Char 11. 제품 소프트웨어 패키징

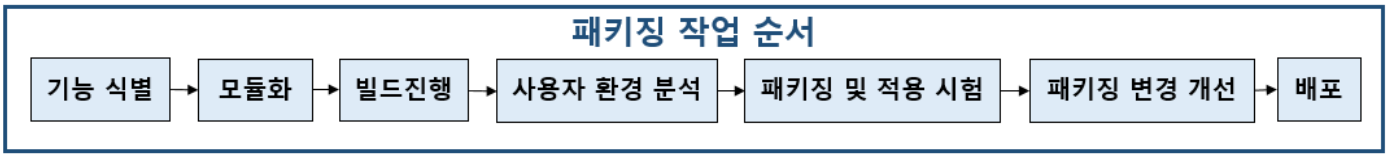
소프트웨어 패키징

모듈별로 생성한 실행 파일들을 묶어 배포용 설치 파일을 만드는 것

사용자 중심, 사용자의 편의성 및 실행 환경 우선적 고려

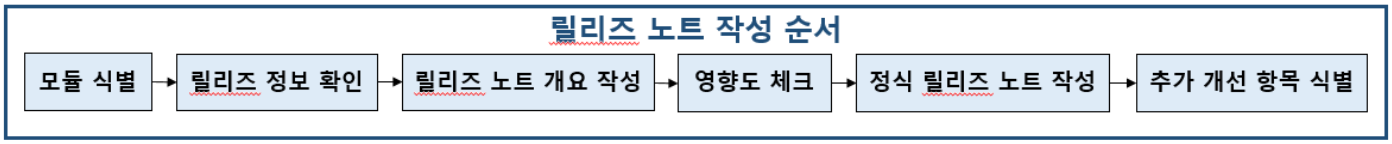
UI는 시각적인 자료와 함께 제공하고 매뉴얼과 일치시켜 패키징

Managed Service 형태로 제공, 안정적인 배포가 중요



릴리즈 노트 Release Note

개발과정에서 정리된 릴리즈 정보를 소프트웨어의 최종 사용자인 고객과 공유하기 위한 문서



빌드 자동화 도구

Jenkins

- Java 기반의 오픈 소스 형태로, 가장 많이 사용되는 빌드 자동화 도구이다.
- 서블릿 컨테이너에서 실행되는 서버 기반 도구이다.
- SVN, Git 등 대부분의 형상 관리 도구와 연동이 가능하다.
- 여러 대의 컴퓨터를 이용한 분산 빌드나 테스트가 가능하다.

Gradle

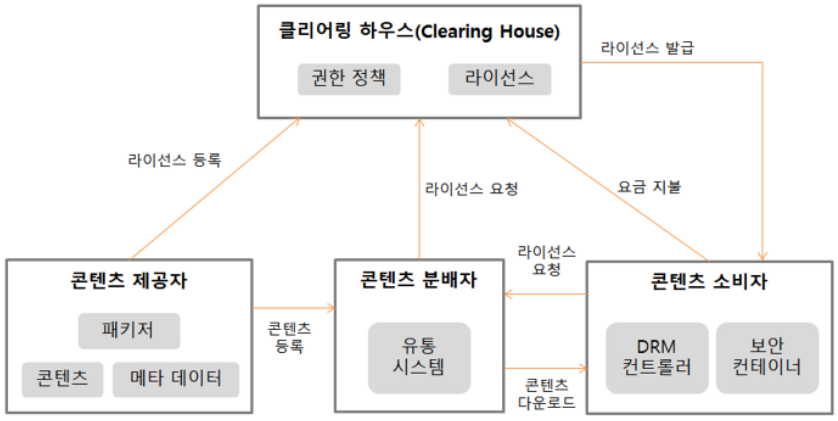
- Groovy를 기반으로 한 오픈 소스 형태의 자동화 도구로, 안드로이드 앱 개발 환경에서 사용된다.
- 안드로이드 뿐만 아니라 플러그인을 설정하면, Java, C/C++, Python 등의 언어도 빌드가 가능하다.
- Groovy를 사용해서 만든 DSL(Domain Specific Language)을 스크립트 언어로 사용한다.
- Gradle은 실행할 처리 명령들을 모아 태스크(Task)로 만든 후 태스크 단위로 실행한다.

디지털 저작권 관리 DRM Digital Right Management

저작권자가 배포한 디지털 콘텐츠가 저작권자가 의도한 용도로만 사용되도록 디지털 콘텐츠의 생성, 유통, 이용까지의 전 과정에 걸쳐 사용되는 디지털 콘텐츠 관리 및 보호 기술

- 원본 콘텐츠가 아날로그인 경우에는 디지털로 변환 한 후 **패키저Packager**에 의해 DRM 패키징을 수행
- **콘텐츠의 크기에** 따라 음원이나 문서와 같이 크기가 작은 경우에는 사용자가 콘텐츠를 요청하는 시점에서 **실시간으로 패키징**을 수행하고 **크기가 큰 경우**에는 미리 **패키징**을 수행한 후 배포
- 패키징을 수행하면 콘텐츠에서는 **암호화된 저작권자의 전자 서명**이 포함되고 저작권자가 설정한 라이선스 정보가 **클리어링 하우스 Clearing House**에 의해 등록
- 사용자가 콘텐츠를 사용하기 위해서는 **클리어링 하우스**에 등록된 **라이선스 정보**를 통해 **사용자 인증과 콘텐츠 사용 권한 소유 여부**를 확인
- **종량제 방식**을 적용한 소프트웨어의 경우 **클리어링 하우스**를 통해 서비스의 실제 사용량을 측정하여 이용한 만큼의 요금을 부과

디지털 저작권 관리의 흐름도



클리어링 하우스(Clearing House)	저작권에 대한 사용 권한, 라이선스 발급, 사용량에 따른 결제 관리 등을 수행하는 곳
콘텐츠 제공자(Contents Provider)	콘텐츠를 제공하는 저작권자
패키저(Packager)	콘텐츠를 메타 데이터와 함께 배포 가능한 형태로 묶어 암호화하는 프로그램
콘텐츠 분배자(Contents Distributor)	암호화된 콘텐츠를 유통하는 곳이나 사람
콘텐츠 소비자(Customer)	콘텐츠를 구매해서 사용하는 주체
DRM **컨트롤러**(DRM Controller)	배포된 콘텐츠의 이용 권한을 통제하는 프로그램
보안 컨테이너(Security Container)	콘텐츠 원본을 안전하게 유통하기 위한 전자적 보안 장치

* 메타 데이터 : 데이터에 대한 데이터, 즉 데이터에 대한 속성 정보 등을 설명하기 위한 데이터

구성요소	설명
암호화(Encryption)	콘텐츠및 라이선스를 암호화하고 전자 서명을 할 수 있는 기술
키 관리(Key Management)	콘텐츠를 암호화한 키에 대한 저장 및 분배 기술
암호화 파일 생성(Packager)	콘텐츠를 암호화된 콘텐츠로 생성하기 위한 기술
식별 기술(Identification)	콘텐츠에 대한 식별 체계 표현 기술
저작권 표현(Right Expression)	라이선스의 내용 표현 기술
정책 관리(Policy Management)	라이선스 발급 및 사용에 대한 정책 표현 및 관리 기술
크랙 방지(Tamper Resistance)	크랙에 의한 콘텐츠 사용 방지 기술
인증(Authentication)	라이선스 발급 및 사용의 기준이 되는 사용자 인증 기술

소프트웨어 설치 매뉴얼

개발 초기에서부터 적용된 기준이나 사용자가 소프트웨어를 설치하는 과정에 필요한 내용을 기록한 설명서와 안내서

-설치메뉴얼 주석: 주의사항과 참고사항을 기술

-설치도구 구성: exe, dll, ini, chm 등의 설치 관련 파일에 대해 설명, 설치과정과 결과가 기록된 log 폴더 설명

소프트웨어 사용자 매뉴얼

사용자가 소프트웨어를 사용하는 과정에서 필요한 내용을 문서로 기록한 설명서와 안내서

컴포넌트 단위로, 컴포넌트 명세서와 컴포넌트 구현 설계서를 토대로 작성

소프트웨어 패키징 형상 관리 SCM Software Configuration Management

소프트웨어의 개발 과정에서 소프트웨어의 변경 사항을 관리하기 위해 개발된 일련의 활동

- 소프트웨어 변경의 원인을 알아내고 제어하며, 적절히 변경되고 있는지 확인하여 해당 담당자에게 통보
- 형상 관리는 소프트웨어 개발의 전 단계에 적용되는 활동이며, 유지보수 단계에서도 수행됨
- 형상 관리는 개발의 전체 비용을 줄이고, 개발 과정의 여러 방해 요인이 최소화되도록 보증하는 것을 목적

형상관리의 중요성

1. 변경 사항을 체계적으로 추적 및 통제
2. 무절제한 변경 방지
3. 버그나 수정 사항 추적 가능
4. 진행 정도를 확인하기 위한 기준으로 사용

형상관리의 기능

- 형상 식별 : 형상 관리 대상에 이름과 관리 번호를 부여하고, 계층 구조로 구분해 수정 및 추적이 용이하도록 하는 작업
- 버전 제어 : 다른 버전의 형상 항목을 관리하고, 이를 위해 특정 절차와 도구를 결합시키는 작업
- 형상 통제(변경 관리)
- 형상 감사 : 기준선의 무결성을 평가하기 위해 확인, 검증, 검열 과정을 통해 공식적으로 승인하는 작업
- 형상 기록(상태 보고)

저장소(Repository)	최신 버전의 파일들과 변경 내역에 대한 정보들이 저장되어 있는 곳
가져오기(import)	버전 관리가 되고 있지 않은 아무것도 없는 저장소에 처음으로 파일을 복사
체크아웃(Check-Out)	프로그램을 수정하기 위해 저장소에서 파일을 받아옴, 소스 파일과 함께 버전 관리를 위한 파일들도 받아옴
체크인(Check-In)	체크아웃 한 파일의 수정을 완료한 후 저장소의 파일을 새로운 버전으로 갱신
커밋(Commit)	체크인을 수행할 때 이전에 갱신된 내용이 있는 경우에 충돌을 알리고 diff 도구를 이용해 수정한 후 갱신을 완료
동기화(Update)	저장소에 있는 최신 버전으로 자신의 작업 공간을 동기화 함

소프트웨어 버전 관리 도구

공유 폴더 방식

- 버전 관리 자료가 로컬 컴퓨터의 공유 폴더에 저장되어 관리되는 방식
- 개발자들은 개발이 완료된 파일을 약속된 공유 폴더에 매일 복사함
- 담당자는 공유 폴더의 파일을 자기 PC 로 복사한 후 이상 유무를 확인
- 이상 유무 확인 중 파일의 오류가 확인되면, 해당 파일을 등록한 개발자에게 수정 의뢰
- 파일을 잘못 복사하거나 다른 위치로 복사하는 것에 대비하기 위해 파일의 변경사항을 데이터베이스에 기록해 관리
- ex) SCCS, RCS, PVCS, QVCS

클라이언트/서버 방식

- 버전 관리 자료가 서버에 저장되어 관리되는 방식
- 서버의 자료를 개발자별로 자신의 PC 로 복사해 작업한 후 변경된 내용을 서버에 반영
- 모든 버전 관리는 서버에서 수행
- 하나의 파일을 서로 다른 개발자가 작업할 경우 경고 메시지 출력
- 서버에 문제 발생 시 협업 및 버전 관리 작업 중단 됨
- ex) CVS, SVN(Suvbersion), CVSNT, Clear Case, CMVC, Perforce 등

분산 저장소 방식

- 버전 관리 자료가 하나의 원격 저장소와 분산된 개발자 PC 의 로컬 저장소에 함께 저장되어 관리되는 방식
- 개발자별로 원격 저장소의 자료를 자신의 로컬 저장소로 복사해 작업한 후 변경된 내용을 로컬 저장소에서 우선 반영한 다음 이를 원격 저장소에 반영 함
- 원격 저장소에 문제가 생겨도 로컬 저장소의 자료를 이용해 작업 할 수 있음
- ex) Git, GNU arch, DCVS, Bazaar, TeamWare, Bitkeeper 등

Subversion(SVN)

CVS(ConcurrentVersionSystem)를 개선한 것으로, 클라이언트/서버구조이며, 아파치 소프트웨어재단에서 2000년에 발표하였음

- add : 새로운 파일이나 디렉터리를 버전 관리 대상으로 등록
- commit : 버전 관리 대상으로 등록된 클라이언트의 소스 파일을 서버의 소스 파일에 적용
- update : 서버의 최신 commit 이력을 클라이언트의 소스 파일에 적용
- checkout : 버전 관리 정보와 소스 파일을 서버에서 클라이언트로 받아옴
- lock/unlock : 서버의 소스 파일이나 디렉터리를 잠그거나 해제
- Import : 아무것도 없는 서버의 저장소에 맨 처음 소스 파일을 저장
- export : 버전 관리에 대한 정보를 제외한 순수한 소스 파일만을 서버에서 받아 옴
- info : 지정된 파일에 대한 정보 표시
- diff : 지정된 파일이나 경로에 대해 전 리비전과의 차이 표시
- merge : 다른 디렉터리에서 작업된 버전 관리 내역을 기본 개발 작업과 병합

Git

리누스토발즈(LinusTorvalds)가 2005 년 리눅스 커널 개발에 사용할 관리도구로 개발한 이후 주니오 하마노(Junio Hamano)에 의해 유지보수되고 있음

- add : 작업 내역을 지역 저장소에 저장하기 전 스테이징 영역에 추가
- commit : 작업 내역을 지역 저장소에 저장
- branch : 새로운 브랜치 생성
- checkout : 지정한 브랜치로 이동
- merge : 브랜치 병합
- init : 지역 저장소 생성
- remote add : 원격 저장소에 연결
- push : 로컬 저장소의 변경 내역을 원격 저장소에 반영
- fetch : 원격 저장소의 변경 이력만을 지역 저장소로 가져옴
- clone : 원격 저장소의 전체 내용을 지역 저장소로 복제
- fork : 지정한 원격 저장소의 내용을 자신의 원격 저장소로 복제

계정 설정하기

$ git config –global user.name “Kelly”

$ git config –global user.email “Kelly.kim19951110@gmail.com”

지역저장소 만들기

$ git init

작업 내역을 지역저장소에 저장하기 전, 스테이징 영역에 추가

$ git add –all

지역 저장소에 저장

$ git commint –m “first commit is done”

새로운 브랜치 생성

$ git branch new_branch

포인터를 현재 작업 중인 마스터 브랜치에서 새로운 브랜치로 이동

$ git checkout new_branch

마스터 브랜치와 새로운 브랜치를 병합하고 새로운 브랜치 제거

$ git checkout master

$ git merge new_branch

$ git branch –d new_branch

원격 저장소 위치를 별명으로 지정 후, 지역저장소 변경 내용을 원격 저장소에 저장

$ git remote add <별칭> https://github.com/…

$git push <별칭> master