1. 애플리케이션 배포환경 구성하기
2. 애플리케이션 소스 검증하기
3. 애플리케이션 빌드하기
4. 애플리케이션 배포하기

★애플리케이션 배포환경 구성하기

● 소스코드 빌드과정
빌드: 소스 코드를 실행하 ㄹ수 있는 상태로 변환하는 과정
1. 컴파일 언어(C, C++ 등)
기계어로 바로 변환되어 실행되기 때문에 가장 속도가 빠르고 보안에 유리.
소스 변경시 마다 컴파일 과정을 통해서 빌드 작업 수행, 빌드과정이 오래 걸림.
빌드 과정은
전처리기-> 파싱-> 번역-> 어셈블리-> 링킹 과정을 통해 진행.

2. Byte코드 언어(Java, C#등)
컴파일의 결과물이 실행파일이 아닌 'class'라는 바이트 코드 파일로 생성,
가상 실행환경인 JRE(Java Runtime Enviroment), CLI(Common Language Infrastructure)에서 한줄씩 실행하는 방식으로 빌드.
JRE, CLI환경에서 실행될 때 기계어로 변환되며, 컴파일 언어에 비해 빌드과정이 빠름

3. 인터프리터언어(Javascript, Python, Ruby등)
컴파일 언어와 다르게 한 줄씩 번역.
컴파일하는 과정에서 메모리가 훨씬 적게 소모되고 빠른 시간에 컴파일을 진행할 수 있다.

●애플리케이션 배포(Release)환경
개발자 또는 사용자가 애플리케이션을 실행, 테스트할 수 있도록 컴파일된 프로그램, 실행에 필요한 리소스를 서버상의 적합한 위치로 이동하는 작업

1. 웹 서버
사용자의 http요청을 받아, 웹컨테이너에 요청을 전달하고, 결과값을 받아와 사용자에게 전송하는 역할 수행.
애플리케이션 배포 시 이미지, 단순 HTML과 같은 **정적인 리소스를 배포, 정적 리소스**를 빠르고 안정적으로 처리한다.

2. WAS(Web Application Server)

사용자의 요청을 받아 **동적인 처리**를 수행하는 프로그램 실행부분을 배포.

WAS구성 및 운용을 위한 국제 표준 규격을 정하고 있으며, 제품마다 배포 방식과 설정이 일부 다르다.

웹 애플리케이션 경우, UI배포영역(JSP, Servlet등)과 Biz배포영역(EJB, POJO서비스 등)으로 구분되어 있다.

●애플리케이션 배포단위

단순히 컴파일된 실행파일 또는 byte code를 복사하는 방식 이 외에도 다양한 단위로 묶음, 패키징을 통해서 배포할 수 있다.

Java 의경우 jar, war, ear등의 방식으로 패키징 하여 배포할 수 있다.

1. jar(Java Archive)

Java 라이브러리, 리소스, property 파일들을 포함.

프로그램에서 참조하는 라이브러리, 구현된 비즈니스 서비스를 배포할 때 jar단위로 패키징, 배포

2. war(Web Archive)

웹 컨테이너에 배포되는 배포형식으로 Servlet, jar파일과 WEB-INF폴더에 있는 web.xml파일로 구성.

웹 컨테이너상에 배포되어 독립적인 UI를 단 웹 애플리케이션 서비스를 제공할 수 있다.

3. ear(Enterprise Archive)

jar와 war을 묶어서 하나의 완성된 웹 애플리케이션 서비스를 제공할 수 있다.

●형상관리(Configuration Management)시스템

서비스 제공대상 형상항목을 식별하여 **기준선(Baseline)**을 설정

형상항목 변경과정에서 점검, 검증 등의 체계적인 통제를 통해 형상항목 간의 **일관성과 추적성**을 확보하기 위한 시스템

소프트웨어의 개발 및 운영/유지, 보수에 필요한 문서관리, 변경관리, 버전관리, 배포관리 및 작업 산출물에 대한 형상관리를 포함.

범위

신규 프로젝트 및 보완개발, 업무시스템의 운영/유지, 보수, 전산설비 및 시스템 소프트웨어 등과 관련된 작업, 사용자 파일 관리 등

●형상관리 시스템에서 사용하는 용어

1. 형상관리 = 소프트웨어의 전체 생명 주기

2. 형상항목 : 형상관리 대상이 되는 항목 , 유일한 식별자가 부여되어 개별적으로 관리되는 소스파일, 문서, 기타사항 등

3. 기준선 : 공식적으로 검토되고 협의되어 향후 기준이 되는 형상항목의 집합체

4. 마이그레이션: 개발 완료된 시스템이 운영 단계로 전환될 때, 관련 소스파일을 저장공간(Repository)로 **이관**시키는 작업

5. 리포지터리: 관리 대상을 형상관리 시스템으로 일괄 전송하여 압축, 암호화한 후에 저장, 관리하는 **저장공간**을 의미. 업무 또는 디렉터리 단위.

6. 워크플로: 형상관리 활동을 수행하기 위해 **미리 정해진 절차가 형상관리 시스템 안에 구현되어 있는 것**을 의미.

7. 반출(Check out)

형상항목을 변경하기 위해 형상 리포지터리로 부터 **전송받는 것**을 의미.

반출된 형상항목에 대해서는 잠금 상태가 유지.

8. 반입(Check in)

반출된 형상항목을 변경 후 다시 **형상 리포지터리로 전송하는 것** 의미.

반입 시 버전관리는 자동적으로 이루어짐.

<반입/반출 햇갈리니까 다시 확인>

●수행순서

[1] 프로젝트의 배포관리 요건을 분석하고 배포환경 구성 방안을 계획

1. 프로젝트의 애플리케이션 기술적인 특성 확인, 배포방식 정의

(1)개발언어 확인

(2)개발도구, 개발환경 확인

(3)유형 확인

(4)배포대상 리소스의 유형 확인

배포정의서를 통해 배포단위 확인 / 배포대상 리소스의 유형 확인

(5)배포시 연계하여 처리하고자 하는 프로젝트 요건 확인

2. 개발하려는 애플리케이션의 인프라 환경을 확인하고 배포대상 서버를 확인

(1)개발하려는 애플리케이션의 서버구성환경을 호가인

(2)리소스 유형별 배포대상서버를 확인

(3)분산 배포대상을 확인

(4)통합 빌드 배포 구성도 작성

[2]애플리케이션 배포를 위한 도구와 시스템을 선정하고 설치.

1. 애플리케이션 빌드, 배포를 위해 적합한 도구를 선정한다.

(1) 소스 형상관리 도구를 선정한다.

(2) 빌드도구를 선정한다.

(3) 빌드 스케줄 관리도구를 선정한다.

(4) 테스트 도구를 선정
(5) 검토 대상 코드 인스펙션 점검 도구 파악
(6) 애플리케이션 개발에 적합한 코드 인스펙션 점검 도구를 선정

2. 빌드 배포 환경 정의, 선정된 도구 설치
(1)빌드, 배포 환경을 정의
(2)선정된 빌드, 배포도구를 설치(서버단)
(3)선정된 빌드 배포 도구를 설치(클라이언트단)

[3] 애플리케이션 개발 기간(개발 프로젝트) 중 적용할 배포 절차 및 역할을 정의

1. 애플리케이션 개발 기간 중 사용할 배포 유형 및 기준을 정의
2. 애플리케이션 개발 기간 중 사용할 애플리케이션 배포 절차 정의
3. 애플리케이션 개발 기간 중 배포를 위한 관계자들의 역할과 책임 정의

★애플리케이션 소스 검증

● 소스코드 검증도구 (정적 테스트 도구 + 동적 테스트 도구)
● 정적 테스트 도구: 구현된 SW를 실행하지 않고 테스트
테스트 하기 전 코딩오류, 성능저하, 보안 취약점 등 결함을 조기에 발견할 수 있도록 지원
개발의 생산성 향상, 운영환경에서 프로그램의 품질 향상 제고, 정량적인 품질 관리시스템 구축

● 동적 테스트 도구: 구현된 SW를 실행하여 동작을 보면서 테스트
테스트 미수행 코드를 확인하고 분기(결정)문 등 특정 유형의 코드 구조가 충분히 테스트 되었는 지 확인하여 추가 테스트를 진행함으로써 애플리케이션의 안정성을 제고하고, 소스 품질 관리(통제)활동을 할 수 있는 정량적인 품질 관리시스템을 구축할 수 있게 한다.

●코드 인스펙션
정적테스트의 가장 일반적인 유형
사전에 정의된 코드 작성규칙 기반으로 소스코드를 점검하여 작성 규칙에 위반되는 소스코드를 추출하는 분석도구

1. 코드 인스펙션 Rule유형
(1)성능 개선
(2)코드작성규칙
(3)에러발생가능성

2.코드 작성 Rule심각도 구분

(1)필수, Blocker

에러 발생 간으성이 매우 높거나 메모리 누수가 발생되는 코드

반드시 수정되어야 하는 위반 사항

(2)권고 상, Critical

에러 발생 가능성이 높거나 일반적으로 수정되어야 하는 심각한 위반 사항

(3)권고 중, Major

에러발생이 있거나 수정을 권고하는 중요 위반 사항

(4)권고 하, Minor

소스코드의 가독성, 유지보수성향상을 위해 수정을 권고하는 위반사항

(5)정보, Info

정보성으로 제공되는 위반 사항으로 개발자가 참고하여 적용할 수 있다

3. 정규 표현식(Requlat Expression)

특정한 규칙을 가진 문자열의 집합을 표현하는 범용적인 방식

코드 인스펙션 도구의 코드 작성 규칙은 일반적으로 정규식으로 표현

정규식의 내용을 수정해서 점검Rule의 내용을 수정할 수 있다.

● 테스트 프레임워크(동적 분석 도구)의 구성

테스트케이스를 별도의 테스트 코드로 작성하고 동작 시킬수 있는 환경을 제공하는 도구

-구성:

(1)테스트 코드

테스트 코드 작성 및 자동화된 운영환경을 구성

빌드 도구와 연계하여 빌드 수행시 테스트 코드를 동작시켜 자동환된 테스트 환경을 제공

(2)테스트 저장소

테스트 수행을 위한 테스트 코드,테스트 데이터, 관련 테스트 스크립트,테스트 수행 결과를 저장, 관리

● JUnit 테스트 프레임 워크

Java, 오픈소스 기반의 테스트 프레임워크로 Java개발 환경의 범용적인 표준으로 사용

Eclipse개발도구에 기본 내장기능
Assert함수를 제공

★애플리케이션 소스 빌드

● 지속적인 통합(CI:Continuous Integration)환경
애플리케이션 개발 과정 중 지속적으로 개발된 프로그램을 통합, 빌드, 배포하여 애플리케이션의 개발 내역을 검증, 테스트 할 수 있는 환경
통합 빌드과정 중 테스트코드, 소스코드 품질측정도구 등과 연계할 수 있으며, 자동화된 스케줄 관리를 통해서 지속적이고 반복적인 프로그램 빌드, 테스트를 진행할 수 있다.

1. 빌드도구(Ant, Maven 등)
애플리케이션의 배포단위
형식에 따라 소스코드를 컴파일, 패키징 하며, 배포하는 스크립트를 제공하고 수행하는 도구

2. 테스트 도구(Junit, DBUnit, StrutsTestCase등)
개발된 소스코드를 테스트할 수 있는 테스트코드를 작성, 동작시킬수 있는 도구
통합빌드수행시 연결할 수 있다.

3. 소스코드 품질 측정도구(코드인스펙션)(PMD, FindBugs등)
정해진 소스코드 작성 규칙에 따라 소스코드를 점검하고 규칙 위반 여부를 체크하는 도구
통합 빌드 수행 시 연결할 수 있다.

4.테스트 커버리지 측정도구(Clover, JCoverage, ElcEmma등)
소스코드 내 테스트 가능한 경로 중 테스트 도구를 통해서 테스트된 커버리지를 측정하는 도구

5. 빌드 스케줄 관리도구(Anthill, CruiseControl, Hudson 등)
작성된 빌드 스크립트를 정해진 조건, 시간에 기동하고 진행상태, 수행결과를 관리하는 도구

●테스트 커버리지
의미: 전체 프로그램의 범위 대비 테스트 수행 시 해당 테스트 수행을 위해 동작된 프로그램의 범위 비율

●테스트 커버리지 측정 유형
(1) 라인 커버리지(또는 구문 커버리지)
개발 소스의 각 라인이 수행되었는지를 확인하는 측정지표
(2)분기 커버리지

개발소스의 각 분기문이 수행되었는지를 확인하는 측정지표
소스 내에 if문에 대한 true/false조건이 있다면, 두가지 경우가 모두 테스트되어야 100%로 측정된다.
(3)조건 커버리지
각 분기문 내에 존재하는 조건식이 모두 테스트되었는지를 확인하는 측정지표
조건식 간의 조합에 대해서는 체크하지 않는다.

●빌드 스케줄 관리도구
별도의 웹 애플리케이션으로 구성되어 웹 서버상에 배포, 관리자 화면을 통해서 빌드 스크립트, 형상관리 도구 등과 연계
이메일을 통해서 관련 개발자, 관리자들에게 빌드 수행 결과를 제공

●빌드 스케줄 관리도구의 기능
(1)빌드 작업 스케줄링
(2)빌드 작업상태 및 이력관리
(3)빌드 도구 연계관리
(4)빌드 수행 결과 리포팅

★애플리케이션 배포

●운영환경의 특징
1. 네트워크 관점
2. 계정관리부문
3. 보안취약점 부문