

# 스프링 프로젝트

# 스프링 프로젝트

- ✓ 스프링 프로젝트를 생성하고, 스프링의 기능을 하나씩 추가해서 하나의 프로젝트를 완성

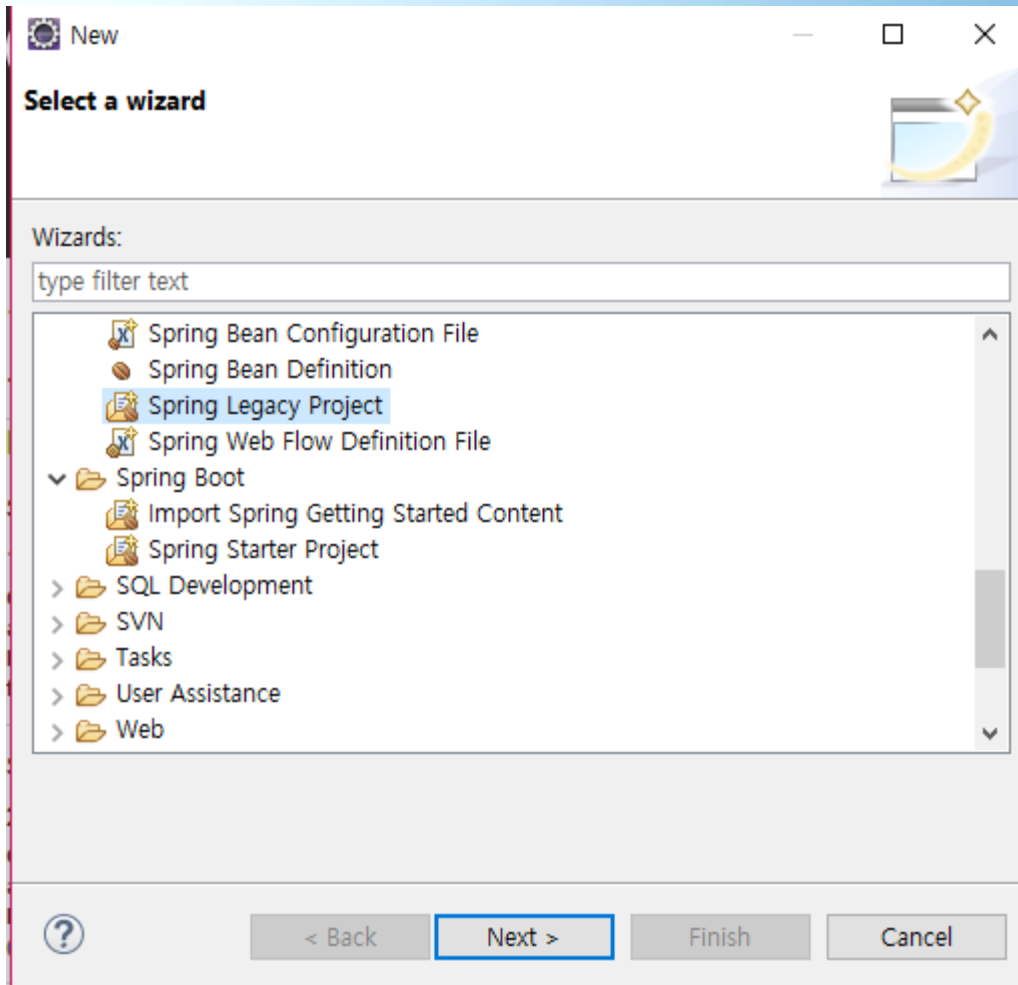
간단히 스프링 프레임워크를 살펴보겠습니다.

- ❖ 스프링은 POJO (Plain Old Java Object) 방식의 프레임워크로서, 일반적인 J2EE 프레임워크에 비해 특정 라이브러리를 사용할 필요가 없어서 개발이 쉬우며, 기존 라이브러리의 지원이 용이
  - ❖ 스프링은 관점지향프로그래밍, AOP(Aspect Oriented Programming)를 지원  
트랜잭션, 로깅, 보안 등 여러 모듈, 여러 계층에서 적용되는데, 이런 코드들을 실제 비즈니스 로직과 분리할 수 있도록 도와 줌. 한때, AOP가 OOP(Object Oriented Programming)를 대체하는 기술로 생각되기도 했지만, 실제로 AOP는 OOP를 더욱 OOP스럽게 보완해 주는 기술
  - ❖ 스프링은 의존성 주입, DI (Dependency Injection)를 지원합니다. 이는 객체간의 의존관계를 관리하는 기술. 어떤 객체가 필요로 하는 객체를 자기 자신이 직접 생성하는것이 아니라, 외부에 있는 다른곳에서 자신이 필요로 하는 객체를 주입 받는 것을 말함
  - ❖ 스프링은 제어 반전, IoC (Inversion of Controller)를 지원합니다. 컨트롤의 제어권이 개발자가 아니라 프레임워크에 있음을 말합니다. 즉, 객체의 생성부터 모든 생명주기의 관리까지 객체의 제어권이 바뀐것을 의미
- ✓ 스프링의 가장 큰 특징은 AOP, POJO, DI, PSA (Portable Service Abstraction)

# 스프링 프로젝트

## 1. 스프링 프로젝트 생성

- 1) File > New > Other 를 선택
- 2) Spring > Spring Legacy Project를 선택

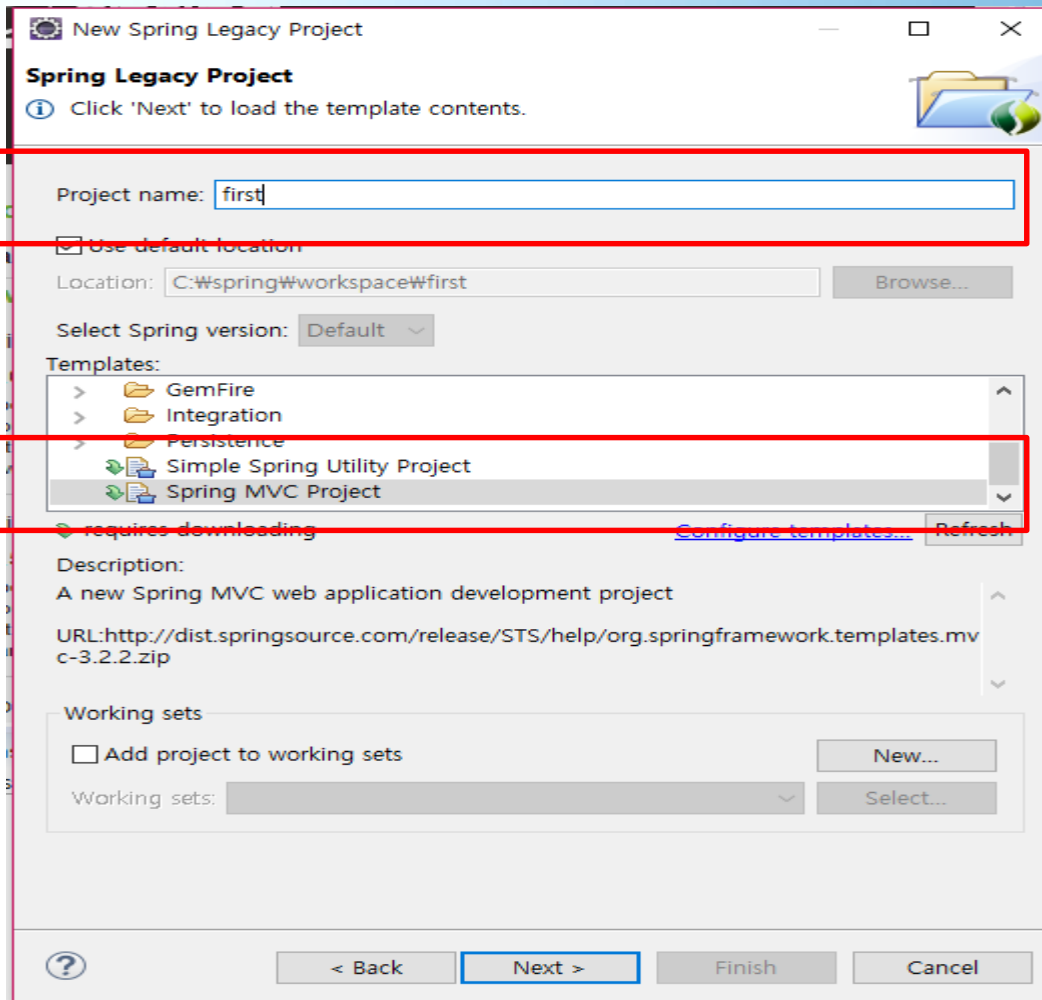


# 스프링 프로젝트

## 1. 스프링 프로젝트 생성

3) 프로젝트의 이름을 입력하고, Spring MVC Project를 선택한다.

✓ 최초의 스프링 프로젝트이니, 프로젝트의 이름을 first 로 지정



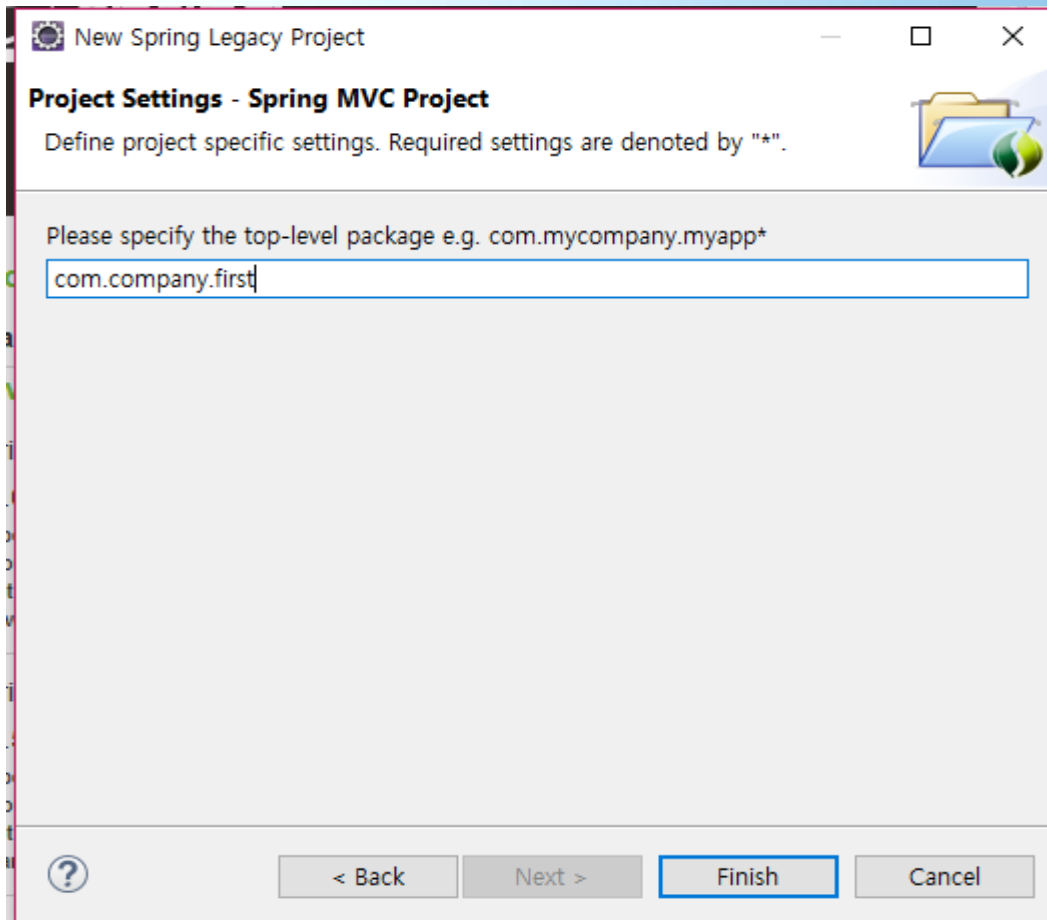
# 스프링 프로젝트

1. 스프링 프로젝트 생성

4) package를 입력한다.

✓ package는 최소 3레벨 이상 ( [1레벨].[2레벨].[3레벨] )로 구성

✓ "com.company.first"라는 package를 사용





# 스프링 프로젝트

## 1. 스프링 프로젝트 생성

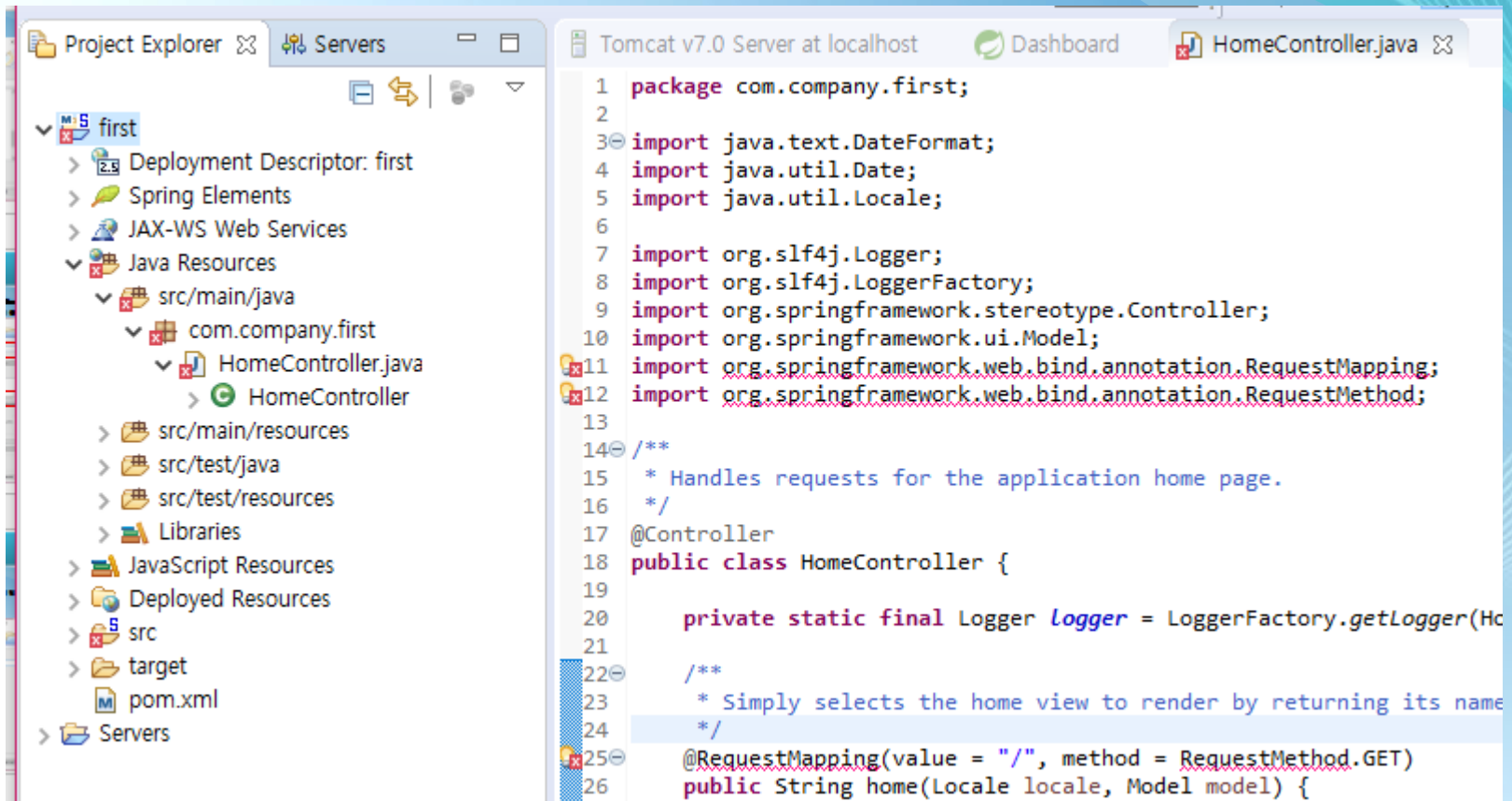
5) Finish를 누르면 프로젝트가 생성이 되고, 인터넷에서 스프링 프로젝트에 필요한 라이브러리를 자동으로 다운받기 시작

- ✓ 생성한 Spring MVC Project에는 여러 가지 라이브러리들이 필요한데, 프로젝트의 생성과 동시에 메이븐이 자동적으로 인터넷에서 필요한 라이브러리를 다운받는 과정

# 스프링 프로젝트

## 1. 스프링 프로젝트 생성

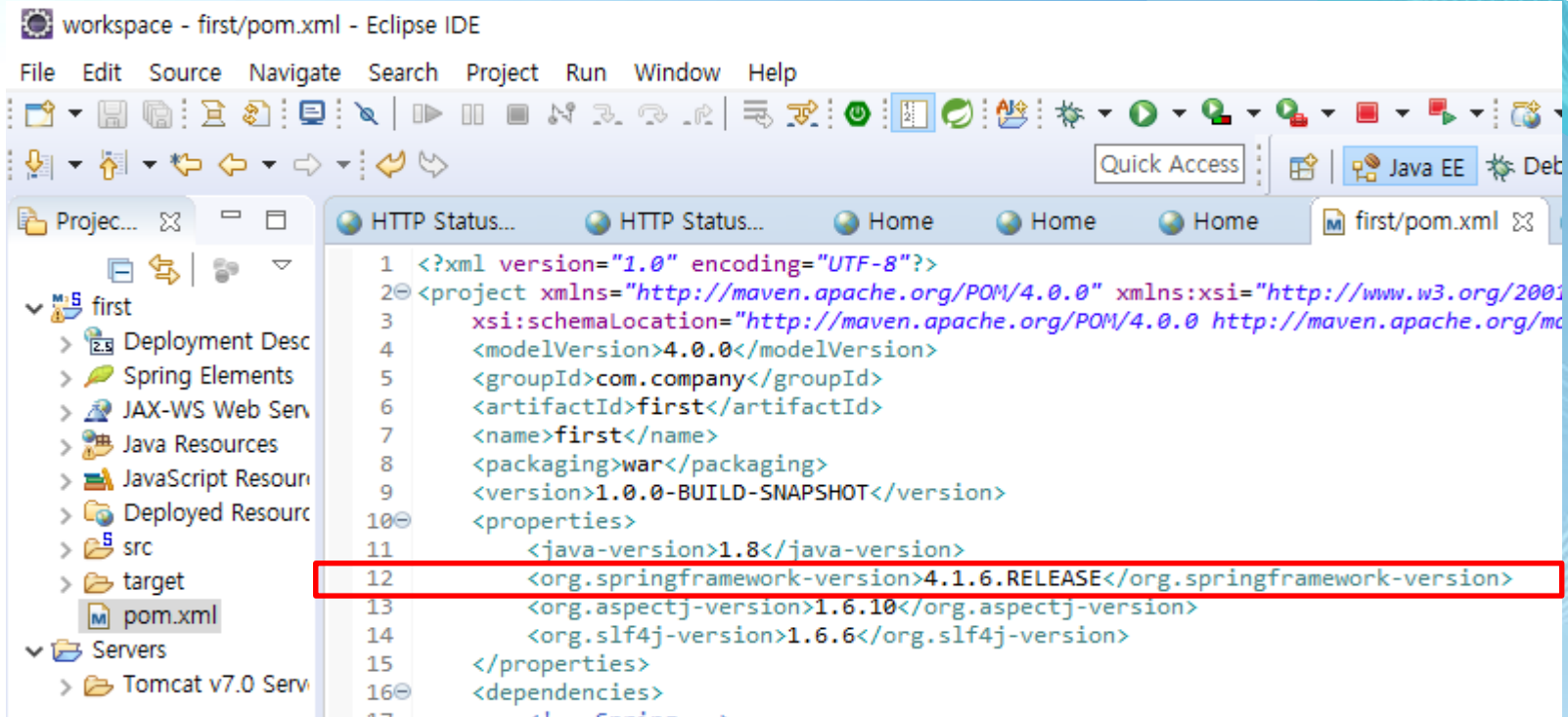
### 6) 다음과 같은 에러시 해결



# 스프링 프로젝트

## 1. 스프링 프로젝트 생성

### 6) pom.xml 수정



workspace - first/pom.xml - Eclipse IDE

File Edit Source Navigate Search Project Run Window Help

Project... first

- Deployment Desc
- Spring Elements
- JAX-WS Web Serv
- Java Resources
- JavaScript Resourc
- Deployed Resourc
- src
- target
- pom.xml

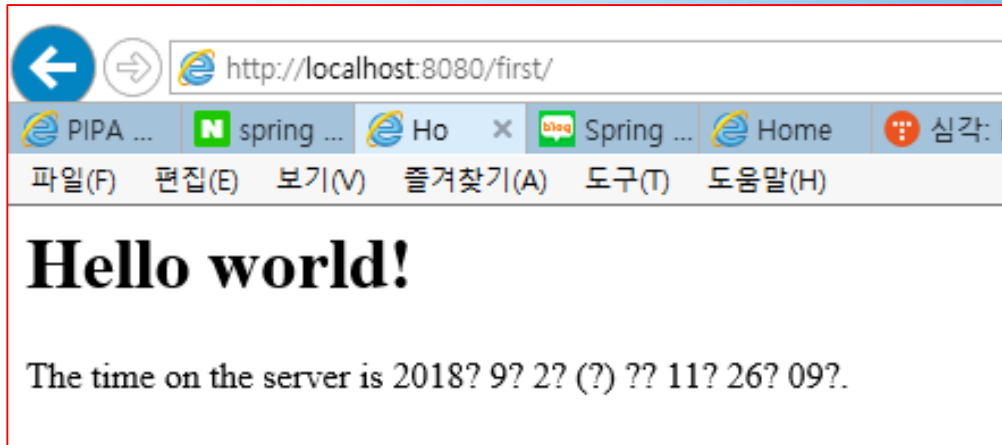
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5   <groupId>com.company</groupId>
6   <artifactId>first</artifactId>
7   <name>first</name>
8   <packaging>war</packaging>
9   <version>1.0.0-BUILD-SNAPSHOT</version>
10  <properties>
11    <java-version>1.8</java-version>
12    <org.springframework-version>4.1.6.RELEASE</org.springframework-version>
13    <org.aspectj-version>1.6.10</org.aspectj-version>
14    <org.slf4j-version>1.6.6</org.slf4j-version>
15  </properties>
16  <dependencies>
```



# 스프링 프로젝트

## 1. 스프링 프로젝트 생성

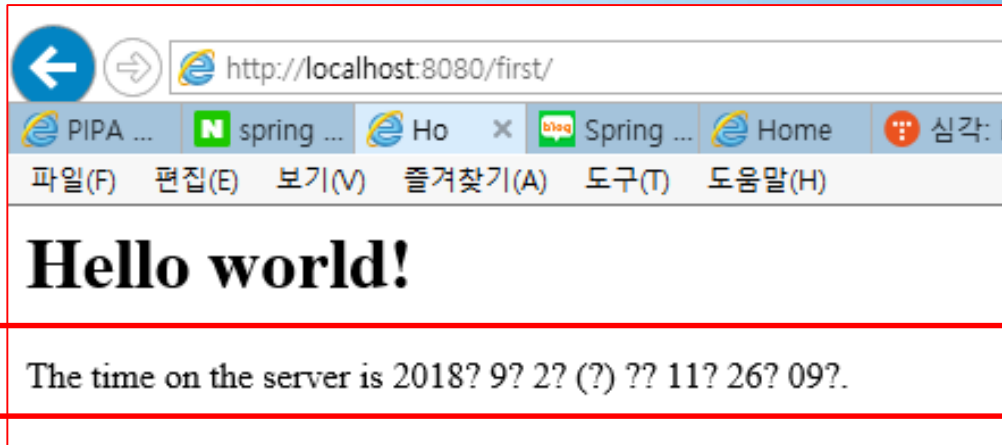
7) 브라우저를 실행시키고, 주소창에 <http://localhost:8080/first/> 를 입력



위와 같이 나오면 성공

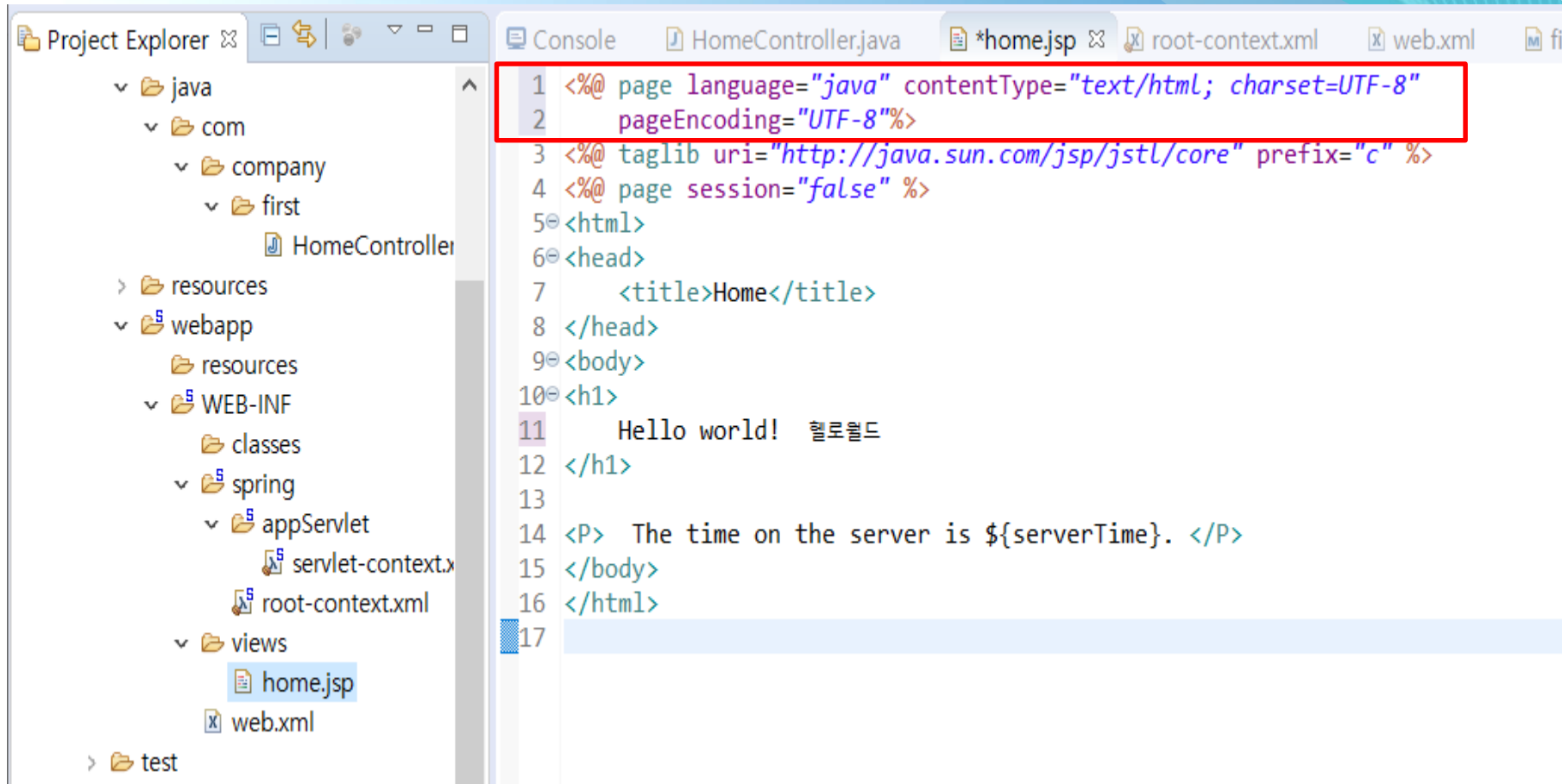
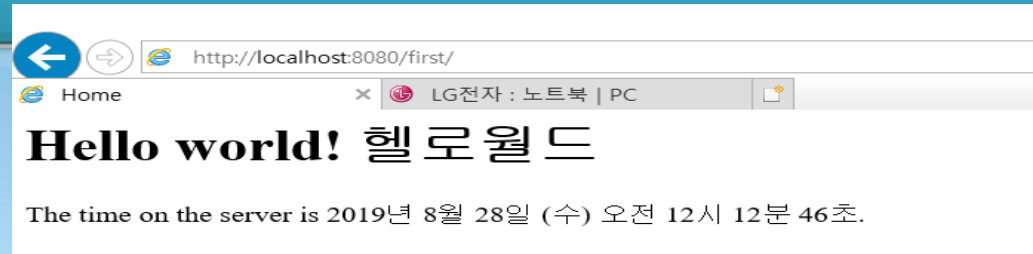
# 스프링 프로젝트

1. 스프링 프로젝트 생성
- 7) 한글 깨짐 현상 해결



# 스프링 프로젝트

1. 스프링 프로젝트 생성
  - 7) 한글 깨짐 현상 해결
- ✓ 다음과 같이 추가



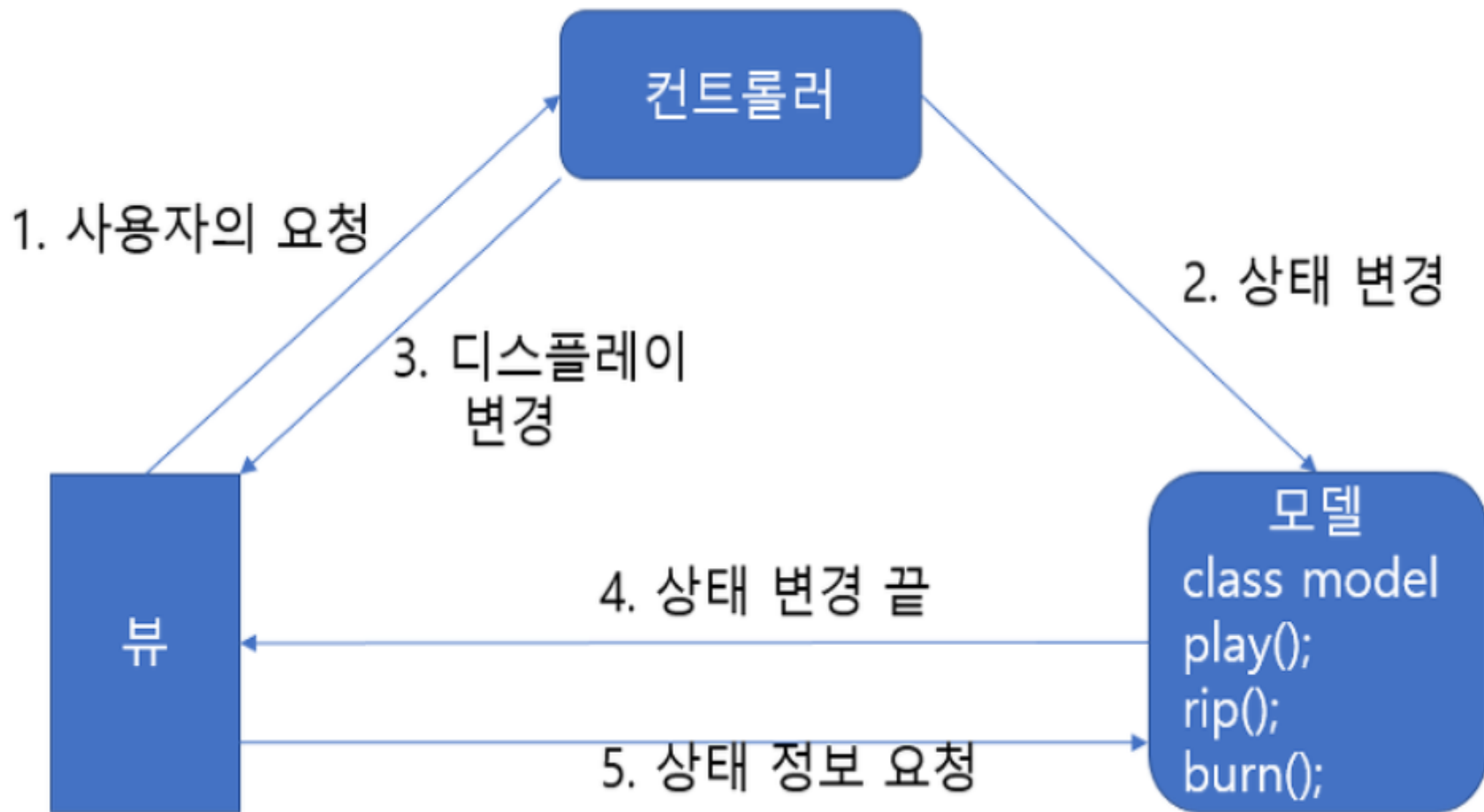
# 스프링 프로젝트

1. 스프링 프로젝트 생성

8) 스프링프레임워크를 수행할 수 있는 환경이 갖추어 졌다.

# 스프링 프로젝트

## 2. MVC 패턴





# 스프링 프로젝트

## 2. MVC 패턴

### (1) View

- ✓ 모델을 표현하는 방법을 제공하는 사용자 인터페이스.
- ✓ 일반적으로 화면에 표시하기 위해 필요한 상태 및 데이터를 모델에서 직접 가져온다.

### (2) Model

- ✓ 모든 데이터, 상태 및 어플리케이션 로직이 들어있다.
- ✓ 뷰와 컨트롤러에서 모델의 상태를 조작하거나 가져오기 위한 인터페이스를 제공하고 모델에서 자신의 상태 변화에 대해서 옵저버들에게 알려주긴 하지만 기본적으로 모델은 뷰와 컨트롤러에 별 관심이 없다.

### (3) Controller

- ✓ 뷰와 모델 사이에서 위치하며 사용자로부터 입력을 받아서 그것이 모델에게 어떤 의미가 있는지 파악한다.
- ✓ 단순히 모델한테 전달하는 역할만 하는 것이 아니라, 사용자가 입력한 것을 해석해서 그것을 바탕으로 모델을 조작하는 임무를 맡고있다.

# 스프링 프로젝트

## 2. MVC 패턴

### (1) View

- ✓ 모델을 표현하는 방법을 제공하는 사용자 인터페이스.
- ✓ 일반적으로 화면에 표시하기 위해 필요한 상태 및 데이터를 모델에서 직접 가져온다.

### (2) Model

- ✓ 모든 데이터, 상태 및 어플리케이션 로직이 들어있다.
- ✓ 뷰와 컨트롤러에서 모델의 상태를 조작하거나 가져오기 위한 인터페이스를 제공하고 모델에서 자신의 상태 변화에 대해서 옵저버들에게 알려주긴 하지만 기본적으로 모델은 뷰와 컨트롤러에 별 관심이 없다.

### (3) Controller

- ✓ 뷰와 모델 사이에서 위치하며 사용자로부터 입력을 받아서 그것이 모델에게 어떤 의미가 있는지 파악한다.
- ✓ 단순히 모델한테 전달하는 역할만 하는 것이 아니라, 사용자가 입력한 것을 해석해서 그것을 바탕으로 모델을 조작하는 임무를 맡고있다.

# 스프링 프로젝트

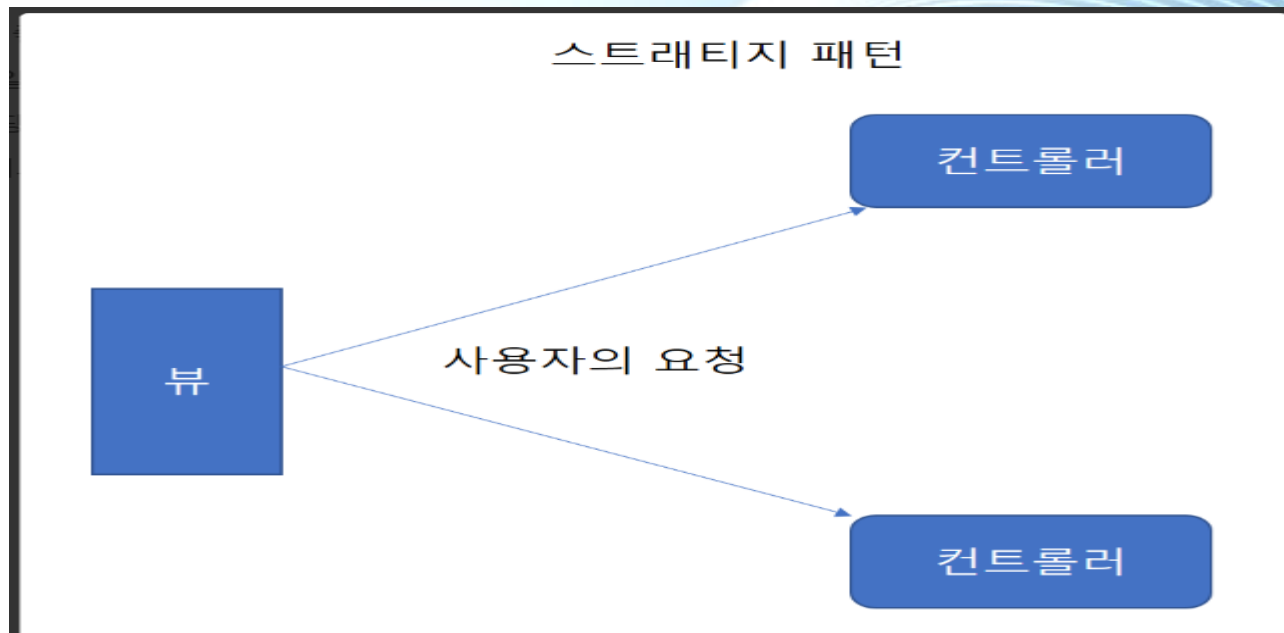
## 2. MVC 패턴

- ✓ 모델에서는 자신의 상태를 알려야 하기 때문에 옵저버 패턴을 사용해서 상태가 바뀔 때마다 뷰와 컨트롤러에게 연락한다.
- ✓ 컨트롤러는 뷰의 행동에 해당하기 때문에 스트래티지 패턴을 이용해서 다른 행동을 원할 시 다른 컨트롤러로 교환하기만 하면 된다.
- ✓ 뷰에서는 내부적으로 컴포지트 패턴을 써서 윈도우, 버튼 같은 다양한 구성요소를 관리한다

# 스프링 프로젝트

## 2. MVC 패턴

- ✓ 뷰와 컨트롤러는 고전적인 스트래티지 패턴으로 구현되어 있다.
- ✓ 때문에 뷰 객체를 여러 전략을 써서 설정할 수 있으며 컨트롤러가 전략을 제공한다.
- ✓ 뷰에서는 애플리케이션의 겉모습에만 신경쓰고, 인터페이스의 행동에 대한 결정은 모두 컨트롤러에게 맡긴다.
- ✓ 스트래티지 패턴을 이용하므로 뷰를 모델로부터 분리시키는 데에도 도움이 된다.
- ✓ 사용자가 요청한 내역을 처리하기 위해서 모델하고 얘기를 해야 하는 부분이 컨트롤러기 때문에 뷰는 그 방법을 전혀 알지 못한다.

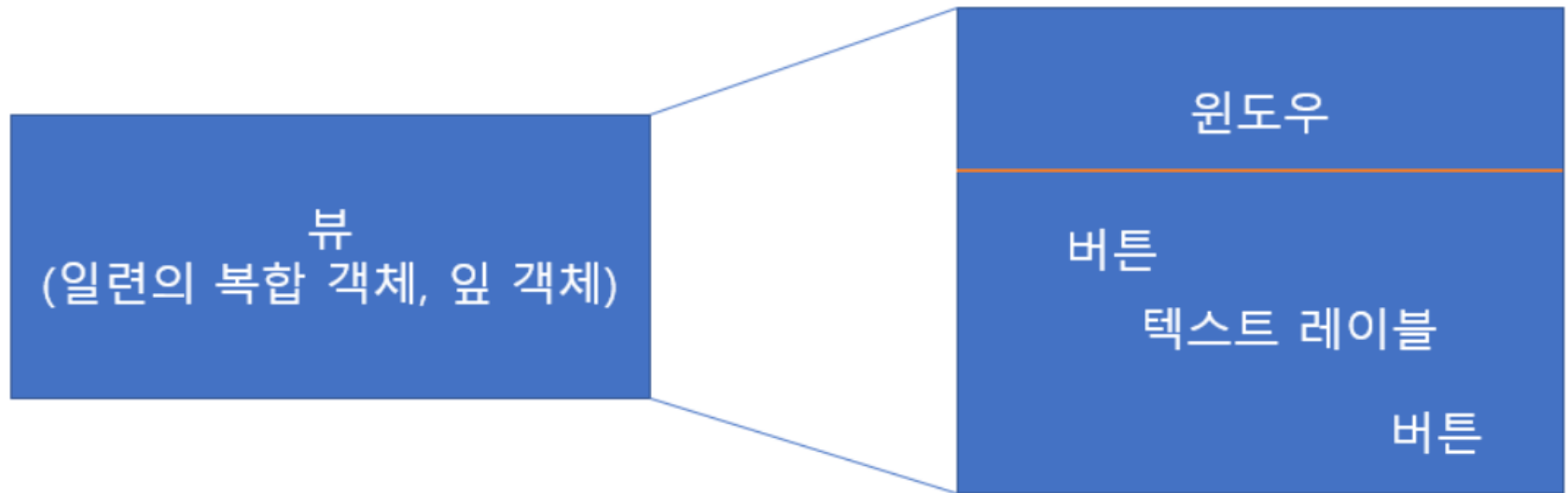


# 스프링 프로젝트

## 2. MVC 패턴

- ✓ 뷰에 속하는 디스플레이는 여러 단계로 겹쳐져 있는 일련의 윈도우, 패널, 버튼, 텍스트 레이블 등으로 구성된다.
- ✓ 각 디스플레이 항목은 복합 객체(윈도우 등) 또는 잎(버튼 등)이 될 수 있다.
- ✓ 컨트롤러에서 뷰한테 화면을 갱신해 달라고 요청하면 최상위 뷰 구성요소한테만 갱신하라고 얘기하면 된다.
- ✓ 나머지는 컴포지트 패턴에 의해 자동으로 처리된다.

### 컴포지트 패턴

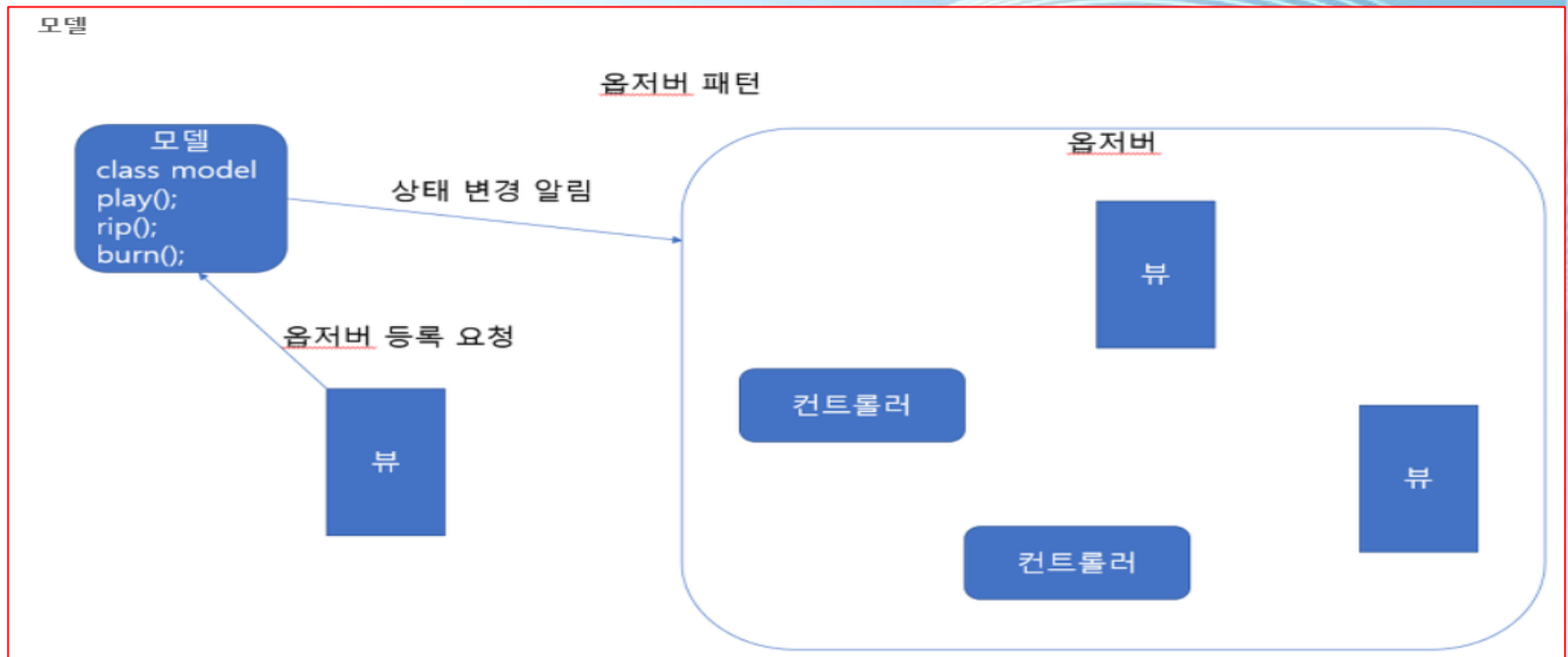




# 스프링 프로젝트

## 2. MVC 패턴

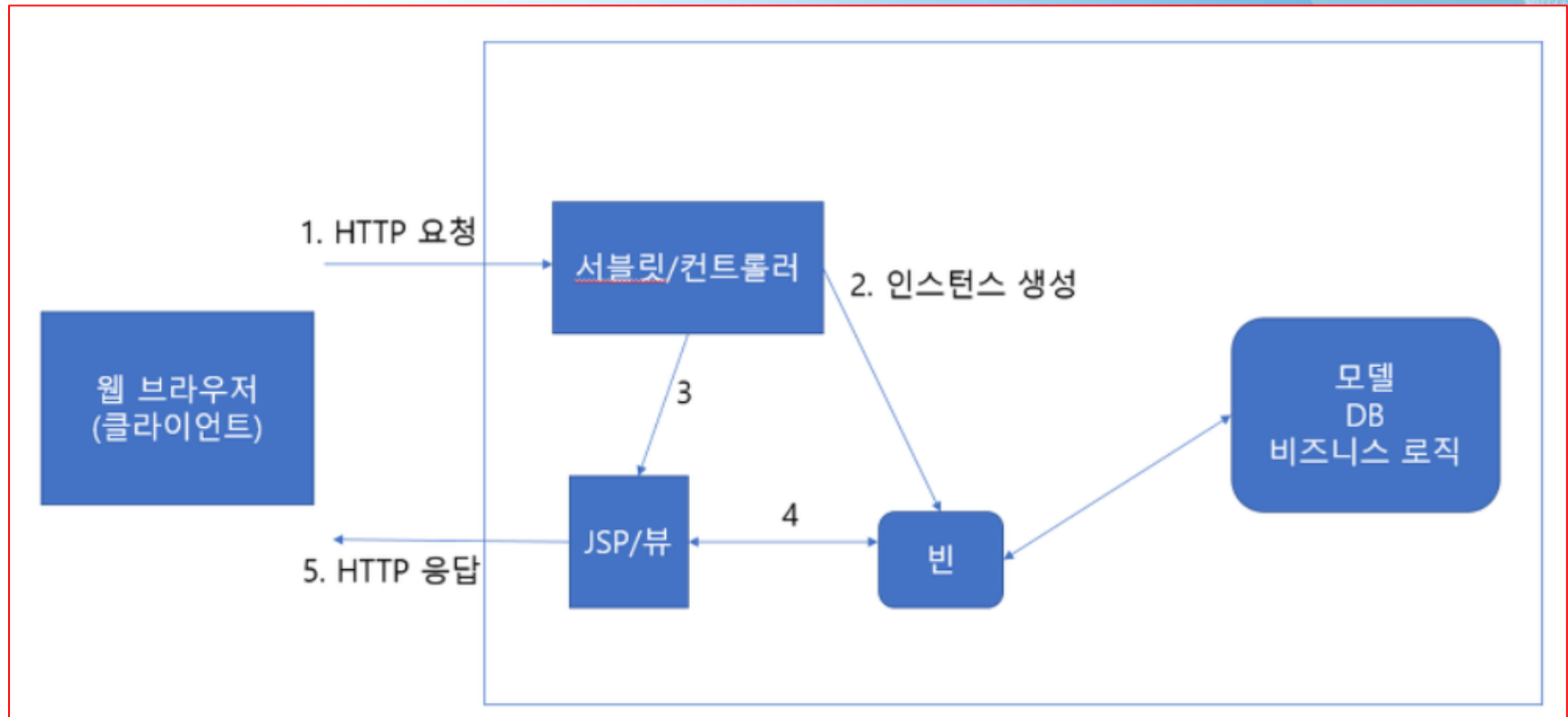
- ✓ 모델에서는 옵저버 패턴을 써서 상태가 변경되었을 때 그 모델하고 연관된 객체들에게 연락을 한다.
- ✓ 옵저버 패턴을 이용해서 뷰 및 컨트롤러로부터 완전히 독립시킬 수 있다.
- ✓ 한 모델에서 서로 다른 뷰를 사용할 수도 있고, 여러 개의 뷰를 동시에 사용하는 것도 가능하다.



# 스프링 프로젝트

## 3. MVC와 웹 환경

- ✓ 웹의 등장으로 여러 개발자들이 MVC를 브라우저/서버 모델에 맞게 변형시켜서 사용하기 시작했다.
- ✓ "모델2(Model 2)" 라는 방법이 가장 많이 쓰이는데, 서블릿과 JSP 기술을 사용하여 일반적인 GUI와 마찬가지로 모델, 뷰, 컨트롤러를 분리해서 디자인할 수 있다.



# 스프링 프로젝트

## 3. MVC와 웹 환경

(1) 사용자가 HTTP 요청을 하면 서버에서 그 요청을 수신한다.

- ✓ 사용자가 웹 브라우저를 이용해 HTTP 요청을 한다.
- ✓ 보통 사용자 ID와 비밀번호와 같은 폼 데이터가 함께 전달된다.
- ✓ 서버에서는 이런 폼 데이터를 받아서 파싱한다.

(2) 서버가 컨트롤러 역할을 한다.

- ✓ 서버는 컨트롤러 역할을 맡아서 사용자 요청을 처리하고, 대부분의 경우에
- ✓ 모델(DB 등)에 어떤 요청을 하게 된다. 요청을 처리한 결과는 일반적으로 자바빈 형태로 포장된다.

(3, 4) 컨트롤러에서는 컨트롤을 뷰한테 넘긴다.

- ✓ 뷰는 JSP에 의해 표현된다. JSP에서는 (4 자바빈을 통해 얻은) 모델의 뷰를 나타내는 페이지만 만들어주면 된다.
- ✓ 그 페이지를 만드는 과정에서 다음 단계의 작업을 위해 몇 가지 더 제어해야 할 일이 있을 수도 있다.

(5) 뷰에서 HTTP를 통해서 브라우저한테 페이지를 전달한다.

- ✓ 페이지가 브라우저한테 전달되면, 그 웹 페이지가 사용자의 화면에 표시된다.
- ✓ 사용자가 또 다른 요청을 할 수도 있으며, 새로운 요청도 지금까지의 과정과 같은 방식으로 처리한다.

# 스프링 프로젝트

## 4. 모델2의 장점

- ✓ 제작 책임까지도 분리시켜 주기 때문에 모델 2의 장점은 단순히 디자인적인 면에서 각 구성요소를 분리해주는 것에 그치지 않는다.
- ✓ 옛날에는 JSP에 접근할 수 있는 사람이라면 JAR에 대해서는 전혀 몰라도 누구든 자바 코드를 마음대로 만들 수 있었다.
- ✓ 하지만 대부분의 웹 제작자들은 콘텐츠와 HTML에 대해서는 잘 알지만 소프트웨어에 대해서는 그리 잘 알기 힘들다는 문제가 있었다.
- ✓ 모델 2가 등장하면서 개발자들은 서블릿에만 전념하면 되고, 웹 제작자들은 간단한 모델2 스타일의 JSP만 다루면 되는 환경이 조성되었다.
- ✓ 그래서 웹 제작자들은 HTML과 간단한 자바빈즈만 만지면 된다.

# 스프링 프로젝트

## 4. 모델2의 장점

