4장 서버 프로그램 구현

062 개발 환경 구축

- 응용 소프트웨어 개발을 위해 개발 프로젝트를 이해하고 소프트웨어 및 하드웨어 장비를 구축하는 것을 의미

하드웨어 환경 – 사용자와의 인터페이스 역할을 하는 클라이언트, 클라이언트와 통신하여 서비스를 제공하는 서버로 구성

클라이언트 종류 : 개인용 컴퓨터 / 스마트폰

서버의 종류 : 웹 서버 / 웹 애플리케이션 서버 / 데이터베이스 서버 / 파일 서버

소프트웨어 환경 – 클라이언트와 서버 운영을 위한 시스템 소프트웨어와 개발에 사용되는 개발 소프트웨어로 구성

시스템 소프트웨어 종류 : 운영체제 / 웹 서버 및 WAS 운용을 위한 서버 프로그램 / DBMS

개발 소프트웨어 종류 : 요구사항 관리 도구 / 설계/모델링 도구 / 구현 도구 / 빌드 도구 / 테스트 도구 / 형상 관리 도구

웹 서버의 기능 – HTTP/HTTPS 지원 / 통신 기록 / 정적 파일 관리 / 대역폭 제한 / 가상 호스팅 / 인증

개발 언어 선정 기준 – 적정성 / 효율성 / 이식성 / 친밀성 / 범용성

063 소프트웨어 아키텍처

- 소프트웨어를 구성하는 요소들 간의 관계를 표현하는 시스템의 구조 또는 구조체

모듈화 – 소프트웨어의 성능 향상, 시스템의 수정 및 재사용, 유지 관리 등이 용이하도록 시스템의 기능들을 모듈 단위로 나누는 것

추상화 – 전체적이고 포괄적인 개념을 설계한 후 차례로 세분화하여 구체화시켜 나가는 것

추상화의 유형 – 과정 추상화 / 데이터 추상화 / 제어 추상화

단계적 분해 – 문제를 상위의 중요 개념으로부터 하위의 개념으로 구체화시키는 분할 기법

정보 은닉 – 한 모듈 내부에 포함된 절차와 자료들의 정보가 감추어져 다른 모듈이 접근하거나 변경하지 못하도록 하는 기법

소프트웨어 아키텍처의 품질 속성 – 품질 평가 요소의 종류

시스템 측면 – 성능, 보안, 가용성, 기능성, 사용성, 변경 용이성, 확장성

비즈니스 측면 – 시장 적시성, 비용과 혜택, 예상 시스템 수명, 목표 시장, 공개 일정

아키텍처 측면 – 개념적 무결성, 정확성, 완결성, 구축 가능성, 변경성, 시험성

소프트웨어 아키텍처의 설계 과정

설계 목표 설정 -> 시스템 타입 결정 -> 아키텍처 패턴 적용 -> 서브시스템 구체화 -> 검토

064 아키텍처 패턴

- 아키텍처를 설계할 때 참조할 수 있는 전형적인 해결 방식 또는 예제

레이어패턴 – 시스템을 계층으로 구분하여 구성하는 고전적인 방법의 패턴

클라이언트-서버 패턴 – 하나의 서버 컴포넌트와 다수의 클라이언트 컴포넌트로 구성되는 패턴

파이프-필터 패턴 – 데이터 스트림 절차의 각 단계를 필터로 캡슐화하여 파이프를 통해 전송하는 패턴

모델-뷰-컨트롤러 패턴 – 서브시스템을 모델, 뷰, 컨트롤러로 구조화하는 패턴

기타 패턴 – 마스터-슬레이브 패턴 / 브로커 패턴 / 피어-투-피어 패턴 / 이벤트-버스 패턴 / 블랙보드 패턴 / 인터프리터 패턴

065 객체지향

- 소프트웨어의 각 요소들을 객체로 만든 후, 객체들을 조립해서 소프트웨어를 개발하는 기법

객체지향의 구성 요소 : 객체 / 클래스 / 메시지

객체지향의 특징 : 캡슐화 / 상속 / 다형성 / 연관성

객체 – 데이터와 이를 처리하기 위한 함수를 묶어 놓은 소프트웨어 모듈

클래스 – 공통된 속성과 연산을 갖는 객체의 집합

메시지 – 객체들 간의 상호작용을 하는데 사용되는 수단

캡슐화 – 외부에서의 접근을 제한하기 위해 인터페이스를 제외한 세부 내용을 은닉하는 것

상속 – 상위 클래스의 모든 속성과 연산을 하위 클래스가 물려받는 것

다형성 – 하나의 메시지에 대해 각각의 객체가 가지고 있는 고유한 방법으로 응답할 수 있는 능력

연관성 – 두 개 이상의 객체들이 상호 참조하는 관계

Is member of / is instance of / is part of / is a

066 객체지향 분석 및 설계

객체지향 분석 – 사용자의 요구사항과 관련된 객체, 속성, 연산, 관계 등을 정의하여 모델링하는 작업

객체지향 분석의 방법론 – Rumbaugh 방법 / Booch 방법 / Jacobson 방법 / Coad와 Yourdon 방법 / Wirfs-Brock 방법

럼바우 분석 기법 – 모든 소프트웨어 구성 요소를 그래픽 표기법을 이용하여 모델링하는 기법

객체 / 동적 / 기능 모델링

객체 지향 설계 원칙 – 단일책임원칙 / 개방-폐쇄 원칙 / 리스코프 치환 원칙 / 인터페이스 분리 원칙 / 의존 역전 원칙

067 모듈

- 모듈화를 통해 분리된 시스템의 각 기능

결합도 – 모듈간의 상호 의존하는 정도

내용 결합도 – 공통 결합도 – 외부 결합도 – 제어 결합도 – 스탬프 결합도 – 자료 결합도

결합도 강함 <- -> 결합도 약함

응집도 – 기능적 응집도 – 순차적 응집도 – 교환적 응집도 – 절차적 응집도 – 시간적 응집도 – 논리적 응집도 – 우연적 응집도

응집도 강함 <- -> 응집도 약함

팬인 – 어떤 모듈을 제어하는 모듈의 수

팬 아웃 – 어떤 모듈에 의해 제어되는 모듈의 수

N-S 차트 – 논리의 기술에 중점을 두고 도형을 이용해 표현하는 방법

068 단위 모듈

- 소프트웨어 구현에 필요한 여러 동작 중 한 가지 동작을 수행하는 기능을 모듈로 구현하는 것

구현 과정

단위 기능 명세서 작성 -> 입출력 기능 구현 -> 알고리즘 구현

IPC – 모듈 간 통신 방식을 구현하기 위해 사용되는 대표적인 프로그래밍 인터페이스 집합

메소드 5가지

Shared Memory / Socket / Semaphores / Pipes&named Pipes / Message Queueing

단위 모듈 테스트 – 모듈이 정해진 기능을 정확히 수행하는 지 검증하는 것

테스트 케이스 – 소프트웨어가 사용자의 요구사항을 정확하게 준수했는지를 확인하기 위한 테스트 항목에 대한 명세서

식별자 / 테스트 항목 / 입력 명세 / 출력 명세 / 환경 설정 / 특수 절차 요구 / 의존성 기술

069 공통 모듈

- 여러 프로그램에서 공통으로 사용할 수 있는 모듈

명세 기법의 종류 – 정확성 / 명확성 / 완전성 / 일관성 / 추적성

재사용 – 이미 개발된 기능들을 새로운 시스템이나 기능 개발에 사용하기 적합하도록 최적화하는 작업

재사용 규모에 따른 분류 – 함수와 객체 / 컴포넌트 / 애플리케이션

070 코드

- 자료의 분류, 조합, 집계, 추출을 용이하게 하기 위해 사용하는 기호

코드의 주요 기능 – 식별 기능 / 분류 기능 / 배열 기능 / 표준화 기능 / 간소화 기능

코드의 종류 – 순차 코드 / 블록 코드 / 10진 코드 / 그룹 분류 코드 / 연상 코드 / 표의 숫자 코드 / 합성 코드

071 디자인 패턴

- 모듈 간의 관계 및 인터페이스 설계할 때 참조할 수 있는 전형적인 해결 방식 또는 예제

생성 패턴 – 추상 팩토리 / 빌더 / 팩터리 메소드 / 프로토타입 / 싱글톤

구조 패턴 – 어댑터 / 브리지 / 컴포지트 / 데코레이터 / 퍼싸드 / 플라이웨이트 / 프록시

행위 패턴 – 책임 연쇄 / 커맨드 / 인터프리터 / 반복자 / 중재자 / 메멘토 / 옵서버 / 상태 / 전략 / 템플릿 메소드 / 방문자

072 개발 지원 도구

통합 개발 환경 – 개발에 필요한 환경, 편집기, 컴파일러, 디버거 등의 다양한 툴을 하나의 인터페이스로 통합하여 제공하는 환경

통합 개발 환경 도구의 종류 – 이클립스 / 비주얼 스튜디오 / 엑스 코드 / 안드로이드 스튜디오 / IDEA

빌드 도구 – 소스 코드 파일들을 컴퓨터에서 실행할 수 있는 제품 소프트웨어로 변환하는 과정 또는 결과물

빌드 도구 종류 – Ant / Maven / Gradle

협업 도구 – 개발에 참여하는 사람들이 서로 다른 작업 환경에서 원활히 프로젝트를 수행할 수 있도록 도와주는 도구

073 서버 개발

- 웹 애플리케이션의 로직을 구현할 서버 프로그램을 제작하여 웹 애플리케이션 서버에 탑재하는 것

서버 개발 프레임 워크 – 다양한 네트워크 설정, 요청 및 응답 처리, 아키텍처 모델 구현 등을 손쉽게 처리할 수 있도록 클래스나 인터페이스를 제공하는 소프트웨어

종류 – Spring / Node.js / Django / Codeignter / Ruby on Rails

서버 개발 과정 – DTO/VO / SQL / DAO / Service / Controller를 각각 구현하는 과정