**CHAPTER1**

**1.1 자바 개발 간소화**

POJO(Plain Old Java Object) : 간단한 형태에도 불구하고 매우 강력하다.

DI, AOP를 이용하여 조립할 수 있다.

DI(Dependency Injection)는 주입을 통해서 결합도를 낮춰준다.

애플리케이션 컨텍스트(application context)는 애플리케이션을 구성하는 객체의 생성과 와이어링을 전적으로 책임진다.

애스팩트 지향 프로그래밍은 애플리케이션 전체에 걸쳐 사용되는 기능을 재사용할 수 있는 컴포넌트에 담는다.

애스팩트 지향 프로그래밍은 소프트웨어 시스템 내부의 관심사들을 서로 분리하는 기술이라고 설명한다.

템플릿을 이용하여 자바 개발의 복잡성을 줄일 수 있다. 템플릿 내부에서 처리된다.

**1.2 빈을 담는 그릇, 컨테이너**

스프링 기반 애플리케이션에서는 스프링 컨테이너 안에서 객체가 태어나고, 자라고, 소멸한다,

스프링 컨테이너는 크게 두 가지로 분류된다.

1) 빈 팩토리(org.springframework.beans.factory.BeanFactory) 이는 DI에 대한 기본적인 지원을 제공하는 가장 단순한 컨테이너다.

2) 애플리케이션 컨텍스트(org.springframework.context.ApplicationContext) 빈 팩토리를 확장해 프로퍼티(property) 파일에 텍스트 메시지를 읽고 해당 이벤트 리스너에 대한 애플리케이션 이벤트 발생 같은 애플리케이션 프레임워크 서비스를 제공하는 컨테이너다.

빈 팩토리는 지나치게 저수준의 기능을 제공하기 때문에 애플리케이션 컨텍스트를 선호한다.

**1.2.1 또 하나의 컨테이너, 애플리케이션 컨텍스트**

-AnnotationConfigApplicationContext – 하나 이상의 자바 기반 설정 클래스에서 스프링 애플리케이션 컨텍스트를 로드한다.

-AnnotationConfigWebApplicationContext – 하나 이상의 자바 기반 설정 클래스에서 스프링 웹 애플리케이션 컨텍스트를 로드한다.

-ClassPathXmlApplicationContext – 클래스패스(classpath)에 위치한 XML 파일에서 컨텍스트 정의 내용을 로드한다.

-FileSystemXmlApplicationContext – 파일 시스템에서, 즉 파일 경로로 지정된 XML 파일에서 컨텍스트 정의 내용을 로드한다.

-XmlWebApplicationContext – 웹 애플리케이션에 포함된 XML 파일에서 컨텍스트 정의 내용을 로드한다.

**1.2.2 빈의 일생**

보통의 자바 애플리케이션에서 빈의 생명 주기는 매우 단순하다.

자바의 new 키워드를 이용해 빈을 인스턴스화하고 이를 바로 사용한다. 빈이 더 이상 사용되지 않으면 가비지 컬렉션 후보가 되어 언젠가는 사라질 것이다.

반면 스프링 컨테이너 내에서 빈의 생명 주기는 좀더 정교 하다.

**BeanFactory 컨테이너 내에서 빈이 갖는 구동 생명주기를 보여 준다.**

1. 스프링이 빈을 인스턴스화 한다.

2. 스프링이 값과 빈의 레퍼런스를 빈의 프로퍼티에 주입한다.

3. 빈이 BeanNameAware를 구현하면 스프링이 빈의 ID를 setBeanName() 메소드에 넘긴다.

4. 빈이 BeanFactoryAware를 구현하면 setBeanFactory() 메소드를 호출하여 빈 팩토리 자체를 넘긴다.

5. 빈이 ApplicationContextAware를 구현하면 스프링이 setApplicationContext() 메소드를 호출하고 둘러싼 애플리케이션 컨텍스트에 대한 참조를 넘긴다.

6. 빈이 BeanPostProcessor 인터페이스를 구현하면 스프링은 postProcessBeforeInitialization() 메소드를 호출한다.

7. 빈이 InitializingBean 인터페이스를 구현하면 스프링은 afterPropertiesSet() 메소드를 호출한다. 마찬가지로 빈이 init-method와 함께 선언됐으면 지정한 초기화 메소드가 호출된다.

8. 빈이 BeanPostProcessor를 구현하면 스프링은 postProcessAfterInitialization() 메소드를 호출한다.

9. 이 상태가 되면 빈은 애플리케이션에서 사용할 준비가 된 것이며, 애플리케이션 컨텍스트가 소멸될 때까지 애플리케이션 컨텍스트에 남아 있다.

10. 빈이 DiposableBean 인터페이스를 구현하면 스프링은 destroy() 메소드를 호출한다. 마찬가지로 빈이 destroy-method와 함께 선언됐으면 지정된 메소드가 호출된다.

**1.3 스프링 현황**



4.0은 20개의 서로 다른 모듈로 이루어져 있다. 그리고 6개의 기능 카테고리에 속하게 된다.

1) 데이터 액세스와 통합

2) 웹과 리모팅

3) 애스펙트 지향 프로그래밍

4) 인스트루멘테이션

5) 코어 스프링 컨테이너

6) 테스팅

**코어 스프링 컨테이너**

스프링 프레임워크의 핵심은 스프링 애플리케이션의 빈의 생성, 설정 그리고 처리 방법을 관리하는 컨테이너다.

모든 스프링의 모듈은 코어 컨테이너 위에 구축된다.

**스프링의 AOP 모듈**

스프링은 AOP 모듈을 통해 애스펙트 지향 프로그래밍을 풍부하게 지원한다.

**데이터 액세스와 통합**

JDBC를 이용하면 반복 코드가 생기는데, 스프링의 JDBC와 데이터 액세스 객체(DAO) 모듈은 이렇게 반복되는 코드를 추상화하므로 간단하게 만들 수 있고, 데이터베이스 리소스를 닫지 않아서 발생할 수 있는 문제를 예방한다. 또한, 이 모듈에는 여러 종류의 데이터베이스 서버가 제공하는 오류 메시지에 의미 있는 예외 계층이 추가되어 있다.

스프링의 ORM 모듈은 JDBC보다 객체 관계 매핑(ORM) 도구를 선호하는 사람들을 위한 것이다.스프링은 고유한 ORM 솔루션을 구현하지 않고, 하이버네이트, 자바 퍼시스턴스, iBatis 등 널리 사용되는 ORM 프레임워크와의 연결 고리를 제공한다.

**웹과 리모팅**

스프링이 다양한 유명 MVC 프레임워크와 잘 통합되기는 하지만, 웹과 리모팅 모듈에는 애플리케이션의 웹 계층에서 결합도를 낮추는 MVC 프레임워크가 별도로 만들어져 있다.

사용자 접촉 웹 애플리케이션뿐만 아니라 다른 애플리케이션과 상호작용하는 애플리케이션을 개발하기 위한 다양한 리모팅 옵션도 제공한다. 그리고 스프링은 REST API에 대해 최고의 지원을 제공한다.

**인스트루멘테이션**

스프링의 인스트루멘테이션 모듈은 JVM에 에이전트를 추가하는 기능을 제공한다. 정확히는 톰캣용 위빙 에이전트를 제공하는데, 톰캣은 클래스로더에 의해 로드되는 클래스 파일을 변환한다.

**테스팅**

개발자가 작성하는 테스트의 중요성을 인식하여 스프링은 스프링 애플리케이션 테스트에 전념하는 모듈을 제공한다.

통합 테스트의 경우, 이 모듈은 스프링 애플리케이션 컨텍스트에서 빈을 로드하고 이 컨텍스트에 있는 빈과의 작업을 지원한다.

**1.4 스프링의 새로운 기능**

스프링 프레임워크는 세 개의 중요한 릴리스(3.1, 3.2, 4.0)를 선보였고, 각각의 릴리스는 새로운 특징과 개선으로 편리한 애플리케이션 개발을 가져왔다.

**1.4.1 스프링 3.1에서 새로워진 기능**

3.1에서는 여러 가지 유용한 새 기능과 개선이 있었으며, 이 중 대부분이 설정을 간소화하고 개선하는데 초점이 맞춰졌다.

-다양한 환경에서 서로 다른 설정을 선택하는 일반적인 문제를 다루기 위해 환경 프로파일을 도입했다.

-스프링에도 차츰 선언적 캐싱 지원이 쓰이게 되었다.

-새로운 c-네임스페이스 덕분에 생성자 주입이 가능해졌다.

-스프링의 JPA 지원이 개선되었다.

스프링 MVC에 대한 부분 또한 강화되었다.

-애트리뷰트 모델에 대한 경로 변수의 자동 바인딩이 추가되었다.

-요청을 위한 Accept와 Content-Type 헤더에 매칭하기 위한 @RequestMapping의 produces와 consumes 애트리뷰트를 포함한다.

-@RequestPart 애너테이션으로 핸들러 메소드 파라미터에 대한 다중 요청 부분의 바인딩이 가능하다.

**1.4.2 스프링 3.2에서 새로워진 기능**

스프링 3.1이 설정 개선에 초점을 맞추고, 스프링 MVC를 포함한 기타 개선점이 적었던 반면, 스프링 3.2는 주로 스프링 MVC에 초점을 맞춘 릴리즈였다.

-스프링 3.2 컨트롤러는 서블릿 3의 비동기식 요청을 이용하여 요청 처리를 분리시키고, 분리된 스레드 내에서 처리되도록 한다.

-스프링 2.5 이후로 스프링 MVC 컨트롤러 테스트가 POJO처럼 용이해졌다고는 하지만, 스프링 3.2에서는 스프링 MVC 테스트 프레임워크가 포함됨으로써 컨트롤러에 대한 더욱 다양한 테스트 작성이 가능해졌다.

-ResponseEntityExceptionHandler 클래스는 DefaultHandlerExceptionResolver의 대안으로 사용되기 위해 추가되었다.

-매핑된 인터셉터들은 인터셉터 프로세싱으로부터 배제되기 위해 URL 패턴을 지원한다.

**MVC 말고도 다른 개선점들이 추가되었다. 3.2이 가장 흥미로운 새로운 특징을 보자.**

-@Autowired, @Value 그리고 @Bean 애너테이션은 커스텀 주입과 빈 선언 애너테이션을 생성하기 위한 메타-애너테이션으로 사용된다.

-DateTimeFormat 애너테이션은 더 이상 JodaTime에 강력한 의존성을 갖지 않는다.

-스프링의 선언적 캐싱 지원은 JCache 0.5를 위한 초기 지원을 가진다.

-날짜와 시간을 파싱하고 렌더링하는 글로벌 포맷을 정의한다.

-통합 테스트는 WebApplicationContext를 설정하고 로드한다.

-통합 테스트는 요청 범위와 세션 범위의 빈에 대하여 테스트한다.

**1.4.3 스프링 4.0에서 새로워진 기능**

스프링 4.0에는 다음과 같이 흥미롭고 새로운 기능들이 많이 있다.

-스프링은 이제 JSR-356의 웹 소켓을 위한 자바 API 지원을 포함하여 웹 소켓 프로그래밍을 지원한다.

-스프링 4.0은 자바8의 특징인 람다식을 지원하는 첫 번째 프레임워크 중 하나이다. 무엇보다도 읽기에 훨씬 깨끗하고 쉬운 콜백 인터페이스로 작업할 수 있다.

-JSR-310의 데이터 및 시간 API를 포함하는 자바8 지원을 통해 개발자들이 java.util.Date 또는 java.util.Calendar에서 제공되는 것보다 훨씬 풍부한 API로 날짜와 시간을 작업하는 기회를 제공받을 수 있다.

-조건부 빈 생성에 대한 일반적 지원이 추가되면서 개발자가 정의한 조건이 만족되었을 때만 빈이 생성되도록 선언할 수 있다.

-스프링 4.0이 포함하는 또 한 가지는 스프링의 RestTemplate에 대한 새로운 비동기식 구현으로, RestTemplate는 응답을 반환하고, 연산이 끝났을 때 콜백을 호출한다.