1. Java의 특징 및 장/단점

-자바 : 객체지향 언어

-> 객체 : class TV의 여러 변수와 메소드 = TV 틀

TV틀을 통해 만든 TV = 객체

new를 통해 객체를 실체화 = 인스턴스

-자바 VM(Vritual Machine/가상머신)(JVM)

-자바가 생성한 바이트코드를 가상머신을 통해 해석/실행

-운영체제에 상관없이 동일한 프로그램을 개발/실행할 수 있도록 해줌

-자바 Application은 운영체제(OS)에 독립적이지만 JVM에 종속적임

-자바 플랫폼

-자바 플랫폼? 자바 프로그램이 실행되는 특정 환경

-ex) Java SE(Standard Edition), Java ME(Micro Edition), Java EE(Enterprise Edition),
Java Card, Java TV 등

=> 개발시 Java EE 사용했음. 이유? servlet-api 사용,
app server와 n-tier구조의 application 개발을 하려면 Java EE로 개발

-JDK(Java Development Kit) : 자바 프로그램 개발에 필요한 구성요소

-JRE(Java Runtime Environment) : 자바 프로그램 실행에 필요한 구성요소

=> 1.7 사용. 이유?

-API(Application Programming Interface) : 자바 응용 프로그램 개발에 사용하는 라이브러리

-자바의 장점

-객체지향 언어(OOP(Object Oriented Programming) Language)

-상속, 캡슐화, 다형성

-캡슐화 : 필요한 속성(Attribute)과 행위(Method)를 하나로 묶고 그중 일부를 외부에서 사용하지 못하도록 은닉하는 것

-> 사용자는 각 객체를 사용만 하고 세부 내용은 알 필요가 X

-접근제어자를 이용한 캡슐화

-다형성 : 조상타임의 참조변수로 자식타입의 인스턴스를 참조할 수 있다.

-인터페이스의 다형성 : 인터페이스 타입의 참조변수로 이를 구현한 클래스의 인스턴스를 참조할 수 있다.

-재사용성과 유지보수의 용이성 (객체지향언어의 특징)

-자동 메모리 관리(GC : Garbage Collection)

-멀티쓰레드(Multi-thread) 지원 : 여러 thread에 대한 스케쥴링(scheduling)을 자바 interpretor가 담당

-동적로딩(Dynamic Loading) 지원 : 모든 클래스를 로딩하지 않고 필요한 시점에 클래스를 로딩하여 사용

-자바의 단점

-자바는 실행 시에 해석(interpret)하기 때문에 속도가 느림

-> 바이트코드를 하드웨어의 기계어로 변환해주는 JIT 컴파일러와 Hotspot(향상된 최적화 기술)과 같은 기술 적용으로 JVM 기능이 향상되어 속도문제 개선

2. HTML5 (HTML : Hyper Text Markup Language)

-W3C

-표준화 기구와 업체 간 협력을 통해 제정된 차세대 웹 표준

-HTML5 마크업 영역 : 웹 문서의 구조 정의

-CSS3 : 문서의 디자인과 스타일 표현

-JavaScript API 확장 영역 : 다양한 동적/상호작용 기능 제공

-Markup = tag

-HTML

- 웹 페이지상에서 문단, 제목, 표, 이미지, 동영상 등을 정의하고 그 구조와 의미를 부여하는 마크업 언어
- 웹 문서의 구조 정의
- Markup = tag

-CSS

- 배경색, 폰트, 컨텐츠의 레이아웃 등을 지정하여, HTML 컨텐츠를 꾸며주는 스타일 규칙 언어
- 문서의 디자인과 스타일 표현

-JavaScript

- 웹 페이지와 상호작용 하도록 만들어진 스크립트 언어
- 동적으로 컨텐츠를 바꾸고, 멀티미디어를 다루며, 움직이는 이미지 등 웹페이지를 꾸며주도록 하는 프로그래밍 언어
- 특징
  - 객체 기반의 언어
    - -객체 : 속성과 메소드를 갖는 것
    - -사용자 정의 객체 : 사용자가 정의하여 사용
    - -내장 객체 : 기본적으로 제공되는 객체
    - -브라우저 객체 : 브라우저 지원을 이용하는 객체
  - 인터프리터 언어 : 클라이언트의 웹 브라우저에 의해 해석/실행
  - HTML 문서 내에 기술되며 HTML 문서와 함께 수행됨
- 구성요소
  - 객체 : 사용자 정의 객체, 내장 객체, 브라우저 객체
  - 문법 : 자료형, 변수, 제어문, 함수
  - 이벤트와 이벤트핸들러
- jQuery : JavaScript library
  - -JavaScript와 Asynchronous JavaScript + XML (Ajax) 프로그래밍 단순화
  - -장점
    - 크로스 브라우저 지원
    - 다양한 플로그인 지원
    - CSS Selector 등
- Ajax(Asynchronous Javascript And XML)
  - -웹서버와 비동기적으로 데이터를 교환하고 조작하기 위한 XML, XSLT, XMLHttpRequest
  - -비동기? 클라이언트가 서버에 요청을 보낸 후 응답여부와 상관없이 프로그램 작동
    - -> 먼저 요청한 것에 대한 콜백함수가 먼저 실행되지 않음
  - -XML? Extensible Markup Language = 확장가능한 표시 언어
  - -장점
    - 새로운 컨텐츠를 보여주기 위해 웹페이지를 리로딩 하지 않아도 됨
    - 서버처리를 기다리지 않고 비동기 요청 가능
    - 수신하는 데이터 양을 줄일 수 있으며, 클라이언트에게 처리 위임 가능
  - -단점
    - 요청 남발시 서버에 부하
    - Cross-Domain 문제 (동일-출처 정책으로 다른 도메인과는 통신 불가)
    - Ajax를 사용할 수 없는 브라우저에 대한 문제
    - 페이지 이동 없는 통신으로 인한 보안상의 문제
    - 지원하는 Charset 한정

3. JSP(Java Server Pages)

-HTML내에 자바코드를 삽입하여 웹서버에서 동적으로 웹페이지를 생성하고 웹브라우저에 돌려주는 언어

-Java EE 스펙 중 일부로 웹 어플리케이션 서버에서 동작

-JSP는 실행시 Java Servlet으로 변환된 후 실행되므로 서블릿과 유사함, but HTML 표준에 따라 작성되므로 웹 디자인이 편리

-자바가 제공하는 모든 기능을 자유롭게 사용할 수 있으면서 플랫폼(OS)에 영향을 주지 않음

-JSP spec 2.0 : EL(Expression Language) 사용 가능 -> Java코드를 사용해야 했던 곳에서 EL 사용 가능

-EL -> JSP 스코프(page,request,session,application)에 저장된 어떤 bean이라도 EL 변수로 사용 가능

-> 내장변수 지원

4. Spring Framework

*Spring Framework

-정의

-자바 플랫폼을 위한 오픈소스 어플리케이션 프레임워크

-자바 엔터프라이즈 개발을 편하게 해주는 오픈소스 경량급 애플리케이션 프레임워크

-자바 객체를 담고 있는 경량 컨테이너 : 자바 객체의 생성, 소멸과 같은 라이프사이클을 관리하며, 스프링으로부터 필요한 객체를 가져와 사용 할 수 있음

-자바 개발을 위한 프레임워크로 종속 객체를 생성해주고 조립해주는 도구

-POJO(Plain Old Java Object) Bean Container

-스프링 컨테이너에 저장되는 자바 객체는 특정한 인터페이스를 구현하거나 클래스를 상속받지 않아도 됨. -> 기존에 작성한 코드를 수정할 필요 없이 스프링에서 사용 가능

-POJO(Plain Old Java Object)

-EJB이전의 방식으로 돌아가 자바 개발자가 사용하기 편하고 간단한 Object를 만들어 사용하자는 개념

-필수요소 : light-weight(possibly), flexible, simple, supported by separate optional components such as hiberate or spring

-spring : POJO기반의 프레임워크

=> POJO를 이용한 애플리케이션 개발이 가진 특징과 장점을 살리면서, EJB에서 제공하는 엔터프라이즈 서비스와 기술을 그대로 사용할 수 있도록 도와주는 프레임워크

-특징

-크기와 부하의 측면에서 경량

-DI 패턴 지원

-DI(Dependency Injection) : 스프링은 설정파일을 통해 객체간의 의존관계를 설정할 수 있도록 하기 때문에, 객체는 직접 의존하고 있는 객체를 생성하거나 검색할 필요 없음

-제어의 역행(IOC)를 통해 어플리케이션의 느슨한 결합을 도모

-IOC(Inversion Of Control) : 스프링 프레임워크가 객체의 생성 및 생명주기 관리

-관점지향 프로그래밍(AOP)을 위한 풍부한 지원

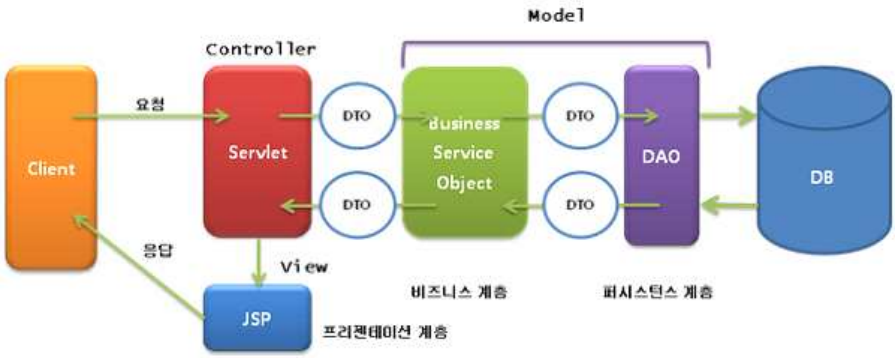
-AOP(Aspect Oriented Programming) : 공통 관심 사항(cross-cutting concern)을 Aspect를 이용하여 핵심 로직을 구현한 각 클래스에 공통 기능 적용

*Aspect 설정

1) pom.xml에 aspectj 추가
2) Aspect 파일 작성
3) xml에 bean 등록.....

-어플리케이션 객체의 생명주기와 설정을 포함한다고 관리한다는 점에서 일종의 컨테이너

*MVC(Model – View – Controller) Pattern



-Model, View, Controller로 로직이 분리되어 유지보수에 용이

-인터페이스와 비즈니스 로직을 분리하여 웹 개발을 하는 것을 가장 큰 장점으로 함

-Model : Application의 정보, 즉 데이터 나타냄

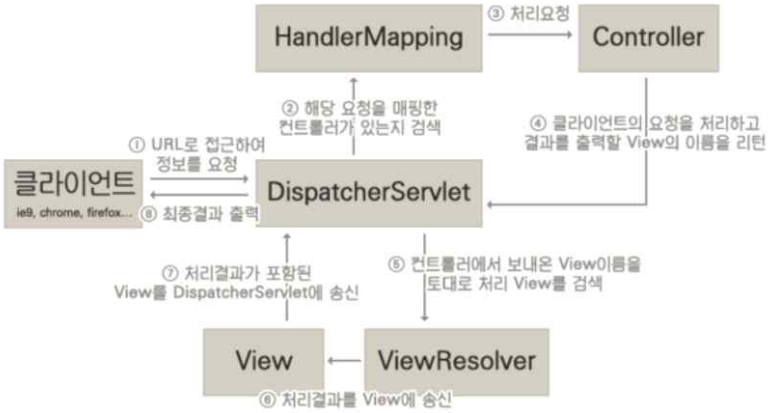
-View : 사용자 인터페이스, 즉 화면을 나타냄. 자바 웹 애플리케이션에서는 JSP 의미

-Controller : 비즈니스 로직과 모델의 상호동작 조정 역할. MVC2에서는 Servlet이 Controller 역할

-DispatcherServlet : FrontController

-Dispatcher = 분배하다, 배치하다, 보급하다

-Servlet Container에서 HTTP프로토콜을 통해 들어오는 모든 요청을 Presentation계층의 제일 앞에서 중앙집중식으로 처리해주는 컨트롤러



*Spring Bean Scope

-singleton = default scope

-prototype : 어플리케이션 요청시마다 새 인스턴스 생성

-request : HTTP요청별로 인스턴스화. 요청 종료시 소멸

-session : HTTP 세션별로 인스턴스화. 세션 종료시 소멸

-global session : 포틀릿 기반의 웹 어플리케이션 용도

-thread : 새 thread에서 요청시 새로운 bean 인스턴스 생성.

-custom : org.springframework.beans.factory.config.Scope를 구현하고 custom scope를 스프링의 스프링의 설정에 등록하여 사용

-설정방법

1) XML설정파일 -> bean 정의시 scope 명시적 지정

<bean id="normalBean" class="com.java.SomeClass" scope="singleton"/>

2) Annotation사용 -> @Scope("singleton")

5. slf4j

-Simple Logging Facade for Java

-Java 로깅 모듈의 추상체

-다양한 로깅 모듈을 변경해 사용 가능

-slf4j와 log4j 연결해 사용했음

6. log4j

-Java 기반의 로깅 유틸리티로 Apache에서 만든 오픈소스 라이브러리

-시스템 성능에 큰 영향을 미치지 않으면서, 옵션 설정을 통해 다양한 로깅 방법 제공

-설정방법 : pom.xml에 Log4j 추가 -> log4j.xml파일 변경을 통해 설정

-Tomcat 7.0 -> Servlet 3.0을 지원

*WebSocket 사용 조건

-Spring 4.0.1 이상

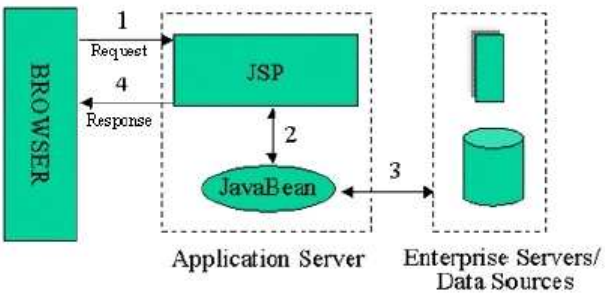
-Servlet 3.0 이상

-EgovFramework 3.6.0

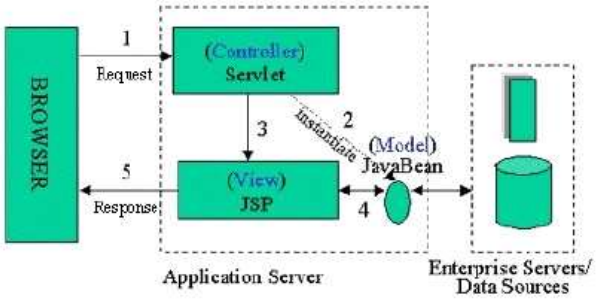
-jdk 1.7 이상

7. Model 1 & Model 2 개발 방식

7-1) Model 1 개발방식



7-2) Model 2 개발방식



-Model

-Application Logic 담당 부분 -> Database나 Legacy System과의 로직 담당

-Application으로부터 UI가 분리된 영역

-View

-사용자가 직접 사용하는 부분으로 Presentation로직을 담당

-Controller와 Model에 의해 생성된 결과물을 보여주는 역할

-Controller

-Business Logic 담당 부분

-사용자의 요청을 받아 요청에 해당하는 작업을 한 후 작업 결과에 따라 응답을 결정하는 역할

-Model과 View 사이에서 데이터를 전달하는 역할

8. iBATIS

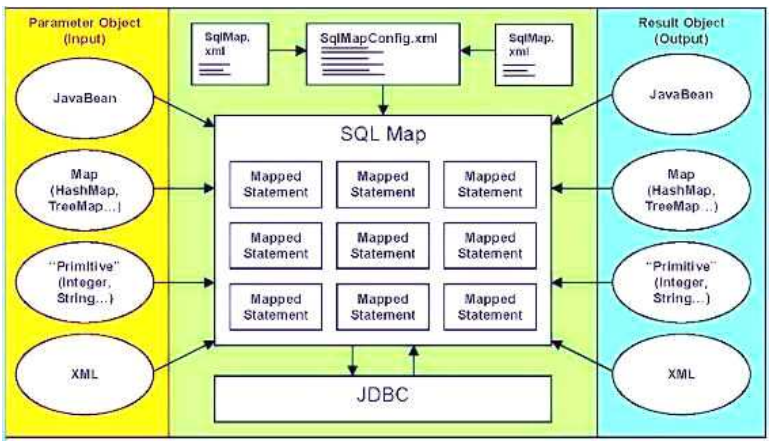
-SQLdp 기반한 데이터베이스와 자바, 닷넷, 루비 등을 연결시켜주는 역할을 하는 Persistence Framework

-iBATIS Data Mapper Framework

-xml을 사용해 Java를 SQL Statement에 Mapping

-iBATIS Data Mapper API -> PreparedStatement 파라미터와 ResultSets로 쉽게 Mapping 하도록 함

-ibatis-common.jar, ibatis-sqlmap.jar, ibatis-dao.jar는 Spring Framework에서 지원



*JDBC(Java Database Connectivity) : 자바에서 데이터베이스에 접속할 수 있도록 하는 자바 API

-단계

1) 드라이버 로드 (Class.forName("oracle.jdbc.driver.OracleDriver");)
2) Database 연결 (Connection conn = Drivermanager.getConnection("db주소","ID","PW");)
3) SQL을 위한 Statement객체 생성 (Statement stmt = conn.createStatement();)
4) SQL문장 실행 (
stmt.execute("SQL문");
ResultSet result = stmt.getResultSet();
)
5) 질의결과 ResultSet처리 (
Result result = stmt.executeQuery("SQL문");
while(result.next()){String str = result.getString(1);}
)
6) JDBC 연결 해제 (
result.close();
stmt.close();
conn.close();
)

8. Tiles Framework

-웹페이지의 상단/하단과 같이 반복적으로 사용되는 부분들에 대한 코드를 분리해 관리 가능
-view를 모듈화하여 필요한 view를 동적으로 배치
-Composite View 패턴 : 모듈 단위의 뷰들을 조합해 하나의 뷰 구성 => 트리구조(구현체=leaf)
-> 화면이 바뀌어도 중복되는 부분의 코드를 줄일 수 있고, 일관성을 높여 유지보수와 제어를 쉽게 한다.
-변하는 부분만 변경됨 -> 나머지 부분은 재사용 가능 = 재사용성 ↑
-런타임시에도 설정 변경 가능

9. Bootstrap

-웹사이트를 쉽게 만들 수 있도록 도와주는 HTML, CSS, JS 프레임워크

10. WebSocket

-ws:// => 컴퓨터 네트워크용 통신규약의 하나
-W3C(World Wide Web Consortium)와 IETF(Internet Engineering Task Force)가 Web server와 Web browser간의 통신을 위한 규정을 정의한 쌍방향통신(Duplex)용 기술 규약
-서버와 클라이언트 간에 Socket Connection을 유지해 언제든 양방향 통신 또는 데이터 전송이 가능하도록 하는 기술
-Real-time web application 구현에 주로 사용됨
(Real-time web application : 서버/클라이언트 쪽 데이터가 실시간으로 업데이트 되는 웹 어플리케이션)
-HTTP통신은 Request/Response 기반의 Stateless protocol
-> 서버-클라이언트 간 연결이 영구적이지 않으며, Request시에만 서버에서 Response를 하는 방식의 프로토콜 (클라이언트 쪽에서만 대화 시작 가능한 단방향통신)
-WebSocket Handshake -> WebSocket 초기화
-기존의 통신방법과의 차이
-프로토콜 : WebSocket 독자의 프로토콜로 접속 이루어짐
-overhead가 적음(header가 작음)
-장시간 접속을 전제로 함 -> 접속한 상태라면 클라이언트/서버로부터 데이터 송신 가능
-하나의 커넥션으로 데이터 송수신 가능
-URL 형태 : ws://www.sample.com

11. Server-Sent Events(SSE)
    -전통적인 HTTP를 통해 통신하므로 WebSocket과 달리 프로토콜이 필요하지 않음
    -HTML과 Javascript만으로 구현 가능
    -대부분의 브라우저 지원
    -푸쉬 알림 구현시 많이 이용
    -서버가 클라이언트에게 단방향 채널로 데이터를 보낼 수 있음(클라이언트가 메시지를 구독)

12. RMI(Remote Method Invocation)
    -원격 인터페이스 안에 정의된 리모트 객체의 메소드를 호출하는 것
        *원격 객체(Remote Object)
         : 다른 호스트에 있는 자바VM으로부터 호출되어지는 메소드를 가지고 있는 객체
        *원격 인터페이스(Remote Interface)
         : 원격 객체의 메소드를 선언하는 자바 인터페이스
        *원격 메소드 호출(Remote Method Invocation)
         : 원격 인터페이스 안에 정의된 리모트 객체의 메소드를 호출하는 것
    -분산되어있는 객체 간의 메시지 교환 가능

13. WYSIWYG 에디터
    -WYSIWYG = What You See Is What You Get
             = 화면에 포맷된 낱말, 문장이 출력물과 동일하게 나오는 방식
    -CKEditor
        -장점
            -다양한 기능 구현 가능
            -높은 브라우저 호환성
        -단점
            -다양한 기능으로 인한 복잡한 UI
            -투박한 UI
    -그 외 WYSIWYG Editor 종류 : Redactor, Summernote, TinyMCE, Trix

14. oracle 10g, 11g 차이
    -oracle 11g xe(express) 버전
    -기존의 9i 버전의 I = internet
    -10g,11g의 g = Grid Computing
        (Grid Computing : 원거리 통신망으로 연결된 서로 다른 기종의 컴퓨터를 묶어 가상의 대용량 고성능 컴퓨터를 구성하여 고도의 연산 혹은 대용량 연산을 구행하는 것)

15. sqldeveloper
    -SQL 명령어를 GUI(Graphical User Interface) 환경에서 작업할 수 있도록 도와주는 도구

16. starUML
    -UML(Unified Modeling Language) : 객체지향 프로그램 설계를 표현하기 위해 사용하는 표준설계 언어
        -소프트웨어 개발 과정에서 산출되는 산출물들을 명시, 개발, 문서화하기 위한 모델링 언어
        -시각적인 모델링 언어 제공
    -UseCaseDiagram
        -시스템이 제공하는 기능과 이용자의 관계 표현
        -사용자의 시각에서 시스템의 범위와 기능 정의
        -Actor(행위자)가 어떤 기능을 사용할 수 있는지 보여줌
        -Use Case : 사용자 입장에서 바라본 시스템의 특성을 설명한 구조로서, 행위자(actor) 즉, 사람, 시간의 흐름, 또는 다른 시스템에 의해 개시되는 시나리오 집합의 형태를 갖추고 있음
        -Actor(행위자)
            -시스템 외부에 존재하며 시스템과 교류/상호작용 하는 것
            -시스템이 서비스 해주기를 요청하는 존재
            -시스템에서 정보를 제공하는 대상
            -사용자 액터 : 시스템이 제공하는 기능인 유스케이스의 권한을 가지는 대상/역할

ex) |학생(Actor)| -> |도서구매(Use Case)|

-시스템 액터 : 사용자액터가 사용한 유스케이스를 처리해주는 외부의 시스템

ex) |도서구매(Use Case)| --> |도서주문시스템(Actor)|

-Association(연관) : 행위자(Actor)와 유스케이스(Use Case)간의 관계

-Generalization(일반화)

-행위자와 행위자, 유스케이스와 유스케이스 사이의 정의

-상속의 특성

-is-a 관계

-두 개체가 일반화 관계에 있음을 의미

-ex) 즉시이체 ◁- 자동이체 (자동이체는 즉시이체의 구체적은 유스케이스 개념)

-Include(포함)

-한 유스케이스가 자신의 서비스 수행 도중 다른 유스케이스의 서비스 사용이 필요할 때 정의

=> 서비스는 반드시 사용이 되어야 함

ex) 사용자 – 로그인 --<<include>>--> 통합인증

-Extend(확장)

-한 유스케이스가 다른 유스케이스의 서비스 수행을 요청. but, 선택적 유스케이스 관계

=> 서비스가 수행되지 않을 수 있음

ex) 고객목록조회 <--<<extend>>-- 고객상세조회

-클래스다이어그램(클래스 명세와 클래스 간의 관계 표현),
시퀀스다이어그램(인스턴스 간의 상호작용을 시계열로 표현) 작성 가능

17. ERwin Data Modeler

-데이터 모델링(Data Modeling) : 관리하고자 하는 정보를 표현하기 위해 데이터구조를 논리적으로 묘사하기 위해 사용되는 도구

-요소

-엔티티 : 정보로 관리되어야 하는 식별 가능한 사람, 장소, 사물, 개념, 사건

-속성 : 어떤 것이 갖는 고유한 특성이나 성격, 성질

-관계 : 하나 혹은 두 개의 어떤 것 간에 업무와 관련된 상황

-ERD : 말로서 되어 있는 업무요구사항을 그림으로 그려내어 관계를 표현한 것으로 엔티티타입과 엔티티간의 관계를 이해하기 쉽게 나타낸 것

18. PL, AA, TA, UA, BA, DA 각 역할

-PL(Project Leader) : 적절한 업무분담 및 프로젝트 계획 총괄, 프로젝트 전체 흐름 파악

-AA(Application Architect) : 애플리케이션에 대한 표준가이드 및 아키텍처 구조 설계 담당

-TA(Technical Architect) : 프로젝트 전반에 대한 하드웨어 및 네트워크 아키텍처 설계

-> 네트워크 구성 및 OS 설치. WEB/WAS/DB 설치

-UA(User-interface Architect) : 화면 설계, 화면 설정 담당(BootStrap)

-BA(Business Architect) : 업무 프로세스 설계, 업무수행에 활용할 새로운 정보 정의

-> 신규 정보시스템의 상세 요구명세서 창출

-DA(Data Architect) : DB설계자 -> 데이터 표준 관리, 데이터 구조 관리, 데이터 품질 관리

19. 프로젝트 산출물 각 단계 및 정의

-프로젝트 수행계획서

-주제 선정 이유

-필요 기술(API) 명명

-프로젝트 상세 내용 및 진행 계획 작성

-요구사항정의서

-고객의 요구사항을 세분화하여 작성

-각 요구사항 및 기술 구현에 소요되는 시간 설정

-유스케이스 다이어그램

-사용자의 입장에서 시스템의 사용을 그림 형태로 작성

-시스템의 기능 중심

-단위업무정의서

-각 기능을 단위업무로 나누어 상세화하고 배분

-프로세스정의서
	-시스템의 각 기능을 단위프로세스로 세분화 하여 각 단위프로세스에 대한 흐름 상세 설명
-프로세스 흐름도
	-시스템의 기능에 따른 흐름을 도식화
-메뉴구성도
	-각 기능을 실제 사용자가 보게 되는 화면의 depth에 따라 구성 및 작성
-화면정의서
	-화면의 흐름 및 이동에 따라 각 기능에 대한 설명과 함께 ppt로 작성
	-프로세스 정의서에 작성된 모든 기능이 포함
-DB설계
	-단어사전 - 요구분석에서 명사만 추출
	-용어사전 – 단어사전의 단어 기반으로 각 테이블의 엔티티명 속성명 결정 및 정의
	-개념적 설계
		-요구분석에서 나온 결과를 개념적으로 모델링
		-E-R다이어그램 형태로 표현(Entity-Relationship Diagram)(개체-관계 다이어그램)
		-개념스키마, 트랜잭션 모델링, E-R 모델
	-논리적 설계
		-데이터 모델링
		-현실 세계에서 발생하는 자료를 컴퓨터가 이해하고 처리할 수 있는 물리적 저장장치에 저장할 수 있도록 변환하기 위해 특정 DBMS가 지원하는 논리적구조로 변환시키는 과정
		-목표 DBMS에 맞는 논리 스키마 설계, 트랜잭션 인터페이스 설계
	-물리적 설계
		-데이터 구조화
		-논리적 구조로 표현된 데이터를 디스크 등의 물리적 저장장치에 저장할 수 있는 물리적 구조의 데이터로 변환하는 과정
		-목표 DBMS에 맞는 물리적 구조의 데이터로 변환
	-테이블명세서
		-테이블의 속성명 및 설명, 자료형, 식별자 등의 특징 작성
-프로그램 설계
	-프로그램 목록
	-테이블 관계도
	-클래스 다이어그램
		-시스템의 논리적인 구조(클래스)를 표현
		-시스템의 요구사항에 표현된 작업에 대한 책임 분할
	-프로그램 사양서
-테스트
	-단위테스트
		-각 화면마다 수행되어야 하는 기능들의 정상 작동 여부 테스트
	-통합테스트
		-프로세스흐름도에 따라 진행했을 때 정상 작동 여부 테스트
-간트차트
-사용자매뉴얼, 관리자매뉴얼