서버 프로그램 구현

chap1. 개발 환경 구축

- 개발환경 구축
  - 개발 환경 구성 시 구현 될 시스템 요구사항의 명확한 이해가 필요하다.
  - 개발 도구의 분류 (**빌구테형**)
    - 구현 도구
      - 개발자의 코드 작성과 디버깅, 수정 등과 같은 작업을 지원하는 도구
      - 프로그램을 개발할 때 가장 많이 사용
    - 테스트 도구
      - 코드의 기능 검증, 전체의 품질을 높이기 위해 사용하는 도구
      - 코드의 테스트, 테스트에 대한 계획, 수행 및 분석 등의 작업 가능
    - 형상관리 도구
      - 개발자들이 작성한 코드와 리소스 등 산출물에 대한 버전 관리를 위한 도구
      - 프로젝트 진행 시 필수로 포함되는 도구
    - 빌드 도구
      - 작성한 코드의 빌드 및 배포를 수행
      - 각각의 구성요소와 모듈에 대한 의존성 관리를 지원
    - 개발환경 구성요소
      - 서버 하드웨어 개발환경
        - 프로젝트 구성에 따라 웹 서버, 웹 애플리케이션 서버, 데이터베이스 서버, 파일 서버로 구분.
      - 클라이언트 하드웨어 개발 환경
        - 서버 개발환경에서 제공된 서비스를 사용하기 위해 UI 제공
        - 클라이언트 프로그램
          - 설치를 통해 사용자와 커뮤니케이션하는 프로그램
          - visual basic, C#, Delphi 등으로 개발
        - 웹 브라우저
          - 웹 서비스 형태로 서버에서 웹 애플리케이션 응답 시 브라우저를 통해 사용자와 커뮤니케이션
        - 모바일 앱
          - 모바일 디바이스에 설치되어 활용
        - 모바일 웹
          - 웹 브라우저와 동일한 형태로 모바일상 웹 브라우저로 서비스 제공
      - 소프트웨어 개발 환경
        - 개발환경 구성 분류 (**운미D**)
          - 운영체제
            - 서버의 하드웨어를 사용자 관점에서 편리,유용하게 사용하기 위한 소프트웨어
            - 프로젝트 성격에 따른 운체 사용
          - 미들웨어
            - 컴퓨터-컴퓨터 간 연결을 쉽고 안전하게 해줌.
            - 자바 기반 환경에서 JVM을 설치해 컨테이너로의 기능을 주로 이용

        - DBMS
            - 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해주고, 데이터베이스 관리해주는 소프트웨어
            - 데이터 저장 및 활용을 위해 DBMS설치
- 형상 관리 (Configuration Management)
    - 소프트웨어 개발을 위한 전체 과정에서 발생하는 모든 항목의 변경 사항을 관리하기 위한 활동
    - 목적
        - 생명주기 동안 제품의 무결성과 변경에 대한 추적성 확보
        - 변경 시 처리하는 메커니즘 제공
        - 형상 관리대상 파악, 베이스라인 지정, 형상 관리, 접근 제어
    - 형상 관리 절차 (**식통감기**)
        - 형상 **식별**
            - 형상 관리 대상을 정의 및 식별
            - 추적성 부여를 위한 id와 관리번호 부여
        - 형상 **통제**
            - 형상 항목 버전 관리를 위한 형상통제위원회 운영
            - 변경 요구 관리, 변경제어, 형상 관리 등 통제 지원
        - 형상 **감사**
            - 소프트웨어 베이스라인의 무결성 평가
        - 형상 **기록**
            - 소프트웨어 형상 및 변경관리에 대한 각종 수행 결과 기록
            - 형상 결과 보고서 작성
    - 소프트웨어 형상 관리 도구 유형 (**공클분**)
        - 공유 폴더 방식 (RCS, SCCS)
        - 클라이언트/서버 방식 (CVS, SVN)
        - 분산 저장소 방식(Git)
    - 소프트웨어 형상 관리 도구별 특징
        - CVS - 서버와 클라이언트로 구성, 다수 인원이 동시에 범용적 운영체제로 접근 가능한 형상 관리 도구
        - SVN - 하나의 서버에서 소슬르 쉽고 유용하게 관리할 수 있게 도와주는 도구
        - RCS - CVS와 달리 소스 파일 수정을 한 사람만으로 제한, 다수의 사람이 수정을 동시에 할 수 없도록 함.
        - Bitkeeper - SVN과 비슷한 중앙통제 방식, 대규모 프로젝트에서 빠른 속도 낼 수 있음.
        - Git - 속도에 중점을 둔 분산형 버전 관리 시스템. 커밋은 로컬에서, 푸시는 원격 저장소에 반영
        - Clear Case - 복수 서버, 복수 클라이언트 구조로 서버가 부족할 때 필요한 서버를 하나씩 추가하여 확자성을 기할 수 있음.
    - 소프트웨어 형상 관리 도구 사용 시 유의사항
        - 버전에 대한 쉬운 정보 접근성
        - 불필요한 사용자에 대한 접근 제어
        - 동일 프로젝트에 대한 동시 사용성
        - 빠른 오류 복구
    - 개발환경 구축 절차
        - 통합 개발 환경 설치

        - 형상 관리 도구 설치
        - 빌드 도구 설치

---

chap2. 공동 모듈 구현

- 공통 모듈 구현
    - 모듈의 개념
        - 그 자체로 하나의 완전한 기능을 수행할 수 있는 독립된 실체
        - 모듈화를 통해 분리된 시스템의 각 기능들로 서브프로그램, 서브 루틴, 소프트웨어 내의 단위 프로그램, 작업 단위 등과 같은 의미로 사용된다.
    - 모듈의 특징
        - 각각의 모듈은 상대적으로 독립성을 가지고 있다.
        - 모듈 내부에는 그 모듈을 하나로 통합하는 수많은 조합이 존재할 수 있다.
        - 단독으로 컴파일 가능하고, 재사용할 수 있다.
        - 독립성이 높은 모듈일수록 모듈 수정 시에도 다른 모듈들에는 영향을 거의 미치지 않고, 오류 발생 시에도 쉽게 해결할 수 있다.
        - 모듈의 독립성은 결합도와 응집도에 의해 측정되며, 독립성을 높이려면 **모듈의 결합도는 약하게(낮게), 응집도는 강하게(높게), 모듈의 크기는 작게**만들어야 한다.
    - 모듈화의 개념 및 기법
        - 소프트웨어의 성능을 향상, 복잡한 시스템의 수정, 재사용, 유지 관리 등이 용이하도록 기능 단위의 모듈로 분해하는 설계 및 구현 기법이다.
        - 모듈화 기법
            - 루틴
                - 소프트웨어에서 특정 동작을 수행하는 일련의 코드로 기능을 가진 명령들의 모임
            - 메인 루틴
                - 프로그램의 주요한 부분, 전체의 개략적인 동작 절차를 표시하도록 만들어진 루틴
                - 메인 루틴은 서브 루틴을 호출
            - 서브 루틴
                - 메인 루틴에 의해 필요할 때마다 호출되는 루틴
    - 공통 모듈 구현의 개념
        - 기능을 분할, 추상화하여 성능을 향상시키고 유지보수를 효과적으로 하기위한 공통 컴포넌트 구현 기법
        - 인터페이스 모듈, 데이터베이스 접근 모듈 등 필요한 공통 모듈을 구현한다.
        - 모듈 간 결합도를 줄이고, 응집도는 높인 공통 모듈 구현 권장

- 소프트웨어 모듈 응집도
    - 응집도 개념
        - 모듈의 독립성을 나타내는 정도로, 모듈 내부 구성요소 간 연관 정도
        - 하나의 모듈은 하나의 기능을 수행할수록 응집도가 높다.
    - 응집도 유형 (**우논시절 통순기**)
        - 우연적 응집도 - 모듈 내부의 각 구성요소가 연관이 없을 경우의 응집도
        - 논리적 응집도 - 유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들이 한 모듈에서 처리되는 경우의 응집도
        - 시간적 응집도 - 연관된 기능이라기보단 특정 시간에 처리되어야 하는 활동들을 한 모듈에서 처리할 경우 응집도

        - 절차적 응집도 - 모듈이 다수의 관련 기능을 가질 때 모듈 안의 구성요소들이 그 기능을 순차적으로 수행할 경우
        - 통신적 응집도 - 동일한 입력과 출력을 사용하여 다른 기능을 수행하는 활동들이 모여있을 경우의 응집도
        - 순차적 응집도 - 모듈 내 한 활동으로부터 나온 출력값을 다른 활동이 사용할 경우의 응집도
        - 기능적 응집도 - 모듈 내부의 모든 기능ㅇ ㅣ단일한 목적을 위해 수행되는 경우의 응집도
        - 우연적 -> 기능적 순으로 **응집도가 높다 (품질 좋다)**

- 소프트웨어 모듈 결합도

    - 결합도 개념
        - 모듈 내부가 아닌 외부의 모듈과읜 연관도 또는 모듈 간의 상호의존성
        - 소프트웨어 구조에서 모듈 간의 관련성을 측정하는 척도
    - 결합도 유형 (**내공 외제 스자**)
        - 내용 결합도 - 다른 모듈 내부에 있는 변수나 기능을 다른 모듈에서 사용하는 경우의 결합도
        - 공통 결합도 - 파라미터가 아닌 모듈 밖에 선언되어 있는 전역 변수를 참조, 전역 변수를 갱신하는 식으로 상호작용하는 경우
        - 외부 결합도 - 두 개의 모듈이 외부에서 도입된 데이터 포맷, 통신 프로토콜, 또는 디바이스 인터페이스를 공유할 경우의 결합도
        - 제어 결합도 - 단순처리 대상인 값만 전달되는게 아니라 어떻게 처리를 해야 한다는 제어 요소가 전달되는 경우의 결합도
        - 스탬프 결합도 - 모듈 간의 인터페이스로 배열이나 객체, 구조 등이 전달되는 경우의 결합도
        - 자료 결합도 - 모듈 간의 인터페이스로 전달되는 파라미터를 톧ㅇ해서만 모듈간의 상호 작용이 일어나는 경우의 결합도
        - 자료 --> 내용 결합도 순으로 **결합도가 높다 (낮은 품질)**

- 공통 모듈 구현 대상

    - 화면 모듈, 화면에서 입력받은 데이터 처리를 위한 서비스 컴포넌트, 비즈니스 트랜잭션 컴포넌트 등이 있다.
    - 통합 구현에서 공통 모듈 구현은 상세설계된 공통 모듈, 환경변수를 실제 프로그래밍 언어로 구현한다.

- 공통모듈 구현 절차에 MVC 패턴을 사용한다.

    - MVC 패턴 역할
        - 모델
            - 애플리케이션이 무엇을 할 것인지 정의
            - 내부 비즈니스 로직을 처리하기 위한 역할
        - 뷰
            - 화면에 무엇인가를 보여주기 위한 역할
            - 모델,컨트롤러가 보여주려고 하는 것들을 화면에 처리
        - 컨트롤러
            - 모델이 어떻게 처리할지 알려주는 역할
            - 뷰에 명령을 보내어 화면 요청 결과를 전달

- 팬인, 팬아웃

    - 모듈을 계층적으로 분석하기 위해 활용

- 공통 모듈 테스트

    - 공통 모듈 테스트 개념
        - IDE 도구를 활용하여 개별 공통 모듈에 대한 디버깅을 수행
        - 화이트박스 기법 활용
        - 대표적 단위테스트 도구 JUnit를 활용해 테스트 코드 구현
    - 공통 모듈 테스트 종류 (**화메화테**)
        - 화이트박스 테스트
        - 메서드 기반 테스트
        - 화면 기반 테스트
        - 테스트 드라이버/테스트 스텁

---

chap3. 서버 프로그램 구현

- 서버 프로그램 구현
    - 서버 프로그램 구현의 개념
        - 업무 프로세스를 기반으로 개발언어와 도구를 이용하여 서버에서 서비스 제공에 필요한 기능을 구현하는 활동
        - 웹 서비스 기반의 회원정보를 조회하고 등록하는 서버 프로그램을 차례대로 구현한다.
    - 서버 프로그램 구현 절차
        - 서버 프로그램은 백엔드와 프론트엔드를 구분하여 구현한다.
            - 백앤드 (**디스 다써클**)
                - DTO/VO구현 -> SQL문 구현 -> DAO구현 -> Service 구현 -> Controller 구현
    - 서버 프로그램 세부 구현
        - DTO (Data Transfer Object), VO (Value Object) 구현
            - 화면에서 전달받은 회원정보로 데이터베이스에 저장하는 객체를 구현한다.
        - SQL 문 구현
            - 데이터베이스 테이블 정의 및 생성
                - VO에서 정의한 객체 정보에 맞춰 정보가 저장될 테이블 정보를 생성한다.
            - Mybatis XML 구현
        - 데이터 접근 객체 (DAO Data Access Object) 구현
        - 서비스 클래스 구현
        - 컨트롤러 클래스 구현
        - 입/출력 검증 로직 구현

---

chap4. 배치 프로그램 구현

- 배치 프로그램
    - 배치 프로그램 개념
        - 배치 프로그램은 사용자와의 상호작용 없이 일련의 작업들을 작업 단위로 묶어 정기적으로 반복 수행하거나 정해진 규칙에 따라 일괄 처리하는 방법이다.
    - 배치 프로그램 유형 (**이온정**)
        - 이벤트 배치 - 사전에 정의해 둔 조건 충족 시 자동으로 실행
        - 온디맨드 배치 - 사용자의 명시적 요구가 있을 때마다 실행
        - 정기 배치 - 정해진 시점에 정기적으로 실행
    - 배치 스케줄러

        - 일괄 처리(Batch processing)을 위해 주기적으로 발생하거나 반복적으로 발생하는 작업을 지원하는 도구
        - 배치 스케줄러 종류
            - 스프링 배치 - 스프링 프레임워크의 DI, AOP, 서비스 추상화 등 스프링 프레임워크의 3대 요소를 모두 사용할 수 있는 대용량 처리를 제공하는 스케줄러 배치 애플리케이션
            - 쿼츠 스케줄러 - 스프링 프레임워크에 플러그인되어 수행하는 작업과 실행 스케줄을 정의하는 트리거를 분리하여 유연성을 제공하는 오픈소스 기반 스케줄러
        - Cron 표현식
            - 스케줄러를 실행시키기 위해 작업이 실행되는 시간 및 주기 등을 설정하게 되는데, 크론 표현식을 통해 배치 수행시간을 설정한다.
            - **초분시일 월요연**
                - 초, 분, 시간, 일, 월, 요일, 연도
            - 리눅스/유닉스 크론과 쿼츠 크론 차이는 초(seconds)의 유무.
            - **크론 표현식 특수문자 들어있는 식 해석 가능하도록 공부하기**
    - 배치 프로그램 설계
    - 배치 프로그램 작성
