Spring 개요 및 특징

Bok, Jong Soon
javaexpert@nate.com
https://github.com/swacademy/Spring5

Software 재 사용 방안들

- Copy & Paste
- Method(or Function)
- Class(include Inheritance)
- AOP(Aspect Oriented Programming)
- Design Pattern
- Framework

Software 재 사용 방안들 (Cont.)

- **Copy & Paste**
  - 초보적인 재사용 방식
  - 비슷한 예제를 다른 source에서 복사해서 사용.

```
GregorianCalendar date = (GregorianCalendar)Calendar.getInstance();
SimpleDateFormat df = new SimpleDateFormat("yyyyMMdd");
String date = df.format(date);
```

Software 재 사용 방안들 (Cont.)

- **Copy & Paste**
  - 예를 들어 A라는 class에서 Date type을 String type으로 변환하는 coding을 하고, class B에서 동일한 logic이 필요하여 복사했다고 가정한 경우
  - **JDK version이 바뀌어 동일한 기능을 제공하는 향상된 interface가 나오면 위의 code를 사용한 A, B class를 모두 변경해야 한다.**

Software 재 사용 방안들 (Cont.)

- **Method(or Function)**
  - 자주 사용되고, 유사한 기능들을 모아 method로 정의하여 재사용.

```
public class DateUtility{
    public static String toStringToday(String format){
        GregorianCalendar date =
                              (GregorianCalendar)Calendar.getInstance();
        SimpleDateFormat df = new SimpleDateFormat(format);
        String date = df.format(date);
        return date;
    }
}
String sdate = DateUtility.toStringToday("yyyyMMdd");
```

Software 재 사용 방안들 (Cont.)

- **Method(or Function)**
  - JDK version이 바뀌거나 method의 내용이 수정되더라도 해당 class를 모두 수정할 필요 없이 `toStringToday()` method의 내용만 수정하면 된다.
  - `toStringToday()` method의 signature를 변경하면 이 method를 사용하는 모든 class에 영향을 준다.
  - **Method 재사용방법은 'Copy & Paste'보다는 진보된 방식이지만, 작업 영역간의 결합도(Coupling) 문제 여전히 존재함.**

Software 재 사용 방안들 (Cont.)

■ Class(include Inheritance)

```
public class Person{
      public String printBirthDate(String format){
            return DateUtility.toStringToday(birthDate, format);
      }
}
```

- Person을 상속받은 모든 class들은 자동적으로 변경된 **`printBirthDate()`** method를 사용하게 된다.
- **`DateUtility`** class의 method가 변경되더라도 **`printBirthDate()`** method의 interface가 변하지 않으면 나머지 class들은 영향을 받지 않는다.
- **부모 class가 바뀌면 자식 class를 변경해야 한다.**

Software 재 사용 방안들 (Cont.)

- **AOP(Aspect Oriented Programming)**
  - 관심의 분리(Separation of Concerns)
  - AOP는 OOP를 더욱 OOP 답게 만들어 줄 수 있다.
  - AOP는 OOP 뿐만 아니라 기존의 절차적 Programming에도 적용될 수 있다.
  - AOP가 핵심관심 Module의 code를 직접 건드리지 않고 필요한 기능이 동작하도록 하는데, Weaving이라고 하는 특수한 작업이 필요하다.
  - 즉, AOP에서 Weaving 작업을 통해 핵심 Module 사이 사이에 필요한 횡단 관심 code가 동작하도록 엮어지게 만든다.

Software 재 사용 방안들 (Cont.)

- **Design Pattern**
  - Program 개발에서 자주 나타나는 과제를 해결하기 위한 방법 중 하나
  - Software 개발과정에서 발견된 know-how를 축적하여 이름을 붙여 이후에 재사용하기 좋은 형태로 특정 규약을 묶어서 정리한 것.
  - 이 용어를 Software 개발 영역에서 구체적으로 처음 제시한 곳은, GoF(Gang of Four)라 불리는 네 명의 Computer 과학 연구자들이 쓴 서적 'Design Patterns : Elements of Reusable Object-Oriented Software'(재사용 가능한 객체지향 software의 요소-Design Pattern)이다.

Software 재 사용 방안들 (Cont.)

- **Design Pattern**
  - 사용 이유
    - 요구사항은 수시로 변경 → 요구사항 변경에 대한 Source code 변경 최소화
    - 여러 사람이 같이 하는 Team Project 진행 → 범용적인 Coding Style 적용
    - 상황에 따라 인수 인계하는 경우도 발생 → 직관적인 Code를 사용

Software 재 사용 방안들 (Cont.)

- **Framework**
  - 비 기능적(Non-Functional) 요구사항(성능, 보안, 확장성, 안정성 등)을 만족하는 구조와 구현된 기능을 안정적으로 실행하도록 제어해 주는 잘 만들어진 구조의 Library의 덩어리
  - Framework는 Application들의 최소한의 공통점을 찾아 하부 구조를 제공함으로써 개발자들로 하여금 System의 하부 구조를 구현하는데 들어가는 노력을 절감하게 해 줌.

Software 재 사용 방안들 (Cont.)

- **Framework**
  - 사용 이유
    - 비기능적인 요소들을 초기 개발 단계마다 구현해야 하는 불합리함 극복
    - 기능적인 요구사항에 집중할 수 있도록 해줌.
    - Design Pattern과 마찬가지로 반복적으로 발견되는 문제를 해결하기 위한 특화된 Solution을 제공한다. 예)한글 Encoding
    - 개발자는 각 개개인의 능력 차이가 큰 직종이고, 따라서 개발자의 구성에 따라 Project의 결과 역시 차이가 크다.
    - 그래서 이런 상황을 극복하기 위한 Code의 결과물이 Framework이다.

Software 재 사용 방안들 (Cont.)

- **Framework**
  - 사용 이유
    - Framework를 이용한다는 의미는 Program의 기본 흐름이나 구조를 정하고, 모든 팀원이 이 구조에 자신의 Code를 추가하는 방식으로 개발하게 됨.
    - Framework의 최대 장점은 개발에 필요한 구조를 이미 Code로 만들어 놓았기 때문에, 실력이 부족한 개발자라 하더라도 반쯤 완성한 상태에서 필요한 부분을 조립하는 형태의 개발이 가능하다는 점.
    - 회사의 입장에서 Framework를 사용하면 일정한 품질이 보장되는 결과물을 얻을 수 있고, 개발자의 입장에서는 완성된 구조에 자신에 만든 Code를 개발해서 넣어주는 형태이므로 개발시간 단축 가능

Software 재 사용 방안들 (Cont.)

- **Design Pattern vs. Framework**
  - Design Pattern은 Framework의 핵심적인 특징
  - Framework를 사용하는 Application에 그 Pattern이 적용
  - 하지만 Framework는 Design Pattern이 아니다.
  - Design Pattern은 Application을 설계할 때 필요한 구조적인 Guideline이 되어 줄 수는 있지만 구체적으로 구현된 Base Code를 제공하지 않는다.
  - Framework는 Design Pattern과 함께 Pattern이 적용된 기반 Class library를 제공해서 Framework를 사용하는 구조적인 틀과 구현 Code를 함께 제공한다.

Software 재 사용 방안들 (Cont.)

■ 결론

- 개발자는 Framework의 Base Code를 **확장**하여 사용하면서 자연스럽게 그 Framework에서 사용된 Pattern을 **적용**할 수 있게 된다.

Framework 구성요소와 종류

- IoC(Inversion of Control)
- Class Library

Framework 구성요소와 종류 (Cont.)

- **IoC(Inversion of Control)**
  - '***제어의 역전***' 즉, Instance 생성부터 소멸까지의 생명주기 관리를 개발자가 아닌 Container가 대신 해준다는 뜻임.
  - Container 역할을 해주는 Framework에게 제어하는 권한을 넘겨서 개발자의 Code가 신경 써야 할 것을 줄이는 전략.
  - Framework의 동작원리를 제어흐름이 일반적인 Program 흐름과 반대로 동작하므로 IoC라고 설명함.
  - Spring Container는 IoC를 지원하며, Meta-data를 통해 Beans를 관리하고 Application의 중요부분을 형성함.
  - Spring Container는 관리되는 Bean들을 의존성 주입(Dependency Injection)을 통해 IoC를 지원함.

Framework 구성요소와 종류 (Cont.)

- **Class Library**
  - Framework는 특정 부분의 기술적인 구현을 library 형태로 제공
  - Class library라는 구성요소는 Framework의 정의 중 하나인 '***Semi Complete(반제품)***'이다 라고 해석하게 만들었다.

Framework 구성요소와 종류 (Cont.)

- **Class Library vs Framework**

특징	Framework	Library
User Code 작성	Framework class를 Sub-classing 해서 작성	독립적으로 작성
호출 흐름	Framework코드가 User code를 호출	User code가 library를 호출
실행흐름	Framework가 제어	User code가 제어
객체의 연동	구조 Framework가 정의	독자적으로 정의

Framework 구성요소와 종류 (Cont.)

■ 결론

- Framework와 Library를 구분하는 방법은 실행제어가 **어디서** 일어나는가에 달려있다.
- Library는 개발자가 만든 Class에서 직접 호출하여 사용하므로 실행의 흐름에 대한 제어를 개발자의 Code가 관장하고 있다.
- Framework는 반대로 Framework에서 개발자가 만든 Class를 호출하여 실행의 흐름에 대한 제어를 담당한다.
- **Design Pattern + Library = Framework**

Framework 구성요소와 종류 (Cont.)

종류

- Architecture 결정 = 사용하는 Framework의 종류 + 사용전략
- **Web(MVC)** : Spring MVC, Struts2, Webwork, PlayFramework
- **OR(Object-Relational) Mapping** : MyBatis, Hibernate, JPA, Spring JDBC
- **AOP(Aspect Oriented Programming)** : Spring AOP, AspectJ, JBoss AOP
- **DI(Dependency Injection)** : Spring DI, Google Guice
- **Build & Library Management** : Ant + Ivy, Maven, Gradle
- **Unit Testing** : jUnit, TestNG, Cactus
- **JavaScript** : jQuery, AngularJS, Node.js, Vue.js, React.js

What is Spring

- EJB(Enterprise Java Beans)를 Main Framework로 사용할 때 불편했던 점들을 해소하기 위한 목적
- *Rod Johnson*, 2002, **Expert One-on-One J2EE Design and Development**, Wrox
- 2003년 6월 Apache 2.0 License
- "The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform."
- "스프링 프레임워크는 최신 자바-기반 엔터프라이즈 애플리케이션을 위한 포괄적인 프로그래밍 및 구성 모델을 제공한다."

What is Spring (Cont.)

- Enterprise Application 구축을 위한 Lightweight Solution.
- J2EE를 대체하는 Framework
- Module화가 되어 있어서 필요한 부분만 사용 가능하다.
- 선언적 Transaction 관리가 가능하다.
- 완전한 기능을 갖춘 MVC Framework를 제공한다.
- AOP 기능을 사용할 수 있다.
- Java Application 개발을 위한 포괄적인 Infra 지원을 제공

Spring Framework History

- 1 October 2002 : Initial Release
- March 2004 : 1.0
- October 2006 : 2.0
- November 2007 : 2.5
- December 2009 : 3.0
- December 2013 : 4.0
- February 2017 : 5.0
- https://www.quora.com/What-is-the-history-of-The-Spring-Framework
- https://www.slideshare.net/AliakseiZhynhiarousk/spring-framework-5-history-and-reactive-features

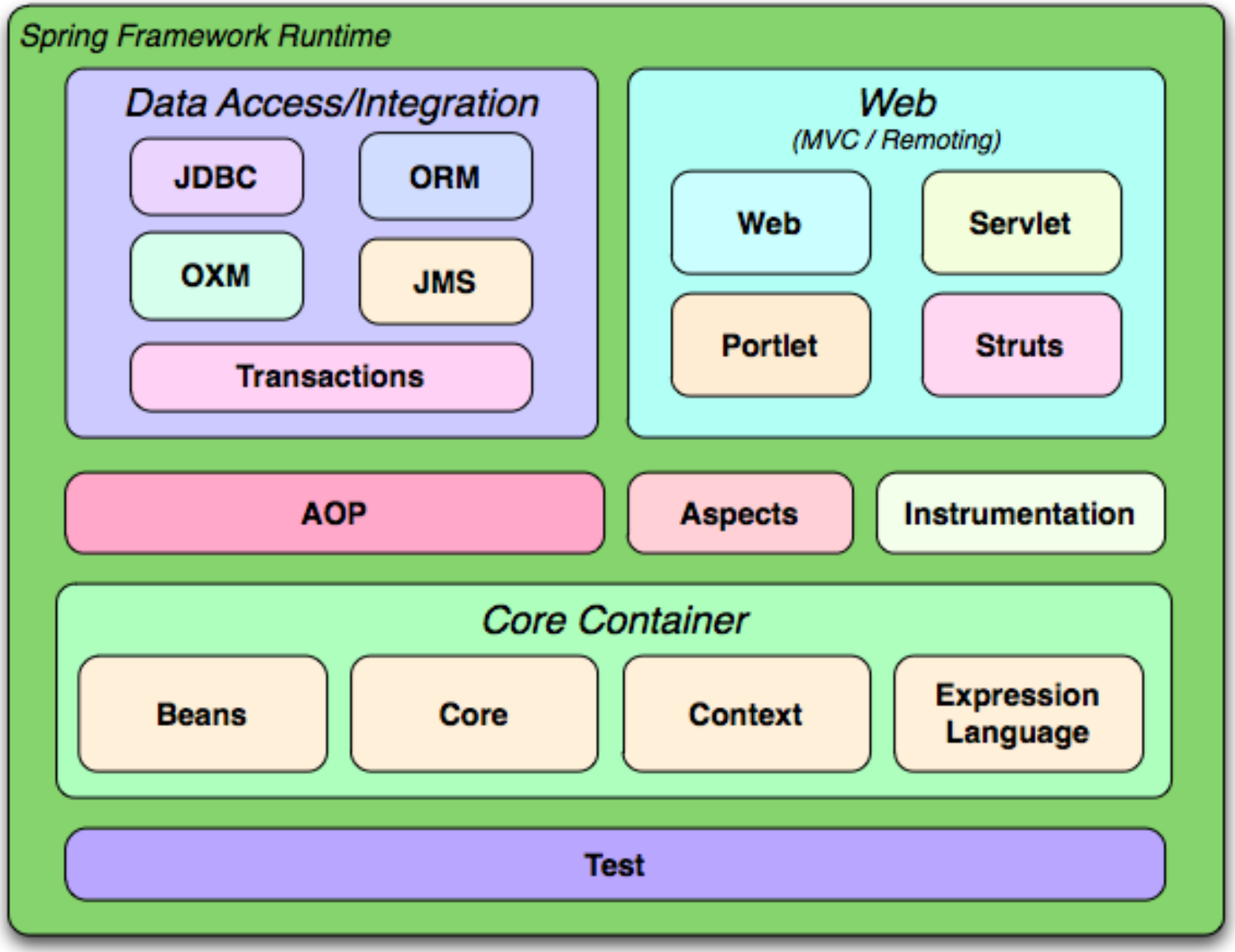
Spring Framework Strategy

- Enterprise Application 개발의 복잡함을 상대하는 Spring 전략

Spring Framework Strategy (Cont.)

- Portable Service Abstraction
  - Transaction 추상화, OXM 추상화, Data access의 Exception 변환기능 등 기술적인 복잡함은 추상화를 통해 low level의 기술 구현부분과 기술을 사용하는 interface로 분리한다.
- OO & DI
  - Spring은 객체지향에 충실한 설계가 가능하도록 단순한 객체 형태로 개발할 수 있고, DI는 유연하게 확장 가능한 객체를 만들어 두고 그 관계는 외부에서 dynamic 하게 설정해 준다.
- AOP
  - Application logic을 담당하는 code에 남아있는 기술 관련 code를 분리해서 별도의 module로 관리하게 해 주는 강력한 기술이다.
- POJO
  - 객체지향 원리에 충실하면서 특정 환경이나 규약에 종속되지 않고 필요에 따라 재활용될 수 있는 방식으로 설계된 객체이다.

Spring Modules



Spring Modules (Cont.)

Modules	Description
spring-beans	Spring Container를 이용해서 객체를 생성하는 기본 기능을 제공
spring-context	객체 생성, Life cycle 관리, Schema 확장 등의 기능 제공
spring-aop	Proxy 기반 AOP 기능 제공
spring-web	REST Client, Data 변환 처리, Servlet Filter, File Upload 지원 등 Web 개발에 필요한 기반 기능 제공
spring-webmvc	Spring 기반의 Web MVC Framework. Web Application을 개발하는데 필요한 Controller와 View 구현을 제공
spring-websocket	Spring MVC에서 Web Socket을 사용하기 위한 기능을 지원
spring-oxm	XML과 Java 객체 간의 Mapping을 처리하기 위한 API 제공
spring-tx	Transaction 처리를 위한 추상 Layer 제공

Spring Modules (Cont.)

Modules	Description
spring-jdbc	JDBC Programming을 보다 쉽게 할 수 있는 Template을 제공. 이를 이용하면 JDBC Programming에서 반복적으로 입력해야 하는 code를 줄일 수 있음.
spring-orm	Hibernate, JPA, MyBatis 등과의 연동을 지원
spring-jms	JMS Server와 Message를 쉽게 주고받을 수 있도록 Template과 Annotation 등을 제공
spring-context-support	Scheduling, Mail 발송, Cache 연동, Velocity 제공

Spring 5 New Features

- Baseline Upgrades
- JDK 9 Runtime Compatibility
- Usage of JDK 8 Features
- Reactive Programming Support
- A Functional Web Framework
- Kotlin Support
- Dropped Features

Spring 5 New Features (Cont.)

- **Baseline Upgrades**
  - Minimum JDK 8 and Java EE 7.
  - Previous JDK and Java EE versions are not supported anymore.
  - To elaborate, Java EE 7 includes
    - Servlet 3.1
    - JMS 2.0
    - JPA 2.1
    - JAX-RS 2.0
    - Bean Validation 1.1

Spring 5 New Features (Cont.)

- **Baseline Upgrades**
  - Similar to Java baseline, there are changes in baselines of many other frameworks as well. e.g.
    - Hibernate 5
    - Jackson 2.6
    - EhCache 2.10
    - JUnit 5
    - Tiles 3
  - Also, note down the minimum supported versions of various servers.
    - Tomcat 8.5+
    - Jetty 9.4+
    - WildFly 10+
    - Netty 4.1+
    - Undertow 1.4+

Spring 5 New Features (Cont.)

- **JDK 9 Runtime Compatibility**
  - `@Nullable` and `@NotNull` annotations will explicitly mark nullable arguments and return values.
  - This enables dealing null values at compile time rather than throwing `NullPointerExceptions` at runtime.
  - On the logging front, Spring Framework 5.0 comes out of the box with Commons Logging bridge module, named spring-jcl instead of the standard Commons Logging.
  - Also, this new version will auto detect Log4j 2.x, SLF4J, JUL (java.util.logging) without any extra bridges.

Spring 5 New Features (Cont.)

- **Usage of JDK 8 Features**
  - Until Spring 4.3, JDK baseline version was 6.
  - So Spring 4 had to support Java 6, 7 and 8.
  - Spring 5 has baseline version 8, so it uses many new features of Java 8 and 9 as well. e.g.
  - Java 8 default methods in core Spring interfaces
  - Internal code improvements based on Java 8 reflection enhancements
  - Use of functional programming in the framework code – lambdas and streams
  - Refer to https://howtodoinjava.com/java8/default-methods-in-java-8/

Spring 5 New Features (Cont.)

- **Reactive Programming Support**
  - Provides an alternate style of programming focused on building applications that react to events.
  - Spring Framework 5 embraces Reactive Streams (language-neutral attempt to define reactive APIs) and Reactor (java implementation of Reactive Streams provided by the Spring Pivotal team) for its own reactive use as well as in many of its core APIs.
  - Spring Web Reactive lives in the new **spring-web-reactive** module next to the existing (and popular!) Spring Web MVC that lives in the **spring-webmvc** module.

Spring 5 New Features (Cont.)

- **A Functional Web Framework**
  - Provides features to define endpoints using functional programming style.
  - This framework introduces two fundamental components: `HandlerFunction` and `RouterFunction`.
    - The `HandlerFunction` represents a function that handles incoming requests and generates responses.
    - `RouterFunction` serves as an alternative to the `@RequestMapping` annotation.
    - It's used for routing incoming requests to handler functions. e.g.

```
RouterFunction<String> route =
    route(GET("/hello-world"),
    request -> Response.ok().body(fromObject("Hello World")));
```

Spring 5 New Features (Cont.)

- **Kotlin Support**
  - Kotlin is a statically typed JVM language that enables code that is expressive, short, and readable.
  - Spring framework 5.0 has good support for Kotlin.

Spring 5 New Features (Cont.)

- **Dropped Features**
  - Spring Framework 5 removed support for a few frameworks. e.g.
    - Portlet
    - Velocity
    - JasperReports
    - XMLBeans
    - JDO
    - Guava