01_Spring Framework

Framework란?

- 애플리케이션 개발 시 필요한 **기본 구조와 뼈대를 제공하는 틀**
- 제공되는 구조에 맞게 코딩을 진행하면 되기 때문에
  개발 시간이 줄고, 효율적인 개발을 할 수 있음
- 코드의 재사용성을 증가 시키기 위해 일련의 클래스 묶음이나 뼈대, 틀을 라이브러리 형태로 제공함

Libarary란?

- 자주 사용 되는 **기능을 모아둔 코드의 집합**
- 개발자 또는 회사에서 개발한 기능 (코드)를 라이브러리로 만들어서 배포
  → 해당 기능이 필요한 개발자가 라이브러리를 자신의 코드에 추가해서 사용

API(Application Programming Interface)

- 서로 다른 소프트웨어 구성 요소 간의 상호 작용을 정의하는 규약
- API를 통해 서로 **다른 프로그램들이 데이터를 주고받거나 기능을 호출하여 통신할 수 있음**
-

Spring Framework 란?

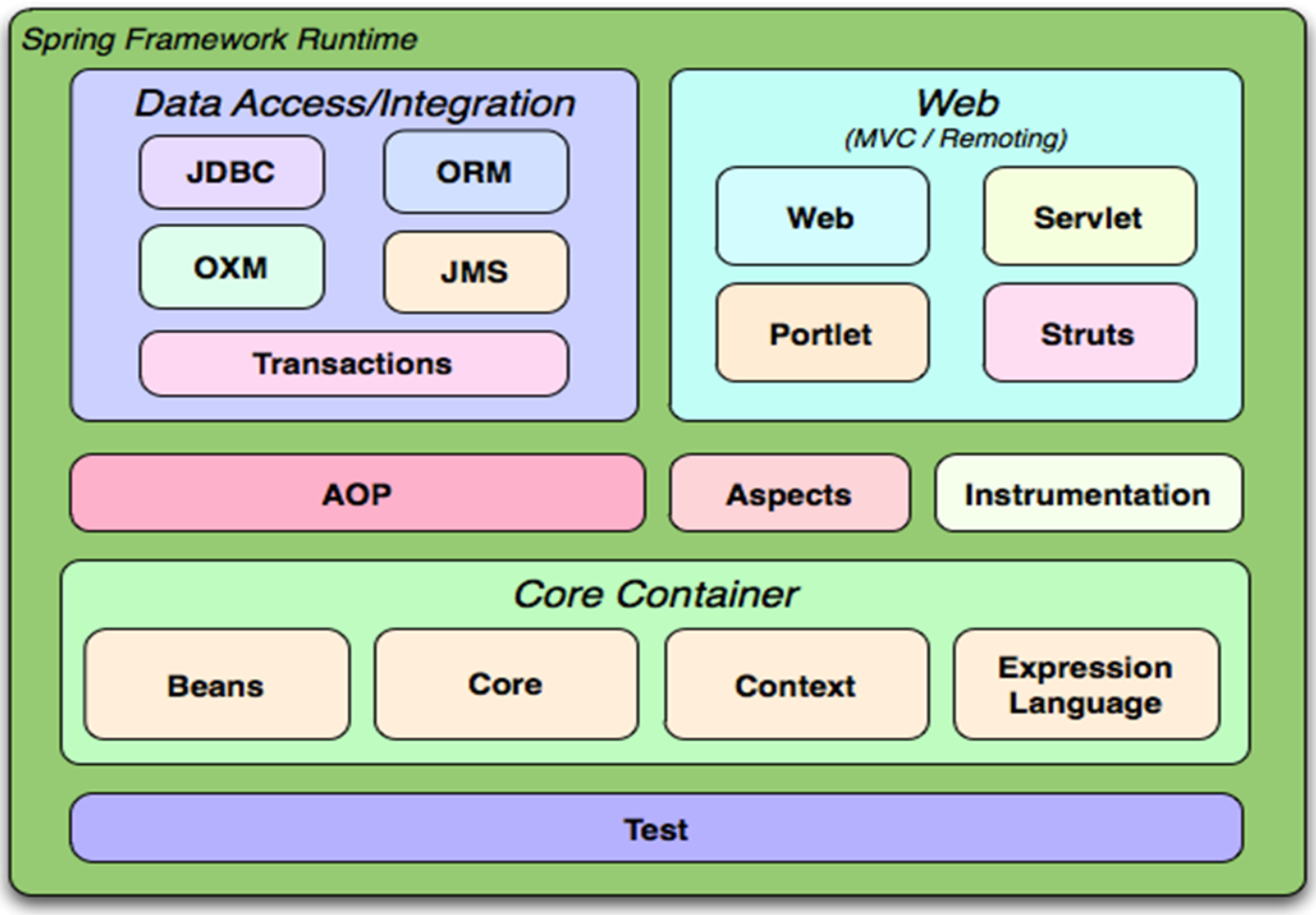
- **자바 플랫폼을 위한 오픈 소스 애플리케이션 프레임워크**로 간단하게 **스프링(Spring)**이라고도 불림
- **동적인 웹 사이트를 개발**하기 위한 여러 가지 서비스를 제공함
- 대한민국 공공기관의 웹 서비스 개발 시 사용을 권장하고 있는
- **전자정부 표준 프레임워크(Spring MVC Project 기반 프레임워크)**의 기반 기술로서 사용

(Spring 공식 페이지 : https://spring.io/)

Spring Framework의 주요 특징

IOC **(Inversion of Control)** **제어 반** 전	컨트롤의 제어권이 개발자가 아니라 프레임워크에 있다는 뜻으로 **객체의 생성부터 모든 생명주기의 관리까지 프레임워크가 주도함.** 개발자가 객체를 생성하고, 직접 호출하지 않고, Spring Container가 만들어둔 객체를 필요할 때 얻어와서 사용. 클래스간의 결합도를 낮추고 유지보수성을 향상시킬 수 있음.
DI **(Dependency Injection)** **의존성** 주입	설정 파일(XML)이나 어노테이션을 통해 객체 간의 의존 관계를 설정하여 **개발자가 직접 객체를 생성하지 않고** **Spring Container가 만들어둔 객체를 필요한 위치에서 주입함.** (직접 객체를 만들지 않고 Spring이 만든 객체를 얻어와 사용 == 의존)
POJO **(Plain Old Java Object** **)**	Spring은 J2EE, EJB와 같은 특정 기술이나 라이브러리의 내용을 상속 받아 클래스를 구현하지 않고, **일반적인 자바 객체(POJO)를 사용할 수 있도록 지원함.** 이를 통해 개발자는 프레임워크 학습 곡선을 낮추고 코드의 가독성을 높일 수 있음. * J2EE(Java2 Enterprise Edition) : Servlet, JSP 레벨의 서버 프로그래밍 인터페이스 * EJB(Enterprise Java Bean) : 쉽게 웹 개발이 가능한 기술, 객체지향 장점을 포기해야 하는 문제점 발생
AOP **(Aspect Oriented Programming)** **관점 지향 프로그** 래밍	로깅, 보안, 트랜잭션 관리 등 공통적인 관심사를 분리하여 코드 중복을 줄이고 유지보수성을 향상 시키는 기능을 지원함.
Spring MVC	MVC(Model, View, Controller) 디자인 패턴을 적용할 수 있는 어노테이션을 지원하고, IOC / DI를 이용해 의존 관계를 관리하여 개발자가 아닌 서버가 객체들을 관리하는 웹 애플리케이션을 구축 할 수 있음.
PSA **(Portable Service Abstraction)**	스프링은 다른 여러 모듈을 사용함에 있어 별도의 추상화 레이어를 제공하여 특정 기술에 종속되지 않으면서 다양한 기술 스택을 쉽게 사용할 수 있음 (외부 라이브러리, API 등 외부 기술을 쉽게 이용할 수 있도록 여러 인터페이스, 추상 클래스를 제공)

Spring Container 구성 모듈(기본 구성)



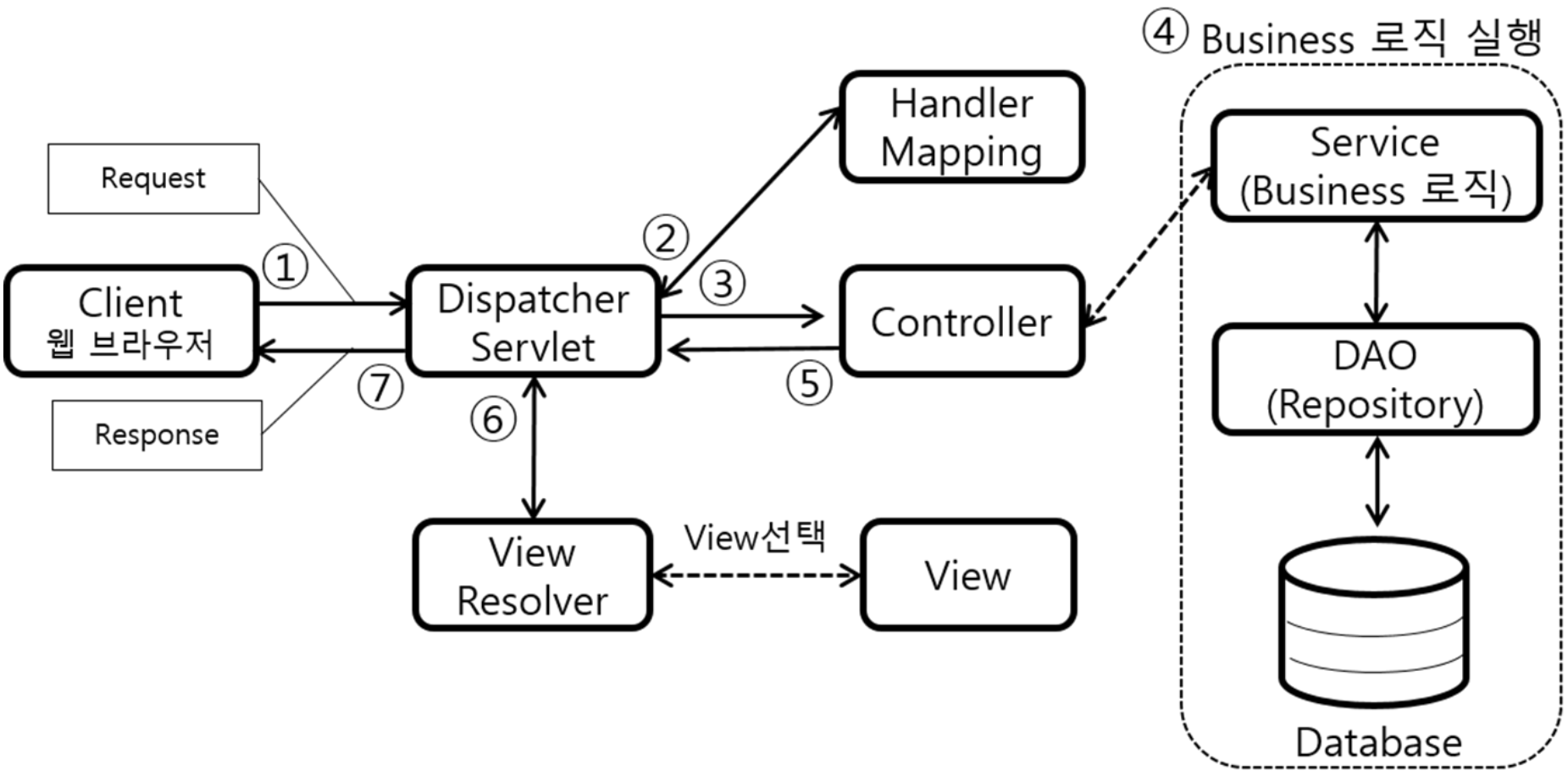
Data 접근 계층	JDBC나 데이터베이스에 연결하는 모듈로, Data 트랜잭션에 해당하는 기능을 담당하여 영속성 프레임워크의 연결을 담당.
Web 계층 (MVC / Remoting)	Spring Framework에서 Servlet등의 웹 구현 기술과의 연결점을 Spring MVC 구성으로 지원하기 위해 제공되는 모듈 계층. 스프링의 리모팅 기술로 RMI, Hessian, Burlap, JAX-WS, HTTP 호출자 그리고 REST API 모듈을 제공.
AOP 계층	Spring에서 각 흐름 간 공통된 코드를 한 쪽으로 빼내어 필요한 시점에 해당 코드를 첨부하게 하기 위해 지원하는 계층으로, 별도의 proxy를 두어 동작. 이를 통해 객체간의 결합도를 낮출 수 있음.
Core Container	Spring의 핵심 부분이라고 할 수 있으며 모든 스프링 관련 모듈은 이 Core Container 기반으로 구축됨. Spring의 근간이 되는 IOC 기능을 지원하는 영역을 담당. BeanFactory를 기반으로 Bean 클래스들을 제어할 수 있는 기능을 지원.
spring-beans	스프링 컨테이너를 이용해서 객체를 생성하는 기본기능을 제공
spring-context	객체생성, 라이프 사이클 처리, 스키마 확장 등의 기능을 제공
spring-aop	AOP 기능을 제공
spring-web	REST 클라이언트 데이터 변환 처리, 서블릿 필터, 파일 업로드 지원 등 웹 개발에 필요한 기반 기능을 제공
spring-webmvc	스프링 기반의 MVC 프레임워크, 웹 애플리케이션을 개발하는데 필요한 컨트롤러, 뷰 구현을 제공
spring-websocket	스프링 MVC에서 웹 소켓 연동을 처리할 수 있도록 제공

spring-oxm	XML과 자바 객체간의 매핑을 처리하기 위한 API 제공
spring-tx	트랜잭션 처리를 위한 추상 레이어를 제공
spring-jdbc	JDBC 프로그래밍을 보다 쉽게 할 수 있는 템플릿 제공
spring-orm	Hibernate, JPA, Mybatis 등과의 연동을 지원
spring-jms	JMS 서버와 메시지를 쉽게 주고 받을 수 있도록 하기위한 템플릿
spring-context-support	스케쥴링, 메일발송, 캐시연동, 벨로시티 등 부가 기능을 제공

Spring MVC

- Spring Framework 에서는 클라이언트의 화면을 표현하기 위한 View와 서비스를 수행하기 위한 개발 로직 부분을 나누는 **MVC2 패턴**을 지원함.

Spring MVC 요청 처리 과정



DispatcherServlet	클라이언트의 요청(Request)을 전달 받고, 요청에 맞는 컨트롤러가 리턴 한 결과 값을 View에 전달하여 알맞은 응답(Response)을 생성
HandlerMapping	클라이언트의 요청 URL을 어떤 컨트롤러가 처리할지 결정
Controller	클라이언트의 요청을 처리한 뒤, 결과를 DispatcherServlet에게 리턴
ModelAndView	컨트롤러가 처리한 결과 정보 및 뷰 선택에 필요한 정보를 담음

ViewResolver	컨트롤러의 처리 결과를 생성할 View를 결정
View	컨트롤러의 처리 결과 화면을 생성, JSP나 Velocity 템플릿 파일 등을 View로 사용