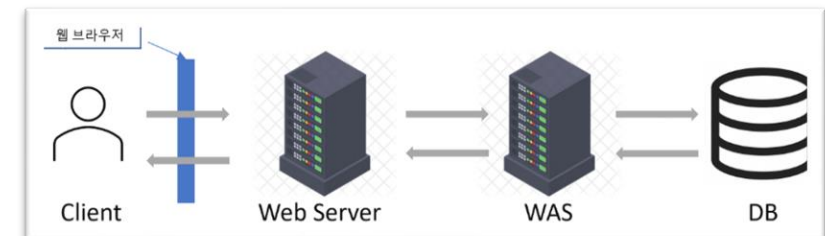
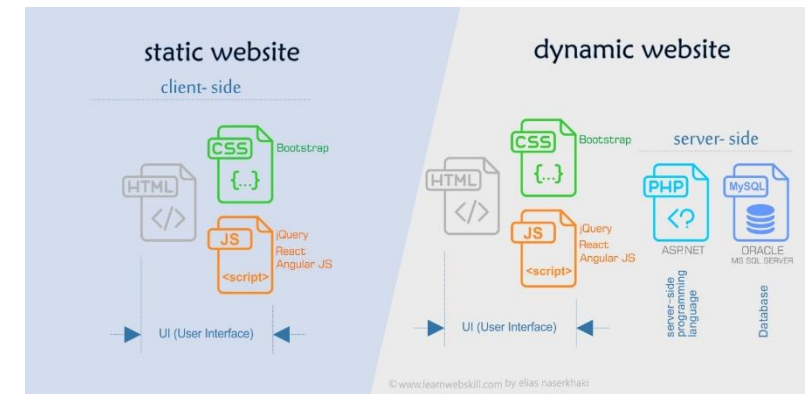
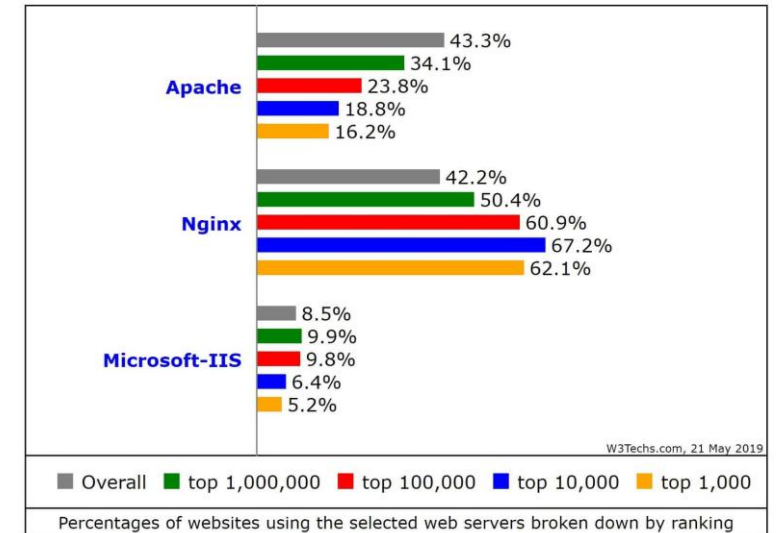


+



# 스프링 부트 동작 과정

- 웹 서버와 WAS
  - 대표 웹 서버
    - 기본적인 웹 요청/응답 기능 수행
    - NGINX, APACHE, IIS 등
  - 변화 : 정적 → 동적 웹 페이지 개발
    - DB, 서버 사이드 언어 확장
- 즉, 현존 대부분 웹 서버는?
  - WAS를 내장하고 DB 연동 필요
  - 스프링 프레임워크도 WAS를 기반으로 동작



# 스프링 부트 동작 과정

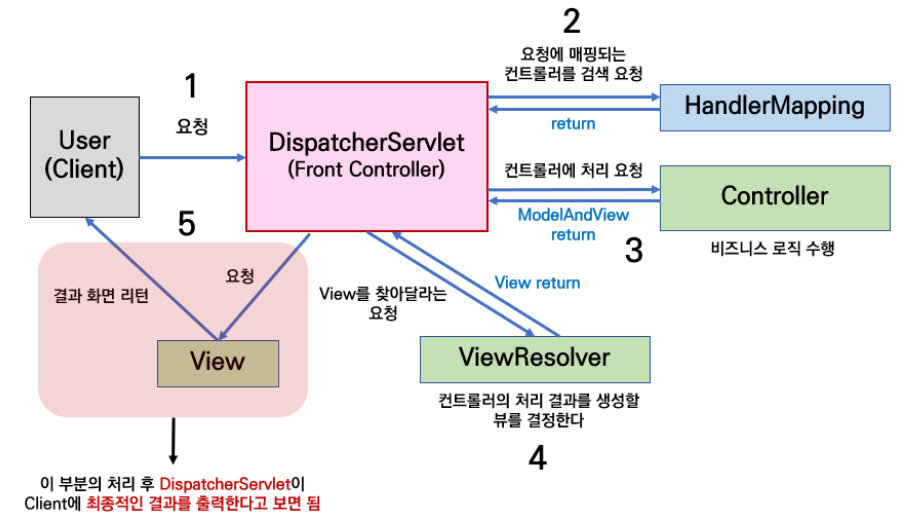
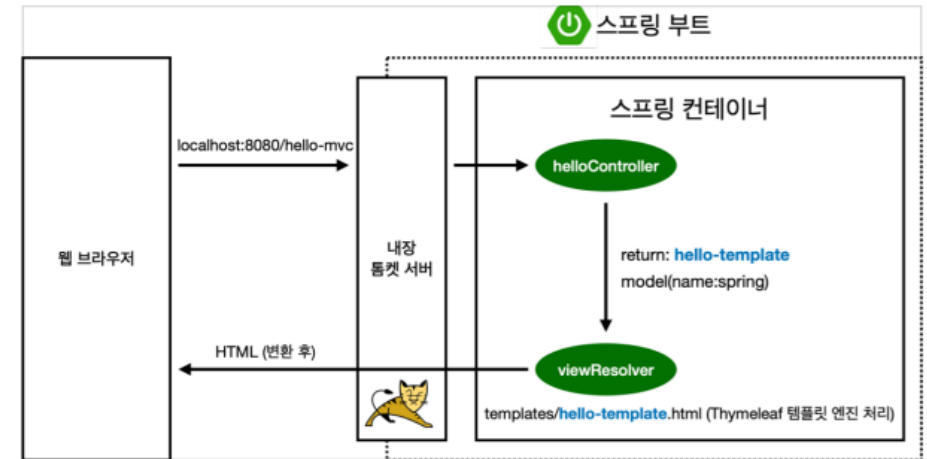
- 스프링 부트 프레임워크는?
  - 아파치 웹 서버 + 톰캣(TOMCAT)
    - 백엔드 개발 필수
    - 내부 객체의 생성부터 초기화, 호출, 종료 관리
    - 멀티 스레딩 지원 등

## 컨테이너 전체 동작 과정 요약

- (1) 클라이언트 요청(URI 매핑)
- (2) 어댑터 조회 및 컨트롤러 호출
- (3) 응답 반환(응답)
- (4) 뷰 리졸버 실행(뷰 렌더링)
- (5) 화면 리턴

## 뭔가 복잡해 보이지만?

- 중요, 서블릿 컨테이너를 통해서 동작한다.

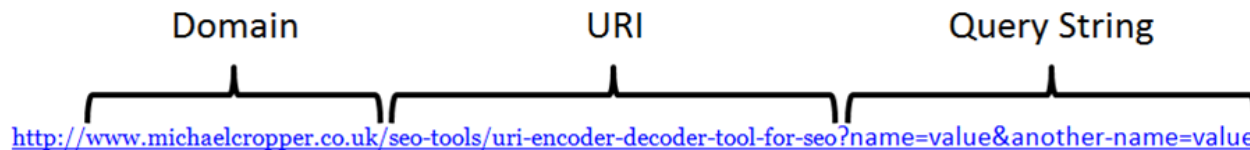


# 스프링 부트 동작 과정

- 서블릿(DispatcherServlet) 내부 동작

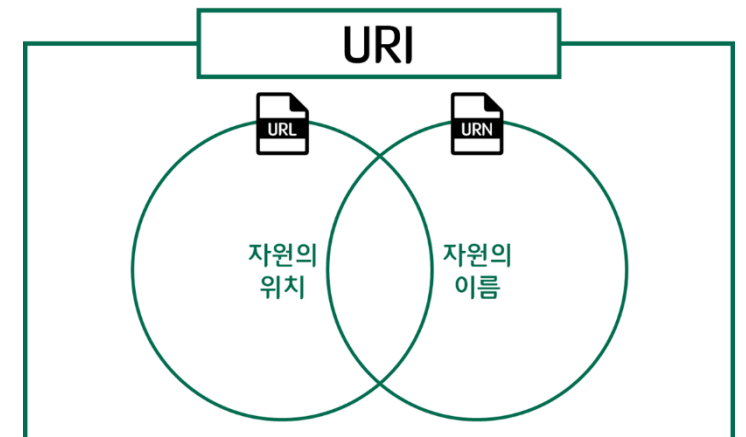
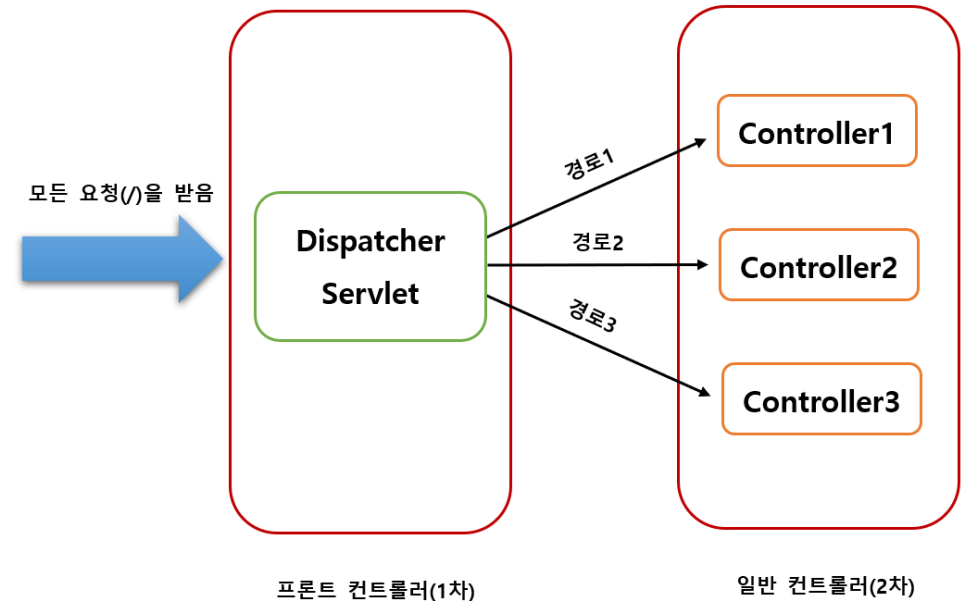
- 프론트 컨트롤러 역할(페이지 요청/응답 처리)
  - 1. html 페이지 요청
  - 2. uri 읽은 후 .java 파일 → .class 변환
  - 3. html 파일 생성 후 페이지 응답

- 참고 : URI 또는 URL(잘 살펴보자.)



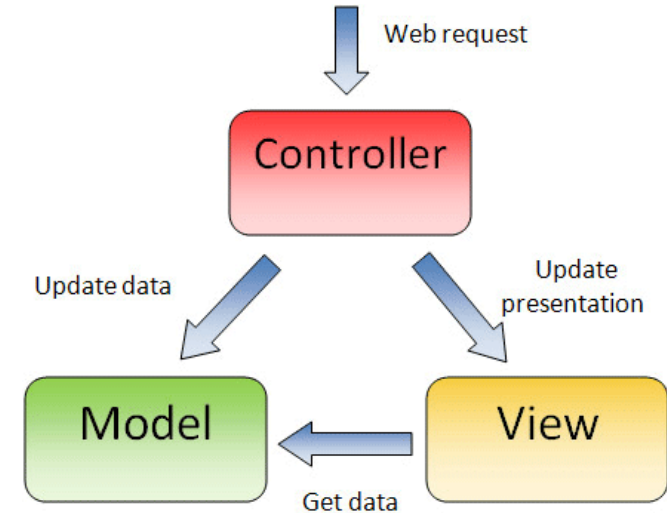
Domain:            The physical server where your website is hosted  
URI:                The identifier which maps to files on your server  
Query String:     Part of a GET request to easily pass in values to customise the output

\* Note: URI stands for Uniform Resource Identifier

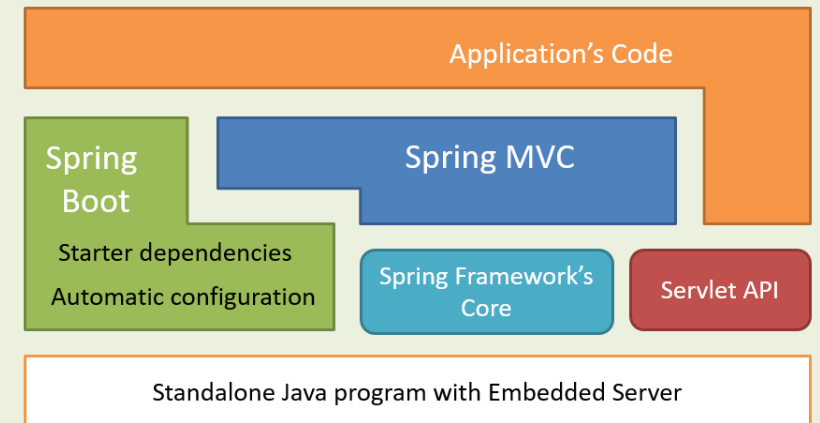


# 스프링 부트 동작 과정

- 기본 **MVC** 디자인 패턴 적용
  - M(MODEL) : 페이지 정보 저장
  - V(VIEW) : 페이지 화면 구성
    - 템플릿 엔진 : jsp, thymeleaf 등
  - C(CONTROLLER) : 페이지 흐름 제어
- 참고 : 디자인 패턴
  - 개발에 앞서 설계 구조 정형화(방법론)



## Spring Boot's Architecture



# Thymeleaf란?

- HTML 기반 템플릿 엔진
  - 템플릿 엔진은 다양한 엔진 존재
    - 고전 JSP는 선호 X
    - HTML, XML, JS, CSS 등 지원
- MVC에서 화면 출력 : 뷰(V) 역할
  - DispatcherServlet을 통해 동작
  - TemplateEngine 내장
- 주요 기능
  - 데이터 바인딩, 연산, 객체 호출 등
  - 템플릿 Fragment
  - 전용 제어문(반복/조건 등)



Thymeleaf

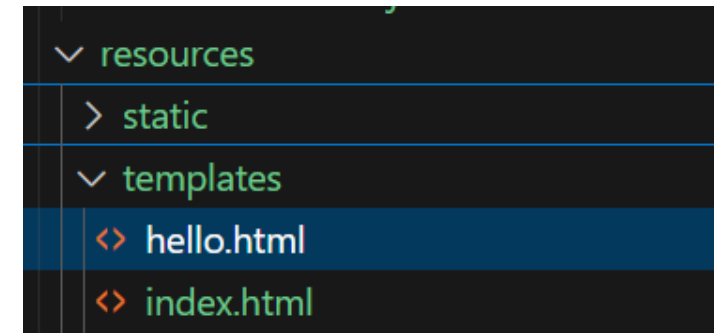
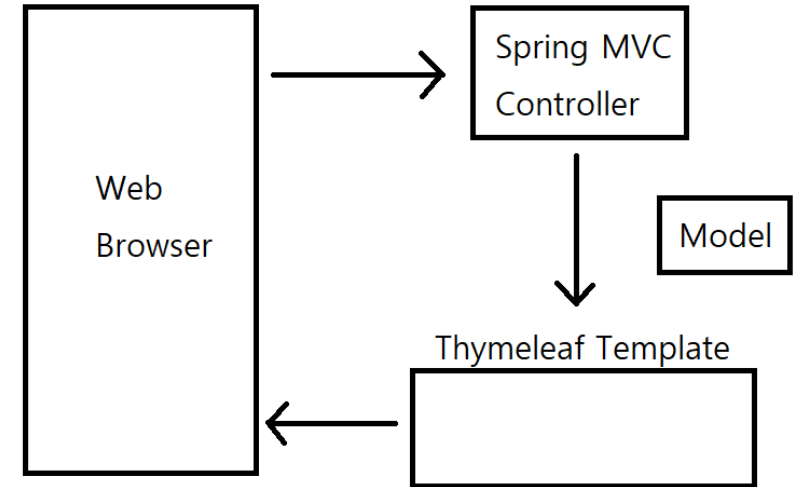


JSP

Template	Spring Boot starter dependency
FreeMarker	spring-boot-starter-freemarker
Groovy templates	spring-boot-starter-groovy-templates
JavaServer Pages (JSP)	None (provided by Tomcat or Jetty)
Mustache	spring-boot-starter-mustache
Thymeleaf	spring-boot-starter-thymeleaf

# Thymeleaf란?

- 모든 링크는 컨트롤러를 통해 동작
  - URL 요청 이후 컨트롤러는 viewName 리턴
    - 소스 코드 다시 살펴보기
- 내부 구조 및 설정 확인
  - 메인화면 index.html의 위치 확인
    - resources 폴더 안에 templates
    - static 폴더에는 이외 자원들을 위치
  - 리턴 받은 이름의 페이지 연결
    - viewName.html
- 참고 : 기본 템플릿 엔진
  - 최상위 폴더 : pom.xml 설정 파일



```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```



# Thymeleaf란?



- **오늘의 문법 사용하기(매 주 업데이트)**
  - .html에서 어떻게 사용되나?
    - 선언
      - `<html lang="en" xmlns:th="http://www.thymeleaf.org">`
    - 주석
      - `<!--/* This code will be removed at thymeleaf parsing time! */-->`
    - 텍스트 출력
      - `<sapn th:text=${data}>`
      - 참고 : `${...}` 변수 표현식
  - 참고 : 공식 웹사이트
    - <https://www.thymeleaf.org/>

# URL 맵핑과 컨트롤러

- 루트 폴더 하위의 resource 폴더
  - **Templates의 index.html 파일을 수정한다.**
- 임시로 hello.html를 생성한다.
  - 방갑습니다. 입력
  - 빈 파일이어도 상관없음

안녕하세요!

[홈페이지 메인](#)

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>index 메인페이지</title>
</head>
<body>
  <h1>안녕하세요!</h1>
  <a href="/hello.html">홈페이지 메인</a>
</body>
</html>
```

잘못 연결? 무엇이 문제인가? 페이지 404 에러

## Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Jul 30 16:39:51 KST 2024

There was an unexpected error (type=Not Found, status=404).

No static resource hello.html.

```
org.springframework.web.servlet.resource.NoResourceFoundException: No static resource hello.html.
    at org.springframework.web.servlet.resource.ResourceHttpRequestHandler.handleRequest(ResourceHttpRequestHandler.java:585)
    at org.springframework.web.servlet.mvc.HttpRequestHandlerAdapter.handle(HttpRequestHandlerAdapter.java:52)
    at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1089)
    at org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:979)
    at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1014)
    at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:903)
    at jakarta.servlet.http.HttpServlet.service(HttpServlet.java:564)
    at org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:885)
    at jakarta.servlet.http.HttpServlet.service(HttpServlet.java:658)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:195)
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:140)
```

# URL 맵핑과 컨트롤러

- 루트 폴더 하위의 demo 폴더
  - DemoController.java 파일을 생성한다.
    - 대 소문자 및 이름이 일치해야 한다. 주의!
  - 다음과 같이 작성하고 저장한다.
    - 패키지 : 프로젝트 폴더 구조 구분

```
package com.example.demo;
```

```
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;
```

```
@Controller // 컨트롤러 어노테이션 명시  
Public class DemoController {  
    @GetMapping(" /hello ") // 전송 방식 GET  
    public String hello(Model model) {  
        model.addAttribute("data", " 반갑습니다."); // model 설정  
        return "hello"; // hello.html 연결  
    }  
}
```

```
src  
└─ main  
    └─ java\com\example\demo  
        ├─ DemoApplication.java  
        └─ DemoController.java
```

## 클래스 이름 작성 규칙

작성 규칙	예
하나 이상의 문자로 이루어져야 한다.	Car, SportsCar
첫 번째 글자는 숫자가 올 수 없다.	3Car (X)
'\$', '_' 외의 특수 문자는 사용할 수 없다.	\$Car, _Car, @Car (X), #Car (X)
자바 키워드는 사용할 수 없다.	int (X), for (X)

# URL 맵핑과 컨트롤러

- 루트 폴더 하위의 resource 폴더
  - templates의 index.html 파일을 수정한다.
    - 기존 링크를 /hello 로 수정한다.
  - hello.html 내용도 추가한다.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>index 메인페이지</title>
</head>
<body>
  <h1>안녕하세요!</h1>
  <a href="/hello">홈페이지 메인</a>
</body>
</html>
```

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Hello 페이지</title>
</head>
<body>
  <h1>안녕하세요!</h1>
  <p th:text="${data}"></p>
  <a href="/">홈페이지 메인</a>
</body>
</html>
```

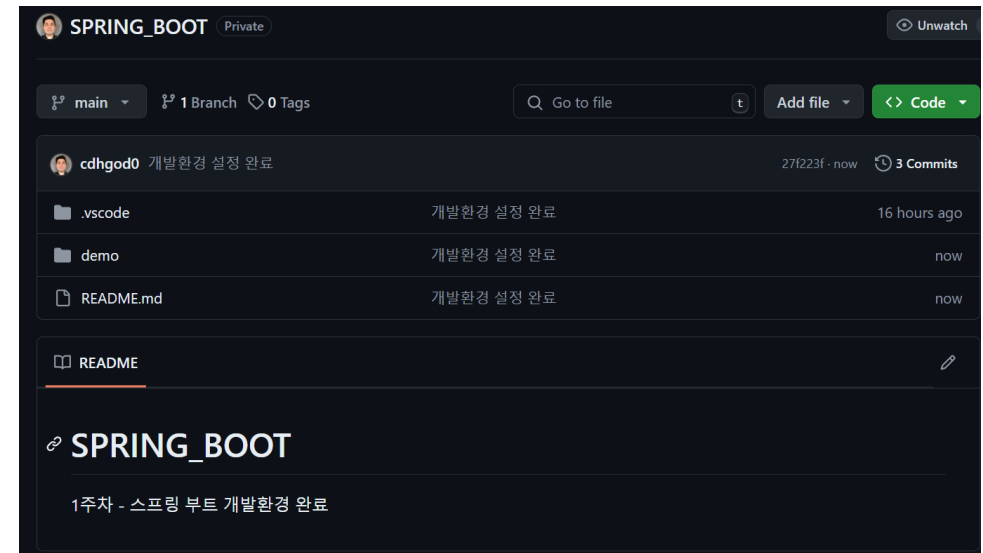
**안녕하세요!**

방갑습니다.

[홈페이지 메인](#)

# 잠깐! GIT 연동 - 소스코드 업로드

- README.MD 파일 열기 수정 예)
  - 1주차 스프링 부트 개발환경, 테스트 완료
  - [index.html : 실행/수정 완료]([깃 허브 소스코드 URL](#))
  - 추가된 파일 + 로 추가 → COMMIT (메인화면)
  - PUSH 하여 서버에 파일 업로드
- 앞으로 실습 코드를 계속 업데이트 하자.
  - 폴더 또는 소스코드를 링크 가능하다.
  - 참고 : 깃 허브 소스코드 주소?
    - <https://github.com/> 접속
      - 저장소 → 소스 파일 확인
    - 업로드 되어 있는 소스의 주소를 복사(오른쪽 클릭)



## • 2주차 연습문제

URL 맵핑과 컨트롤러 추가하기



# URL 맵핑과 컨트롤러 추가하기

- 기존 hello.html 링크를 수정한다.
  - 링크 : 두번째 헬로 페이지
  - url 맵핑 : /hello2
- 컨트롤러에 맵핑을 추가 한다.
  - hello2 메서드를 작성한다.
    - 5개의 속성을 각자 추가한다.
- hello2.html 작성한다.
  - 5개 속성 변수를 출력한다.
- 실습 결과 확인 – Q / A

**안녕하세요!**

방갑습니다.

[두번째 헬로 페이지](#)

**안녕하세요!**

홍길동님.

방갑습니다.

오늘.

날씨는.

매우 좋습니다.

[홈페이지 메인](#)

# Q & A

- 무엇이든 물어보세요!

