**프레임워크 프로그래밍**

**2주차 Report**

**그리기이(가) 표시된 사진

자동 생성된 설명**

**한국산업기술대학교**

**201715404컴퓨터공학과**

**권한길**

|  |
| --- |
| **Q1. 메이븐의 프로젝트 관리 기능인 의존성 관리 기능에 대해 설명하시오** |
| A1.  **의존성 관리는** 프로젝트 내에서 어떤 외부 라이브러리들을 사용할 것인지에 대해 정의해 놓고 이걸 관리 한다고 생각하면 된다.  Maven 이전에는 보통 필요한 라이브러리 파일(jar)들을 lib 폴더로 집어 넣고 사용을 했다.  그렇기 때문에 lib에 들어가 있는 jar파일들이 어떤 라이브러리 인지 나중에는 파악이 안되는 경우도 있었고, 일부 라이브러리들은 파일명에 버전 정보가 표시 안되는 파일도 있어 버전 확인도 어려운 문제가 있다.  이를 개선하고자 Maven에서는 **pom.xml파일 안에 라이브리 목록을 작성**해 놓고 필요한 jar 파일들을 자동으로 찾아서 추가해주는 의존성 관리 기능을 포함시켰습니다.  **pom.xml 파일의 의존성(dependency)항목에 필요한 라이브러리를 적어 주기만 하면 되는 것이다**. 각각의 라이브러리는 같이 배포가 될지, 컴파일 때만 사용될지, 테스트 때만 사용될지를 지정해 주면 자동으로 관리가 된다. |

|  |
| --- |
| **Q2. 메이븐의 빌드 기능 용어중에서 빌드라이프 사이클, 골, 플러그인에 대해 설명하시오.** |
| A2.  **Maven 빌드 라이프사이클(lifecycle)**  - Maven은 프로젝트 생성에 필요한 단계(phases)들을 Build Lifecycle이라 정의하고 **default, clean, site** 세가지로 표준 정의한다.    라이프 사이클(lifecycle)  **\* default** : 프로젝트 배포 절차, 패키지 타입 별로 다르게 정의 됨   1. validate : 프로젝트 상태 점검, 빌드에 필요한 정보 존재 유무 체크 2. initialize : 빌드 상태를 초기화, 속성 설정, 작업 디렉터리 생성 3. generate-sources : 컴파일에 필요한 소스 생성 4. process-sources : 소스코드를 처리 5. generate-resources : 패키지에 포함될 자원 생성 6. compile : 프로젝트의 소스코드를 컴파일 7. process-classes : 컴파일 후 후처리 8. generate-test-source : 테스트를 위한 소스 코드를 생성 9. process-test-source : 테스트 소스코드를 처리 10. generate-test-resources : 테스팅을 위한 자원 생성 11. process-test-resources : 테스트 대상 디렉터리에 자원을 복사하고 가공 12. test-compile : 테스트 코드를 컴파일 13. process-test-classes : 컴파일 후 후처리 14. test : 단위 테스트 프레임워크를 이용해 테스트 수행 15. prepare-package : 패키지 생성 전 사전 작업 16. package : 개발자가 선택한 war, jar 등의 패키징 수행 17. pre-integration-test : 통합 테스팅 전 사전 작업 18. integration-test : 통합 테스트 19. post-integration : 통합 테스팅 후 사후 작업 20. verify : 패키지가 품질 기준에 적합한지 검사 21. install : 패키지를 로컬 저장소에 설치 22. deploy : 패키지를 원격 저장소에 배포   **\* clean** : 빌드 시 생성되었던 산출물을 삭제   1. pre-clean : clean 작업 전에 사전 작업 2. clean : 이전 빌드에서 생성된 모든 파일 삭제 3. post-clean : 사후 작업   **\* site** : 프로젝트 문서화 절차   1. pre-site : 사전 작업 2. site : 사이트 문서 생성 3. post-site : 사후 작업 및 배포 전 사전 작업 4. site-deploy : 생성된 문서를 웹 서버에 배포   MAVEN GOAL  goal은 **내가 실행할 명령**이라고 생각하면 된다.  메이븐에서 기본 명령어 예시 : (mvn [-option] [<goal(s)>] [<phase(s)>] )  clean 페이즈를 실행하고, compiler 플러그인의 compile 골을 실행한다.  phase에 goal이 연결되어 있기 때문에 phase를 통해 메이븐 빌드를 실행하면 해당 phase에 연결되어 있는 goal이 실행된다.  MAVEN PLUGIN  메이븐에서 제공하는 모든 기능은 플러그인을 기반으로 동작한다.  메이븐은 자체는 기본적인 기능만 가지고 있고 대부분의 기능들은 플러그인을 통해 제공하도록 되어있기 때문이다. 플러그인들은 몇가지 goal을 가지고 있고 goal은 플러그인에 포함되어 있는 명령이다.  즉, **플러그인은 하나 이상의 goal의 집합체**이다. (사진 참고)  스크린샷이(가) 표시된 사진  자동 생성된 설명 |
| **Q3. maven 첨부 예제를 참고하여 실습하고 결과 출력 페이지를 PrintScreen하여 첨부하시오.** |
| **<pom.xml 에 코드 추가>**  스크린샷이(가) 표시된 사진  자동 생성된 설명  **<STS 툴에서 실행>**  스크린샷이(가) 표시된 사진  자동 생성된 설명  **<Runable.jar 파일 생성 후 실행>** |