#스프링이 뭔지 간단히 설명해보세요

스프링은 자바 플랫폼을 위한 오픈소스 애플리케이션 프레임워크입니다. 자바 SE로 된 자바 객체 POJO를 자바 EE에 의존적이지 않게 연결해주는 역할을 합니다. 스프링의 특징으로는 크기와 부하 측면에서 경량 시킨 것과, IOC 기술로 애플리케이션의 느슨한 결합을 도모시킨 것이 있습니다.

#스프링이랑 스프링 부트는 차이점이 뭔가요?

스프링 부트는 스프링에서 사용하는 프로젝트를 간편하게 셋업할 수 있는 서브 프로젝트입니다. 독립 컨테이너에서 동작할 수 있기 때문에 임베디드 톰캣이 자동으로 실행되구요. 임베디드 컨테이너에서 애플리케이션을 실행시키기에는 다소 불안전해서 큰 프로젝트는 사용하지 않는 것이 좋습니다.

#MVC패턴이란?

MVC 패턴은 코드의 재사용에 유용하며, 사용자 인터페이스와 응용 프로그램 개발에 소요되는 시간을 줄여주는 효과적인 설계 방식을 말합니다.

구성요소로는 Model, View, Controller가 있는데요. 모델은 핵심적인 비즈니스 로직을 담당하여 데이터베이스를 관리하는 부분이고, 뷰는 사용자에게 보여주는 화면, 컨트롤러는 모델과 뷰 사이에서 정보 교환을 할 수 있도록 연결시켜주는 역할을 합니다.

#MVC1이랑 MVC2 패턴 차이에 대해 설명해주세요.

모델1은 JSP페이지 안에서 로직 처리를 위해 자바 코드가 함께 사용됩니다. 요청이 오면, 직접 자바빈이나 클래스를 이용해 작업을 처리하고, 이를 클라이언트에 출력해줍니다. 구조가 단순한 장점이 있지만, JSP 내에서 html 코드와 자바 코드가 같이 사용되면서 복잡해지고 유지보수가 어려운 단점이 있습니다.

모델2는 이와는 다르게 모든 처리를 JSP에서만 담당하는 것이 아니라 서블릿을 만들어 역할 분담을 하는 패턴입니다. 요청 결과를 출력해주는 뷰만 JSP가 담당하고, 흐름을 제어해주고 비즈니스 로직에 해당하는 컨트롤러의 역할을 서블릿이 담당하게 됩니다. 이처럼 역할을 분담하면서 유지보수가 용이해지는 장점이 있지만 습득하기 힘들고 구조가 복잡해지는 단점도 있습니다.

#스프링 MVC 구조 흐름에 대해 과정대로 설명해보세요.

우선, 디스패처 서블릿이 클라이언트로부터 요청을 받으면, 이를 요청할 핸들러 이름을 알기 위해 핸들러맵핑에게 물어봅니다.

핸들러맵핑은 요청 url을 보고 핸들러 이름을 디스패처 서블릿에게 알려줍니다. 이때 핸들러를 실행하기 전/후에 처리할 것들을 인터셉터로 만들어 줍니다.

디스패처 서블릿은 해당 핸들러에게 제어권을 넘겨주고, 이 핸들러는 응답에 필요한 서비스를 호출하고 렌더링해야 하는 뷰 이름을 판단하여 디스패처 서블릿에게 전송해줍니다.

디스패처 서블릿은 받은 뷰 이름을 뷰 리졸버에게 전달해 응답에 필요한 뷰를 만들라고 명령합니다.

이때 해당하는 뷰는 디스패처 서블릿에게 받은 모델과 컨트롤러를 활용해 원하는 응답을 생성해서 다시 보내줍니다.

디스패처 서블릿은 뷰로부터 받은 것을 클라이언트에게 응답해줍니다.

#스프링 필터랑 인터셉터의 차이점은 뭔가요?

필터와 인터셉터는 실행되는 시점에서 차이가 있습니다. 필터는 웹 애플리케이션에 등록을 하고, 인터셉터는 스프링의 context에 등록을 합니다. 따라서 컨트롤러에 들어가기 전 작업을 처리하기 위해 사용하는 공통점이 있지만, 호출되는 시점에서 차이가 존재합니다.

#IOC란?

IOC란, 인스턴스의 생성부터 소멸까지 개발자가 아닌 컨테이너가 대신 관리해주는 것을 말합니다. 인스턴스 생성의 제어를 서블릿과 같은 bean을 관리해주는 컨테이너가 관리합니다.

#Dispatcher-Servlet이란?

서블릿 컨테이너에서 HTTP 프로토콜을 통해 들어오는 모든 요청을 제일 앞에서 처리해주는 프론트 컨트롤러를 말합니다.

따라서 서버가 받기 전에, 공통처리 작업을 디스패처 서블릿이 처리해주고 적절한 세부 컨트롤러로 작업을 위임해줍니다.

디스패처 서블릿이 처리하는 url 패턴을 지정해줘야 하는데, 일반적으로는 .mvc와 같은 패턴으로 처리하라고 미리 지정해줍니다.

디스패처 서블릿으로 인해 web.xml이 가진 역할이 상당히 축소되었습니다. 기존에는 모든 서블릿을 url 매핑 활용을 위해 모두 web.xml에 등록해 주었지만, 디스패처 서블릿은 그 전에 모든 요청을 핸들링해주면서 작업을 편리하게 할 수 있도록 도와줍니다. 또한 이 서블릿을 통해 MVC를 사용할 수 있기 때문에 웹 개발 시 큰 장점을 가져다 줍니다.

#DI(Dependency Injection)란?

스프링 컨테이너가 지원하는 핵심 개념 중 하나로, 설정 파일을 통해 객체간의 의존관계를 설정하는 역할을 합니다.

각 클래스 사이에 필요로 하는 의존관계를 Bean 설정 정보 바탕으로 컨테이너가 자동으로 연결합니다.

객체는 직접 의존하고 있는 객체를 생성하거나 검색할 필요가 없으므로 코드 관리가 쉬워지는 장점이 있습니다.

#AOP(Aspect Oriented Programming)란?

공통의 관심 사항을 적용해서 발생하는 의존 관계의 복잡성과 코드 중복을 해소해줍니다.

각 클래스에서 공통 관심 사항을 구현한 모듈에 대한 의존관계를 갖기 보단, Aspect를 이용해 핵심 로직을 구현한 각 클래스에 공통 기능을 적용합니다.

간단한 설정만으로도 공통 기능을 여러 클래스에 적용할 수 있는 장점이 있으며 핵심 로직 코드를 수정하지 않고도 웹 애플리케이션의 보안, 로깅, 트랜잭션과 같은 공통 관심 사항을 AOP를 이용해 간단하게 적용할 수 있습니다.

#AOP 용어들을 설명해보세요

Advice : 언제 공통 관심기능을 핵심 로직에 적용할지 정의

Joinpoint : Advice를 적용이 가능한 지점을 의미 (before, after 등등)

Pointcut : Joinpoint의 부분집합으로, 실제로 Advice가 적용되는 Joinpoint를 나타냄

Weaving : Advice를 핵심 로직코드에 적용하는 것

Aspect : 여러 객체에 공통으로 적용되는 공통 관심 사항을 말함. 트랜잭션이나 보안 등이 Aspect의 좋은 예

#DAO(Data Access Object)란?

DB에 데이터를 조회하거나 조작하는 기능들을 전담합니다.

Mybatis를 이용할 때는, mapper.xml에 쿼리문을 작성하고 이를 mapper 클래스에서 받아와 DAO에게 넘겨주는 식으로 구현합니다.

#Annotation이란?

소스코드에 @어노테이션의 형태로 표현하며 클래스, 필드, 메소드의 선언부에 적용할 수 있는 특정기능이 부여된 표현법을 말합니다.

애플리케이션 규모가 커질수록, xml 환경설정이 매우 복잡해지는데 이러한 어려움을 개선시키기 위해 자바 파일에 어노테이션을 적용해서 개발자가 설정 파일 작업을 할 때 발생시키는 오류를 최소화해주는 역할을 합니다.

어노테이션 사용으로 소스 코드에 메타데이터를 보관할 수 있고, 컴파일 타임의 체크뿐 아니라 어노테이션 API를 사용해 코드 가독성도 높여줍니다.

@Controller : dispatcher-servlet.xml에서 bean 태그로 정의하는 것과 같음.
@RequestMapping : 특정 메소드에서 요청되는 URL과 매칭시키는 어노테이션
@Autowired : 자동으로 의존성 주입하기 위한 어노테이션
@Service : 비즈니스 로직 처리하는 서비스 클래스에 등록
@Repository : DAO에 등록

#Spring JDBC란?

데이터베이스 테이블과, 자바 객체 사이의 단순한 매핑을 간단한 설정을 통해 처리하는 것

기존의 JDBC에서는 구현하고 싶은 로직마다 필요한 SQL문이 모두 달랐고, 이에 필요한 Connection, PrepareStatement, ResultSet 등을 생성하고 Exception 처리도 모두 해야하는 번거러움이 존재했습니다.

Spring에서는 JDBC와 ORM 프레임워크를 직접 지원하기 때문에 따로 작성하지 않아도 모두 다 처리해주는 장점이 있습니다.

#MyBatis란?

객체, 데이터베이스, Mapper 자체를 독립적으로 작성하고, DTO에 해당하는 부분과 SQL 실행결과를 매핑해서 사용할 수 있도록 지원함

기존에는 DAO에 모두 SQL문이 자바 소스상에 위치했으나, MyBatis를 통해 SQL은 XML 설정 파일로 관리합니다.

설정파일로 분리하면, 수정할 때 설정파일만 건드리면 되므로 유지보수에 매우 좋습니다. 또한 매개변수나 리턴 타입으로 매핑되는 모든 DTO에 관련된 부분도 모두 설정파일에서 작업할 수 있는 장점이 있습니다.