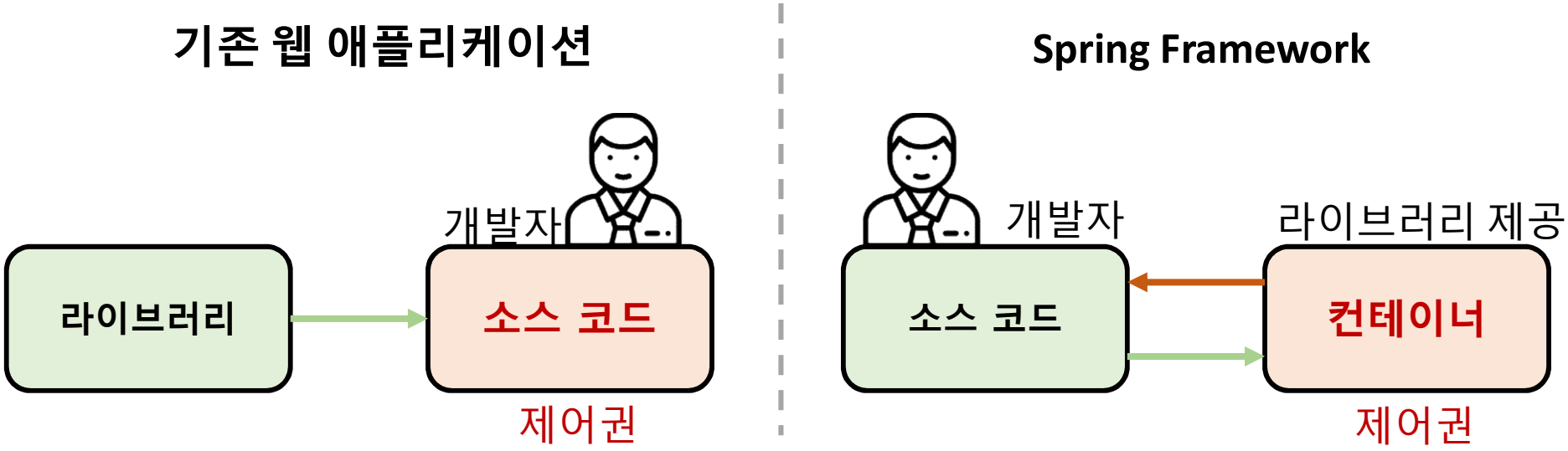
Spring
IoC, DI

Spring IoC

▶ Spring IoC

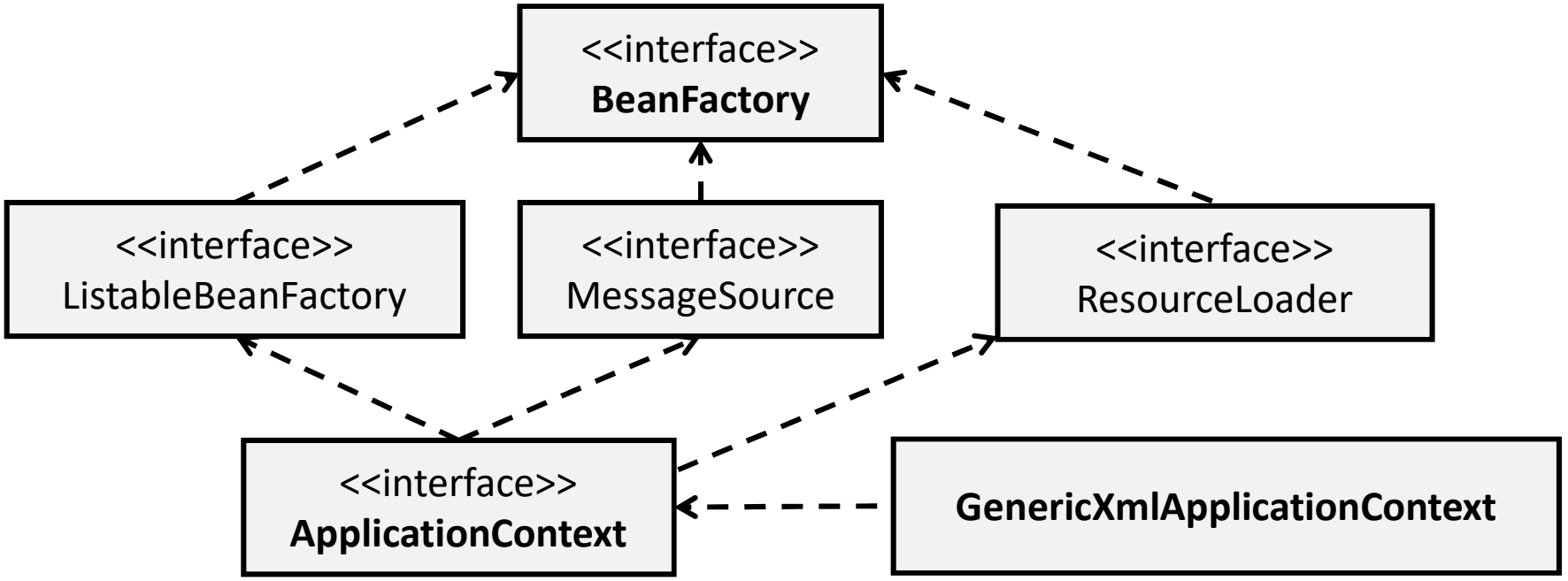
IoC(Inversion of Controller)는 프로그램을 구동하는데 필요한 객체에 대한 생성, 변경 등의 관리를 프로그램을 개발하는 사람이 아닌 프로그램을 구동하는 컨테이너에서 직접 관리하는 것을 말함
스프링은 IoC구조를 통해 구동 시 필요한 객체의 생성부터 생명주기까지 해당 객체에 대한 관리를 직접 수행함



▶ Spring IoC 컨테이너

스프링에서 관리하는 객체를 Bean(빈)이라고 하고

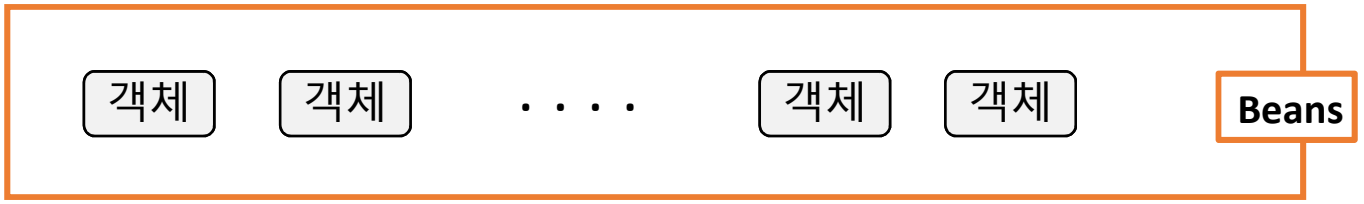
해당 빈들을 관리한다는 의미로 컨테이너를 Bean Factory라고 함

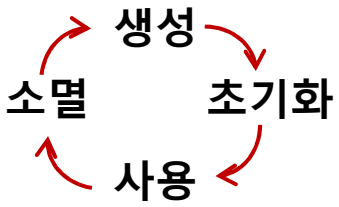


▶ Spring IoC 컨테이너

✓ IoC 컨테이너 역할

1. 객체의 생명주기와 의존성 관리
2. VO(DTO / POJO) 객체의 생성, 초기화, 소멸 등의 처리 담장
3. 개발자가 직접 객체를 생성할 수 있지만 해당 권한을
   컨테이너에 맡김으로써 소스 코드 구현 시간 단축





▶ Spring IoC 컨테이너

✓ IoC 컨테이너와 Bean 객체

Bean **빈**	스프링이 IoC방식으로 관리하는 Class 스프링이 직접 생성과 제어를 담당하는 객체
Bean Factory **빈 팩토리**	스프링의 IoC를 담당하는 핵심 컨테이너 Bean 등록, 생성, 조회, 반환하는 기능 담당
ApplicationContext **애플리케이션 컨텍스트**	BeanFactory를 확장한 IoC 컨테이너 Bean을 등록하고 관리하는 기능은 BeanFactory와 동일하지만 스프링이 제공하는 각종 부가 서비스를 추가로 제공
GenericXmlApplicationContext	ApplicationContext를 구현한 Class 일반적인 XML형태의 문서를 읽어 컨테이너 역할
Configuration metadata **설정 메타 정보**	ApplicationContext 또는 BeanFactory가 IoC를 적용하기 위해 사용하는 설정 정보 설정 메타 정보는 IoC 컨테이너에 의해 관리되는 Bean 객체를 생성하고 구성할 때 사용

Spring
DI

▶ Spring DI

DI(Dependency Injection)는 IoC 구현의 핵심 기술이라고 할 수 있음
사용하는 객체를 직접 생성하여 만드는 것이 아니라 컨테이너가
빈의 설정 정보를 읽어와 자동으로 해당 객체에 연결하는 것 의미
이렇게 의존성을 주입 받게 되면 이후 해당 객체를 수정해야 할 상황이
생겼을 때 소스 코드의 수정을 최소화 할 수 있음

✓ DI의 장점

1. 개발자가 작성해야 할 코드가 단순해짐
2. 각 객체 간의 종속 관계(결합도) 해소 가능

* 객체간의 종속 관계(결합도) : 한 클래스에서 필드 객체를 생성 할 때 발생하는
두 객체 간의 관계를 말하며, 각 객체 간의 내용이 수정될
경우 영향을 미치는 정도를 나타냄

▶ Spring DI 종류

✓ Setter메소드를 통한 의존성 주입

의존성을 주입 받는 Setter메소드를 만들고 이를 통해 의존성 주입

✓ 생성자를 통한 의존성 주입

필요한 의존성을 포함하는 클래스에 생성자를 만들고
이를 통해 의존성 주입

✓ 메소드를 통한 의존성 주입

의존성을 입력 받는 일반 메소드를 만들고 이를 통해 의존성 주입

▶ Spring DI 종류

✓ Setter메소드를 통한 의존성 주입

Setter메소드를 통해 의존 관계가 있는 Bean을 주입하기 위해서는
<property>태그 사용

✓ XML 선언 방법

```
<bean id="객체 이름" class="클래스 풀 네임">
        <property name="name" value="OOO"/>
        <property name="name" ref="OOO"/>
</bean>
```

* name 속성 : Class에서 선언한 필드 변수 이름
* value 속성 : 단순 값 또는 Bean이 아닌 객체를 주입할 때 사용
* ref 속성 : Bean이름을 이용해 주입할 Bean을 찾음

▶ Spring DI 종류

✓ Setter메소드를 통한 의존성 주입 예시

```
<bean id="student" class="com.kh.spring.person.model.vo.Student">
        <property name="name" value="홍길동"/>
        <property name="wallet" ref="money"/>
</bean>
<bean id="money" class="com.kh.spring.wallet.model.vo.Wallet"/>
```

▶ Spring DI 종류

✓ 생성자를 통한 의존성 주입

Constructor를 통해 의존 관계가 있는 Bean을 주입하기 위해서는
<constructor-arg>태그 사용

✓ XML 선언 방법

```
<bean id="불러올 객체 이름" class="클래스 풀 네임">
        <constructor-arg index="0" value="OOO"/>
        <constructor-arg name="OOO" ref="OOO"/>
</bean>
```

* Constructor 주입 방식은 생성자의 파라미터를 이용하기 때문에 한 번에 여러 개의 객체 주입이 가능하며 필드 선언 순서에 따라 index 속성을 통해서도 접근 가능

▶ Spring DI 종류

✓ 생성자를 통한 의존성 주입 예시

```
<bean id="student" class="com.kh.spring.person.model.vo.Student">
        <constructor-arg index="0" value="홍길동"/>
        <constructor-arg index="1" ref="money"/>
</bean>
<bean id="money" class="com.kh.spring.wallet.model.vo.Wallet"/>
```