장 희 정

Spring Framework

Spring Framework 시작하기에 앞서…

Framework 탄생 전

- SW 재 사용성을 높일 수 있는 방안들

  : 복사(Copy) & 붙이기(Paste)

  : 자주 사용되는 유사한 기능들을 모아 메소드로 정의 하여 재사용.

  (메소드호출) – 복사,붙이기 보다는 진보된 방식이지만 작업 영역간의 결합도(Coupling)문제는 여전히 존재.

  : 클래스의 재사용(상속)

  : AOP(Aspect Oriented Programming)

디자인 패턴이란?

프로그램 개발에서 자주 나타나는 과제를 해결하기 위한 방법 중 하나로 소프트웨어 개발과정에서 발견한 Know-How를 축적하여 이름을 붙여 이후에 재사용하기 좋은 형태로 특정 규약을 묶어서 정리한 것이다.

* 이 용어를 소프트웨어 개발영역에서 구체적으로 처음 제시한 곳은

GoF(Gang of Four)라 불리는 네 명의 컴퓨터 과학 연구자들이 쓴 서적

'Design Patterns : Elements of Reusable Object Oriented Software' (재사용 가능한 객체지향 소프트웨어의 요소 – 디자인패턴) 이다.

디자인패턴을 사용해야 하는 이유

- 요구사항은 수시로 변경되기 때문에 요구사항 변경에 대한 소스코드 변경을 최소화 필요.

- 여러 사람이 같이 하는 팀 프로젝트진행에 있어 범용적인 코딩 스타일 필요.

- 상황에 따라 인수 인계하는 경우도 발생하는데 직관적인 코드를 사용이 필요.

프레임워크의 정의

-비 기능적(Non-Funtional) 요구사항(성능, 보안, 확장성, 안전성등) 을 만족하는 구조와 구현된 기능을 안정적으로 실행하도록 제어해주는 잘 만들어진 구조의 라이브러리의 덩어리.

- 프레임 워크는 애플리케이션들의 최소한의 공통점을 찾아 하부구조를 제공함으로써 개발자들로 하여금 시스템의 하부구조를 구현하는데 들어가는 노력을 감소시켜 주다. – Business Logic에 좀더 집중할 수 있다.

프레임워크를 사용해야 하는 이유

- 비 기능적인 요소들을 초기개발 단계마다 구현 해야 하는 번거로움을 해결.

- 기능적인(Functional)요구사항에 좀더 집중 할 수 있도록 해준다.

- 디자인 패턴과 마찬가지로 반복적으로 발견되는 문제를 해결하기 위한 특화된 Solution을 제공한다.

디자인패턴 + 프레임워크의 조합

- 디자인 패턴은 프레임워크의 핵심적인 특징이고 프레임워크를 사용하는 애플리케이션에 그 패턴이 적용된다는 특징을 가지고 있다.

- 디자인 패턴은 어플리케이션을 설계할 때 필요한 구조적인 가이드라인이 되어 줄 수 있지만 구체적은 구현된 기반코드를 제공하지는 않는다.

- 프레임워크는 디자인 패턴과 함께 제공해서 프레임워크를 사용하는 구조적인 틀과 구현코드의 패턴이 적용된 기반 클래스라이브러리를 함께 제공한다.

개발자는 프레임워크의 기반코드를 확장하여 사용면서 자연스럽게 그 프레임워크에서 사용된 패턴을 적용 할 수 있게 된다.

라이브러리 란?

-프레임워크는 특정 부분의 기술적인 구현을 라이브러리 형태로 제공한다.

-Class Library라는 구성요소는 프레임워크의 정의 중 하나인 Semi Complete(반제품) 이다.

특징	프레임워크	라이브러리
유저코드의 작성	프레임워크 클래스를 서브 클래싱 해서 작성	독립적으로 작성
호출흐름	프레임워크코드가 유저코드를 호출	유저코드가 라이브러리를 호출
실행흐름	프레임워크가 제어	유저코드가 제어
객체의 연동	구조프레임워크가 정의	독자적으로 정의

라이브러리 와 프레임워크의 차이점

-프레임워크와 라이브러리는 구분하는 방법은 실행제어가 어디서 일어나는 가에 달려 있다.

-라이브러리는 개발자가 만든 클래스에서 직접 호출하여 사용하므로 실행의 흐름에 대한 제어를 개발자가의 코드가 관장하고 있다.

-프레임워크는 반대로 프레임워크에서 개발자가 만든 클래스를 호출하여 실행의 흐름에 대한 제어를 담당한다.

디자인 패턴

디자인 패턴 + 라이브러리 = 프레임워크

-프레임워크는 디자인 패턴과 그것이 적용된 기반 라이브러리의 결합으로 이해할 수 있다.

-프레임워크의 라이브러리를 살펴볼 때도 적용된 패턴을 주목해서 살펴 본다면 그 구성을 이해하기 쉽다.

-특히 프레임워크를 확장하거나 커스텀마이징할때는 프레임워크에 적용된 패턴에 대한 이해가 꼭 필요하다.

Spring Framework 시작하기

Spring 준비하기

스프링공식사이트 : https://spring.io

- JDK준비
- IDE 준비(STS – SpringSource Tool Suite)

: STS는 Sprig 개발업체인 SpringSource가 직접 만들어 제공하는 이클립스 확장판으로 최신 이클립스를 기반으로 주요한 Spring지원 플러그인과 관련된 도구를 모아서 Spring 개발에 최적화 되도록 만들어진 IDE이다.

- Tomcat 준비
- Oracle 준비

Maven과 Library관리

http://maven.apache.org : 라이브러리 관리 + 빌드 툴

https://mvnrepository.com/ : 라이브러리 Dependency 정보

- Maven을 사용하는 이유

: 편리한 Dependent Library관리 – Dependency Management

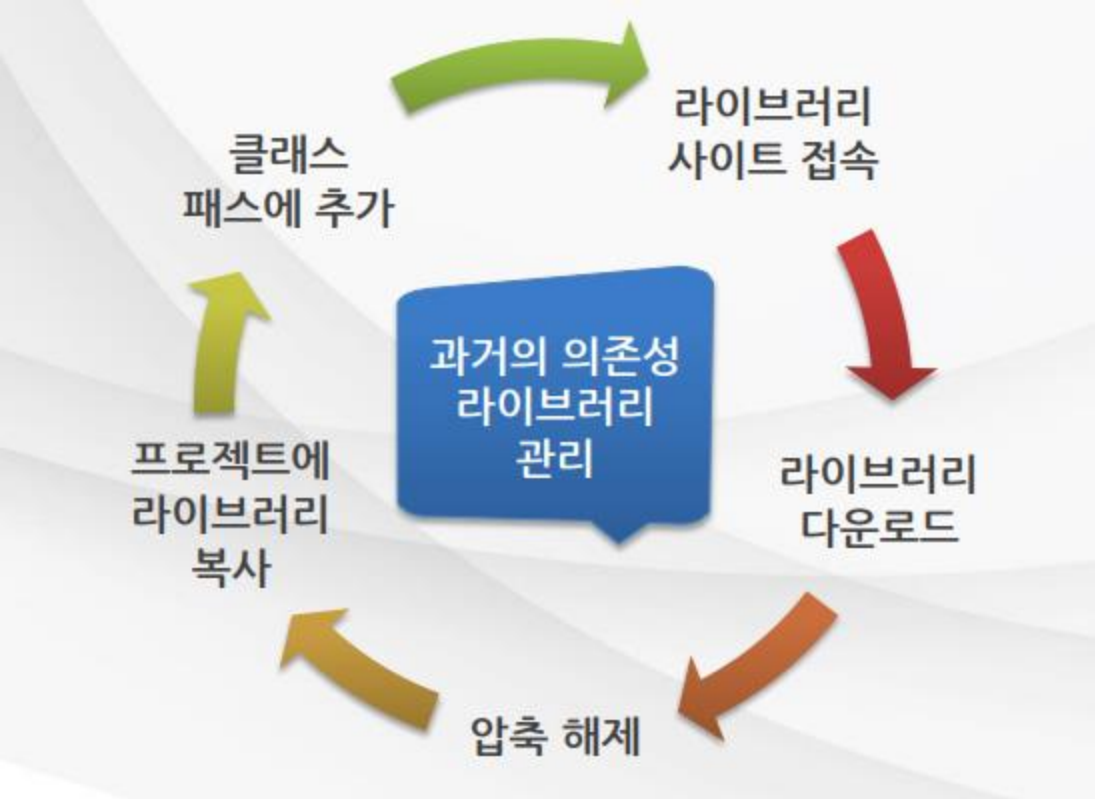
: 여러 프로젝트에서 프로젝트 정보나 jar파일들을 공유하기 쉬움.

: 모든 프로젝트 빌드 프로세스를 일관되게 가져 갈수 있음.

* Maven과 Gradle은 비슷함.

Maven과 Library관리

Maven이전의 Library관리 방법



Maven과 Library관리

Maven의 Library관리 방법

Maven과 Library관리

pom.xml

: Maven프로젝트를 생성하면 pom.xml파일이 생성됨

: pom.xml파일은 Project Object Model 정보를 담고 있음.

: pom.xml문서에 의존관계(dependency) 추가

Spring Framework이란?

Rod Johnson이 만든 오픈 소스 프레임워크로 Java 엔터프라이즈 개발을 편하게 해주는 오픈소스 경량급 애플리케이션 프레임워크이다.

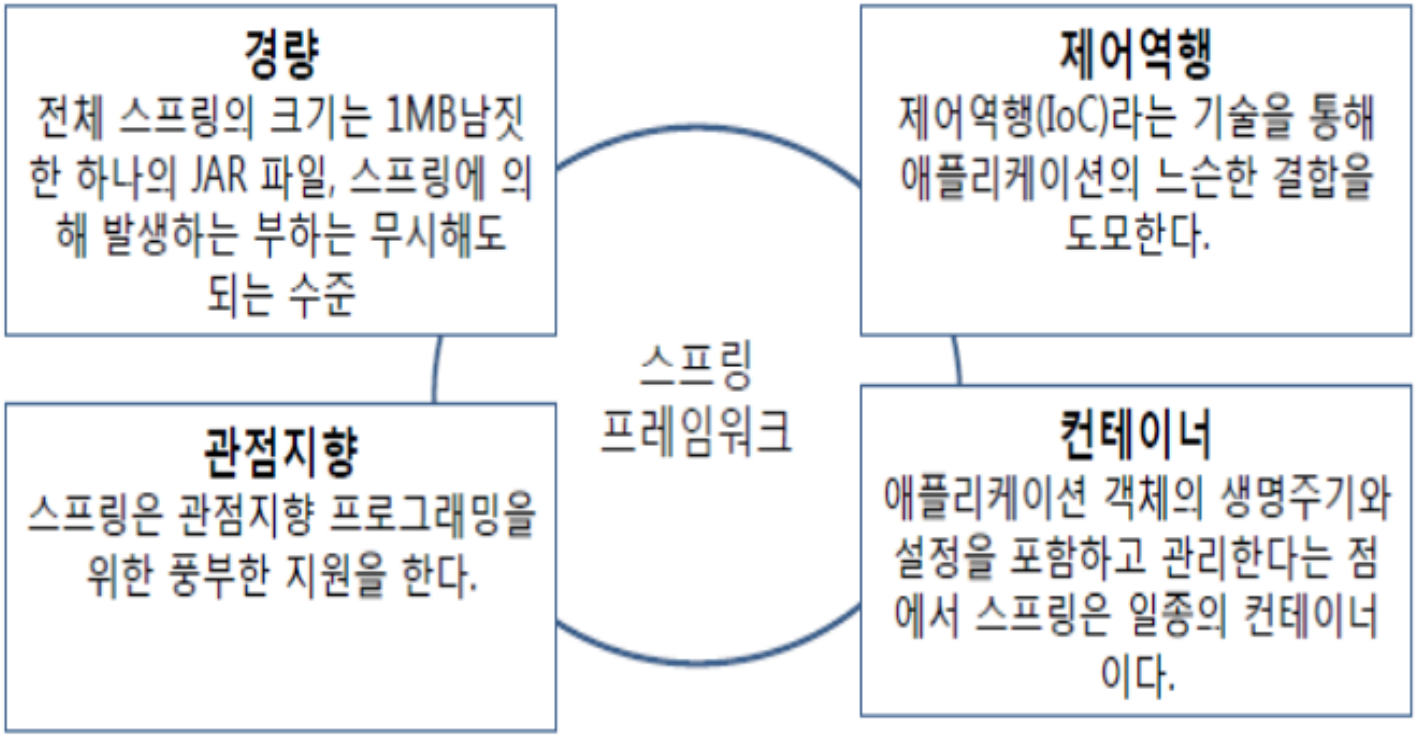
: 복잡한 엔터프라이즈 애플리케이션개발을 겨냥해서 탄생함.

: 2002년 책을 통해 소개

Ejb기반으로 개발을 하지 않고 POJO기반으로 개발을 하더라도 가볍고 제어가 가능한 상호 관련이 적은 AOP를 지원하고 컨테이너를 통해 라이프사이클을 관리하고 xml기반으로 컴포넌트를 개발 할 수 있도록 지원해주는 프레임웍이다.

Spring Framework의 특징

스프링은 경량의 제어 역행과 관점지향 컨테이너 프레임워크이다.



Spring Framework전략

엔터프라이즈 개발의 복잡함을 상대하는 Spring의 전략

: Portable Service Abstraction, DI, AOP, POJO

Spring Framework전략

Portable Service Abstraction 란?

: 트랜잭션 추상화, 데이터액세스 Exception변환 기능등 기술적인 복잡함은 추상화를 통해 Low Level의 기술구현 부분과 기술을 사용하는 인터페이스로 분리한다.

Spring Framework전략

DI(Dependency Injection) 란?

: Spring은 객체지향에 충실한 설계가 가능하도록 단순한 객체형태로 개발 할 수 있고, DI는 유연하게 확장 가능한 객체를 만들어 두고 그 관계는 외부에서 다이나믹하게 설정해준다.

Spring Framework전략

AOP(Aspect Oriented Programming)란?

: AOP는 애플리케이션 로직을 담당하는 코드에 남아 있는 기술 관련 코드를 분리해서 별도의 모듈로 관리하게 해주는 강력한 기술이다.

Spring Framework전략

POJO(Plain Old Java Object)란?

: POJO는 객체지향 원리에 충실하면서, 특정 환경이나 규약에 종속되지 않고 필요에 따라 재 활용 될 수 있는 방식으로 설계된 객체이다.

Spring Framework 알아야 할 필수 개념!

-Spring Container의 개념

: Spring 컨테이너는 Java객체의 LifeCycle을 관리하며, Spring 컨테이너로 부터 필요한 객체를 가져와서 사용 한다.

-Spring IoC & DI 개념

: Spring은 설정파일이나 어노테이션을 통해서 객체간의 의존관계를 설정 할 수 있다.

-Spring AOP 개념

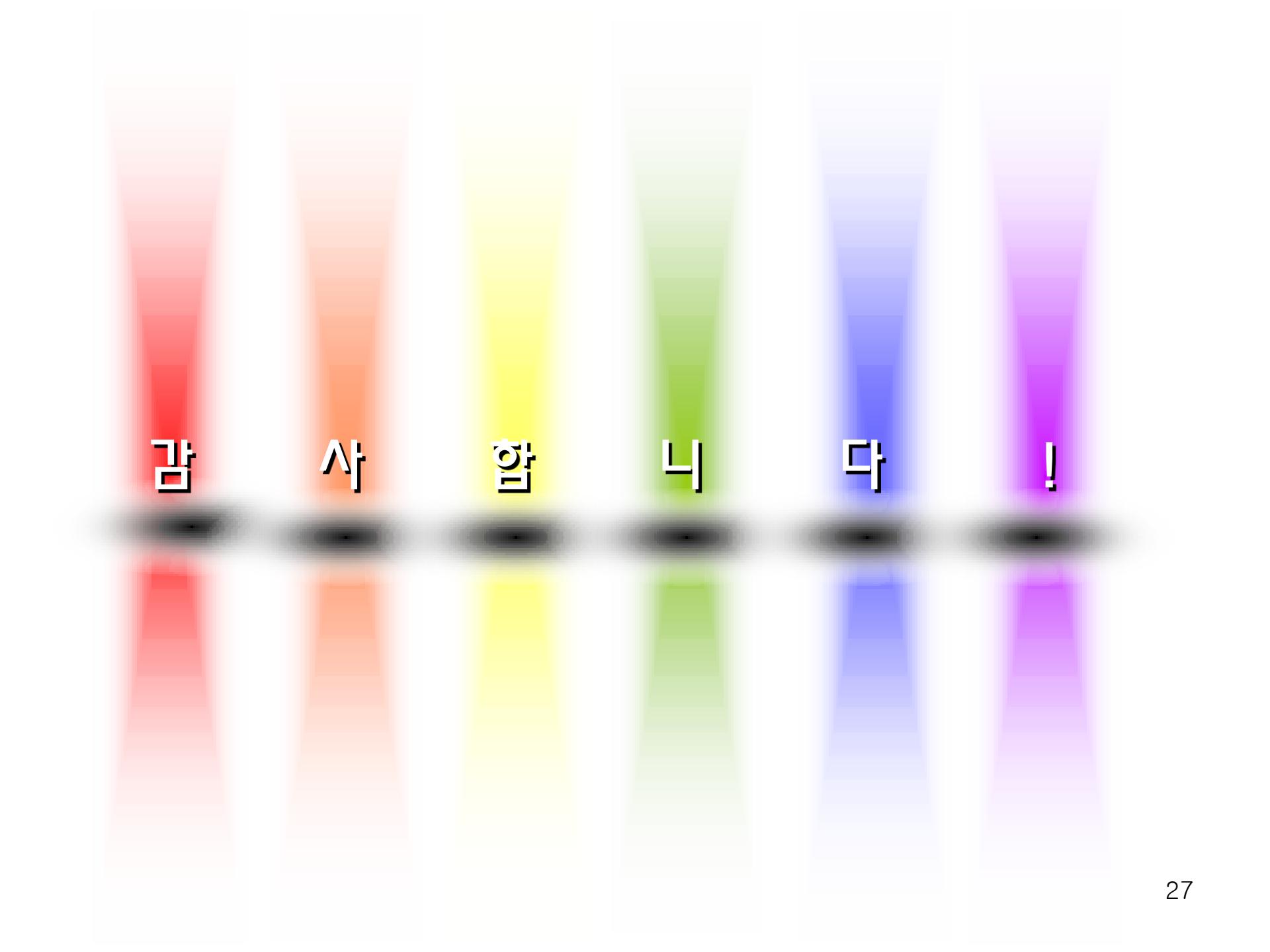
: Spring은 트랜잭션이나 로깅, 보안과 같이 공통적으로 필요로 하는 모듈들을 실제 핵심 모듈에서 분리해서 적용한다.

Spring Framework를 구성하는 기능 요소

Spring은 필요한 기능을 모듈로 제공하고 있기 때문에 필요한 모듈만 가져다 사용하면 된다. (기본 모듈은 Spring-context)

필요한 모듈은 pom.xml문서에 등록하여 의존관계에 해당하는

모든 라이브러리를 한번에 관리할 수 있다.


감 사 합 니 다 !