**Spring framework의 이해**

**I. Spring 개념**

**(1) spring 이란?**

Spring Framework는 자바 플랫폼을 위한 오픈소스 [애플리케이션 프레임워크](http://ko.wikipedia.org/wiki/%EC%95%A0%ED%94%8C%EB%A6%AC%EC%BC%80%EC%9D%B4%EC%85%98_%ED%94%84%EB%A0%88%EC%9E%84%EC%9B%8C%ED%81%AC)로서 간단히 Spring이라고도 불린다. 동적인 웹 사이트를 개발하기 위한 여러 가지 서비스를 제공하고 있다. 대한민국 [공공기관](http://ko.wikipedia.org/wiki/%EA%B3%B5%EA%B3%B5%EA%B8%B0%EA%B4%80)의 웹 서비스 개발 시 사용을 권장하고 있는 [전자정부 표준프레임워크](http://ko.wikipedia.org/wiki/%EC%A0%84%EC%9E%90%EC%A0%95%EB%B6%80_%ED%91%9C%EC%A4%80%ED%94%84%EB%A0%88%EC%9E%84%EC%9B%8C%ED%81%AC)의 기반 기술로서 쓰이고 있다.

**(2) 역사**

로드 존슨이 2002년에 출판한 자신의 저서인 [Expert One-on-One J2EE Design and Developement](http://www.wrox.com/WileyCDA/WroxTitle/productCd-0764543857.html)에 선보인 코드를 기반으로 시작하여 점점 발전하게 되었다..

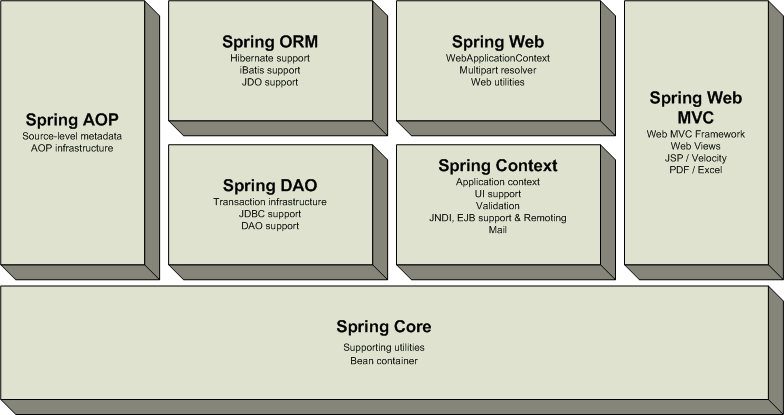
* 1.0 : 2004년 3월
* 2.0 : 2006년 10월
* 2.5 : 2007년 11월
* 3.0 : 2009년 12월
* 3.1 : 2011년 12월
* 4.0 : 2013년 12월

**(3) spring framework의 특징**

* 경량 컨테이너로서 자바 객체를 직접 관리한다. 각각의 객체 생성, 소멸과 같은 라이프 사이클을 관리하며 스프링으로부터 필요한 객체를 얻어올 수 있다.
* 스프링은 [POJO](http://ko.wikipedia.org/wiki/POJO) (Plain Old Java Object: 평범한 자바객체) 방식의 프레임워크이다. 일반적인 [J2EE](http://ko.wikipedia.org/wiki/J2EE) 프레임워크에 비해 구현을 위해 특정한 [인터페이스](http://ko.wikipedia.org/wiki/%EC%9D%B8%ED%84%B0%ED%8E%98%EC%9D%B4%EC%8A%A4)를 구현하거나 상속을 받을 필요가 없어 기존에 존재하는 [라이브러리](http://ko.wikipedia.org/wiki/%EB%9D%BC%EC%9D%B4%EB%B8%8C%EB%9F%AC%EB%A6%AC) 등을 지원하기에 용이하고 객체가 가볍다.
* 스프링은 [제어 반전](http://ko.wikipedia.org/wiki/%EC%A0%9C%EC%96%B4_%EB%B0%98%EC%A0%84)(IoC : Inversion of Control)을 지원한다. 컨트롤의 제어권이 사용자가 아니라 프레임워크에 있어서 필요에 따라 스프링에서 사용자의 코드를 호출한다.
* 스프링은 [의존성 주입](http://ko.wikipedia.org/wiki/%EC%9D%98%EC%A1%B4%EC%84%B1_%EC%A3%BC%EC%9E%85)(DI : Dependency Injection)을 지원한다. 각각의 계층이나 서비스들 간에 의존성이 존재할 경우 프레임워크가 서로 연결시켜준다.
* 스프링은 [관점 지향 프로그래밍](http://ko.wikipedia.org/wiki/%EA%B4%80%EC%A0%90_%EC%A7%80%ED%96%A5_%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D)(AOP : Aspect-Oriented Programming)을 지원한다. 따라서 [트랜잭션](http://ko.wikipedia.org/wiki/%ED%8A%B8%EB%9E%9C%EC%9E%AD%EC%85%98)이나 로깅, 보안과 같이 여러 모듈에서 공통적으로 사용하는 기능의 경우 해당 기능을 분리하여 관리할 수 있다.
* 스프링은 영속성과 관련된 다양한 서비스를 지원한다. MyBatis, [iBatis](http://ko.wikipedia.org/wiki/IBatis)나 [Hibernate](http://ko.wikipedia.org/w/index.php?title=Hibernate&action=edit&redlink=1) 등 이미 완성도가 높은 데이터베이스 처리 라이브러리와 연결할 수 있는 인터페이스를 제공한다.
* 스프링은 확장성이 높다. 스프링 프레임워크에 통합하기 위해 간단하게 기존 라이브러리를 감싸는 정도로 스프링에서 사용이 가능하기 때문에 수많은 라이브러리가 이미 스프링에서 지원되고 있고 스프링에서 사용되는 라이브러리를 별도로 분리하기도 용이하다



**(4) spring framework의 모듈**



∎ Spring Core : spring framework의 핵심기능인 의존성 주입기능을 지원하는 모듈이다.

∎ Spring Context : Spring Core에서 지원하는 기능 외의 추가적인 기능을 지원하고, JNDI연결이나 ejb연계를

위한 Adapter를 제공하는 모듈이다.

∎ Spring Web : Web Application개발에 적합한 Web Application Context를 제공하며, Mulitpart 요청을 처리

할 수 있다. Struts와 같은 다른 프레임워크와의 연동을 지원한다.

∎ Spring DAO : jdbc를 기반으로 DAO클래스 개발할 때 연결취득, statement 생성, ResultSet 처리, 연결 끊기

등의 반복적인 작업을 줄일 수 있는 템플릿 기반의 클래스를 제공한다.

∎ Spring ORM : ORM(Object-Relation Mapping)툴과의 연동을 지원하는 모듈이다.

∎ Spring AOP : 관점지향 프로그래밍을 지원하는 모듈이다.

∎ Spring MVC : Model-View-Controller 기반의 웹 어플리케이션 개발을 지원하는 모듈이다.

**(5) Spring의 3대 핵심 기술**



POJO(Plain Old Java Objects):오래된 방식의 객체지향적인 자바 객체

DI(Dependency Injection): 의존성 주입

AOP(Aspect Oriented Programming):관점 지향 프로그래밍

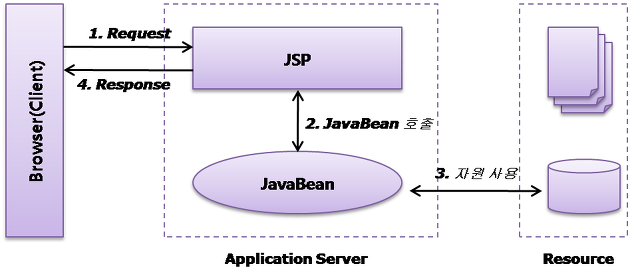
PSA(Portable Service Abstractions): 추상화 서비스

**II.  spring framework의 Web Application 설계방식**

Spring에서 중요한 자바 개념 : 기본 생성자 / interface / Bean

|  |  |
| --- | --- |
| Model | iBatis, MyBatis : SQL Map (Sql문장을 맵핑)   Hibernate : ORM ( Object Relation Mapping; Java & DB Mapping) |
| View | **JSTL 라이브러리**  Velocity   Freemarker   Tiles   sitemesh |
| Control | struts1 /struts2  **spring**  sprite  play |

**1. Model 1**

* JSP 만 이용하여 개발하는 경우
* JSP + Java Bean을 이용하여 개발하는 경우
* Model2의 Controller 개념이 모호
* 
* 장점
  + 개발속도가 빠름
  + 개발자의 기술적인 숙련도가 낮아도 배우기 쉬워 빠르게 적용 가능
* 단점
  + JSP 페이지에서 프레젠테이션 로직과 비즈니스 로직이 혼재되어 복잡
  + 로직의 혼재로 인해 개발자와 디자이너의 작업 분리가 어려움
  + JSP 코드의 복잡도로 인해 유지보수가 어려워짐
* 웹 애플리케이션이 복잡해지고 사용자 요구가 증가함에 따라 새로운 개발방식을 요구

**2. Model II**

* GUI 개발모델인 MVC를 웹 애플리케이션에 적용하여 구현한 방식
* Application의 역할을 Model – View – Controller로 분리
* MVC는 Model-View-Controller 로 나눠져서 모델은 데이터의 타입을 정의하고, View는 인터넷 상에 보이는 부분, Controller는 이 두가지에서 데이터의 흐름과 View를 조절하는 역할.
* Model:

Business Logic을 담당 – **Java Bean**으로 구현

* Business Service(Manager) – Business Logic의 workflow를 관리
* DAO (Data Access Object) – Database와 연동하는 Business Logic을 처리.

**※ Model** 부분은 DB 처리를 하는 DAO 클래스와 그 결과 데이터 (DTO) VO 클래스가 담당한다

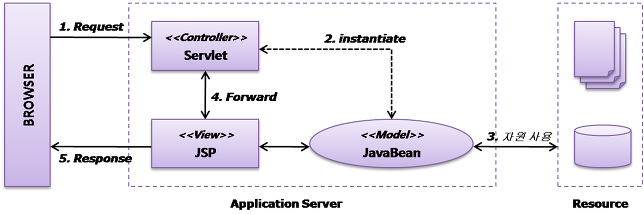
* View:

Client에게 응답을 처리 – **JSP**로 구현

* Controller:

클라이언트의 요청을 받아 Model과 View사이에서 일의 흐름을 조정한다. - **Servlet**으로 구현

* Client의 요청을 받아 Client가 보낸 Data를 읽고 검사한다.
* Model에게 Business Logic을 요청한다.
* Model의 처리 결과에 맞는 View에게 응답을 요청한다.



* 장점
  + 비즈니스 로직과 프리젠테이션의 분리로 인해 어플리케이션이 명료해지며 유지보수와 확장이 용이함
  + 디자이너와 개발자의 작업을 분리해 줌
* 단점
  + 개발 초기에 아키텍처 디자인을 위한 시간의 소요로 개발 기간이 늘어남
  + MVC 구조에 대한 개발자들의 이해가 필요함