1장 프레임워크 기초

단답형

웹 브라우저가 웹 서버로부터 저장된 HTML을 읽어와 표현

웹 브라우저가 웹서버에 동적 페이지를 요청하면 이 요청을 애플리케이션 서버가 실행하고, 처리 결과(데이터베이스 조회 등)를 브라우저가 HTML 형식으로 받아서 표현

동적 콘텐츠. 매번 프로그램 실행으로 성능 저하. 트랜잭션 관리 어려움

웹 컨테이너가 세션 관리. 웹페이지와 비즈니스 로직 분리. 자바 언어 사용, 오브젝트 지향의 장점 '재사용'

분산 트랜잭션 처리 컴포넌트. JSP, Servlet으로 웹 프리젠테이션 처리. ( )분산 객체로 비즈니스 로직 처리. JEEE 버전업과 함께 EJB 복잡도는 높아지고 확장성은 떨어짐.

DI x AOP 컨테이너. 경량 컨테이너, POJO 객체. 웹, 클라우드, 모바일 등 다양한 자바 기반의 애플리케이션을 만들기 위한 프레임워크. POJO(Plain Old Java Object) 지원을 통해 기존 EJB의 문제점인 복잡성을 줄임.

특정 인터페이스 또는 클래스를 상속받을 필요가 없는 가벼운 객체를 의미. 단순한 자바 클래스.

웹 애플리케이션 아키텍처 물리 층 이름. 층 3개

웹 애플리케이션 아키텍처 논리 층 이름. 층 3개. 이건 물리층의 중간 층에서 나온 3개를 의미해.

비즈니스 로직 레이어의 주된 두가지 특징.

정적 컨텐츠, 동적 컨텐츠, CGI, JSP Servlet, EJB, Spring, 티어 클라이언트 층 중간 층 EIS 층, 레이어 프리젠테이션 층 비즈니스 로직 층 데이터 엑세스 층, 서비스 도메인,

표현 층 또는 영속화 메커니즘이 변경되어도 비즈니스 로직 층에는 영향을 최소화. 레이어 층 도입과 함께 결합 부분에 약한 결합 설계 구현(인터페이스 도입).

사용자 인터페이스와 컨트롤러를 제공하는 역할을 하는 층.

사용자 인터페이스를 통해 사용자의 입력을 받아 적정한 로직을 호출하고, 그 결과를 사용자 인터페이스로 변환하는 작업. 프리젠테이션 층에 있음.

JSP만 구현 개발하거나 Java bean을 포함하여 개발하는 방식을 의미. JSP에 뷰와 비즈니스 로직이 혼재되어 복잡도가 높고, 유지보수 어려움.

모델, 뷰, 컨트롤러 방식.

뷰에 필요한 비즈니스 영역의 로직을 처리.

비즈니스 영역에 대한 프레젠테이션 뷰(결과 화면)를 담당

사용자의 입력 처리와 화면의 흐름 제어를 담당.

유즈케이스로 표현되는 특정 업무 처리를 위한 서비스와 도메인으로 구성. 개발, 운영 업무의 경우 웹 애플리케이션의 기능 추가와 변경을 담당.

비즈니스 로직 층의 역할 중 트랜잭션의 ACID 특성 중 ( )과 ( )을 관리.

트랜잭션 경계는 ( )과 ( )층 사이에 존재하는 것이 일반적.

오목형 레이어, 프리젠테이션 층, 컨트롤러, 모델 1 방식. 모델 2 방식, 모델, 뷰, 컨트롤러, 비즈니스 로직, 원자성 독립성, 프리젠테이션 비즈니스 로직

트랜잭션 시작과 커밋, 롤백과 같은 RDB에 대한 트랜잭션 관리를 소스코드로 명시.

프레임워크에서 제공하는 정의 파일 선언을 통해 트랜잭션 관리.

RDB 액세스를 비즈니스 로직으로 숨기고, 비즈니스 로직에 필요한 데이터를 테이블에서 취득해서 오브젝트에 매칭하는 층.

오브젝트와 RDB를 매핑하는 것을 ( )이라고 함.

개발 효율성과 유연성 향상을 위해서는 티어 또는 레이어를 통해 ( )화. ( )이 큰 쪽이 티어, 레이어 그리고 작은 ( )은 패키지 또는 컴포넌트.

부품화를 위해서는 ( )가 중요.

웹 애플리케이션의 문제 3개. 이러한 문제들이 스프링 프레임워크로 해결이 됨.

위의 문제 3개가 각각 스프링 프레임워크의 어떤 기술들로 해결되었지?

Spring Framework의 주요 특징 7개.

명시적 트랜잭션, 선언적 트랜잭션, 데이터 액세스 층, O/R 매핑, 부품, 인터페이스, 오브젝트 생명 주기 부품화 문제 기술 은닉과 부적절한 기술 은닉, DI 컨테이너 DI 컨테이너 AOP, POJO 관리 IoC 경량 컨테이너 DIxAOP 지원 O/R 매핑 프레임워크 지원 스프링 데이터를 통한 다양한 데이터 연동 기능 스프링 배치 스프링 시큐리티

서술형

애플리케이션 아키텍처에서 애플리케이션 서버 티어를 구성하는 3가지 레이어에 대해 설명하고 결합도를 줄이기 위한 방법을 설명하시오.

- ✓ 논리 층인 레이어로 프리젠테이션, 비즈니스 로직, 데이터 엑세스 층 이렇게 3가지가 있다. 프리젠테이션 층은 사용자 인터페이스와 컨트롤러를 제공하는 역할을 한다. 컨트롤러는 사용자 인터페이스를 통해 사용자의 입력을 받아 적정한 로직을 호출하고, 그 결과를 사용자 인터페이스로 변환하는 작업이다. 비즈니스 로직 층은 유즈케이스로 표현되는 특정 업무 처리를 위한 서비스와 도메인으로 구성한다. 개발, 운영 업무의 경우 웹 애플리케이션의 기능 추가와 변경은 비즈니스 로직을 변경하면 된다. 데이터 엑세스 층은 RDB 엑세스를 비즈니스 로직에서 숨기고, 비즈니스 로직에 필요한 데이터를 테이블에서 취득해서 오브젝트에 매칭한다. 결합도를 줄이기 위한 방법은 오목형 레이어로 설계하는 것이다. 표현 층 또는 영속화 메커니즘이 변경되어도 비즈니스 로직 층에는 영향을 최소화하고, 레이어 층 도입과 함께 결합 부분에 약한 결합 설계를 구현(인터페이스)한다.

스프링 프레임워크의 특징 3가지(POJO, IoC, DIxAOP)를 설명하시오.

- ✓ POJO

  특정한 인터페이스를 구현하거나 상속을 받을 필요가 없는 가벼운 객체를 관리한다. 특히 EJB의 복잡성을 POJO를 관리하는 방식으로 변경하여 기존의 많은 클래스를 상속받지 않고 일반 클래스로 구현할 수 있게 한다.

  IoC

  필요에 따라 스프링 프레임워크에서 사용자의 코드를 호출한다. 일반 오브젝트의 생애 주기 관리나 오브젝트 간의 의존관계를 해결하는 아키텍처를 구현한다.

  DI

  개발 효율성과 유연성 향상을 위해서는 티어 또는 레이어를 통해 부품화가 필요하며 부품이 큰 쪽이 티어, 레이어 그리고 작은 부품은 패키지 또는 컴포넌트가 된다. 부품 간에는 인터페이스를 통해 연동할 수 있어야 하는데, 이 의존 관계를 기술적으로 구현한 것이 스프링의 DI 이다. 즉, 각각의 계층이나 서비스들 간에 의존성이 존재할 경우 프레임워크가 서로 연결시켜 준다.

  AOP

  불필요한 기술은 은닉해서 사용하는 것이 좋을 수도 있다. 특히 로깅, 예외 처리 부분은 실제 서비스 코드에 반영되면 코드의 가독성이 떨어지고 테스트 역시 어려워질 수 있다. 그래서

랜잭션이나 로깅, 보안과 같이 여러 모듈에서 공통적으로 사용하는 기능의 경우 해당 기능을 분리하여 관리하는 것이 스프링 AOP이다. 그렇게 함으로써 서비스 코드에서 공통 모듈의 코드를 제거할 수 있게 되었다.

2장 메이븐 기초

단답형

2장에서 배운 프로젝트를 관리하는 도구.

메이븐 기능 2개

의존관계(라이브러리) 설정(pom.xml)시에 최상위 엘리먼트는 무엇일까?

의존관계(라이브러리) 설정(pom.xml)시에 3개의 필수 필드를 가지는데 그 3개는?

의존관계(라이브러리) 설정(pom.xml)시에 프로젝트 의존관계의 라이브러리를 관리하는 필드?

모든 POM은 ( )으로부터 ( )받는다.

메이븐 저장소는 프로젝트에 사용되는 프로젝트 jar 파일, 라이브러리 jar 파일들이 위치하며 3가지 타입이 있는데 뭘까?

메이븐 의존성 검색 절차 3단계 어디서부터 검색하는지 말하시오.

의존 라이브러리 scope속성 중 기본 스코프는?

의존 라이브러리 scope속성 중 컴파일시에는 직접 의존성을 참조하고 런타임시에는 다른 환경에서 의존성을 제공받는 스코프는?

의존 라이브러리 scope속성 중 컴파일 시에는 사용되지 않지만, 애플리케이션을 실행할 때 사용되는 라이브러리일 경우 설정하는 스코프는?

의존 라이브러리 scope속성 중 테스트하는 시점에만 사용하는 라이브러리에 대한 스코프를 설정할 때 사용하는 스코프는?

의존 라이브러리 scope속성 중 provided와 비슷함. 단지 우리가 직접 jar 파일을 제공해야 하는 것.

메이븐, 빌드 자동화 기능 프로젝트 관리 기능, project, groupId artifactId version, dependency, Super POM 상속, 지역 중앙 원격, 지역->중앙->원격, compile, provided, runtime, test, system

의존 라이브러리 scope속성 중 다른 POM 설정 파일에 정의되어 있는 의존 관계 설정을 현재 프로젝트로 가져온다.

메이븐의 빌드 자동화 기능 중 빌드 단계(컴파일, 테스트, 페키징, 배포)들을 ( )이라고 한다.

각 빌드 단계에서 수행되는 작업을 ( )이라고 한다.

실제 골은 그 단계에 연결된 ( )에 의해 실행된다.

메이븐은 ( ), ( ), ( ) 3개의 라이프사이클이 있다.

기본 라이프사이클은 여러 단계의 ( )로 나뉘어져 있으며, 각 ( )는 의존관계를 가진다.

각 페이즈는 ( ), ( ), ( ) 순서로 진행된다.

빌드 라이프사이클은 하나 이상의 ( )을 수행하는 ( )들로 구성. 각 ( ) 별로 ( ) 이 작업을 수행한다. 이 작업을 ( )이라고 한다.

메이븐 플러그인 중 빌드 이후에 target 디렉토리를 삭제하는 것은?

메이븐 플러그인 중 자바 소스 파일을 컴파일하는 것은?

메이븐 플러그인 중 Junit 단위 테스트를 실행하고, 리포트를 생성하는 것은?

메이븐 플러그인 중 현재 프로젝트로부터 JAR file을 생성하는 것은?

메이븐 플러그인 중 현재 프로젝트로부터 WAR file을 생성하는 것은?

메이븐 플러그인 중 프로젝트의 Javadoc 문서를 생성하는 것은?

메이븐 플러그인 중 빌드 페이지 단계에서 특정 ant 작업을 실행하는 것은?

import, 라이프 사이클, 골, 플러그인, 페이즈, compile test package deploy, 골 페이즈 페이즈 플러그인 골, clean, compiler, surefire, jar, war, Javadoc, antr

서술형

메이븐의 프로젝트 관리 기능인 의존성 관리 기능에 대해 설명하시오.

✓ 의존성 관리 기능

프로젝트 빌드에 필요한 라이브러리, 플러그인들을 개발자 PC에 자동으로 다운로드한다. 개발자가 프로젝트 의존관계에 있는 라이브러리를 pom.xml에 설정하면 개발자 PC의 특정 디렉토리에 관련 라이브러리를 자동으로 관리해줘서 매우 편리한 기능이다. 그래서 대규모 프로젝트를 수행하는 개발자들은 설정 파일을 통해 프로젝트 환경의 일관성을 보장받을 수 있다.

메이븐의 빌드 기능 용어 중에서 빌드 라이프 사이클, 골, 플러그 인에 대해 설명하시오.

✓ 메이븐은 소프트웨어 빌드를 위한 공통 인터페이스를 제공한다. 컴파일, 테스트, 패키징, 배포 등의 빌드 단계들을 빌드 라이프 사이클이라 한다. 메이븐은 기본, clean, site 3개의 라이프 사이클이 있다. 기본 라이프 사이클은 여러 단계의 페이즈로 나뉘어져 있으며, 각 페이즈는 의존관계를 가진다.

✓ 각 빌드 단계에서 수행되는 작업을 골이라고 하고 그 골은 그 단계에 연결된 플러그인에 의해 실행된다. 각 페이즈 별로 플러그인이 작업을 수행하는데 이 작업을 Goal이라고 한다.

✓ 골은 그 단계에 연결된 플러그인에 의해 실행된다. 각 페이즈 별로 플러그인이 작업을 수행한다.

3장 스프링 DI

단답형

의존 관계 주입. 오브젝트 간의 의존 관계를 만드는 것. 스프링 프레임워크는 런타임시 사용할 객체들의 의존 관계를 부여함. 객체 간의 결합도를 낮춤.

Spring Core = ( ) Container = ( ) Container = ( ) Container

역전제어, 즉 인스턴스를 제어하는 주도권이 역전된다는 의미. 컴포넌트를 구성하는 인스턴스 생성과 의존 관계 연결을 개발자의 소스 코드가 아닌 DI 컨테이너가 대신해 주기 때문에 제어가 역전되었다고 정의함.

스프링 프레임워크가 제공하는 IoC 컨테이너를 통해 인스턴스의 생명주기 관리 및 의존 관계 주입을 처리함.

일반적인 애플리케이션에서의 의존 관리는 ( )를 사용.

DI 컨테이너를 활용한 애플리케이션은 MemberDao 인스턴스를 MemberService에 ( )(의존 관계주입)

MemberSampleMain은 MemberService에 ( )관계가 있고, MemberService는 MemberDao에 ( )관계가 있다.

DI 컨테이너를 활용한 애플리케이션 : ( )의 컴포넌트화

구현 클래스는 인터페이스 이름에 ( )을 덧붙임.

스프링 컨테이너가 관리하는 객체

DI, IoC DI DI&AOP, IoC, IoC 컨테이너, new 연산자, 인젝션, 의존 의존, 인터페이스 기반, 스프링 빈

스프링 빈의 생성, 관계, 조립, 생명주기를 관리하는 스프링 프레임워크의 핵심. 의존관계주입을 이용하여 애플리케이션을 구성하는 컴포넌트들을 관리한다.

스프링 컨테이너 종류 2가지. 한 개가 상속한 것이 나머지 한 개임.

빈의 생성, 빈의 의존관계 관리 등의 DI의 기본 기능을 제공한다. 빈이 많지 않고 경량 컨테이너로 작업할 때 활용. XML 파일로부터 설정 정보를 활용하는 가장 많이 사용되는 클래스.

일반적인 스프링 컨테이너를 의미. BeanFactory 인터페이스를 상속받은 하위 인터페이스로 확장된 기능 제공. XML 파일로부터 설정 정보를 활용하는 가장 많이 사용되는 클래스.

웹 애플리케이션을 위한 ApplicationContext. XML 파일로부터 설정 정보를 활용하는 가장 많이 사용되는 클래스.

2가지 종류의 WAC는 뭐야

Persistence(DAO), Service 관련 스프링 빈들을 등록. 웹 애플리케이션 전체에서 사용할 WAC 객체 생성.

ContextLoadListener를 설정하는 파일 이름

컨트롤러와 같은 서블릿 관련 빈 등록. 해당 서블릿 마다 사용할 WAC 객체 생성.

DispatcherServlet을 설정하는 파일 이름

( )에서 ContextLoadListener, DispatcherServlet를 사용하여 ApplicationContext 생성.

IoC 컨테이너, BeanFactory ApplicationContext, BeanFactory, Application Context, Web Application Context, ContextLoadListener DispatcherServlet, ContextLoadListener, root-context, DispatcherServlet, servlet-context, web.xml

Dependency Injection(의존 관계 주입) : (　　), (　　), (　　) 기반 설정을 통해서 객체간의 의존 관계를 설정한다.

XML 기반 설정 : XML 파일을 사용하는 (　　)를 정의하는 방법.

Annotation 기반 설정 : (　　) 애너테이션이 부여된 클래스를 DI 컨테이너가 Bean으로 자동으로 등록하는 방법

Java 기반 설정 : 자바 클래스에 (　　) 애너테이션을, 메서드에 (　　) 애너테이션을 사용해 Bean을 등록하는 방법

POM.xml에 추가하는 로깅 프레임워크 2개를 말하시오.

Dependency Injection 의존성 주입 방식 2가지

생성자 기반 의존성 주입은 (　　)의 인수를 사용해 의존성을 주입하고, 설정파일 XML에 (　　) 태그를 사용하여 주입할 컴포넌트를 설정한다.

설정자 기반 의존성 주입은 메서드의 인수를 통해 의존성을 주입하고, 설정파일 XML에 (　　) 요소의 (　　) 속성에 주입할 컴포넌트의 이름을 설정한다.

bean 태그속성

오브젝트를 유일하게 하는 ID

오브젝트명을 정의. 오브젝트에 여러 이름을 설정하고 싶을 때나 ID에는 설정할 수 없는 이름을 지정할 때 이용.

id의 실체. 패키지명과 클래스명으로 구성

오브젝트의 스코프를 지정(singleton, prototype, request, session, application)

XML Annotation JAVA, <Bean> 요소 @Component @Configuration @Bean, slf4j-api logback-classic, 생성자 기반 설정자 기반, 생성자 <constructor-arg>, <property> name, id, name, class, scope

설정을 물려받을 오브젝트명을 지정

true : 인스턴스를 만들지 않고 공통 설정을 정의해두고 싶을 때 지정. false : 속성 생략 시 기본값. 인스턴스를 만들고 싶을 때 지정.

true : 속성 생략 시 기본값. getBean 메서드로 얻는 인스턴스는 싱글턴. false : getBean 메서드로 얻는 인스턴스는 매번 인스턴스화된 것.

true : 인스턴스 생성을 지연시킴. false : 속성 생략 시 기본 값. BeanFactory 시작 시 인스턴스를 생성.

의존 관계의 대상이 되는 오브젝트가 있는지 검사.

컨테이너가 빈과 다른 빈과의 의존성을 자동으로 연결하도록 하는 어노테이션은?

@Autowired 설정을 사용하기 위해서는 ( )설정이 필요하다. 단 ( )가 설정되어 있으면 생략 가능하다.

컨테이너가 인젝션을 위한 인스턴스(빈)을 설정하는 어노테이션은?

( )선언은 @Component 애노테이션이 붙은 클래스를 자동으로 빈으로 등록하는 기능이다.

스프링 주요 태그중 @Autowired, @Resource, @Required를 이용할 때 선언한다. XML 파일에 이미 등록된 빈들의 애노테이션 기능을 적용하기 위해 선언한다. 빈을 등록하기 위한 검색 기능은 없다.

위의 태그를 생략하고 싶으면 어떤 태그를 선언해야 될까? 2개 쓰면 좋고 중요한거 한 개 그거

@Component, @Service, @Repository, @Controller등의 컴포넌트를 이용할 때 선언함. 특정 패키지 안에 클래스를 검색해서 빈을 자동으로 찾아서 DI 컨테이너에 등록.

parent, abstract, singleton, lazy-init, depend-on, @Autowired, <context:annotation-config> <context:component-scan />, @component, <context:component-scan base-package="패키지 이름" />, <context:annotation-config />, <mvc:annotation-driven /> <context:component-scan>, <context:component-scan base-package="패키지명"/>,

타사의 외부 라이브러리를 사용하여 DI하고자 할 경우에는 소스를 가지고 있지 않고 있기 때문에 애노테이션을 이용하는 방법은 불가능한데 이때는 자바코드, Annotation, xml 중 어떤 걸 이용해서 DI를 설정해야 할까?

Java로 DI 설정 코드의 장점은 ( )하고, ( )에 매우 적합.

Java를 이용한 DI 설정에서 컨테이너 생성 클래스는 뭘까?

빈 설정 메타 정보를 담고 있는 클래스를 선언할 때 쓰는 애노테이션? 해당 클래스가 스프링 설정으로 사용됨.

클래스 내의 메서드를 정의하여 새로운 빈 객체를 정의할 때 사용하는 어노테이션. name 속성을 사용하여 새로운 빈 이름 적용 가능.

빈 객체의 스코프

기본 설정. 컨테이너당 한 개의 빈 객체만 생성

빈을 요청할 때마다 빈 객체를 생성

각 요청용으로 한 개의 빈 객체를 생성(WAC에만 적용됨)

각 세션용으로 한 개의 빈 객체를 생성(WAC에만 적용됨)

서블릿 컨텍스트 생성될 때 빈 객체 생성(WAC에만 적용됨)

빈 생성 후 초기화 작업과 빈 종료 전 전처리 과정을 수행할 수 있는 방법을 제공하는데

Initialization 일 때 Destruction 일 때 둘다 말해	XML 기반	( )요소의 ( )속성에 메서드 지정
	에너테이션 기반	( ) 애너테이션이 붙은 메서드
	자바 기반	( )의 ( )속성에 지정된 메서드
	인터페이스 구현	( ) 인터페이스의 ( ) 메서드

빈 종료 전의 전 처리 작업은 ( ) 스코프의 빈에서는 동작하지 않는다.

xml로 설정, TypeSafe Refactoring, AnnotationConfigApplicationContext, @Configuration, @Bean, singleton prototype request session application, <bean> init-method @PostConstruct @Bean intiMethod InitializeBean afterPropertiesSet <bean> destroy-method @ProDestroy @Bean destoryMethod DisposableBean destroy, prototype

서술형

스프링 빈, Application Context를 정의하시오.

- ✓ 스프링 컨테이너가 관리하는 객체이다. 의존 관계의 그 대상이라고 말할 수 있다. 스프링 빈은 애플리케이션의 핵심을 이루는 객체이며, Spring IoC 컨테이너에 의해 인스턴스화, 관리, 생성된다. Bean은 우리가 컨테이너에 공급하는 설정 메타 데이터(xml 파일)에 의해 생성된다.
- ✓ 일반적인 스프링 컨테이너를 의미한다. BeanFactory 인터페이스를 상속받은 하위 인터페이스로 확장된 기능을 제공한다. XML 파일로부터 설정 정보를 활용하는 가장 많이 사용되는 클래스이다.

WebApplicationContext가 생성되는 2가지 방식을 설명하시오.

웹 애플리케이션을 위한 ApplicationContext이다. ApplicationContext와 WebApplicationContext는 만드는 방식이 다르다. ApplicationContext는 자바의 main을 실행 즉 개발자가 시작을 할 수 있지만, WebApplicationContext는 개발자가 시작을 하지 못하고 톰캣이 시작하기 때문에 톰캣이 시작하면 그 때 이벤트를 받아 결과를 만들게 된다. WebApplicationContext이 생성되는 방식은 2가지가 있다. 밑의 2가지는 스프링 프레임 워크에서 제공이 된다. (이거 구지 안 써도 될 듯)

- ✓ ContextLoadListener에 의해 생성되는 WAC

  Persistence(DAO), Service관련 스프링 빈들을 등록하고, 웹 애플리케이션 전체에서 사용할 WAC 객체를 생성한다. root-context.xml에서 설정하면 된다.

- ✓ DispatcherServlet에 의해 생성되는 WAC

  ContextLoadListener에 의해 WAC가 생성되면 그것을 상속받아 DispatcherServlet이 WAC를 만든다. 컨트롤러와 같은 서블릿 관련 빈을 등록하고 해당 서블릿 마다 사용할 WAC 객체를 생성한다. 사용자가 요청을 하게 되면 이 DispatcherServlet에 의해 생성되는 WAC가 먼저 받게 된다. Servlet-context.xml 파일에서 설정을 한다.

DI ANNOTATION 설정 방법에서 <context:annotation-config />와 <context:component-scan base-package="명"/> 차이점을 설명하시오.

- ✓ @Autowired, @Resource, @Required를 이용할 때 선언한다. XML 파일에 이미 등록된 빈들의 애노테이션 기능을 적용하기 위해 선언한다. 컨테이너가 빈과 다른 빈과의 의존성을 자동으로 연결하도록 해주는 수단이다. 빈을 등록하기 위한 검색 기능은 없다. <context:component-scan base-package=""/> 태그가 선언되었을 경우는 생략 가능하다.

✓ @Component, @Service, @Repository, @Controller를 이용할 때 선언한다. 특정 패키지 안에 클래스를 검색해서 빈을 자동으로 찾아서 DI 컨테이너에 등록한다. 컨테이너가 인젝션을 위한 인스턴스(빈)을 설정하는 수단이다. 클래스 선언 앞에 @Component를 붙이면 컨테이너가 찾아서 관리하고 @Autowired가 붙은 인스턴스 변수(빈)에 주입시켜준다. 이 설정을 사용할 경우에는 <context:annotation-config />를 선언할 필요가 없다.

(뇨바기 pick) Dependency Injection 의존성 주입 방식 2가지인 생성자 기반 의존성 주입과 설정자 기반 의존성 주입에 대해 설명하여라

✓ 생성자의 인수를 사용해 의존성을 주입한다. 설정파일 XML에 <constructor-org> 태그를 사용하여 주입할 컴포넌트를 설정한다.

✓ 메서드 인수를 통해 의존성을 주입한다. 설정파일 XML에 <property> 요소의 name 속성에 주입할 컴포넌트의 이름을 설정

JAVA를 이용한 DI 설정 방식의 2가지 주요 애노테이션을 설명하시오.

✓ 빈 설정 메타 정보를 담고 있는 클래스를 선언할 때 쓰는 애노테이션이다. 해당 클래스가 스프링 설정으로 사용된다.

✓ 클래스 내의 메서드를 정의하여 새로운 빈 객체를 정의할 때 사용한다. name 속성을 사용하여 새로운 빈 이름을 적용 가능하다.

빈 객체 라이프 사이클에서 빈을 생성 후 초기화하는 방법과 빈 종료 전 전처리 과정을 수행할 수 있는 방법을 4가지를 정리하시오.

(Initialization) 빈 생성 후 초기화	XML 기반	<bean> 요소의 init-method 속성에 메서드 지정
	에너테이션 기반	@PostConstruct 애너테이션이 붙은 메서드
	자바 기반	@Bean의 initMethod 속성에 지정된 메서드
	인터페이스 구현	InitializeBean 인터페이스의 afterPropertiesSet 메서드
(Destruction) 빈 종료 전 전처리	XML 기반	<bean> 요소의 destroy-method 속성에 메서드 지정
	에너테이션 기반	@PreDestroy 에너테이션이 붙은 메서드
	자바 기반	@Bean의 destroyMethod 속성에 지정된 메서드
	인터페이스 기반	DisposableBean 인터페이스의 destroy 메서드

4장

단답형

데이터 액세스 처리를 비즈니스 로직 층에서 분리하는 층

데이터 액세스 처리에 특화된 오브젝트

데이터 취득과 변경에 데이터 처리를 DAO 오브젝트로 분리하는 패턴

자바의 데이터 액세스 기술 4개

데이터 액세스 기술 종류와 상관없이 데이터베이스 접속을 관리해주는 인터페이스

업무용 어플리케이션은 ( )에 의해 커넥션 오브젝트를 재사용한다.

우리가 프로젝트를 하면서 사용했던 DBCP(DB Connection Pool)은?

JDBC 이용의 문제점? ( )를 기술, 다양한 ( )을 파악하기 위한 코딩이 필요, 코드의 ( )가 어려움

스프링 JDBC는 ( )를 제공해 ( )를 단순화. JDBC를 직접 사용할 때 발생하는 장황한 코드를 ( ).

스프링 JDBC가 제공하는 중요 템플릿(클래스) 2개.

데이터 액세스 층, 데이터 액세스 오브젝트, DAO 패턴, JDBC Hybernate MyBatis JPA, 데이터 소스, 커넥션 풀, Apache Commons DBCP, 대량의 소스코드 에러 원인 일관성 유지, JDBC를 래핑한 API 소스코드 은닉, jdbcTemplate NamedParameterTemplate

메서드 명	설명
	하나의 결과 레코드 중에서 하나의 컬럼 값을 가져올 때 RowMapper와 함께 사용하면 하나의 레코드 정보를 객체에 매핑.
	하나의 결과 레코드 정보를 Map 형태로 매핑할 수 있다.
	여러 개의 Map 형태의 결과 레코드를 다룰 수 있음
	여러 개의 레코드를 객체로 변환하여 처리(ResultSetExtractor)
	데이터의 변경(INSERT, UPDATE, DELETE)을 실행할 때

SELECT 문을 도메인으로 변환할 경우 ( )를 가져올 때는 ( )메서드를 이용하고, ( )를 가져올 때는 ( )메서드를 이용

도메인으로 변환할 때 queryForObject 메서드에서 제2인수로 도메인 자동 변환을 위한 스프링 제공 클래스는?

BeanPropertyRowMapper를 사용할 경우 두개 어떤 명(이름)이 같아야 할까?

도메인으로 변환할 때 query 메서드에서 ( ) 인터페이스를 구현한 익명 클래스를 정의한다. 클래스 내의 ( ) 추상 메서드를 정의한다.

스프링 프레임워크에서 만든 클래스를 테스트하는 모듈. 단위 테스트, 통합 테스트를 지원하기 위한 매커니즘이나 편리한 기능을 제공.

POM.xml에 추가하는 스프링 테스트 프레임 웍 2개는?

queryForObject queryForMap queryForList query update, 한 레코드 queryForObject 여러 레코드 query, BeanPropertyRowMapper StudentVO(도메인 객체)의 프로퍼티 명 테이터베이스 테이블 컬럼 명, RowMapper mapRow(), 스프링 테스트, spring-test junit

서술형

(뇨바기pick)도메인으로 변환할 경우 SELECT문은 2가지의 메서드를 이용한다. 이 두가지 메서드를 말하고 설명하시오.

- ✓ queryForObject 메서드는 한 레코드를 가져올 때 사용한다. 제1, 3인수로 SELECT문과 SELECT문의 파라미터가 들어가고 제2인수로 도메인 자동 변환을 위한 스프링 제공 클래스 BeanPropertyRowMapper를 호출한다. 이 때는 StudentVO(도메인 객체)의 프로퍼티 명과 테이블 컬럼 명이 같아야 한다.
- ✓ query 메서드는 여러 레코드를 가져올 때 사용한다. RowMapper 인터페이스를 구현한 익명 클래스를 정의하고, 클래스내 mapRow() 추상 메서드를 정의한다.

5장

단답형

스프링 MVC는 (　　)에 기초한 MVC 프레임 워크이다.

모델, 뷰, 컨트롤러는 각각의 (　　)가 정의되어 있어 서로 구현에 (　　) 않아 (　　) 결합도로 구성되어 (　　), (　　) 쉽다.

자바 웹 어플리케이션 설계 방식으로 JSP만 사용하여 개발하거나 Java bean을 포함하여 개발하는 방식을 의미한다. JSP에 뷰와 비즈니스 로직이 혼재되어 복잡도가 높다. 따라서 유지보수가 어렵다.

자바 웹 어플리케이션 설계 방식으로 Model – View – Controller로 분리한다. 뷰와 비즈니스 로직의 부리로 유지보수성 및 확장이 용이하다.

클라이언트 요청을 별도의 프론트 컨트롤러에 집중. 모든 요청의 공통 부분을 별도(프론트 컨트롤러)로 처리하는 자바 웹 어플리케이션 설계 방식은?

스프링 MVC에서 프론트 컨트롤러는 결국 프론트 컨트롤러 = (　　)

프론트 컨트롤러 패턴, 인터페이스 의존적이지 약한 유연하고 확장하기, Model 1 방식, Model 2 방식, 프론트 컨트롤러 패턴, DispatcherServlet

프론트 컨트롤러 패턴 실행 프로세스

① (　　)이 HTTP 요청을 받는다.

② (　　)은 (　　)로 HTTP 요청 위임한다.

③ (　　) 클라이언트의 요청 처리를 위해 (　　)를 호출한다.

④ (　　)는 리소스를 엑세스하여 (　　)를 생성 후 요청 결과를 리턴한다.

⑤ (　　)은 처리 결과에 적합한 (　　)에 (　　)를 요청한다.

⑥ 선택된 (　　)는 화면에 (　　)를 가져와 (　　)을 처리한다.

⑦ HTTP (　　)이 이루어진다.

스프링 MVC의 주요 구성 요소 중 프론트 컨트롤러를 담당. 클라이언트의 요청을 받아서 Controller에게 클라이언트의 요청을 전달하고, 리턴한 결과값을 View에게 전달하여 알맞은 응답을 생성한다.

스프링 MVC의 주요 구성 요소 중 URL과 요청 정보를 기준으로 어떤 컨트롤러를 실행할지 결정하는 객체이다. DispatcherServlet은 하나 이상의 핸들러 매핑을 가질 수 있음. 애노테이션을 이용할 때에는 mvn:annotation-driven 태그를 설정해야 함.

스프링 MVC의 주요 구성 요소 중 클라이언트의 요청을 처리한 뒤 그 결과를 DispatcherServlet에게 알려줌

스프링 MVC의 주요 구성 요소 중 컨트롤러가 뷰에게 넘겨줄 데이터를 저장하기 위한 객체

스프링 MVC의 주요 구성 요소 중 Cotroller가 리턴한 뷰 이름을 기반으로 Controller 처리 결과를 생성할 뷰를 결정

스프링 MVC의 주요 구성 요소 중 Controller의 처리 결과 화면을 생성한다.

DispatcherServlet DispatcherServlet 서브 Controller 서브 Controller DAO 객체 Model DispatcherServlet 뷰 화면처리 뷰 모델 객체 화면 응답, DispatcherServlet, HandelerMapping, Controller, Model, ViewResolver, View

스프링 스테레오 타입 애노테이션 4개?

POM.xml에서 라이브러리를 설정할 때, (　　)모듈을 설정하면 스프링 웹과 기타 스프링 프레임워크 의존 모듈에 대한 의존 관계도 함께 처리한다.

POM.xml에서 라이브러리를 설정할 때, 스프링 MVC는 (　　) 구조체(　　)를 이용해 입력 값의 유효성을 검증한다.

웹 애플리케이션에 사용할 2가지 애플리케이션 컨텍스트 2가지와 등록은 어떤 파일에서 설정할까?

ContextLoadListener 클래스는 서비스 계층 이하의 빈((@　　), (@　　))을 등록하기 위한 클래스

DispatcherServlet 클래스는 컨트롤러((@　　), (@　　)) 빈을 등록하기 위한 클래스.

모든 컨트롤 클래스에는 @Controller 애노테이션을 설정. 메서드 별로는 (　　) 애노테이션을 사용하여 URL 매핑을 설정. (　　) 애노테이션은 HTTP 요청 파라미터를 메소드의 파라미터로 전달받을 때 사용. 메서드 반환 값으로는 (　　) 이름을 반환.

DispatcherServlet 클래스 설정에서 스프링 MVC를 이용하는데 필요한 컴포넌트 빈 정의가 자동으로 수행. 패키지 내부에서 찾은 빈(컨트롤러)과 URI를 맵핑.

DispatcherServlet 클래스 설정에서 base-package 내부의 클래스에서 @Controller로 지정된 컨트롤러를 검색하여 빈으로 등록

DispatcherServlet 클래스 설정에서 정적 리소스 파일 설정은 (　　)태그 사용.

@Component @Service @Controller @Repository, spring-webmvc, Bean Validation hibernate-balidator, ContextLoadListener DispatcherServlet web.xml, Service Repository, Controller Component, @RequestMapping @RequestParam View, <annotation-driven /> <context:component-scan base-package="org.kpu.web.Controller" />, resource

DispatcherServlet 클래스 설정에서 뷰 리졸브를 설정할 때, 뷰 객체 이름 생성을 위한 ( ), ( )를 지정한다.

URL과 컨트롤러 메서드의 메핑을 설정하는 애노테이션인 @RequestMapping의 속성으로 URL, HTTP 메서드, 요청 파라미터 유무나 파라미터 값이 있는데 3개 말해.

컨트롤러 메서드 매개변수 5개를 말해봐

매핑 설정 방식1, 2, 3, 4

( ) 값을 ( ) 적용 변수로 전달.

( ) 값을 ( ) 적용 변수로 전달. -> 이 때 페이지로 msg 전달 못해

( ) 값을 ( ) 적용 변수로 전달.

( ) 형태의 값을 지정할 수 있음. ( ) 값을 ( ) 적용 변수로 전달

Autowired 속성에서required=true이면 ( )를 한다.

@PostConstruct -> ( )하는 메서드 위에 달림.

Model은 내장 객체 -> model.( )(“students”, students); 이런 것을 ( )언어를 쓴다고 한다.

RequestMethod.POST에서 해당 폼에 있는 변수(이름)가 해당 도메인 객체의 멤버 변수와 ( ) 도메인 객체에 자동 저장됨. (POST 보내면). 그리고 @ModelAttribute(“student”) StudentVO vo 이것 때문에 모델 객체에도 ( ).

REST는 클라이언트와 서버 사이에 데이터 연동 애플리케이션을 위한 아키텍처 스타일. 웹 상의 정보를 리소스로 파악하고 그 식별자로서 URI를 할당해 고유하게 주소를 지정하는 방법. 이것은?

접두사 접미사, value method params, Model 오브젝트 @ModelAttribute @PathVariable @RequestParam @MatrixVariable @RequestBody, URL 경로 내의 변수 @PathVariable 요청 파라미터 @RequestParam 요청 파라미터 @ModelAttribute 배열 요청 파라미터 @ModelAttribute, 예외처리, 빈 생성 후 초기화, addAttribute 표현, 같은경우 자동저장, REST 아키텍처

REST 컨트롤러를 위한 라이브러리 설정 -> ( )를 사용하면 JSON과 자바빈즈를 교환할 수 있음.

REST 컨트롤러에 사용되는 애노테이션 중

REST API를 제공하는 컨트롤러를 의미하며 @Controller와 @ResponseBody를 합침.

컨트롤러 메서드 매개 변수에 ( )가 애노테이션된 경우, 스프링은 요청된 HTTP request body를 해당 매개 변수에 바인딩한다.

컨트롤러 메소드가 ( )로 어노테이션 된 경우, 스프링은 반환 값을 나가는 HTTP response body에 바인딩한다. 스프링은 요청된 메시지의 HTTP 헤더에 있는 Content-Type을 기반으로 HTTP Message converter를 사용하여 반환 값을 HTTP response body로 변환한다.

전체 HTTP 응답을 나타내며 sttusCode, headers, body 3가지 속성 값을 지정할 수 있다.

Spring MVC에서의 예외처리 방법 크게 보면 2가지 있는데 모지?

컨트롤러 별로 예외 처리에서 별도의 예외 처리 메서드를 정의하고 그 메서드에 ( ) 애노테이션 설정한다.

하나의 웹 애플리케이션 안에서 공통된 예외 처리에서 공통된 예외 처리 클래스를 정의하고 그 클래스에 ( ) 애노테이션을 설정

서술형

프론트 컨트롤러 패턴 실행 프로세스를 스프링 MVC의 주요 구성 요소를 통해 설명하시오.

프론트 컨트롤러는 DispatcherServlet을 의미한다.

① DispatcherServlet이 HTTP 요청을 받는다.

② DispatcherServlet은 서브 Controller로 HTTP 요청 위임을 받는다. 이 단계에서 DispatcherServeet은 서브 Controller를 스프링 빈으로 등록한다.

③ 서브 Controller는 클라이언트의 요청 처리를 위해 DAO 객체를 호출한다.

④ DAO 객체는 리소스를 엑세스하여 Model 객체를 생성 후 요청 결과를 리턴한다.

⑤ DispatcherServlet은 처리 결과에 적합한 뷰에 화면 처리 요청을 한다.

⑥ 선택된 뷰는 화면에 모델 객체를 가져와 화면을 처리한다.

⑦ 이과정을 통해 결국 HTTP요청에 응답을 한다.

(뇨바기 pick) 프론트 컨트롤러 패턴과 관련된 주요 구성 요소 6가지를 말하고 설명하여라.

- ✓ DispatcherServlet

  프론트 컨트롤러를 담당한다. 클라이언트의 요청을 받아서 Controller에게 클라이언트의 요청을 전달하고, 리턴한 결과값을 View에게 전달하여 알맞은 응답을 생성한다.

- ✓ HandlerMapping

  URL과 요청 정보를 기준으로 어떤 컨트롤러를 실행할지 결정하는 객체이다. DispatcherServlet은 하나 이상의 핸들러 매핑을 가질 수 있다. 애노테이션을 이용할 때에는 mvn:annotation-driven 태그를 설정해야 한다.

- ✓ Controller

  클라이언트의 요청을 처리한 뒤 그 결과를 DispatcherServelet에게 알려준다.

- ✓ Model

  컨트롤러가 뷰에게 넘겨줄 데이터를 저장하기 위한 객체이다.

- ✓ ViewResolver

  Controller가 리턴한 뷰 이름을 기반으로 Controller 처리 결과를 생성한 뷰를 결정한다.

- ✓ View

Controller의 처리 결과 화면을 생성한다.

@ModelAttribute, @PathVariable, @RequestParam 차이점을 설명하시오.

✓ @PathVariable

URL 경로 내의 변수 값을 @PathVariable이 적용된 변수로 전달된다. 변수를 페이지로 전달해 요청 페이지에서 변수를 사용할 수 있다.

✓ @RequestParam

요청 파라미터 값을 @RequestParam 적용 변수로 전달한다. 변수를 로컬 함수 내에서만 쓸 수 있다. 요청된 페이지로 변수를 전달하지 못한다.

✓ @ModelAttribute

요청 파라미터 값을 @ModelAttribute 적용 변수로 전달한다. 변수를 페이지로 전달해 요청 페이지에서 변수를 사용할 수 있다.

@Controller, @RequestMapping, @RequestParam, @RestController, @RequestBody, @ResponseBody 의미를 설명하시오.

✓ @Controller

모든 컨트롤러 클래스에 붙여지는 애노테이션이다.

✓ @RequestMapping

URL과 컨트롤러 메서드의 매핑을 설정하는 애노테이션이다.

✓ @RequestParam

HTTP 요청 파라미터를 메소드의 파라미터로 전달받을 때 사용하는 애노테이션이다.

✓ @RestController

REST API를 제공하는 컨트롤러를 의미하며 @Controller와 @ResponseBody를 합친 것을 의미한다.

✓ @RequestBody

컨트롤러 메서드 매개 변수에 @RequestBody가 애노테이션 된 경우, 스프링은 요청된 HTTP request body를 해당 매개 변수에 바인딩한다.

✓ @RequestBody

컨트롤러 메서드 매개 변수에 @RequestBody로 어노테이션 된 경우, 스프링은 반환 값을 나가는 HTTP response body에 바인딩한다. 스프링은 요청된 메시지의 HTTP 헤더에 있는 Content-Type을 기반으로 HTTP Message converter를 사용하여 반환 갑을 HTTP response body로 변환한다.