**IoC (Inversion of Control)**

- 의존 관계 주입

- 의존 객체를 직접 생성이 아닌, 장치(생성자)를 사용해서 주입을 받아서 사용함

**BeanFactory**

- 스프링 IoC 컨테이너의 최상위 인터페이스

- 스프링이 여러가지 기능들을 제공해줄 수 있음

**Bean**

- 스프링 IoC 컨테이너가 관리하는 객체

- 의존성 관리

- 스코프: 싱글톤(메모리 아낄 수 있음)

- 라이프사이클 인터페이스 지원

(부가적인 기능들을 만들 수 있음)

**ApplicationContext**

- BeanFactory를 상속받음

- BeanFactory의 IoC기능을 가지면서 추가적인 기능을가짐

- ApplicationContext가 아닌, 해당 구체적인

인터페이스명으로 주입 받는게 좋음(직관적)

**스프링 IoC 컨테이너**

- Bean 설정 파일이 있어야함(ApplicationContext**)**

: 고전적으로 xml설정 파일이 있고,

후에 java설정 파일이 등장하여 유연해졌으나 일일이 Bean설정을 해야돼서 귀찮음

**@Autowire**

- 생성자, setter, 필드에 의존성 주입을 할 수 있음

- required: 기본값 true

- 같은 타입의 빈이 여러 개인 경우 주입을 못해줌

: @primary, @Qualifier로 원하는 빈을 지정하거나,

해당 타입의 모든 빈을 주입 받아야함

-> 이중에서 primary를 제일 추천

**컴포넌트 스캔**

- Application을 시작점으로 Application을 담고있는 패키지 안의 모든 것부터 해당 패키지 안의 다른 패키지까지 스캔함

- Filter 옵션으로 어디서부터 어디까지 스캔할 것 인가,

또는 어디 부분을 스캔하지 않을 것 인가를 설정함

**@Component**

- @Repository, @Service, @Controller, @Configuration

**Function을 사용한 Bean 등록**

- Spring5부터 나옴

- 프록시를 사용하지 않아 구동 시간이 짧음

- 앱 구동 시 성능상 구동시간이 좋다고해서 컴포넌트

스캔을 버리면 Bean마다 설정을 해야되기 때문에 너무 불편할 것 같아 반대

**BeanFactoryPostProcessor**

- 다른 Bean들이 만들어지기 전에 컴포넌트 스캔을

해서 Bean등록을 해줌

**스코프**

- 싱글톤

: 아무것도 설정하지 않은 Bean의 스코프

: 해당 Bean의 인스턴스가 오직 한 개 일 때

: 프로토타입 Bean을 참조하게 되면 프로토타입의

인스턴트 업데이트가 안돼서 한 개밖에 사용을

못하기때문에 문제가 생김

-> 싱글톤이 바로 프로토타입을 참조하게 되면

매번 바꿀 여유를 안주기 때문에, proxyMode를

설정하여 프록시로 감싸서 프록시를 거친

다음에 프로토타입을 참조하게 만들어 해결할

수 있음

(프록시는 프로토타입을 상속받아 같은 타입임)

- 프로토타입

: 매번 새로운 인스턴스를 만들어서 사용

: 싱글톤 Bean을 참조해도 아무 문제가 없음

**프로파일**

: Bean들의 그룹

: 각각의 환경마다 다른 Bean을 사용해야하는 경우나,

특정 환경에서만 Bean을 따로 등록해야 하는 경우

**프로파일 정의 방법**

- 클래스에 정의

: @Configuraion or @Component들 @Profile(“이름”)

- 메소드에 정의

: @Bean @ Profile(“이름”)

**프로퍼티**

- 다양한 방법으로 정의할 수 있는 설정값

- 계층형으로 접근

- @PropertySource로 프로퍼티를 추가할 수 있음

**MessageSource (ApplicationContext기능)**

- 국제화 기능을 제공하는 인터페이스

: 메시지를 다국화함

**ApplicationEventPublisher (ApplicationContext기능)**

- 이벤트 기반의 프로그래밍할 때 유용한 인터페이스

- 이벤트 클래스

: Bean으로 등록을 안함

: 원하는 데이터를 담아서 전송할 수 있는 이벤트를

만들거나 이벤트를 발생시킨 소스만 전달함

- 이벤트 핸들러

: Bean으로 등록을 함

: 이벤트를 전달 받음

: 여러 개일 경우 동시실행이 아닌 순차적으로 실행

(order와 비동기 실행이 가능)

- 4.2부터 ApplicationEvent를 상속받지 않아도 됨

: 스프링이 추구하는 비침투성을 만족할 수 있다.

(스프링 코드가 내 코드에 노출이 되지 않음)

**ResourceLoader (ApplicationContext기능)**

- 리소스를 읽어오는 인터페이스