

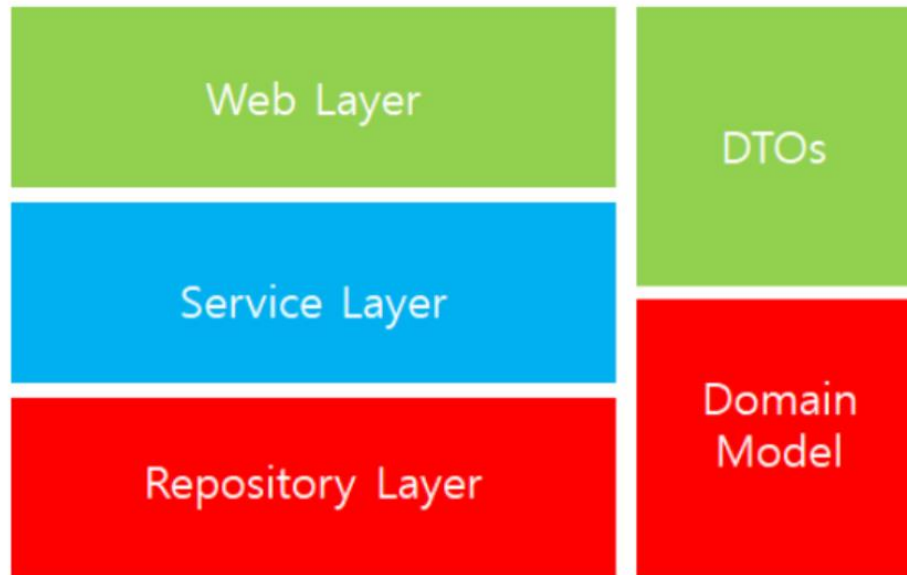
# Spring Boot

# Spring Boot

---

- Spring Boot

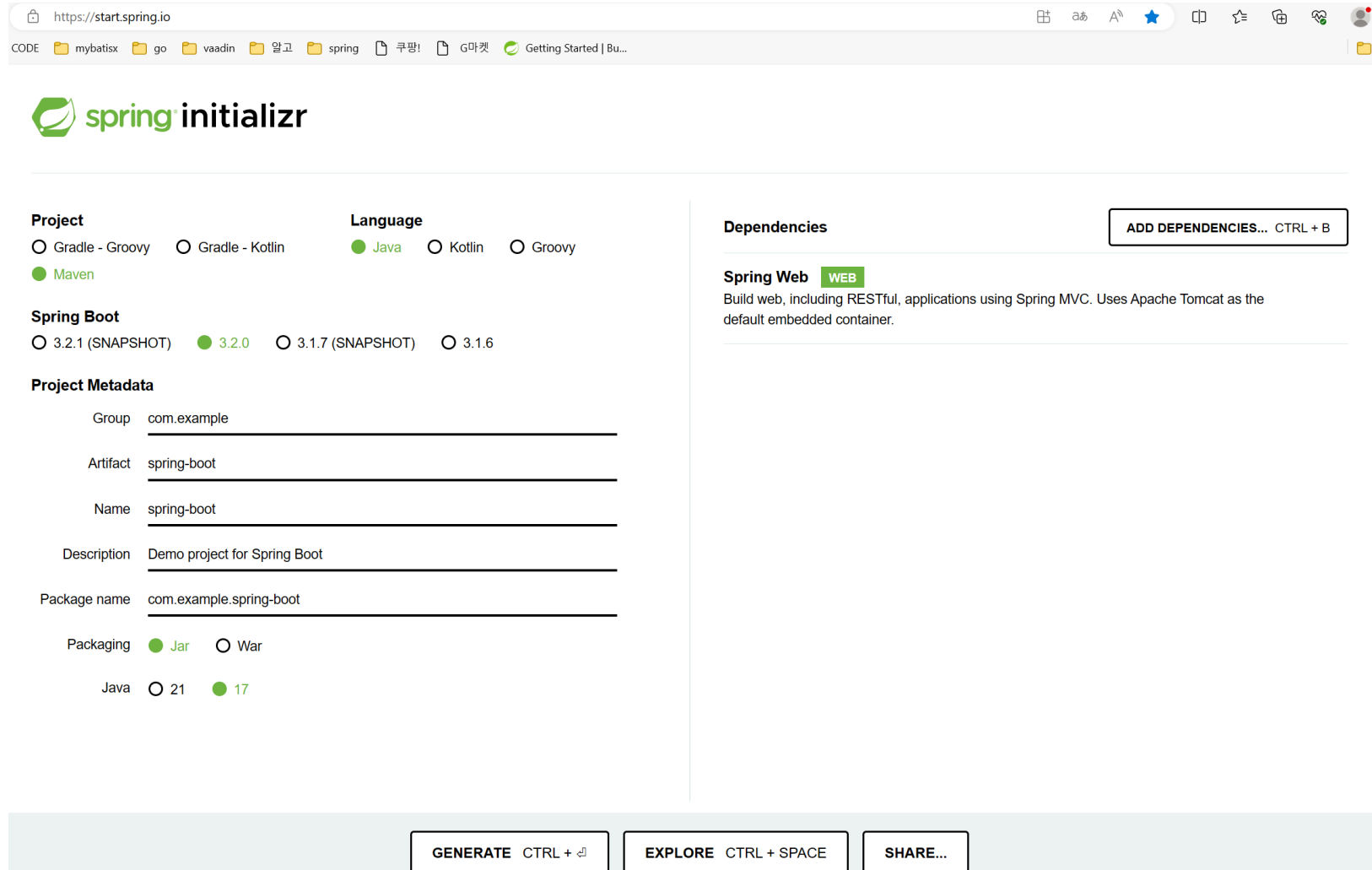
스프링 웹 계층



# Spring Boot

## ■ Spring Boot

<https://start.spring.io/>



The screenshot shows the Spring Boot Start page with the following configuration:

- Project:** ☐ Gradle - Groovy, ☐ Gradle - Kotlin, ☒ **Maven**
- Language:** ☒ **Java**, ☐ Kotlin, ☐ Groovy
- Spring Boot:** ☐ 3.2.1 (SNAPSHOT), ☒ **3.2.0**, ☐ 3.1.7 (SNAPSHOT), ☐ 3.1.6
- Project Metadata:**
  - Group:
  - Artifact:
  - Name:
  - Description:
  - Package name:
  - Packaging: ☒ **Jar**, ☐ War
  - Java: ☐ 21, ☒ **17**
- Dependencies:** 
  - Spring Web** ☒ **WEB**  
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

At the bottom, there are three buttons: , , and

# Spring Boot

---

## ■ Spring Boot

<https://spring.io/projects/spring-boot>

Spring Boot를 사용하면 "그냥 실행"할 수 있는 독립 실행형 프로덕션 등급 Spring 기반 애플리케이션을 쉽게 만들 수 있습니다.

Spring 플랫폼과 타사 라이브러리에 대한 독단적인 관점을 취하므로 최소한의 번거로움으로 시작할 수 있습니다.

대부분의 Spring Boot 애플리케이션에는 최소한의 Spring 구성이 필요합니다.

<https://spring.io/guides/gs>

# Spring Boot

---

## ■ Spring Boot 기능

독립 실행형 Spring 애플리케이션 만들기

Tomcat, Jetty 또는 Undertow 직접 포함(WAR 파일을 배포할 필요 없음)

독단적인 '시작' 종속성을 제공하여 빌드 구성을 단순화합니다.

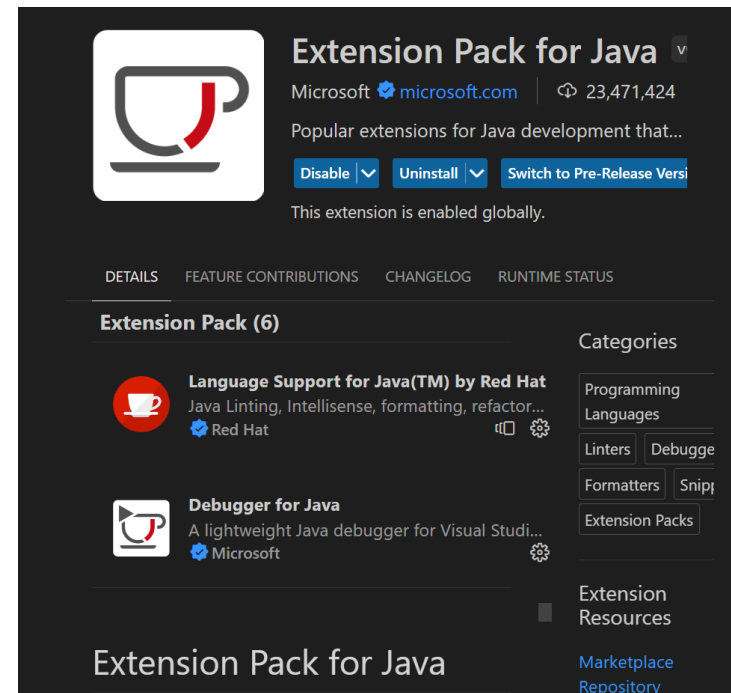
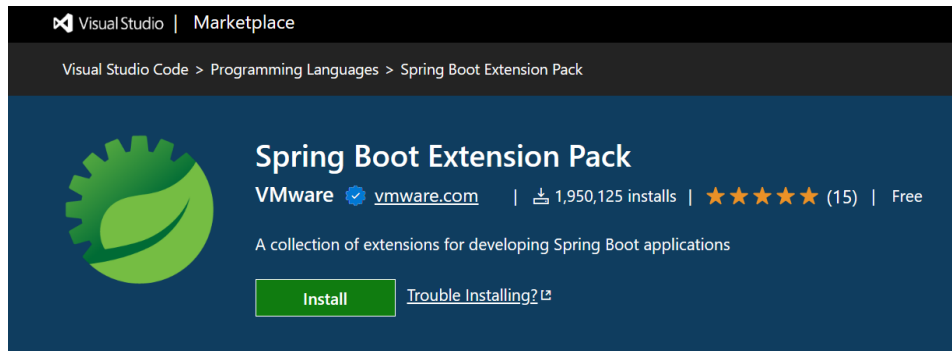
가능할 때마다 Spring 및 3rd 파티 라이브러리를 자동으로 구성합니다.

메트릭, 상태 확인 및 외부화된 구성과 같은 프로덕션에 바로 사용할 수 있는 기능 제공  
코드 생성 및 XML 구성에 대한 요구 사항이 전혀 없습니다.

# Spring Boot

## ■ Spring Boot IDE

<https://spring.io/guides/gs/guides-with-vscode/>



# Spring Boot

## ■ Spring Boot 기능

```
git clone https://github.com/spring-guides/gs-rest-service.git
```

명령 프롬프트

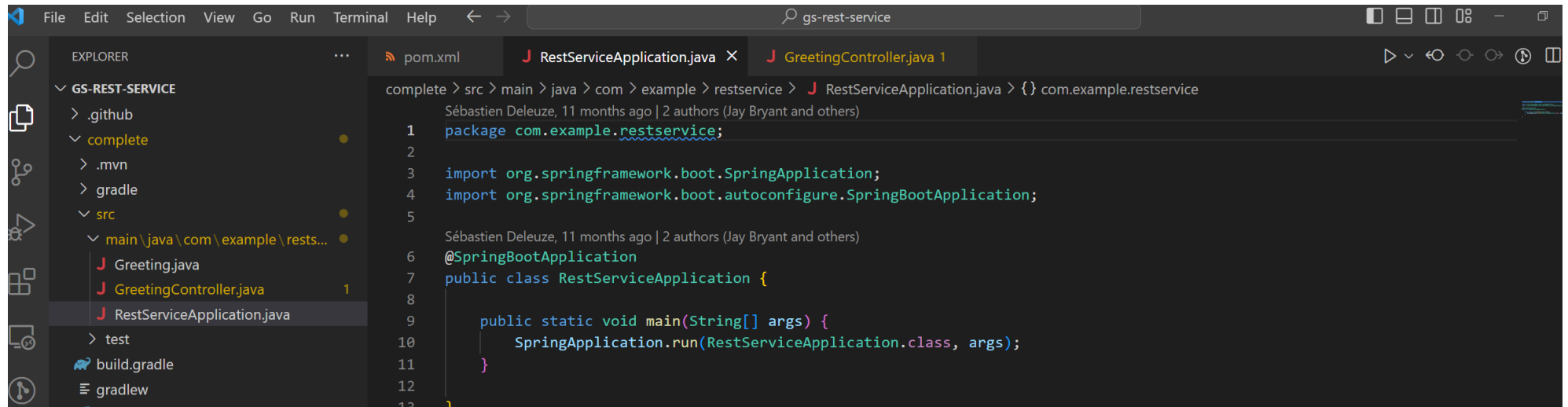
```
E:\springvscode>
E:\springvscode>git clone https://github.com/spring-guides/gs-rest-service.git
Cloning into 'gs-rest-service'...
remote: Enumerating objects: 2076, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (19/19), done.
remote: Total 2076 (delta 12), reused 12 (delta 5), pack-reused 2051 receiving objects: 97% (2014/2076)
Receiving objects: 100% (2076/2076), 1.55 MiB | 14.55 MiB/s, done.
Resolving deltas: 100% (1111/1111), done.
Updating files: 100% (38/38), done.
```

```
E:\springvscode>cd E:\springvscode\gs-rest-service
```

```
E:\springvscode\gs-rest-service>code .
```

# Spring Boot

## ■ Spring Boot 기능



```
er : An incompatible version [1.2.26] of th
ver : Tomcat initialized with port(s): 8080
ce : Starting service [Tomcat]
e : Starting Servlet engine: [Apache Tomcat/10.1
```

```
@RestController
public class GreetingController {

    private static final String tem
    private final AtomicLong counte

    @GetMapping("/greeting")
    public Greeting greeting(@Reque
        return new Greeting(counte
    }
}
```

<http://localhost:8080/greeting>



# Spring Boot

---

## ■ Spring Boot

<https://spring.io/guides/gs/guides-with-vscode/>

VS Code with Java support

If you don't have VS Code installed yet: Coding Pack for Java

If you have VS Code installed already: Java Extension Pack

# Spring Boot

## ■ Spring Boot

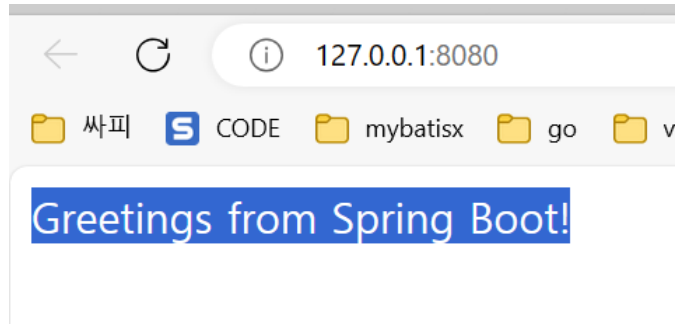
```
E:\springvscode>git clone https://github.com/spring-guides/gs-spring-boot.git
Cloning into 'gs-spring-boot'...
remote: Enumerating objects: 1657, done.
remote: Counting objects: 100% (118/118), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 1657 (delta 47), reused 102 (delta 45), pack-reused 1539
Receiving objects: 100% (1657/1657), 1.06 MiB | 15.08 MiB/s, done.
Resolving deltas: 100% (1050/1050), done.
```

git clone https://github.com/spring-guides/gs-spring-boot.git

```
E:\springvscode>cd E:\springvscode\gs-spring-boot
E:\springvscode\gs-spring-boot>code .
```

# Spring Boot

## ■ Spring Boot



```
E:\springvscode\gs-spring-boot>curl localhost:8080/actuator/health  
{"status":"UP"}  
E:\springvscode\gs-spring-boot>
```

`curl localhost:8080/actuator/health`

# Spring Boot

---

## ■ maven

<https://maven.apache.org/>

<https://maven.apache.org/what-is-maven.html>

지식 축적자라는 뜻의 이디시어 인 메이븐(Maven)은 자카르타 터빈 프로젝트의 구축 프로세스를 단순화하려는 시도로 시작되었습니다. 각각 자체 Ant 빌드 파일이 있는 여러 프로젝트가 있었는데 모두 조금씩 달랐습니다. JAR이 CVS에 체크인되었습니다. 우리는 프로젝트를 빌드하는 표준 방법, 프로젝트 구성에 대한 명확한 정의, 프로젝트 정보를 게시하는 쉬운 방법, 여러 프로젝트에서 JAR을 공유하는 방법을 원했습니다.

Maven의 주요 목표

빌드 프로세스를 쉽게 만들기  
균일한 빌드 시스템 제공  
양질의 프로젝트 정보 제공  
더 나은 개발 관행 장려

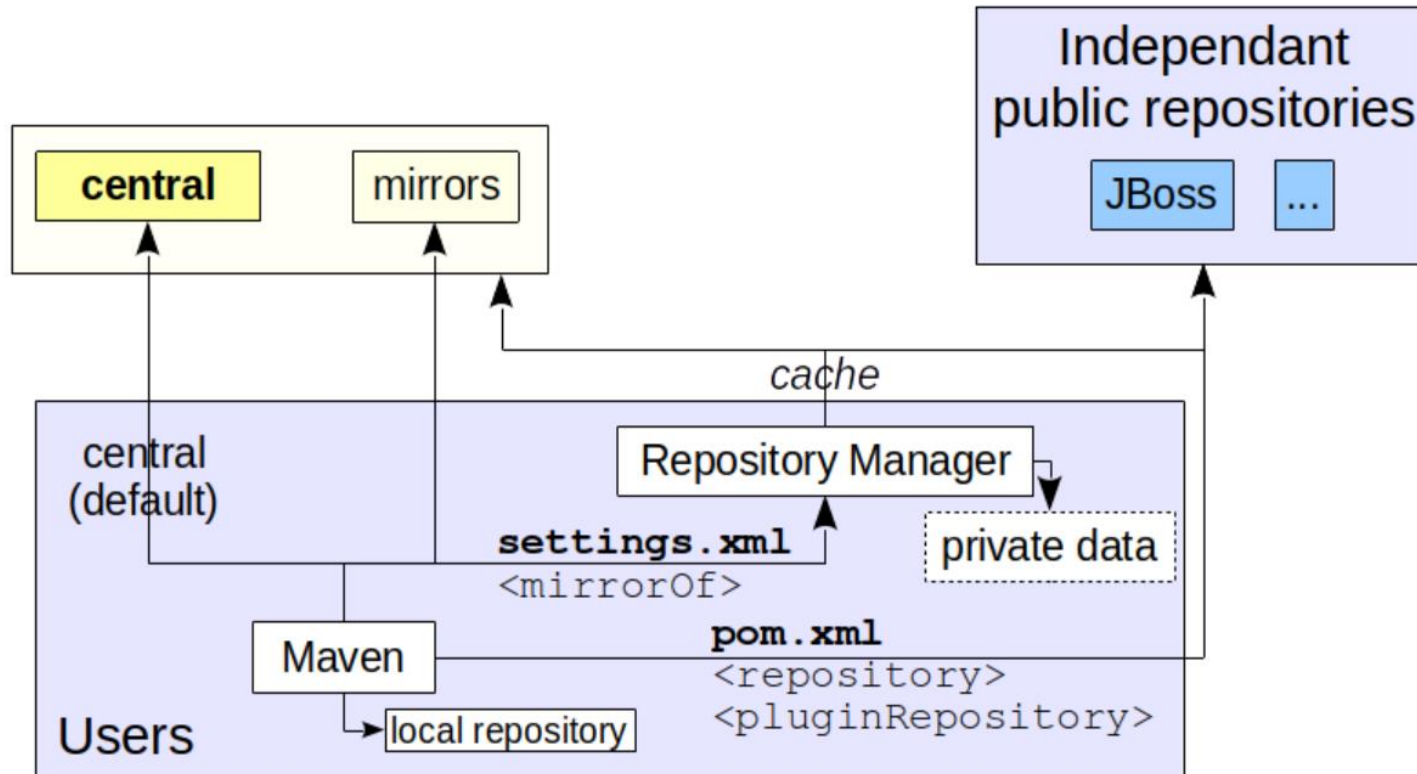
# Spring Boot

## ■ Maven repository

<https://maven.apache.org/repositories/index.html>

## Maven Repositories

Apache Maven uses repositories to store artifacts. Your dependencies are being downloaded from repositories, and artifacts you publish are stored in repositories. These are the fundamental concepts of Maven since its inception: Maven command line tool and Maven Repositories were mold together and designed to work together.



# Spring Boot

---

## ■ Spring Boot

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/kmboard?serverTimezone=Asia/Seoul&useUnicode=yes&characterEncoding=UTF-8
spring.datasource.username=kitri
spring.datasource.password=kitri
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql: true
```

```
    : spring.jpa.open-in-view is enabled
    : To disable this feature, set spring.jpa.open-in-view to false
    : Tomcat started on port(s): 8080 (http)
    : Started AccessingDataMysqlApplication in 1.5 seconds
```

<https://spring.io/guides/gs/accessing-data-mysql>

# Spring Boot

## ■ Spring Boot

```
@Controller // This means that this cl
@RequestMapping(path="/demo") // This
public class MainController {
    @Autowired // This means to get th
    // Which is auto-genera
    private UserRepository userReposit


    @PostMapping(path="/add") // Map C
    public @ResponseBody String addNew
        , @RequestParam String ema
        // @ResponseBody means the ret
        // @RequestParam means it is a

    User n = new User();
    n.setName(name);
    n.setEmail(email);
    userRepository.save(n);
    return "Saved";
}

@GetMapping(path="/all")
public @ResponseBody Iterable<User>
```

# Spring Boot

## ■ Spring Boot

 <http://localhost:8080/demo/add?name=kildong&email=kildong@naver.com>

POST



<http://localhost:8080/demo/add?name=kildong&email=kildong@naver.com>

Params ● Authorization Headers (7) Body Pre-request Script Tests Settings

### Query Params

	Key	Value
<input checked="" type="checkbox"/>	name	kildong
<input checked="" type="checkbox"/>	email	kildong@naver.com
	Key	Value

<http://localhost:8080/demo/add?name=kildong&email=kildong@naver.com>



# Spring Boot

## ■ Spring Boot

The screenshot shows a web browser interface with the address bar displaying `http://localhost:8080/demo/all`. The request method is `GET`. The response body is displayed in the 'Body' tab, showing a JSON array with one object:

```
[{"id": 1, "name": "kildong", "email": "kildong@naver.com"}]
```

`http://localhost:8080/demo/all`

Hibernate: select next\_val as id\_val from user\_seq for update

Hibernate: update user\_seq set next\_val= ? where next\_val=?

Hibernate: insert into user (email,name,id) values (?,?,:) ?

Hibernate: select u1\_0.id,u1\_0.email,u1\_0.name from user u1\_0

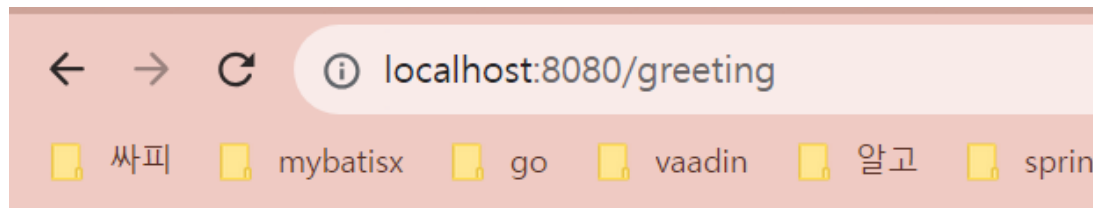
# Spring Boot

## ■ Spring Boot MVC

<https://spring.io/guides/gs/serving-web-content/>

git clone <https://github.com/spring-guides/gs-serving-web-content.git>

```
E:\springvscode>git clone https://github.com/spring-guides/gs-serving-web-content.git
Cloning into 'gs-serving-web-content'...
remote: Enumerating objects: 1266, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (61/61), done.
remote: Total 1266 (delta 69), reused 82 (delta 46), pack-reused 1155R
Receiving objects: 100% (1266/1266), 1.13 MiB | 3.92 MiB/s, done.
Resolving deltas: 100% (791/791), done.
Updating files: 100% (38/38), done.
```



Hello, World!

```
<p th:text="|Hello, ${name}!|" />
```