

# Spring Framework 과정

2021. 9. 1 ~ 3 (총3일, 21시간)

정 준 수 Ph.D

# 과정 목표

Spring Framework 구성 모듈을 기반으로 Maven 프로젝트로 구성하고 Spring의 핵심 기능인 의존성 주입(Dependency Injection), 제어의 역전(Inversion of Control) 및 웹(MVC) 애플리케이션과 RESTful 웹 서비스 프레임워크의 대한 이해와 실습을 통하여 프로그램 개발 역량을 습득

1. Spring Framework 개요 및 환경 설정
2. Spring 프로젝트 생성과 Maven 설정
3. DI(Dependency Injections)과 IoC(Inversion of Coverision)
4. 웹(MVC), JDBC
5. 실습예제

# 1. Spring Framework 개요 및 환경 설정

## Spring Framework란?

Java 엔터프라이즈 개발을 편하게 해주는 오픈소스 경량급 어플리케이션 프레임워크

- 자바(JAVA) 플랫폼을 위한 오픈소스(Open Source) 어플리케이션 프레임워크
- 자바 엔터프라이즈 개발을 편하게 해주는 오픈 소스 경량급 애플리케이션 프레임워크
- 자바 개발을 위한 프레임워크로 종속 객체를 생성해주고, 조립해주는 도구
- POJO(Plain Old Java Object) BEAN CONTAINER

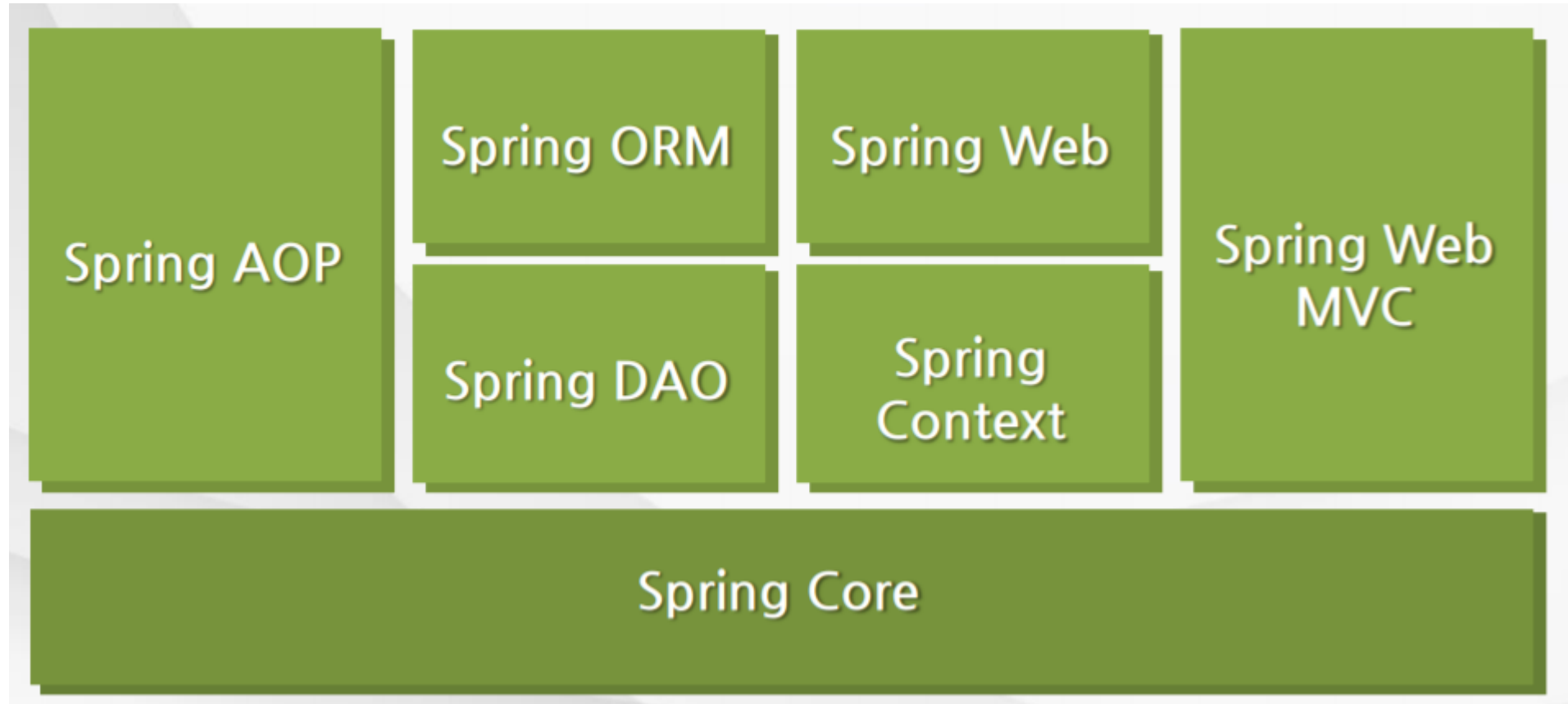
# Spring Framework이 특징

- 경량의 컨테이너로서 자바 객체를 직접 관리
- POJO(Plain Old Java Object) 방식의 프레임워크  
(POJO는 자바 모델이나, 기능, 프레임워크 등에 따르지 않고 홀로 독립적이며 단순한 기능만을 가진 객체를 의미, 자바에서는 이러한 객체를 Bean이라 부른다)
- IoC(Inversion of Control) 지원
- DI(Dependency Injection) 지원
- AOP(Aspect-Oriented Programming) 지원
- iBATIS, myBATIS, Hibernate 등의 데이터베이스 라이브러리 지원

## Spring Framework의 사용 장점

- 스프링 프레임워크를 적용하면 자바 코드 작성 부분을 줄일 수 있는 효과를 얻음
- 반면, 초기 프로젝트 셋팅이 다소 복잡하므로 충분한 연습이 필요함
- 반복되는 작업을 줄일 수 있어 기능 개발에 집중할 수 있음
- 프로젝트 관리가 용이함
- 다수의 개발자와 동시에 프로젝트 개발이 용이함

## Spring 프레임워크를 구성하는 기능 요소



## 프로그램 실습 환경 구축

1. JDK 설치
2. Eclipse 내려받기
3. Tomcat 설치
4. Maven 설치
5. STS(Spring Tools Suite) 설치

## 2. Spring 프로젝트 생성과 Maven 설정

Maven은 자바(Java) 기반 프로젝트의 Life Cycle을 관리를 위한 빌드 도구

- 메이븐은 자바용 프로젝트 관리도구
- 메이븐은 컴파일과 빌드를 동시에 수행
- 테스트를 병행하거나 서버 측 배포(Deploy) 자원을 관리할 수 있는 환경 제공
- Library 관리 기능도 내포하고 있음
- pom.xml 파일 관리를 통해 개발, 유지보수 면에서 프로젝트 관리가 용이



## 메이븐을 사용하는 이유

- 편리한 Dependent Library 관리 – Dependency Management
- 여러 프로젝트에서 프로젝트 정보나 jar 파일들을 공유하기 쉬움
- 모든 프로젝트의 빌드 프로세스를 일관되게 가져갈 수 있음

## 메이븐 프로젝트 생성 입력 내용

- **groupId** - 프로젝트 속하는 그룹 식별 값. 회사, 본부, 또는 단체를 의미하는 값이 오며, 패키지 형식으로 계층을 표현한다. 위에서는 net.madvirus를 groupId로 이용하였다.
- **artifactId** - 프로젝트 결과물의 식별 값. 프로젝트나 모듈을 의미하는 값이 온다. 위에서는 sample을 artifactId로 이용하였다.
- **version** - 결과물의 버전을 입력한다. 위에서는 기본 값인 1.0-SNAPSHOT을 사용하였다.
- **package** - 기본적으로 생성할 패키지를 입력한다. 별도로 입력하지 않을 경우 groupId와 동일한 구조의 패키지를 생성한다.

# Maven 프로젝트의 기본 Directory 구성

## application-core

- **pom.xml**
- **src**
  - **main**
    - **java**
      - **com.package.dir**
    - **resources**
  - **test**
    - **java**
      - **com.package.dir**
    - **resources**

기본적으로 생성되는 폴더를 포함한 Maven 프로젝트의 주요 폴더는 다음과 같다.

- **src/main/java** - 자바 소스 파일이 위치한다.
- **src/main/resources** - 프로퍼티나 XML 등 리소스 파일이 위치한다. 클래스패스에 포함된다.
- **src/main/webapp** - 웹 어플리케이션 관련 파일이 위치한다. (WEB-INF 폴더, JSP 파일 등)
- **src/test/java** - 테스트 자바 소스 파일이 위치한다.
- **src/test/resources** - 테스트 과정에서 사용되는 리소스 파일이 위치한다. 테스트 시에 사용되는 클래스패스에 포함된다.

기본적으로 생성되지 않은 폴더라 하더라도 직접 생성해주면 된다. 예를 들어 **src/main** 폴더에 **resources** 폴더를 생성해주면 Maven은 리소스 폴더로 인식한다.

## Maven과 Library 관리 - 자바 버전 수정을 POM 파일 입력

pom.xml 파일을 열어서 아래 코드를 추가하여 이 설정은 컴파일 자바 버전을 1.8로 설정한다.

...생략

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
        <encoding>UTF-8</encoding>
      </configuration>
    </plugin>
  </plugins>
</build>

</project>
```

## POM 파일 기본 정의

Maven 프로젝트를 생성하면 pom.xml 파일이 프로젝트 루트 폴더에 생성된다. 이 pom.xml 파일은 Project Object Model 정보를 담고 있는 파일로서, 이 파일에서 다루는 주요 설정 정보는 다음과 같다.

- 프로젝트 정보 - 프로젝트의 이름, 개발자 목록, 라이선스 등의 정보를 기술
- 빌드 설정 - 소스, 리소스, 라이프 사이클 별 실행할 플러그인 등 빌드와 관련된 설정을 기술
- 빌드 환경 - 사용자 환경 별로 달라질 수 있는 프로파일 정보를 기술
- POM 연관 정보 - 의존 프로젝트(모듈), 상위 프로젝트, 포함하고 있는 하위 모듈 등을 기술

## POM 파일에서 프로젝트 정보를 기술하는 태그는 다음과 같다.

- <name> - 프로젝트 이름
- <url> - 프로젝트 사이트 URL

POM 연관 정보는 프로젝트간 연관 정보를 기술하는데, 관련 태그는 다음과 같다.

- <groupId> - 프로젝트의 그룹 ID 설정
- <artifactId> - 프로젝트의 Artifact ID 설정
- <version> - 버전 설정
- <packaging> - 패키징 타입 설정. 위 코드의 경우 프로젝트의 결과 Artifact가 jar 파일로 생성됨을 의미한다. jar뿐만 아니라 웹 어플리케이션을 위한 war나 JEE를 위한 ear 등의 패키징 타입이 존재한다.
- <dependencies> - 이 프로젝트에서 의존하는 다른 프로젝트 정보를 기술한다.
  - <dependency> - 의존하는 프로젝트 POM 정보를 기술
    - <groupId> - 의존하는 프로젝트의 그룹 ID
    - <artifactId> - 의존하는 프로젝트의 artifact ID
    - <version> - 의존하는 프로젝트의 버전
    - <scope> - 의존하는 범위를 설정

## POM 파일의 <dependency> 부분 설정

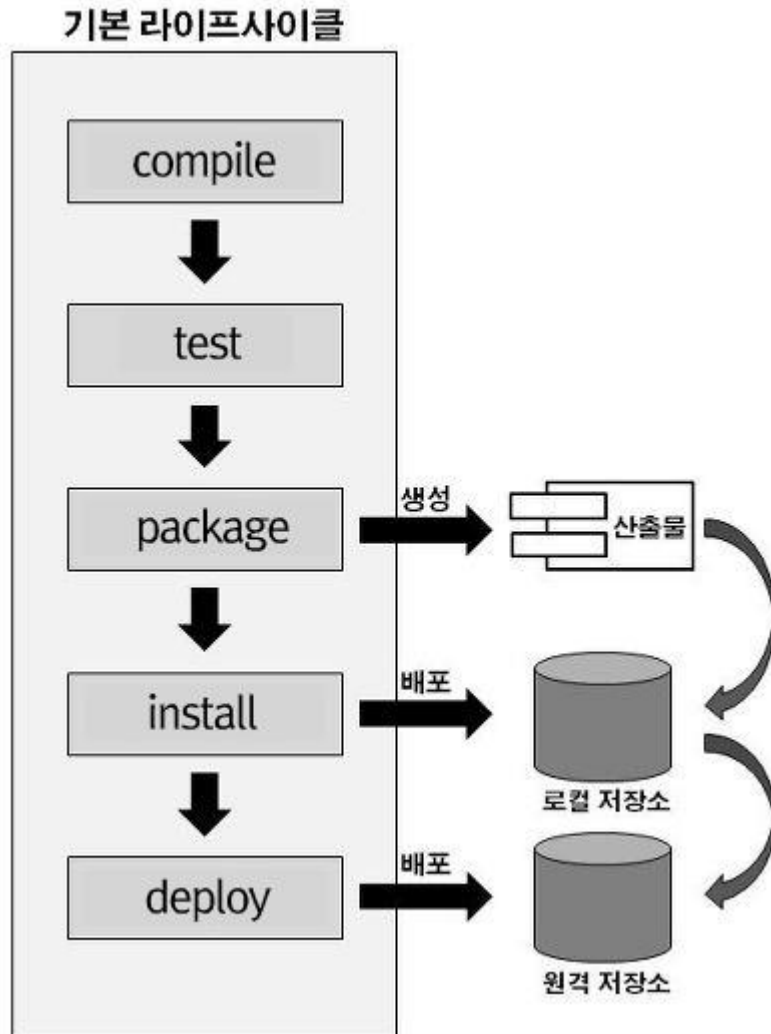
Maven을 사용하지 않을 경우 개발자들은 코드에서 필요로 하는 라이브러리를 각각 다운로드 받아야 한다. 예를 들어 아파치 commons DBCP 라이브러리를 사용하기 위해서는 DBCP뿐만 아니라 common pool 라이브러리도 다운로드 받아야 한다. 물론 commons logging을 비롯한 라이브러리도 모두 추가로 다운로드 받아 설치해 주어야 한다. 즉 코드에서 필요로 하는 라이브러리뿐만 아니라 그 라이브러리가 필요로 하는 또 다른 라이브러리도 직접 찾아서 설치해 주어야 한다.

하지만, Maven을 사용할 경우에는 코드에서 직접적으로 사용하는 모듈에 대한 의존만 추가해주면 된다. 예를 들어 commons-dbcp 모듈을 사용하고 싶은 경우 다음과 같은 <dependency> 코드만 추가해주면 된다.

```
<dependency>  
  <groupId>commons-dbcp</groupId>  
  <artifactId>commons-dbcp</artifactId>  
  <version>1.2.1</version>  
</dependency>
```

그러면, Maven은 commons-dbcp 뿐만 아니라 commons-dbcp가 의존하는 라이브러리를 자동으로 처리해준다.

## 메이븐의 빌드 생명주기(LifeCycle)



Life Cycle을 plugin, phase, goal을 정리하여 이야기하면,  
메이븐은 프로젝트 생성에 필요한 단계 (phase)들을  
Build Life Cycle이라 정의하고 Build Life Cycle은  
default, clean, site 세가지로 표준 정의합니다.

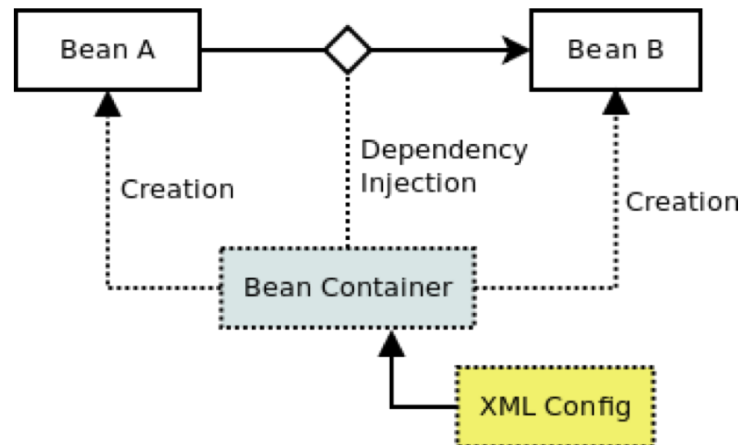


### 3. DI(Dependency Injections)과 IoC(Inversion of Coverision)

#### 의존성 주입 (Dependency Injection)

객체 자체가 아니라 Framework에 의해 객체의 의존성이 주입되는 설계 패턴, 즉 각 클래스간의 의존관계를 Bean 설정 정보를 바탕으로 컨테이너가 자동으로 연결

- Framework에 의해 동적으로 주입되므로 여러 객체 간의 결합이 줄어든다.
- Dependency Injection은 Spring Framework에서 지원하는 IoC의 형태

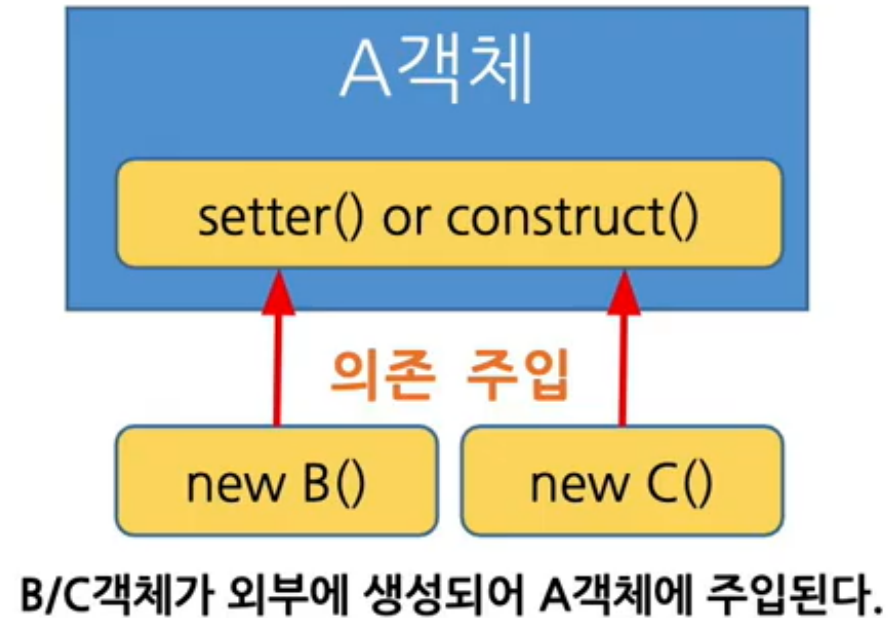


## 프로그램 결합 강도

### 강한 결합



### 약한 결합



## 설정에 명시된대로 Container가

- bean 객체를 생성하고
- 종속성 주입을 수행한다

**Dependency Injection(의존성 주입)과 Inversion Of Control(제어의 역전)은 같은 의미로 사용된다.**

- IoC는 DI를 통해 달성된다.

## **IoC(제어의 역전): 프로그램 제어권을 framework가 가져가는 것**

- 개발자가 모든 제어의 중심이지만 코드 전체에 대한 제어는 framework가 한다.
- 개발자가 설정(xml, annotation 등)만 하면 Container가 알아서 처리한다.
- 즉, 우리는 Framework 속에서 프로그래밍을 하는 것

## 의존성 주입의 세가지 유형

### 1. Contructor Injection – 생성자를 이용한 의존성 주입

- 생성자를 통한 전달
- `<constructor-arg ref="cat"> </constructor-arg>`

### 2. Setter Injection – Setter 메서드를 이용한 의존성 주입

- `setter()`을 통한 전달
- `<property name="myName" value="poodle"> </property>`

### 3. Method Injection – 일반 메서드를 이용한 의존성 주입

- 멤버 변수를 통한 전달

# Spring DI Container

## Spring Framework의 핵심 컴포넌트

- Container는 DI를 사용하여 응용 프로그램을 구성하는 bean 객체를 관리한다.

### 역할

**bean**을 포함하고 관리하는 책임이 있다.

1. 객체(bean)를 생성하고
2. 객체들을 함께 묶고
3. 객체들을 구성하고
4. 객체들의 전체 수명주기(lifecycle)를 관리

# Spring DI Container

## 설정 방법

- Spring Container metadata 설정 방법 (세 가지)

### 1. XML

1. 빈 객체 정의 (Bean Definition)
2. 의존성 주입 (Dependency Injection)

### 2. Java Annotations

### 3. Java Code

# IoC Container(Spring DI Container)의 종류

## BeanFactory

- 클래스를 통해 객체를 생성하고 이를 전달함
- 상속 등 객체 간의 관계를 형성하고 관리함
- Bean에 관련된 설정을 위한 xml 파일은 즉시 로딩하지만 객체는 개발자가 요구할 때 생성함
- XmlBeanFactory

## ApplicationContext

- 클래스를 통해 객체를 생성하고 이를 전달함
- 상속 등 객체 간의 관계를 형성하고 관리함
- 국제화 지원 등 문자열에 관련된 다양한 기능을 제공
- 리스너로 등록되어 있는 Bean에 이벤트를 발생시킬 수 있음
- Bean에 관련된 설정을 위한 xml 파일은 즉시 로딩하면서 객체를 미리 생성해서 가지고 있음
- ClassPathXmlApplicationContext
- FileSystemXmlApplicationContext
- XmlWebApplicationContext
- AnnotationConfigApplicationContext

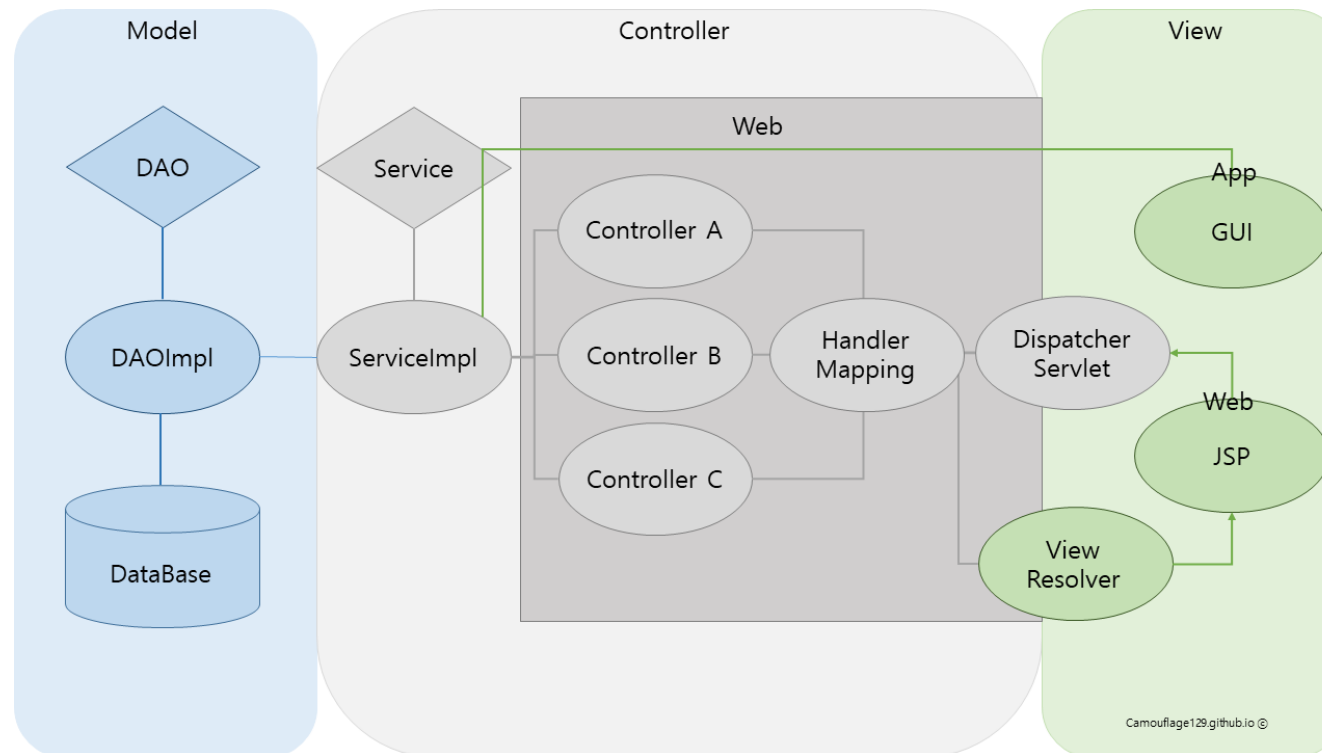
# Spring DI 용어 정리

빈 (Bean)	<ul style="list-style-type: none"><li>▪ 스프링이 IoC 방식으로 관리하는 객체라는 뜻</li><li>▪ 스프링이 직접 생성과 제어를 담당하는 객체를 Bean이라고 부름</li></ul>
빈 팩토리 (BeanFactory)	<ul style="list-style-type: none"><li>▪ 스프링의 IoC를 담당하는 핵심 컨테이너를 가리킴</li><li>▪ Bean을 등록, 생성, 조회, 반환하는 기능을 담당함</li><li>▪ 이 BeanFactory를 바로 사용하지 않고 이를 확장한 ApplicationContext를 주로 이용함</li></ul>
애플리케이션 컨텍스트 (Application Context)	<ul style="list-style-type: none"><li>▪ BeanFactory를 확장한 IoC 컨테이너</li><li>▪ Bean을 등록하고 관리하는 기능은 BeanFactory와 동일하지만 스프링이 제공하는 각종 부가 서비스를 추가로 제공함</li><li>▪ 스프링에서는 ApplicationContext를 BeanFactory 보다 더 많이 사용함</li></ul>
설정 메타정보 (Configuration metadata)	<ul style="list-style-type: none"><li>▪ ApplicationContext 또는 BeanFactory가 IoC를 적용하기 위해 사용하는 메타정보를 말함</li><li>▪ 설정 메타정보는 IoC컨테이너에 의해 관리되는 Bean 객체를 생성하고 구성할 때 사용됨</li></ul>



## 4. 웹(MVC), JDBC

- MVC는 기본적으로 Model View Controller의 약자로 프로그램 개발 시 세가지 역할로 구분하여 개발하는 방법론
- 이렇게 하는 이유는 완벽한 분업화를 통해 해당 역할 개발자가 자신의 역할에만 집중하여 개발 하기 위함



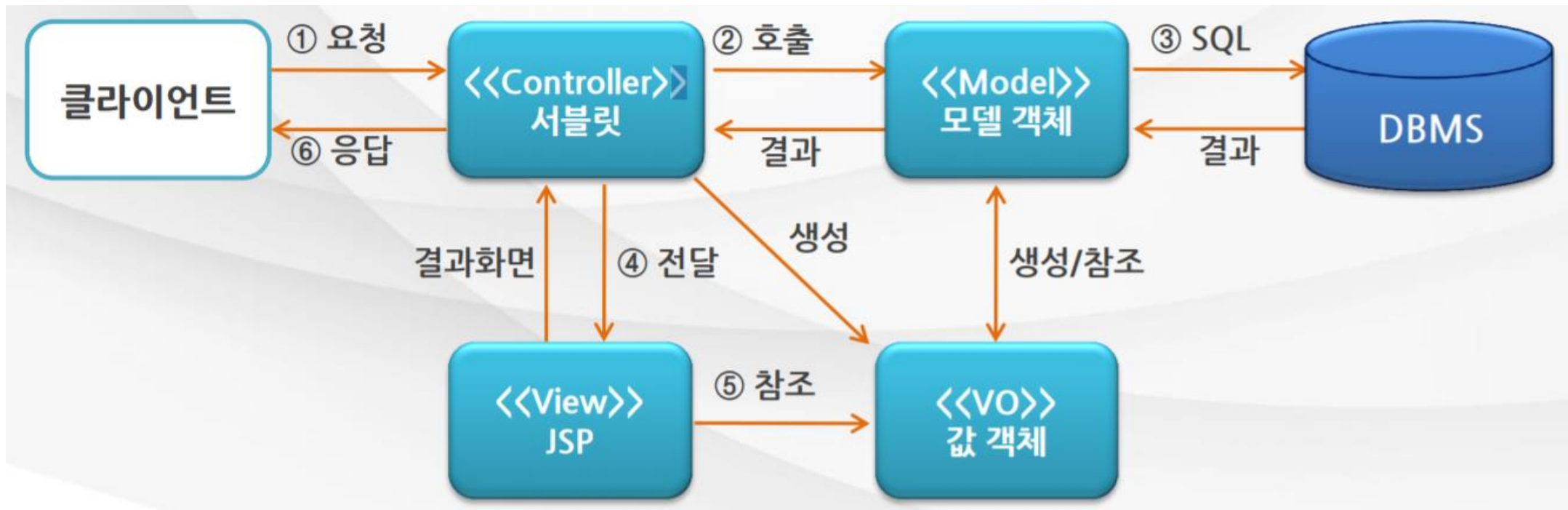
## MVC(Model-View-Controller) 패턴의 개념

MVC 패턴을 사용하면, 사용자 인터페이스로부터 비즈니스 로직을 분리하여 어플리케이션의 시각적 요소나 그 이면에서 실행되는 비즈니스 로직을 서로 영향 없이 쉽게 고칠 수 있는 어플리케이션 개발이 가능

- **Model** : 어플리케이션의 정보(데이터, Business Logic 포함)
- **View** : 사용자에게 제공할 화면(Presentation Logic)
- **Controller** : Model과 View 사이의 상호 작용을 관리

## 모델2 아키텍처 개념

1. 모델 1 아키텍처 : Controller 역할을 JSP가 담당함
2. 모델 2 아키텍처 : Controller 역할을 Servlet이 담당함



## 모델2 아키텍처 호출 순서



- ① 웹 브라우저가 웹 어플리케이션 실행 요청하면, 웹 서버가 그 요청을 받아서 서블릿 컨테이너 (ex: 톰캣서버)에 넘겨준다.  
서블릿 컨테이너는 URL을 확인하고 그 요청을 처리할 서블릿을 찾아서 실행한다.

## 모델2 아키텍처 호출 순서



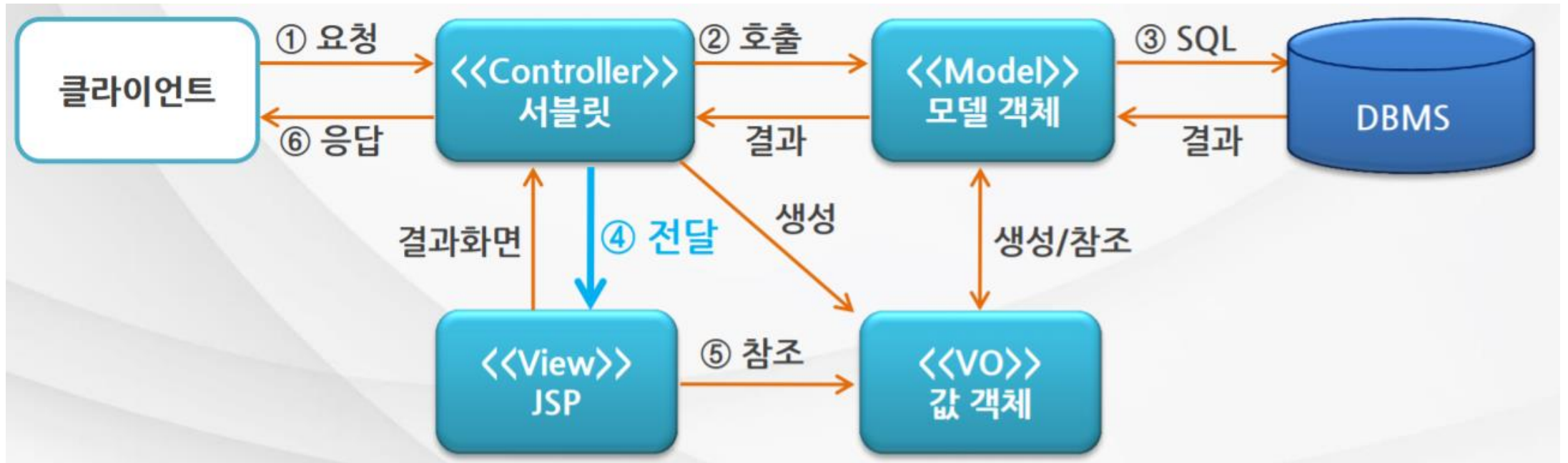
- ② 서블릿은 실제 업무를 처리하는 모델 자바 객체의 메서드를 호출한다.  
만약 웹 브라우저가 보낸 데이터를 저장하거나 변경해야 한다면 그 데이터를 가공하여 VO 객체(Value Object)를 생성하고, 모델 객체의 메서드를 호출할 때 인지 값으로 넘긴다.

## 모델2 아키텍처 호출 순서



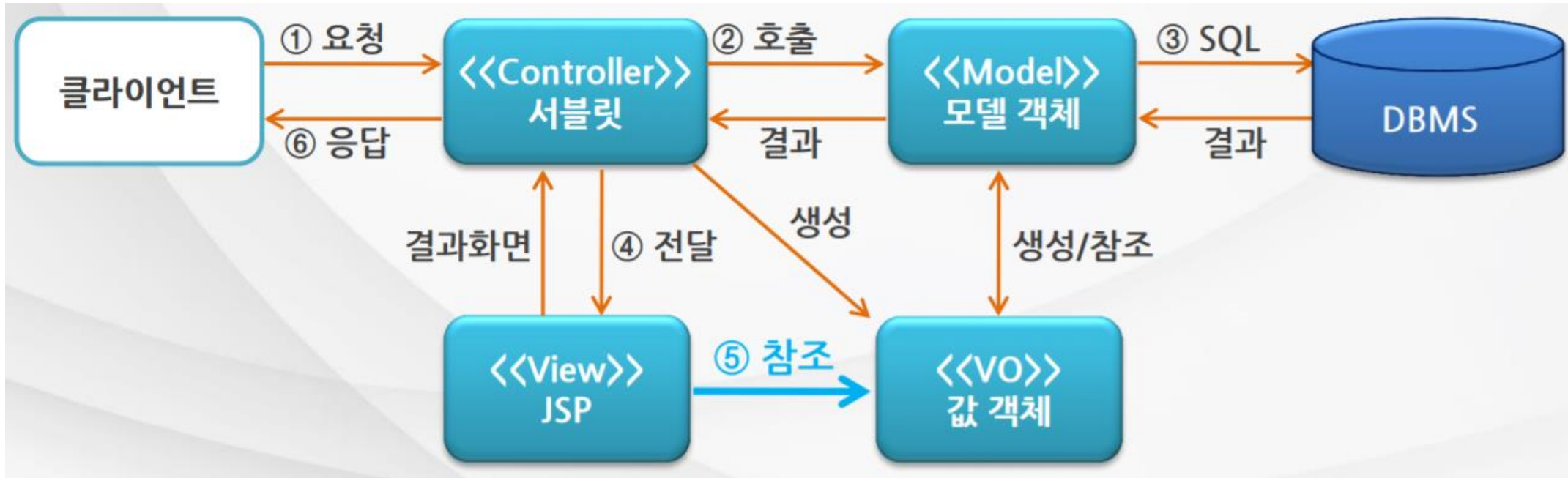
- ③ 모델 객체는 JDBC를 사용하여 매개변수로 넘어온 값 객체를 데이터베이스에 저장하거나 데이터베이스로부터 질의 결과를 가져와서 VO 객체를 만들어 반환한다.

## 모델2 아키텍처 호출 순서



④ 서블릿은 모델 객체로 부터 반환 받은 값을 JSP에 전달한다.

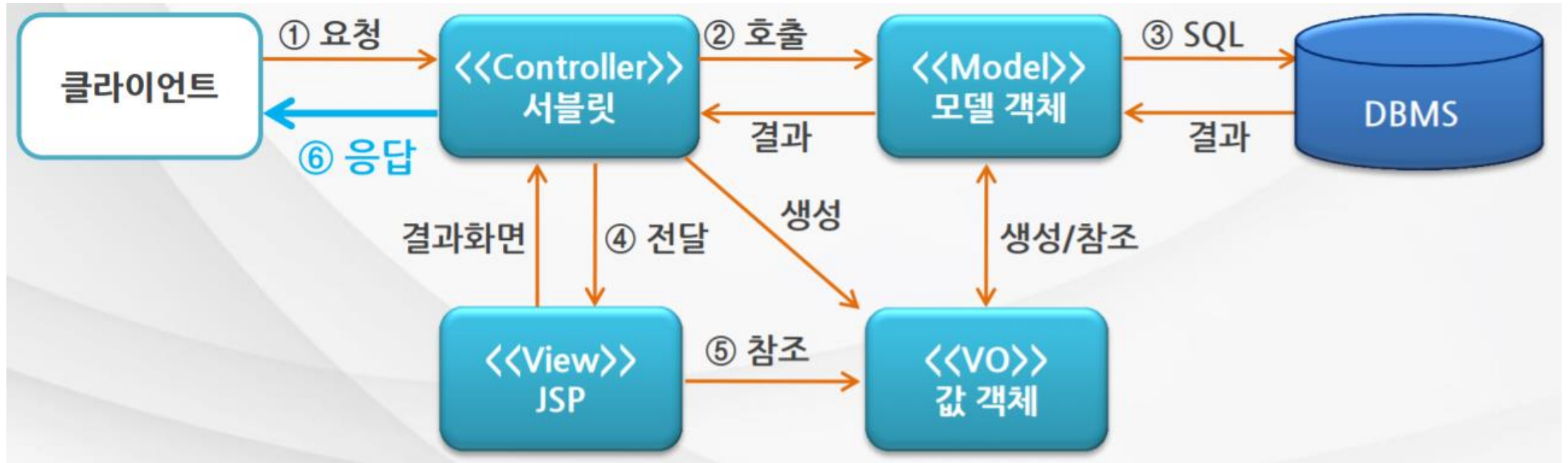
## 모델2 아키텍처 호출 순서



- ⑤ *JSP*는 서블릿으로 부터 전달받은 값 객체를 참조하여 웹 브라우저가 출력할 결과 화면을 만들고, 웹 브라우저에 출력함으로써 요청 처리를 완료한다.



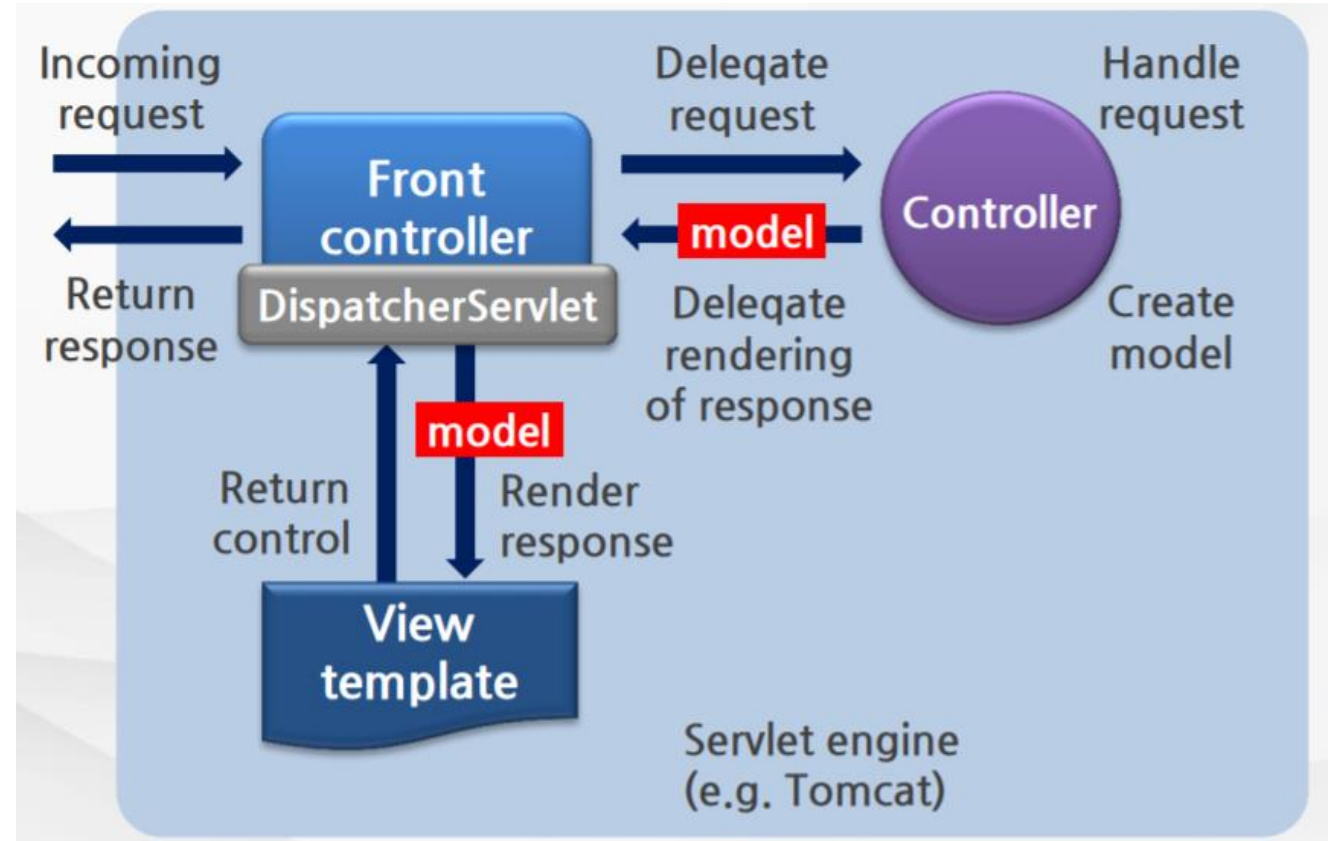
## 모델2 아키텍처 호출 순서



⑥ 웹브라우저는 서버로부터 받은 응답 내용을 화면에 출력한다.

## DispatcherServlet 클래스

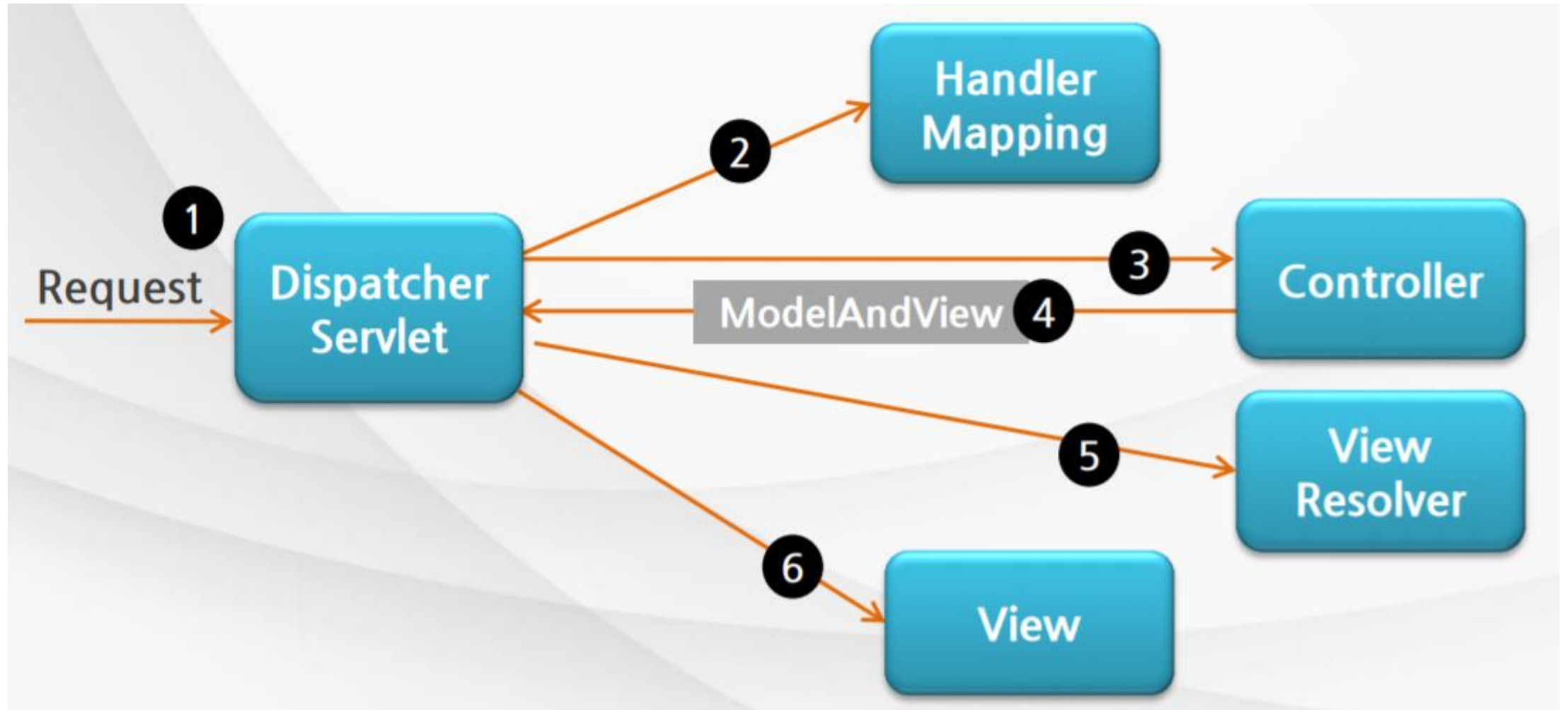
- ① Front Controller 패턴 적용
- ② Web.xml에 설정
- ③ 클라이언트로부터 모든 요청을 전달 받음
- ④ Controller나 View와 같은 Spring MVC의 구성요소를 이용하여 클라이언트에게 서비스를 제공



## Spring MVC의 주요 구성 요소

구성요소	설 명
DispatcherServlet	클라이언트의 요청을 받아서 Controller에게 클라이언트의 요청을 전달하고, 리턴한 결과값을 View에게 전달하여 알맞은 응답을 생성
HandlerMapping	URL과 요청 정보를 기준으로 어떤 핸들러 객체를 사용할지 결정하는 객체이며, DispatcherServlet은 하나 이상의 핸들러 매핑을 가질 수 있음
Controller	클라이언트의 요청을 처리한 뒤, Model를 호출하고 그 결과를 DispatcherServlet에게 알려 줌
ModelAndView	Controller가 처리한 데이터 및 화면에 대한 정보를 보유한 객체
View	Controller의 처리 결과 화면에 대한 정보를 보유한 객체
ViewResolver	Controller가 리턴한 뷰 이름을 기반으로 Controller 처리 결과를 생성할 뷰를 결정

## Spring MVC의 주요 구성 요소의 처리 과정



## Spring MVC의 주요 구성 요소의 처리 과정

- ① 클라이언트의 요청을 받은 DispatcherServlet를 web.xml 에 설정
- ② 클라이언트의 요청을 처리할 Controller를 작성
- ③ Spring Bean으로 Controller 를 등록
- ④ JSP를 이용한 View 영역의 코드를 작성
- ⑤ Browser 상에서 JSP를 실행

.

## Spring MVC기반 웹 어플리케이션 작성 절차

- ① 클라이언트의 요청이 DispatcherServlet에게 전달한다.
- ② DispatcherServlet은 HandlerMapping을 사용하여 클라이언트의 요청을 처리할 Controller를 획득한다.
- ③ DispatcherServlet은 Controller 객체를 이용하여 클라이언트의 요청을 처리한다.
- ④ Controller는 클라이언트 요청 처리 결과와 View 페이지 정보를 담은 ModelAndView 객체를 반환한다.
- ⑤ DispatcherServlet은 ViewResolver로부터 응답 결과를 생성할 View 객체를 구한다.
- ⑥ View는 클라이언트에게 전송할 응답을 생성한다.

.

## 5. 실습예제

### 전자정부 표준프레임워크 3.10 기반 개발 시작하기(Getting Started)

1. 개발환경 설치 : 실습을 위한 개발환경을 구축한다.
2. 프로젝트 생성 : 제공한 샘플 프로젝트를 이용하여 HelloWorld 응용 어플리케이션을 생성하고 실행해 본다.
3. 자세히 들여다 보기 : 생성/실행한 프로젝트의 내부 소스코드를 학습하여 전자정부 표준 프레임워크 기반의 응용 어플리케이션 구현의 원리를 이해한다.

<https://www.egovframe.go.kr/wiki/doku.php?id=egovframework:dev3.10:gettingstarted>

## Step 1. 개발환경 설치

전자정부 표준프레임워크에서 제공하는 구현도구(implementation tool) 및 HelloWorld 응용 어플리케이션을 위한 종속라이브러리를 이용하여 실습에 필요한 개발환경을 설치한다.


### 개발환경설치

먼저 eclipse 기반의 전자정부표준 프레임워크의 [구현도구\(implementation tool\)](#) 설치를 참조하여 설치한다.


### 플러그인 업데이트

설치한 구현도구의 플러그인이 최신 모듈을 사용할 수 있도록 [구현도구\(implementation tool\)](#) 플러그인 업데이트를 참조하여 업데이트를 수행한다.

### Maven 환경설정

시작하기 개발환경은 실제 프로젝트 수행환경과 달리 Nexus를 이용하지 않고, 종속 라이브러리를 Maven 로컬 파일저장소에 수동으로 복사하여 개발환경을 구축한다. Maven의 로컬 파일 저장소를 설정하기 위하여 제공한  maven repository 3.10 파일을 임의의 디렉토리에 압축 해제하여 설치한다.


#### 종속라이브러리 설치순서

- Maven 설정파일 및 종속라이브러리를 포함한  maven repository 3.10 를 다운로드 한다.
- 다운로드 받은 파일은 임의의 디렉토리에서 압축해제한다.(압축해제한 디렉토리는 **[MavenRepository 설치디렉토리]**로 명명한다.)
- 텍스트 에디터를 이용하여 **[MavenRepository 설치디렉토리]/settings.xml** 파일의 localRepository 항목의 값을 다음과 같이 수정한다.




## Step 2. 프로젝트 생성(Core) 및 실행

### 프로젝트 Import

프로젝트 생성 및 실행을 위하여 본 가이드는  HelloWorld 프로젝트 파일을 제공한다. 아래의 순서에 따라 프로젝트를 생성한다.

#### 프로젝트 생성순서

-  HelloWorld 프로젝트 파일을 다운로드 받아서 임의의 디렉토리에 저장한다. (다운로드 받은 파일의 압축을 해제할 필요는 없다.)
- 구현도구에서 File>Import.. 메뉴를 선택한다.

### HelloWorld 테스트 실행

HelloWorld 프로젝트는 JUnit Test Framework 기반의 Test Case를 포함하고 있다. Test Case는 HelloWorldServiceImpl 클래스의 sayHello 메서드의 실행결과에 대한 결과값을 확인하도록 구현되어 있다. 제공한 구현도구에서 아래의 순서로 테스트를 수행한다.

#### HelloWorld 테스트 실행순서

- 프로젝트의 src/test/java에서 HelloWorld 서비스의 junit test case(HelloWorldServiceTest.java)를 마우스 오른쪽 버튼으로 클릭하고 Run As>JUnit test 을 클릭한다.

## 강사 소개

정 준 수 / Ph.D ( heinem@naver.com )

- 前) 삼성전자 연구원
- 前) 삼성의료원 (삼성생명과학연구소)
- 前) 삼성SDS (정보기술연구소)
- 現) (사)한국인공지능협회, AI, 머신러닝 강의
- 現) 한국소프트웨어산업협회, AI, 머신러닝 강의
- 現) 서울디지털재단, AI 자문위원
- 現) 한성대학교 교수(겸)
- 전문분야: Splunk, 시각 모델링, 머신러닝(ML), RPA
- <https://github.com/JSJeong-me/>

