Section 4. 서버 프로그램 구현

1. 서버 프로그램 구현

(1) 업무 프로세스 확인

① 업무 프로세스의 개념

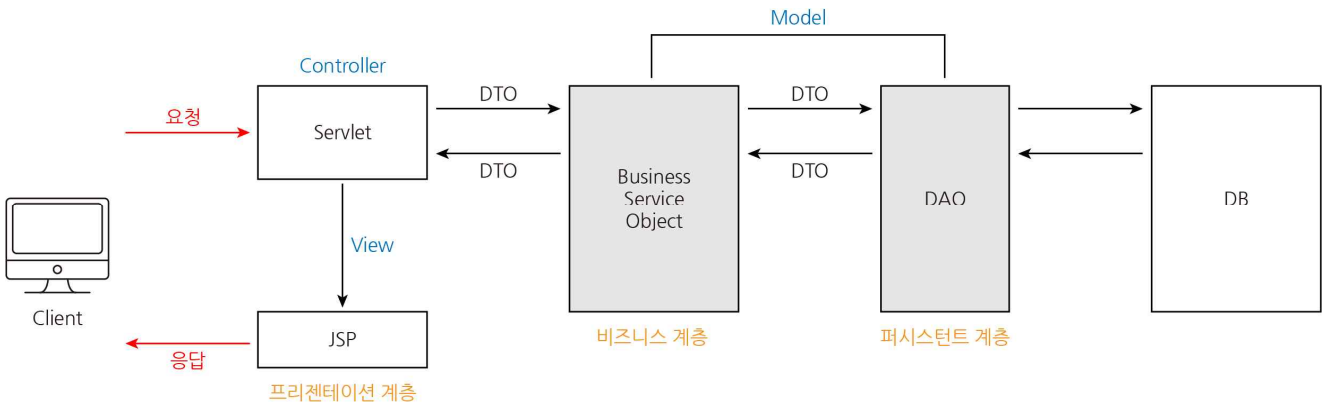
- 개인이나 조직이 한 개 이상의 자원 입력을 통해 가치 있는 산출물을 제공하는 활동

② 업무 프로세스 구성 요소

구성 요소	설명
프로세스 책임자	- 프로세스의 성과와 운영을 책임지는 구성원 - 프로세스를 설계하고 지속적으로 유지한다.
프로세스 맵	- 상위 프로세스와 하위 프로세스의 체계를 도식화하여 업무의 청사진 표현 - 구조적 분석 기법 : 자료 흐름도 - 객체지향 분석 기법 : FRD
프로세스 Task 정의서	- 결과물을 산출하기 위해 Task 들의 운영방법을 문서화한다.
프로세스 성과 지표	- 프로세스의 과정과 결과를 고객 입장에서 정량적으로 표현한 성과 지표
프로세스 조직	- 프로세스를 성공적으로 수행하기 위해 개인들의 업무를 유기적으로 수행하는 구성원
경영자의 리더십	- 경영자는 프로세스의 중요성을 인식한다. - 기업의 경영 방침을 확고하게 한다.

(2) 서버 프로그램 구현

- 업무 프로세스(BusinessLogic)를 기반으로 개발언어, 도구를 이용하여 서버에서 서비스를 제공하는데 필요한 기능을 구현하는 활동
- 서버 프로그램 구현 절차



- 구현 요소

구현 요소	설명
DTO (Data Transfer Object)	- 프로세스 사이에서 데이터를 전송하는 객체 - Getter, Setter 메서드만 포함한다.
VO (Value Object)	- 도메인에서 속성들을 묶어서 특정 값을 나타내는 객체 - DTO와 동일한 개념이나 차이점은 Read-Only 속성 객체이다.
DAO (Data Access Object)	- 실질적으로 DB에 접근하는 객체 - DataBase에 접근하기 위한 로직 & 비지니스 로직을 분리하기 위해 사용
Service	- DAO 캘래스를 호출하는 객체
Controller	- 비즈니스 로직을 수행하는 객체

(3) MVC 모델의 계층

① 프리젠테이션 계층(Presentation Layer) View

- 사용자 인터페이스
- 사용자가 선택할 수 있는 기능 및 부가정보를 전달할 양식을 표현한다.
- JSP 와 JSTL 태그 라이브러리를 결합하는 방식(JAVA의 경우)

② 제어 계층(Control Layer) Controller

- 프리젠테이션 계층과 비즈니스 로직 계층을 분리하기 위한 컨트롤러를 제공
- 어떤 요청이 들어왔을 때 어떤 로직이 처리해야 하는지를 결정한다.
- 사용자 요청을 검증하고 로직에 요청을 전달하는 일과 로직에서 전달된 응답을 적절한 뷰에 연결짓는 역할

③ 비즈니스 로직 계층(Business Logic Layer) Model

- 핵심 업무를 어떻게 처리하는지에 대한 방법을 구현 하는 계층
- 핵심 업무 로직의 구현과 그에 관련된 데이터의 적합성 검증, 트랜잭션 처리 등을 담당한다.

④ 퍼시스턴스 계층(Persistence Layer) Model

- 데이터 처리를 담당하는 계층
- 데이터의 생성/수정/삭제/선택(검색)과 같은 CRUD 연산을 수행한다.

⑤ 도메인 모델 계층(Domain Model Layer)

- 각 계층 사이에 전달되는 실질적인 비즈니스 객체
- 데이터 전송 객체(DTO) 형태로 개발자가 직접 제작해서 데이터를 넘기게 된다.

2. DBMS 접속기술

(1) DBMS 접속기술의 개념

- 프로그램에서 DB에 접근하여 DML을 사용 가능하게 하는 기술
- 프로그램이 DB를 사용할 수 있도록 연결해주는 인터페이스

(2) DBMS 접속기술 종류

① 소켓통신

- 응용프로그램과 DBMS가 주고 받는 통신
- DBMS 사에서 프로토콜을 공개하지 않기 때문에 개발은 어렵다.

② Vender API

- DBMS 사에서 공개한 API를 이용해 DBMS와 통신
- DBMS 사마다 API 사용법이 상이하다.

사실상 불가

③ JDBC(Java DataBase Connectivity)

- Java 에서 DB에 접속하고 SQL문을 수행할 때 사용되는 표준 API
- JDBC는 Java SE(Standard Edition)에 포함되어 있다.
- java.sql 패키지와 javax.sql 패키지에 포함되어 있다.
- 접속하려는 DBMS에 맞는 드라이버가 필요함

④ ODBC(Open DataBase Connectivity)

- 데이터베이스에 접근하기 위한 표준 규격
- 개발언어에 관계없이 사용할 수 있다.
- 모든 DBMS에 접근하는 방법을 통일시킴
- 1990년대 초 MS 사에서 개발됨

3. ORM(Object-Relational Mapping) 프레임워크

(1) ORM 프레임워크의 개념

- 객체와 관계형 데이터베이스의 데이터를 자동으로 매핑(연결)해주는 것
- 객체지향 프로그램에서 클래스를 생성하고, 관계형 데이터베이스의 테이블의 내용을 매핑
- 객체지향 프로그램을 통해 데이터베이스의 데이터를 다룬다.

(2) ORM 장/단점

- 장점
  - 비즈니스 로직에 더 집중할 수 있다.
  - 재사용 및 유지보수의 편리성이 증가한다.
  - DBMS에 대한 종속성이 줄어든다.
- 단점
  - 완벽한 ORM 으로만 서비스를 구현하기 어렵다.
  - 프로시저가 많은 시스템에선 ORM의 객체지향 장점을 활용하기 어렵다.

(3) 매핑 기술 비교

- SQL Mapper
  - SQL을 명시하여 단순히 필드를 매핑 시키는 것이 목적
  - SQL 문장으로 직접 데이터베이스 데이터를 다룬다.
  - SQL 의존적인 방법
  - 종류 : iBatis, Mybatis, jdbc Templetes 등
- OR Mapping (=ORM)
  - 객체를 통해 간접적으로 데이터베이스를 다룬다.
  - 객체와 관계형 데이터베이스의 데이터를 자동으로 매핑
  - ORM을 이용하면 SQL Query 가 아닌 직관적인 코드로 데이터를 조작할 수 있다.
  - 종류 : JPA(Java Persistent API),Hibernate

4. 시큐어 코딩 (Secure Coding)

(1) OWASP(The Open Web Application Security Project)

- 오픈소스 웹 애플리케이션 보안 프로젝트
- 주로 웹에 관한 정보 노출, 악성 파일 및 스크립트 보안 취약점 등을 연구하며 10대 취약점을 발표했다.
- OWASP Top 10
  - 웹 애플리케이션 취약점 중 빈도가 많이 발생하고, 보안상 영향을 줄 수 있는 10가지를 선정하여 발표

(2) 시큐어 코딩 가이드

① 시큐어 코딩 가이드의 개념

- 해킹 등 사이버 공격의 원인인 보안취약점을 제거해 안전한 소프트웨어를 개발하는 SW 개발 기법
- 개발자의 실수나 논리적 오류로 인해 발생할 수 있는 문제점을 사전에 차단하여 대응하고자 하는 것

② 시큐어 코딩 가이드 항목

- 입력 데이터 검증 및 표현
  - 프로그램 입력값에 대한 검증 누락 또는 부적절한 검증, 데이터형식을 잘못 지정하여 발생하는 보안 약점
  - 보안 약점 종류

종류	설명
SQL Injection	- SQL문을 삽입하여 DB로부터 정보를 열람 및 조작할 수 있는 공격
XSS (크로스 사이트 스크립트)	- 악의적인 스크립트를 포함해 사용자 측에서 실행되게 유도하는 공격
자원 삽입	- 외부 입력값이 시스템 자원 접근 경로 또는 자원 제어에 사용되는 공격
위험한 형식 파일 업로드	- 서버측에서 실행될 수 있는 스크립트파일을 업로드 하여 공격
명령 삽입	- 운영체제 명령어 삽입 - XQuery 삽입 - XPath 삽입 - LDAP 삽입
메모리 버퍼 오버프로	- 입력받는 값이 버퍼를 가득 채우다 못해 넘쳐흘러 버퍼 이후의 공간을 침범하는 공격

스택가드

- 보안 기능
  - 보안 기능을 부적절하게 구현하는 경우 발생할 수 있는 보안 약점
  - 보안 약점 종류

종류	설명
적절한 인증 없이 중요기능 허용	- 적절한 인증과정이 없이 중요정보(계좌, 개인정보 등)를 열람 할 때 발생하는 보안 약점
부적절한 인가	- 적절한 접근 제어 없이 외부 입력값을 포함한 문자열로 중요 자원에 접근할 수 있는 보안약점
취약한 암호화 알고리즘 사용	- DES, MD5 등 안전하지 않은 알고리즘 사용
하드코딩된 패스워드	- 소스 코드 내에 비밀번호가 하드코딩되어 있어 소스코드 유출시 노출되는 보안 약점
패스워드 평문 저장	- 계정 정보 탈취 시 패스워드 노출
취약한 패스워드 허용	- 비밀번호 조합규칙(영문, 숫자, 특수문자 등)이 미흡하거나 길이가 충분하지 않아 노출될 수 있는 보안약점

- 시간 및 상태
  - 동시 수행을 지원하는 병렬 시스템이나 하나 이상의 프로세스가 동작하는 환경에서 시간 및 상태를 부적절하게 관리하여 발생할 수 있는 보안 약점
  - 보안 약점 종류

종류	설명
경쟁 조건	- 동일 자원에 대한 검사 시점과 사용 시점이 상이하여 동기화 오류, 교착 상태를 유발
종료되지 않는 반복문 또는 재귀 함수	- 종료 조건이 없어 무한 루프에 빠져 자원 고갈을 유발

- 에러 처리
  - 에러를 처리하지 않거나 불충분하게 처리하여 에러 정보에 중요 정보가 포함될 때 발생할 수 있는 보안 약점
  - 보안 약점 종류

종류	설명
오류 메시지 정보 노출	- 응용 프로그램의 민감 정보가 오류 메시지를 통해 노출됨 - 오류 메시지는 사용자가 필요한 최소한의 정보만을 노출해야 한다.
오류 상황 대응 부재	- 예외처리 미구현
부적절한 예외 처리	- 프로그램 수행 중 발생한 예외 조건을 적절히 검사하지 않음

- 코드 오류
  - 개발자가 범할 수 있는 코딩 오류로 인해 유발되는 보안 약점
  - 보안 약점 종류

종류	설명
널 포인터 역참조	- 널 값을 고려하지 않은 코드에서 발생
부적절한 자원 해제	- 자원을 할당받아 사용한 뒤 반환하기 않는 코드에서 발생
해제된 자원 사용	- 해제한 메모리를 참조하는 코드에서 발생
초기화되지 않은 변수 사용	- 지역변수를 초기화하지 않고 사용하는 코드에서 발생

- 캡슐화
  - 중요한 데이터 또는 기능성을 불충분하게 캡슐화하거나 잘못 사용해 발생하는 보안 약점
  - 보안 약점 종류

종류	설명
잘못된 세션에 의한 정보 노출	- 멀티 스레드 환경에서 서로 다른 세션 간 데이터가 공유될 수 있음
제거되지 않은 디버그 코드	- 배포 단계에 디버그 코드가 남아 있는 경우 민감 정보가 노출될 수 있음
시스템 정보 노출	- 시스템 내부 데이터가 노출될 수 있음
잘못된 접근 지정자	- private, public 잘못된 접근지정자 사용으로 민감 정보 노출될 수 있음

- API 오용
  - 의도된 사용에 반하는 방법으로 API를 사용하거나 보안에 취약한 API를 사용하여 발생할 수 있는 보안 약점
  - 보안 약점 종류

종류	설명
DNS에 의존한 보안 결정	- 공격자가 DNS 정보를 변조하여 보안 결정을 우회 가능함
취약한 API 사용	- 금지되거나 안전하지 않은 함수를 사용함

Section 5. 배치 프로그램 구현

1. 배치 프로그램

(1) 배치의 개념

- 데이터를 일괄적으로 모아서 처리하는 대량의 작업을 처리
- 컴퓨터 흐름에 따라 순차적으로 자료를 처리하는 방식
- 배치 프로그램이란, 대량의 데이터를 모아 정기적으로 반복 처리하는 프로그램이다.

(2) 배치 프로그램의 필수 요소

- 대용량 데이터
  - 대용량의 데이터를 처리할 수 있어야 한다.
- 자동화
  - 심각한 오류 상황 외에는 사용자의 개입없이 동작해야 한다.
- 견고함
  - 비정상적인 동작 중단이 발생하지 않아야 한다.
- 안정성
  - 어떤 문제가 발생했을 때, 해당 문제를 추적하고 복구할 수 있어야 한다.
- 성능
  - 주어진 시간에 작업을 완료해야 하고, 다른 어플리케이션의 동작을 방해하지 않아야 한다.

(3) 스케줄 관리 종류

① 크론탭 (crontab)

- UNIX, LINUX 계열에서 사용
- 크론탭 형식

분 시 일 월 요일 명령어

- 항목의 범위

필드	의미	범위
첫 번째	분	0 ~ 59
두 번째	시	0 ~ 23
세 번째	일	1 ~ 31
네 번째	월	1 ~ 12
다섯 번째	요일	0 ~ 6 (0:일요일, 1:월요일)
여섯 번째	명령어	실행할 명령

- 허용 특수문자

특수문자	설명
*	모든 값 (매시, 매일, 매주)
?	특정 값이 아닌 어떤 값이든 상관 없음
-	범위를 지정할 때 (12-14 : 12시부터 14시)
,	여러 값을 지정할 때 (12, 14 : 12시, 14시)
/	증분값, 즉 초기값과 증가치 설정 (*/20 : 매 20분 마다)

- 설정 예

형식	설명
* * * * * 명령	매분 실행
30 4 * * 0 명령	매주 일요일 4시 30분 실행
10-30 4 * * * 명령	매일 오전 4시 10분부터 30분까지 매분 실행
0,10,20 * * * * 명령	매일 매시간 0분, 10분, 20분 실행
*/30 * * * * 명령	매 30분 마다 실행
30 0 1 1,6 * 명령	1월과 6월, 1일, 0시 30분에 실행

② Spring Batch

- 백엔드의 배치처리 기능을 구현하는데 사용하는 프레임워크
- 대용량 및 고성능 배치 작업을 가능하게 하는 고급 기술 서비스 및 기능을 제공
- 배치가 실패하여 작업 재시작을 하게 된다면 처음부터가 아닌 실패한 지점부터 실행
- Batch Job을 관리하지만 Job을 구동하거나 실행시키는 기능은 지원하진 않는다.
- Batch Job을 실행시키기 위해서는 Quartz, Scheduler, Jenkins등 전용 Scheduler를 사용
- 구성요소 : Job, Job Launcher, Step, Job Repository

③ Quartz Job Scheduler

- 표준 자바 프로그램으로 만들어진 작업을 지정된 일정에 따라 실행시키는데 사용하는 Java 패키지
- 특정한 시간에 특정한 작업을 한다든지 또는 주기적으로 실행해야 할 Java 애플리케이션을 사전에 정해 놓으면 일정에 따라 실행
- 주요 인터페이스 : Scheduler, Job, JobDetail, Trigger, JobBuilder, TriggerBuilder
- 형식

초 분 시 일 월 요일 년(생략가능)

(4) 배치 스케줄러 클래스 작성

① 배치 설계서 확인

② DTO/VO 구현

③ SQL 문 구현

④ DAO 구현

⑤ Schedule 구현 (매일 1시에 배치 실행)

```
@Scheduled(cron="0 0 1 * * ? ")
public void scheduleExecute(){
    String result = "SUCC";
    try{
        /* 수행해야 할 업무를 코딩한다 */
    }
    catch(Exception){
        result = "FAIL";
    }

    logger.info("스케줄 실행 결과 : " + result );
}
```