**48. 개발 환경 구축**

1. 개발 환경 구축의 개요 – 응용 SW 개발을 위해 SW및 HW 장비를 구축하는 것, 응용SW가 운영될 환경과 유사한 구조로 구축, 분석 단계의 산출물을 바탕으로 개발에 필요한 HW와 SW선정

2. 하드웨어 환경 – 사용자와의 인터페이스 역할을 하는 클라이언트, 서비스를 제공하는 서버로 구성

웹 서버(Web Server) : Client로부터 직접 요청을 받아 처리하는 서버, 저용량의 정적 파일 제공

웹 애플리케이션 서버(WAS) : 동적 서비스를 제공하기 위해 웹 서버로부터 요청을 받아 데이터 가공 작업 수행, 웹 서버와 파일 서버 사이에서 인터페이스 역할을 수행하는 서버

데이터베이스 서버 (DB Server) : DB와 이를 관리하는 DBMS를 운영하는 서버

파일 서버(File Server) : DB에 저장하기 비효율적이나 서비스 제공을 목적으로 유지하는 파일들을 저장,

웹 서버(Web Server)의 기능

HTTP/HTTPS 지원 : 브라우저로부터 요청을 받아 응답할 때 사용되는 프로토콜

통신 기록(Communication Log) : 처리한 요청들을 로그 파일로 기록하는 기능

정적 파일 관리(Managing Static Files) : HTML, CSS, 이미지 등의 정적 파일들을 저장하고 관리하는 기능

대역폭 제한(Bandwidth Throttling) : 네트워크 트래픽의 포화 방지를 위해 응답 속도를 제한하는 기능

가상 호스팅(Virtual Hosting) : 하나의 서버로 여러 개의 도메인 이름을 연결하는 기능

인증(Authentication) : 사용자가 합법적인 사용자인지를 확인하는 기능

3. 소프트웨어 환경 – Client와 서버 운영을 위한 시스템 SW와 개발에 사용되는 개발 SW로 구성

시스템 소프트웨어 : OS, 웹 서버 및 WAS 운영을 위한 서버 프로그램, DBMS

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

요구사항 관리 도구 : 요구사항의 수집과 분석, 추적 등을 편리하게 도와주는 소프트웨어

설계 모델링 도구 : UML(통합 모델링 언어)지원, 개발의 전 과정에서 설계 및 모델링을 도와주는 소프트웨어

구현 도구 : 개발 언어를 통해 애플리케이션의 실제 구현을 지원하는 소프트웨어

빌드 도구 : 구현 도구를 통해 작성된 소스의 빌드 및 배포, 라이브러리 관리를 지원하는 소프트웨어

테스트 도구 : 모듈들이 요구사항에 적합하게 구현되었는지 테스트하는 소프트웨어

형상 관리 도구 : 산출물들을 버전별로 관리하여 품질 향상을 지원하는 소프트웨어

개발 언어의 선정 기준

적정성(목적에 적합한지), 효율성, 이식성, 친밀성(개발자들의 높은 이해도와 활용도), 범용성(다른 개발 사례 존재)

**49. 모듈**

1. 모듈(Module)의 개요 – 모듈화를 통해 분리된 시스템의 각 기능, 서브루틴, 서브시스템, 작업단위

단독 컴파일 가능, 재사용 가능, 기능적 독립성은 모듈이 하나의 기능만을 수행하고 다른 모듈과의 과도한 상호작용을 배제함으로써 이루어짐, 독립성이 높을수록 오류를 쉽게 발견하고 해결 가능, 다른 모듈 영향 거의 없음

2. 결합도(Coupling) – 모듈간에 상호 의존하는 정도 또는 두 모듈 사이의 연관 관계를 의미

결합도가 약할수록 품질이 높음, 결합도가 강할수록 시스템 구현 및 유지보수 작업 어려움

(낮음) 자료 – 스탬프 – 제어 – 외부 – 공통 – 내용 (높음)

|  |  |
| --- | --- |
| 자료 결합도  (data Coupling) | 자료 요소로만 구성, 다른 모듈을 호출할 때 매개 변수나 인수로 데이터를 넘겨주고 처리결과를 돌려받음, 가장바람직함 |
| 스탬프 결합도  (검인, Stamp) | 배열이나 레코드 등의 자료 구조가 전달될 때의 결합도.  결합도  ↓  결합도↑  두 모듈이 동일한 자료구조 조회하는 경우의 결합도.  자료구조의 변화는 조회하는 모든 모듈에 영향을 미침 |
| 제어 결합도  (Control) | 어떤 모듈이 다른 모듈 내부의 논리 흐름을 제어하기 위해 신호를 보냄  처리 기능이 두 모듈에 분리되어 설계된 경우에 발생  하위 모듈에서 상위 모듈로 제어 신호가 이동하여 하위 모듈이 상위 모듈에게 처리 명령을 내린다 <-권리 전도 현상 |
| 외부 결합도  (External) | 어떤 모듈에서 선언한 데이터를 외부의 모듈에서 참조할 때의 결합도  참조되는 데이터의 범위를 각 모듈에서 제한가능 |
| 공통 결합도  (공유, Common) | 공유되는 공통 데이터 영역을 여러 모듈이 사용할 때의 결합도  데이터 내용 변경시에 이를 사용중인 모든 모듈에게 영향을 끼침 |
| 내용 결합도  (Content) | 한 모듈이 다른 모듈의 내부 기능 및 그 내부 자료를 직접 참조하거나 수정할 때의 결합도, 내부로 제어가 이동하는 경우에도 해당됨. |

2. 응집도(Cohesion)

정보 은닉 개념을 확장, 모듈이 독립적인 기능으로 정의되어 있는 정도를 의미한다.

응집도↑

응집도↓

|  |  |
| --- | --- |
| 기능(Functional) | 모듈 내부의 모든 기능 요소들이 단일 문제와 연관되어 수행될 경우 |
| 순차적(Sequential) | 모듈 내 하나의 활동으로부터 나온 출력 데이터를 그 다음 활동의 입력 데이터로 사용할 경우의 응집도 |
| 교환(통신)적  (Communication) | 동일한 입력과 출력을 사용하여 서로 다른 기능을 수행하는 구성 요소들이 모였을 경우의 응집도 |
| 절차적 응집도  (Procedural) | 모듈이 다수의 관련 기능을 가질 때 모듈 안의 구성 요소들이 그 기능을 순차적으로 수행할 경우의 응집도 |
| 시간적(Temporal) | 특정 시간에 처리되는 몇 개의 기능을 모아 하나의 모듈로 작성할 경우 |
| 논리적(Logical) | 유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들로 하나의 모듈이 형성되는 경우의 응집도 |
| 우연적(Coinciental) | 모듈 내부의 각 구성 요소들이 서로 관련 없는 요소로만 구성된 경우 |

팬인(Fan-In) – 어떤 모듈을 제어(호출)하는 모듈의 수, 팬인↑ 재사용↑, 장애발생위험↑, 테스트필요

팬아웃(Fan-Out) – 어떤 모듈에 의해 제어(호출)되는 모듈의 수

**50. 공통 모듈**

1. 개요

여러 프로그램에서 공통적으로 사용할 수 잇는 모듈을 의미 ex)자주 사용되는 계산식, 사용자인증

재사용성+중복 개발 회피를 위해 공통부분 식별, 명확한 기능 이해를 이해 명세 기법 준수↓

정확성(Correctness)/기능 필요성 정확히 작성 ,명확성(Clarity)/중의적 해석 방지, 완전성(Completeness)/구현에 필필요 모든 것 기술, 일관성(Consistency)/공통 기능 충돌방지, 추적성(Traceability)/요구사항의 출처 및 관계 파악

2. 재사용

비용, 개발시간 절약하기 위해 이미 개발된 기능들을 재구성하여 제작물에 최적화시키는 작업.

3. 효과적인 모듈 설계 방안

모듈의 독립성↑재사용성↑, 일관성 유지, 유지보수 용이, 예측 가능한 기능, 적당한 크기

51. DBMS 접속(Connection)

1. DBMS 접속의 개요 – 사용자가 데이터를 사용하기 위해 응용 시스템을 이용하여 DBMS에 접근하는 것

응용 시스템 : 사용자로부터 매개 변수를 전달받아 SQL 실행, 결과를 사용자한테 전달해 매개체 역할 수행

웹 응용 시스템 : 웹서버/WAS로 구성, 서비스 규모가 작을 시 서버를 통합해 하나의 서버만으로 운용 가능

2. DBMS 접속 기술 – DBMS에 접근하기 위해 사용하는 API, 프레임워크 등을 의미

JDBC(Java DataBase Connectivity)

Java언어로 다양한 종류의 DB에 접속하고 SQL문을 수행할 때 사용되는 표준 API

ODBC(Open DataBase Connectivity)

데이터베이스에 접근하기 위한 표준 개방형 API, 개발 언어에 관계 없이 사용할 수 있음

MyBatis – JDBC 코드를 단순화 해 사용할 수 있는 SQL Mapping 기반 오픈 소스 접속 프레임워크

다양한 메소드를 호출하고 해제해야 하는 JDBC와는 달리 이를 간소화 해 접속 기능을 더욱 강화

3. 동적 SQL(Dynamic SQL) – 다양한 조건에 따라 SQL 구문을 동적으로 변경하여 처리할 수 있는 처리 방식

**52. 서버 개발**

1. 서버 개발의 개요 – 웹 애플리케이션의 로직을 구현할 서버 프로그램 제작, WAS에 탑재하는 것을 의미

WAS에 구현된 서버 프로그램은 웹 서버로부터 받은 요청을 처리하여 결과를 반환하는 역할을 수행

각 프로그래밍 언어에는 해당 언어로 서버 프로그램을 개발할 수 있도록 지원하는 프레임워크가 있음

2. 서버 개발 프레임워크 – 서버 프로그램 개발 시 다양한 네트워크 설정, 요청 및 응답 처리, 아키텍처 모델 구현 등을 쉽게 처리할 수 있도록 클래스와 인터페이스를 제공하는 SW를 의미

Spring : Java 기반 프레임워크, 전자정부 표준 프레임워크의 기반 기술로 사용되고 있다

Node.js : JS를 기반 프레임워크, 비동기 입 출력 처리와 이벤트 위주의 높은 처리 성능, 입 출력 많은 앱에 적당

Django : Python 기반 프레임워크, 컴포넌트의 재사용과 플러그인화를 강조하여 신속한 개발 가능 지원

Codeigniter : PHP 기반 프레임워크, 인터페이스가 간편하며 서버 자원을 적게 사용

Ruby on Rails : Ruby 기반 프레임워크, 테스트를 위한 웹 서버를 지원하며 DB 작업을 단순화, 자동화 시켜 개발 코드의 길이가 짧아 신속한 개발이 가능

프레임워크의 특성

모듈화(Modularity) : 캡슐화를 통해 모듈화 강화, 설계 및 구현의 변경에 따른 영향 최소화, SW의 품질 향상

재사용성(Reusability) : 재사용 가능한 모듈들을 제공함으로써 개발자의 생산성을 향상

확장성(Extensibility) : 다형성을 통한 인터페이스 확장이 가능하여 다양한 형태와 기능을 가진 앱 개발 가능

제어의 역흐름(Inversion of Control) : 개발자가 관리, 통제해야 하는 객체 제어 권한을 넘겨 생산성 향상

53. 서버 개발 과정

1. 서버 개발 – DTO/VO, SQL, DAO, Service, Controller를 각각 구현하는 과정을 통해 이루어진다

구현 순서는 개발자가 임의로 변경

개발 서버 프로그램의 목적, 개발 언어, 규모 등의 이유로 통합되거나 세분화 될 수 있음

보든 과정에서 보안 약점이 발생하지 않도록 소프트웨어 개발 보안 가이드(정부 배포 지침)를 참고

2. DTO(Data Transfer Object)/VO(Value Object) 구현 - 데이터 교환을 위해 사용할 객체를 만드는 과정

알고리즘 등의 로직은 구현하지 않고, 변수와 데이터를 저장하고 반환하는 메소드만 구현

3. SQL 구현 – 데이터의 삽입, 변경, 삭제 등의 작업을 수행할 SQL문을 생성하는 과정

SQL문은 소스 코드 내에 직접 입력하거나 별도의 XML 파일로 저장하여 관리

XML 파일로 SQL문을 관리하는 경우 중복되는 SQL문을 최소화, 유지보수가 간편해짐

4. DAO(Data Access Object) 구현 – DB에 접근 후 SQL을 활용해 Data를 실제로 조작하는 코드를 구현하는 과정

5. Service 구현 – 사용자의 요청에 응답하기 위한 로직을 구현하는 과정

6. Controller 구현 – 사용자의 요청에 적절한 서비스를 호출, 그 결과를 사용자에게 반환하는 코드를 구현

**54. 배치 프로그램**

1. 배치 프로그램(Batch Program)의 개요 – 사용자와의 상호 작용 없이 여러 작업들을 미리 정해진 일련의 순서에 따라 일괄적으로 처리하는 것을 의미

정기 배치 : 일, 주, 월과 같이 정해진 기간에 정기적으로 수행

이벤트성 배치 : 특정 조건을 설정해두고 조건이 충족될 때만 수행

On-Demand 배치 : 사용자 요청 시 수행

배치 프로그램이 갖추어야 할 필수 요소

대용량 데이터 : 대량의 데이터를 가져오거나, 전달하거나, 계산하는 등의 처리가 가능해야 한다

자동화 : 심각한 오류가 발생하는 상황을 제외하고는 사용자의 개입 없이 수행되어야 한다

견고성 : 잘못된 데이터나 데이터 중복 등의 상황으로 중단되는 일 없이 수행되어야 한다

안정성/신뢰성 : 오류가 발생하면 오류의 발생 위치, 시간 등을 추적할 수 있어야 함

성능 : 다른 응용 프로그램의 수행을 방해하지 않아야 하고, 지정된 시간 내에 처리가 완료되어야 한다.

2. 배치 스케줄러(Batch Scheduler) – 일괄 처리 작업이 설정된 주기에 맞춰 자동으로 수행되도록 지원하는 도구

특정 업무를 원하는 시간에 처리할 수 있도록 지원한다는 특성 때문에 잡 스케줄러(Job Scheduler)라고도 함

스프링 배치(Spring Batch) : Spring framework의특성을 그대로 가져와 스프링이 가지고 있는 다양한 기능 사용

데이터베이스나 파일의 데이터를 교환하는데 필요한 컴포넌트들을 제공

로그 관리, 추적, 트랜잭션 관리, 작업 처리 통계, 작업 재시작 등의 다양한 기능 제공

스프링 배치의 주요 구성 요소와 역할

Job : 수행할 작업 정의 / Job Launcher : 실행을 위한 인터페이스 / Step : Job 처리를 위한 제어 정보

Job Repository : Step 제어 정보를 포함하여 작업 실행을 위한 모든 정보 저장

Quartz – 스프링 프레임워크로 개발되는 응용 프로그램들의 일괄 처리를 위한 다양한 기능 제공 오픈 소스 lib

수행할 작업과 수행 시간 관리하는 요소들을 분리하여 일괄 처리 작업에 유연성 제공

Quartz의 주요 구성 요소와 역할

Scheduler : 실행 환경 관리 / Job : 수행할 작업 정의 / JobDetail : Job의 상세 정보 / Trigger : 실행 스케쥴 정의

Cron – 리눗스의 스케줄러 도구로 crontab 명령어를 통해 작업을 예약할 수 있다

-e : 편집기(Editor)를 호출하여 작업 추가 및 수정, -l : 작업 목록 출력 , -r : 작업 삭제