환경변수

GRADLE\_HOME : 그레들 홈

GRADLE\_USER\_HOME : 그레들 유저홈 여기에 cache라던지 warpper라던지 쌓인다.

메뉴얼.

https://docs.gradle.org/current/dsl/org.gradle.api.tasks.bundling.Jar.html

<https://docs.gradle.org/current/userguide/multi_project_builds.html>

**Chapter 25. Multi-project Builds**

헬로우 그래들

https://examples.javacodegeeks.com/core-java/gradle/gradle-hello-world-tutorial/

멀티프로젝트 셋팅

http://kwonnam.pe.kr/wiki/gradle

http://kwonnam.pe.kr/wiki/gradle/java

setting.gradle

|  |
| --- |
| rootProject.name = 'omnifit' def modulesMap = [:] modulesMap['basic\_java'] = "${System.*getenv*().OMNICNS\_LIB}/basic\_java/common"; modulesMap['basic\_jsp'] = "${System.*getenv*().OMNICNS\_LIB}/basic\_jsp/common"; modulesMap['basic\_spring'] = "${System.*getenv*().OMNICNS\_LIB}/basic\_spring/common"; modulesMap['omnifit\_common']= "${System.*getenv*().OMNICNS\_OMNIFIT}/omnifit\_common/common"; modulesMap.each{ key, value->  println (rootProject.name+' '+key+"-------"+value)  include key  def projectDir = new File(rootDir, key)  if( !projectDir.exists() ) {  projectDir.mkdirs()  }  if (value?.trim()) {  project(":${key}").projectDir = new File(value)  } } |

build.gradle

|  |
| --- |
| apply plugin: 'java' apply plugin: 'war' apply plugin: 'maven' apply plugin: 'eclipse' apply plugin: 'eclipse-wtp' apply plugin: 'idea'   group = 'omnifit\_cms' version = '3.5.0'   sourceCompatibility = 1.8 targetCompatibility = 1.8 // 소스 인코딩 지정방법 1 [compileJava, compileTestJava]\*.options\*.encoding = 'UTF-8' // 소스 인코딩 지정밥법 2 tasks.withType(JavaCompile) {  //options.fork = true  //options.forkOptions.executable = "path/to/javac"  options.encoding = 'UTF-8' }  //위에꺼또는..  compileJava {   options.fork = true  options.forkOptions.executable = "javac" // assumes that javac is on PATH  options.compilerArgs << "-XDignore.symbol.file"// does not exist 오류나면.. } //org.gradle.java.home="${System.getenv().JAVA\_HOME}"  task checkJavaVersion << {  def javaVersion = System.*getProperty*("java.version")  print("java version"+javaVersion)  if (!javaVersion.startsWith("1.6")) {  throw new GradleException("Please update JAVA\_HOME to use a 1.6 JDK")  } }  repositories {  maven { url "https://code.lds.org/nexus/content/groups/main-repo" }  maven { url "http://repo1.maven.org/maven2/" }  maven { url "http://www.egovframe.go.kr/maven/" }  maven { url "http://repo.maven.apache.org/maven2" } } dependencies {  compile(group: 'egovframework.rte', name: 'egovframework.rte.ptl.mvc', version:'3.5.0') { exclude(module: 'commons-logging')  }  compile group: 'egovframework.rte', name: 'egovframework.rte.psl.dataaccess', version:'3.5.0'  compile group: 'egovframework.rte', name: 'egovframework.rte.fdl.idgnr', version:'3.5.0'  compile group: 'egovframework.rte', name: 'egovframework.rte.fdl.property', version:'3.5.0'  compile group: 'javax.servlet', name: 'jstl', version:'1.2'  compile group: 'taglibs', name: 'standard', version:'1.1.2'  compile group: 'org.antlr', name: 'antlr', version:'3.5'  compile group: 'org.hsqldb', name: 'hsqldb', version:'2.3.2'  compile group: 'commons-dbcp', name: 'commons-dbcp', version:'1.4'  compile group: 'com.oracle', name: 'ojdbc6', version: '11.2.0.3'  compile group: 'net.sourceforge.jexcelapi', name: 'jxl', version: '2.6.12'  compile group: 'org.aspectj', name: 'aspectjweaver', version: '1.8.0'  /\*commons-io\*/  compile group: 'commons-io', name: 'commons-io', version: '2.4'  /\*poi\*/  compile group: 'org.apache.poi', name: 'poi', version: '3.14'   /\*groovy\*/  compile group: 'org.codehaus.groovy', name: 'groovy-all', version: '2.4.6'  /\*hibernate\*/  compile group : 'org.hibernate', name:'hibernate-core', version : '4.3.6.Final'  /\* lombok\*/  compile group: 'org.projectlombok', name: 'lombok', version: '1.16.8'  /\* jdbc log \*/  compile group: 'org.lazyluke', name: 'log4jdbc-remix', version: '0.2.7'    /\* httpClient \*/  compile group: 'org.apache.httpcomponents', name: 'httpclient', version: '4.5.2'  compile group: 'commons-httpclient', name: 'commons-httpclient', version: '3.1'    /\* junit \*/  compile group: 'junit', name: 'junit', version: '4.12'  compile group: 'org.hamcrest', name: 'hamcrest-library', version: '1.3'    /\* json \*/  compile group: 'com.bluelinelabs', name: 'logansquare', version: '1.1.0'  compile group: 'com.google.code.gson', name: 'gson', version: '2.6.2'   /\* tiles3 \*/  compile group: 'org.apache.tiles', name: 'tiles-core', version: '3.0.5'  compile group: 'org.apache.tiles', name: 'tiles-servlet', version: '3.0.5'  compile group: 'org.apache.tiles', name: 'tiles-jsp', version: '3.0.5'    /\* email \*/  compile group: 'javax.mail', name: 'mail', version: '1.4.7'    /\* fileUpLoad \*/  compile group: 'commons-fileupload', name: 'commons-fileupload', version: '1.2.2'  compile group: 'commons-net', name: 'commons-net', version: '3.3'    /\* spring-security \*/  compile group: 'org.springframework.security', name: 'spring-security-core', version: '4.0.4.RELEASE'  compile group: 'org.springframework.security', name: 'spring-security-config', version: '4.0.4.RELEASE'  compile group: 'org.springframework.security', name: 'spring-security-taglibs', version: '4.0.4.RELEASE'  compile group: 'org.springframework.security', name: 'spring-security-aspects', version: '4.0.4.RELEASE'    /\* server validation \*/  compile group: 'javax.validation', name: 'validation-api', version: '1.1.0.Final'  compile group: 'org.hibernate' , name: 'hibernate-validator', version: '5.2.4.Final'    /\* PDF LIB \*/  compile group: 'com.itextpdf', name: 'itextpdf', version: '5.0.6'  //json path  compile group: 'com.jayway.jsonpath', name: 'json-path', version: '2.2.0'    providedCompile group: 'javax.servlet', name: 'javax.servlet-api', version:'3.1.0'  providedCompile 'org.apache.tomcat:tomcat-catalina:8.0.32'    **compile project(':basic\_java')  compile project(':basic\_jsp')  compile project(':basic\_spring')  compile project(':omnifit\_common')** }      // 공통 library LinkSoruce //type은 1은 파일 2는 디렉토리이다. //http://kwonnam.pe.kr/wiki/gradle/eclipse eclipse {  project.name = 'cms'  String omnicns\_lib\_home = System.getenv().OMNICNS\_LIB.replaceAll('\\\\','/');  String omnicns\_omnifit\_home = System.getenv().OMNICNS\_OMNIFIT.replaceAll('\\\\','/');  println omnicns\_lib\_home  println omnicns\_omnifit\_home   project {  linkedResource name: 'basic\_java', type: '2', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_java/common/src/main/java'  linkedResource name: 'basic\_jsp', type: '2', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_jsp/common/src/main/java'  linkedResource name: 'basic\_spring', type: '2', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_spring/common/src/main/java'  linkedResource name: 'omnifit\_common', type: '2', locationUri: 'file://'+omnicns\_omnifit\_home+'/omnifit\_common/common/src/main/java'  linkedResource name: 'omnifit\_common\_resources', type: '2', locationUri: 'file://'+omnicns\_omnifit\_home+'/omnifit\_common/common/src/main/resources'  linkedResource name: 'src/main/webapp/js/sizzle.js', type: '1', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_jsp/common/src/main/webapp/js/sizzle.js'  linkedResource name: 'src/main/webapp/js/util.js', type: '1', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_jsp/common/src/main/webapp/js/util.js'  linkedResource name: 'src/main/webapp/js/jquery.extends.js', type: '1', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_jsp/common/src/main/webapp/js/jquery.extends.js'  linkedResource name: 'src/main/webapp/WEB-INF/tld/functions.tld', type: '1', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_jsp/common/src/main/webapp/WEB-INF/tld/functions.tld'  linkedResource name: 'src/main/webapp/WEB-INF/tld/tag.tld', type: '1', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_jsp/common/src/main/webapp/WEB-INF/tld/tag.tld'  }  classpath {  file {  withXml {  def node = it.asNode()  node.appendNode('classpathentry', [kind: 'src', path: 'basic\_java'])  node.appendNode('classpathentry', [kind: 'src', path: 'basic\_jsp'])  node.appendNode('classpathentry', [kind: 'src', path: 'basic\_spring'])  node.appendNode('classpathentry', [kind: 'src', path: 'omnifit\_common'])  node.appendNode('classpathentry', [kind: 'src', path: 'omnifit\_common\_resources'])  }  }  } }  allprojects { } //subprojects 는 공통이다 subprojects { }  //다른프로젝트 설정 만적용시킬때  /project(":basic\_spring") { // dependencies { // compile project(':basic\_jsp') // compile project(':basic\_java') // } //}  idea{ } |
|  |

|  |
| --- |
| project(":basic\_jsp"){  task fatJar(type: Jar) {  manifest {  /\* attributes 'Implementation-Title': 'project',  'Implementation-Version': project.version,  'Main-Class': 'com.sample.CLI'\*///  }  // baseName = project.name + '-all'  from {  configurations.compile.collect {  it.isDirectory() ? it : zipTree(it).matching {  exclude '.\*\*.xml'  //exclude 'META-INF/MANIFEST.MF'  // exclude 'META-INF/log4j-provider.properties'  }  }  }  with jar  } } |

$ gradle :cms:war

dependencies

|  |
| --- |
| dependencies {  compile fileTree(dir: "MyProject/lib/jar")  } |
| dependencies { compile fileTree(dir: "/my/directory/on/computer/libs", includes: ['\*.jar']) } |
| testCompile (...)  providedCompile(...) |
| compile files('libs/"nameOfJarFile".jar') |
| compile fileTree(dir: 'src/main/webapp/WEB-INF/lib', include: '\*.jar') |
| runtime files('libs/mnist-tools.jar', 'libs/gson-2.2.4.jar') |

java compile 및 option setting

|  |
| --- |
| tasks.withType(JavaCompile) {  options.fork = true  options.forkOptions.executable = "path/to/javac"  } |
| tasks.withType(JavaCompile) {  options.encoding = 'UTF-8' } compileJava {  options.fork = true  options.forkOptions.executable = "${System.*getenv*().JAVA\_HOME}/bin/javac" // assumes that javac is on PATH  options.compilerArgs << "-XDignore.symbol.file"// does not exist 오류나면.. } |
| [compileJava, compileTestJava]\*.options\*.encoding = 'UTF-8' |

build.gradle

|  |
| --- |
| import com.sun.corba.se.spi.orbutil.closure.Closure */\*\*  \* 작업자 : doldol  \* 작업날짜 : 2016.12.13  \*/* apply plugin: 'base'  group 'com.nhis.api'  //외부 에서 buildscript {   ext {  springCloudVersion = 'Brixton.SR6'  springBootVersion = '1.4.2.RELEASE'  springDependencyVersion = '0.5.1.RELEASE'  }   repositories {  mavenLocal()  mavenCentral()  jcenter()   maven { url 'http://repo.springsource.org/plugins-release'}  maven { url 'http://repo.spring.io/milestone'}  }   dependencies {  classpath("org.springframework.boot:spring-boot-gradle-plugin:${springBootVersion}")  classpath("io.spring.gradle:dependency-management-plugin:${springDependencyVersion}")   } }  //전체 프로젝트 적용 allprojects {  version = "0.1.0" }  //서브 프로젝트 적용 subprojects { subprojects ->   apply plugin: 'groovy'  apply plugin: 'java'  apply plugin: 'org.springframework.boot'  apply plugin: 'war'  apply plugin: 'idea'  apply plugin: 'io.spring.dependency-management'   targetCompatibility = 1.8  sourceCompatibility = 1.8   //인코딩 모두 적용  def defaultEncoding = 'UTF-8'   //AbstractCompile 모든 JVM을 통한 컴파일일 경우  //하위로는 GroovyCompile JavaCompile AbstractScalaCompile  tasks.withType(AbstractCompile).each {  it.options.encoding = defaultEncoding  }   tasks.withType(Javadoc) { Javadoc javadoc ->  javadoc.options.encoding = defaultEncoding  javadoc.options.addStringOption('Xdoclint:all,-missing', '-quiet')  }   springBoot {  executable = true  }   repositories {  jcenter()  }   dependencyManagement {  imports {  mavenBom "org.springframework.cloud:spring-cloud-dependencies:${springCloudVersion}"  mavenBom "org.springframework.boot:spring-boot-starter-parent:${springBootVersion}"  }  }   dependencies {  compileOnly "org.projectlombok:lombok:1.16.8"  compile "org.springframework.cloud:spring-cloud-starter-eureka"  compile "org.springframework.cloud:spring-cloud-starter-feign"  compile "org.springframework.cloud:spring-cloud-starter-sleuth"  compile "org.springframework.boot:spring-boot-starter-actuator"  compile "org.springframework.boot:spring-boot-starter-security"  compile "org.springframework.security.oauth:spring-security-oauth2" //oauth2  compile 'org.springframework.security:spring-security-jwt' //token  compile "org.springframework.boot:spring-boot-starter-aop"  compile "org.springframework.boot:spring-boot-starter-cache"  compile "org.springframework.boot:spring-boot-starter-data-jpa"  compile "org.springframework.boot:spring-boot-starter-web"  compile "org.hibernate:hibernate-java8"  compile "org.ehcache:ehcache:3.1.+"  compile "javax.cache:cache-api:1.0.0"  compile "com.zaxxer:HikariCP:2.4.+"  compile "com.fasterxml.jackson.datatype:jackson-datatype-hibernate5:2.8.+"  compile "com.fasterxml.jackson.datatype:jackson-datatype-jsr310:2.8.+"  compile "commons-io:commons-io:2.5"  compile "org.apache.commons:commons-lang3:3.4"  compile "com.ibm.icu:icu4j:57.1"  compile "org.jolokia:jolokia-core:1.3.3"  compile fileTree(dir: 'libs', includes: ['\*.jar'])  runtime "com.h2database:h2:1.4.+"  testCompileOnly "org.projectlombok:lombok:1.16.8"  testCompile "org.springframework.boot:spring-boot-starter-test"  } }  project(':rest\_api') {  dependencies {  compile project(':comm\_core')  compile 'org.springframework.data:spring-data-rest-hal-browser' //tools  compile('org.springframework.boot:spring-boot-starter-data-rest') //rest http method to automatic generation  } }  project(':rest\_oauth2') {  dependencies {  compile project(':comm\_core')   } }  /\*repositories {  mavenCentral()  jcenter()  }  dependencies {  compile('org.springframework.boot:spring-boot-starter-thymeleaf')  compile('org.springframework.boot:spring-boot-starter-web')  testCompile('org.springframework.boot:spring-boot-starter-test') }\*/  //==================== 별도의 실행이 필요한 task 시작 ====================  // 하위 디렉토리 생성 eclipse 일경우만 실행 합니다 task initSrc << {  project.sourceSets\*.allSource.srcDirTrees.flatten().dir.each { dir ->  dir.mkdirs()  } }  // javadoc 생성 설정 task docJava(type: Javadoc) {  source subprojects.collect {project ->  project.sourceSets.main.allJava  }  destinationDir = new File(buildDir, 'javadoc')   println "buildDir => ["+buildDir+ "]"   //classpath  classpath = files(subprojects.collect {project ->  project.sourceSets.main.compileClasspath  })   println " ------------------ javadoc end ------------------"  }  //tasks.eclipse.dependsOn(initSrc)   //Wrapper 3.2.1 버전 설치 task wrapper(type: Wrapper) {  gradleVersion = "3.2.1" } //==================== 별도의 실행이 필요한 task 종료 ====================  /\* 테스트 시작 \*/ /\* Closure cl = { task -> println "I'm $task.project.name"}  task hello << cl  project(':common') {  task hello << cl }  subprojects {  hello << { pri ntln " - | depend on com.nhis.api"} } \*/  //project('common').hello << { // println " common 프로젝트에서만 실행되게 테스트 " //}   /\* 테스트 종료 \*/ |

setting.gradle

|  |  |
| --- | --- |
| rootProject.name = 'com.nhis.api'  String [] modules = ['comm\_core','rest\_api','rest\_oauth2'] include modules  // root directory에 하위 modules에 설정된 값들 디렉토리 만들기  modules.each {  def projectDir = new File(rootDir, it)   //sub directory create  if (!projectDir.exists()) {  projectDir.mkdirs()  }  } |  |

각각 서비프로젝트의 build.gradle쪽에 메인클래스 지정!!

|  |
| --- |
| bootRepackage {  mainClass = 'com.nhis.api.ApiApp' } |

sourceSets 할때

## Source Sets

* Java 플러그인에는 Source Set이라는 개념이 들어가 있으며, 이는 함께 컴파일과 실행되는 소스 파일들의 그룹을 뜻한다.
* 소스 셋에는 자바 소스 파일과 리소스 파일들이 들어간다.
* 다른 플러그인들이 그루비나 스칼라 소스를 추가할 수 있다.
* 소스 셋은 컴파일 클래스패스와 런타임 클래스패스와 관련된다.
* 소스 셋의 목적은 소스들를 논리적 그룹으로 묶고 그 목적을 설명하는 데 있다. 통합 테스트용 소스셋, API 인터페이스 클래스들, 구현체 클래스들 형태로 구분 가능하다.
* 기본 Java Source Set
  + main : 실제 작동 소스코드. 컴파일해서 JAR 파일로 들어간다.
  + test : 단위 테스트 소스코드. 컴파일해서 JUnit이나 TestNG로 실행한다.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| <https://gist.github.com/johnrengelman/9a20697b2246a9bfaca2#file-build-gradle-L13-L23>   |  | | --- | | if (!project.plugins.collect { it.class.name}.any { it.endsWith('JetGradlePlugin')}) { | |  |  |  | | --- | |  | |  |  |  | | --- | | //Set the source directories for buildSrc to contain the source from the main project | |  |  |  | | --- | | sourceSets { | |  |  |  | | --- | | main { | |  |  |  | | --- | | java.srcDirs = ['src/main/java', '../src/main/java'] | |  |  |  | | --- | | groovy.srcDirs = ['src/main/groovy', '../src/main/groovy'] | |  |  |  | | --- | | resources.srcDirs = ['src/main/resources', '../src/main/resources'] | |  |  |  | | --- | | } | |  |  |  | | --- | | } | |  |   }  sourceSets {  java.srcDir file('src/main/java')  resources.srcDir file('src/test/java') } |

idea plugin

|  |
| --- |
| // idea { // module { // print(configurations.integrationTestCompile); // scopes.TEST.plus += [configurations.integrationTestCompile] // scopes.TEST.plus += [configurations.integrationTestRuntime]  //testSourceDirs += file('some-extra-test-dir')  //sourceDirs += file('D:\\\\omnicns\\\\project\\\\lib\\\\basic\_java\\\\common\\\\src')  //generatedSourceDirs += file('D:\\omnicns\\project\\lib\\basic\_java\\common\\src') // } // }  idea {  module {  inheritOutputDirs = false  outputDir = file("$buildDir/classes/main/")  } } |
| source sets 할때  sourceSets {  main{  resources{  srcDirs = ['src/main/webapp']  exclude '\*\*/\*.js'  }  }  }  이런식으러 처리가능하고 기존 기본값이 없어지진 않는다. 즉 덮어씌어져서 없어지지 않고 추가된다. |
| sourceSets {  main {  java {  srcDir 'src/model'  }  }  } |

main class설정

|  |
| --- |
| mainClassName = 'com.noxgroup.nitro.NitroApp' |
| jar {  manifest {  attributes 'Main-Class': 'com.foo.bar.MainClass'  }  } |

war 하기

https://docs.gradle.org/current/userguide/war\_plugin.html

|  |
| --- |
| apply plugin: 'war' |
| project(':web') {  apply plugin: 'war'  war {  archiveName 'hello-gradle.war'  }  dependencies {  compile project(':core')  providedCompile 'javax.servlet:servlet-api:2.5'  providedCompile 'org.apache.tomcat:tomcat-jsp-api:7.0.55'  //...  }  }  떨어지는곳 {project}\web\build\libs\hello-gradle.war  https://docs.gradle.org/current/userguide/war\_plugin.html |
| war.from('src/main/resorces'){ include('log4j.properties')} |
| [Artificial Log4j](http://amanick.blogspot.kr/?m=1)Thursday, May 21, 2015 Gradle war overlay The standard way to extend a war artifact using Maven is to use [Maven's Overlay Plugin](https://maven.apache.org/plugins/maven-war-plugin/overlays.html).  Naturally, Gradle users wanted to adopt the same approach. And there are some [solutions](http://tstacked.blogspot.com/2013/06/gradle-war-overlay-plugin.html), including a [gradle overlay plugin](https://github.com/scalding/gradle-waroverlay-plugin).  These all could be used. However, Gradle's powerful dependency management enables you to emulate the overlay plugin ad-hoc, and have a much more precise control on what you want to happen to your war.  Basically, what you need is to include a war as a non-transitive dependency, and copy all its contents to resulting war, preserving the changes you made.  apply plugin: 'war'  repositories {  mavenCentral()  }  configurations {  overlay {  transitive = false  }  }  dependencies {  overlay('your:dependency1:version')  overlay('your:dependency2:version')  }  war {  duplicatesStrategy = 'exclude'    configurations.overlay.filter({it.name.endsWith(".war")}).each {  from zipTree(it)  }  }  Of course, you may want to use other options of gradle war plugin together with this (e.g., include some configuration files into your WEB-INF), you may want to also include some zip dependencies into your resulting war, and exclude some problematic jars from your overlays in case you import a newer version of them.  apply plugin: 'war'  repositories {  mavenCentral()  }  configurations {  overlay {  transitive = false  }  }  dependencies {  compile('org.problematic:problematic:newer\_version')  overlay('your:dependency1:version')  overlay('your:dependency2:version')  }  war {  duplicatesStrategy = 'exclude'  webInf {  from "$projectDir/config"  }    configurations.overlay.filter({it.name.endsWith(".war") || it.name.endsWith(".zip")}).each {  from zipTree(it).matching { exclude '\*\*/\*problematic\*.jar' }  }  }  In case you want to use the imported war's dependencies at compile time, it is also easy.  apply plugin: 'war'  repositories {  mavenCentral()  }  configurations {  overlay  compile.extendsFrom(overlay)  }  dependencies {  overlay('your:dependency1:version')  overlay('your:dependency2:version')  }  war {  duplicatesStrategy = 'exclude'    configurations.overlay.filter({it.name.endsWith(".war") || it.name.endsWith(".zip")}).each {  from zipTree(it).matching { exclude '\*\*/\*.jar' }  }  } |

repositories

|  |
| --- |
| repositories {  flatDir {  dirs 'libs'  }  }  dependencies {  compile name: 'gson-2.2.4'  } |
| repositories {  mavenCentral()  }  dependencies {  compile 'com.google.code.gson:gson:2.2.4'  } |
| dependencies {  compile files('libs/something\_local.jar')  } |
| dependencies {  compile fileTree(dir: 'libs', include: ['\*.jar'])  } |
| compile(group: 'egovframework.rte', name: 'egovframework.rte.ptl.mvc', version:'3.5.0') {  exclude group: "log4j", module: "log4j" }  compile(group: 'egovframework.rte', name: 'egovframework.rte.ptl.mvc', version:'3.5.0') { exclude(module: 'commons-logging')  } |
| allprojects {  ...  repositories {  //All sub-projects will now refer to the same 'libs' directory  flatDir {  dirs "$rootProject.projectDir/libs"  }  mavenCentral()  }  ...  } |

gradle war

|  |
| --- |
|  |

**Runnable JAR jar 만들기**

http://www.gubatron.com/blog/2014/07/29/gradle-how-to-copy-files-from-another-jar-into-your-resulting-output-jar/

|  |
| --- |
| jar {  from files(sourceSets.main.output.classesDir)  from {configurations.compile.collect {zipTree(it)}} {  exclude "META-INF/\*.SF"  exclude "META-INF/\*.DSA"  exclude "META-INF/\*.RSA"  }  manifest {  attributes "Main-Class": "pdfs.ParamPDF"  }  } |
| jar {  baseName = 'org.noxgroup-nitro'  version = '0.1.0'  } |
| jar {  from ("src/main/resources/yang") {  into ("META-INF/yang")  }  } |
| jar {  archiveName='basic\_jsp.jar'  includeEmptyDirs = false  eachFile { details ->  if (details.path.startsWith('yang')) {  details.path = "META-INF/$details.path"  }  }  }  //클래스 컴파일된놈들 리스트 나온다 즉 컴파일되서 jar안에 들어간놈들만.. |
| jar {  exclude "yang"  from ("src/main/resources/yang") {  into ("META-INF/yang")  }  } |
| Jar jarTask = project.jar  jarTask.exclude("yang")  jarTask.from(task.yangFilesRootDir) { into("META-INF/yang") } |
| |  | | --- | | jar { | |  |  |  | | --- | | // From http://cmoz.me/blog/2014/11/service-files-uber-jars-and-gradle/ | |  |  |  | | --- | | doFirst { | |  |  |  | | --- | | def serviceDir = file("$buildDir/META-INF/services") | |  |  |  | | --- | | serviceDir.deleteDir() | |  |  |  | | --- | | serviceDir.mkdirs() | |  |  |  | | --- | |  | |  |  |  | | --- | | // copy all service files from deps to buildDir | |  |  |  | | --- | | for(file in configurations.runtime) { | |  |  |  | | --- | | zipTree(file).matching{ include 'META-INF/services/\*' }.each { f -> | |  |  |  | | --- | | new File(serviceDir, f.name) << f.getText("UTF-8") | |  |  |  | | --- | | } | |  |  |  | | --- | | } | |  |  |  | | --- | | } | |  |  |  | | --- | |  | |  |  |  | | --- | | manifest { | |  |  |  | | --- | | attributes "Main-Class": "com.example.Main" | |  |  |  | | --- | | } | |  |  |  | | --- | |  | |  |  |  | | --- | | from(configurations.runtime.collect{ it.isDirectory() ? it : zipTree(it) }) { | |  |  |  | | --- | | exclude 'META-INF/\*\*' // Don't let Gradle merge service files | |  |  |  | | --- | | } | |  |  |  | | --- | |  | |  |  |  | | --- | | // Include service files from the buildDir. | |  |  |  | | --- | | from fileTree(buildDir).matching{ include 'META-INF/services/\*' } | |  |  |  | | --- | | } | |  | |  | |  | |  | |
| task fatJar(type: Jar) {  manifest {  attributes 'Implementation-Title': 'Gradle Jar File Example',  'Implementation-Version': version,  'Main-Class': 'com.mkyong.DateUtils'  }  baseName = project.name + '-all'  from { configurations.compile.collect { it.isDirectory() ? it : zipTree(it) } }  with jar  } |
| 최종..  jar {  archiveName='basic\_jsp.jar'  includeEmptyDirs = false    filesMatching('\*\*/\*.tld'){details ->  details.path = "META-INF/$details.name" //파일패스바뀜.ㅋㅋ  println(details.name)  };    /\* eachFile { details ->  println(details.name)  def filePattern = ~/${fileSubStr}/  if (filePattern.matcher(it.name).find()) {  details.path = "META-INF/$details.path"  }  }\*/  } |

wrapper

|  |
| --- |
| task wrapper(type: Wrapper) {  gradleVersion = '1.11'  } |

idea

|  |
| --- |
| apply plugin: 'idea'  idea{  project {  languageLevel = '1.7'  }  module {  downloadJavadoc = true // defaults to false  downloadSources = true  }  } |
| idea {  module {  inheritOutputDirs = false  outputDir = file("$buildDir/classes/main/")  }  } |

war

embed tomcat

|  |
| --- |
| buildscript {  repositories {  jcenter()  }  dependencies {  classpath 'com.bmuschko:gradle-tomcat-plugin:2.0'  }  }  apply plugin: 'java'  apply plugin: 'war'  apply plugin: 'com.bmuschko.tomcat'  // JDK version source compatibility  sourceCompatibility = 1.7  // project version  version = '1.0'  // War file name  war.baseName = 'jcg-gradle-war-example'  // Web directory, this overrides the default value "webapp"  project.webAppDirName = 'WebContent'  repositories {  mavenLocal()  mavenCentral()  }  // Set source directory  sourceSets {  main {  java {  srcDir 'src'  }  }  }  // dependencies to run on tomcat, are mandatory for tomcat plugin  dependencies {  def tomcatVersion = '7.0.57'  tomcat "org.apache.tomcat.embed:tomcat-embed-core:${tomcatVersion}",  "org.apache.tomcat.embed:tomcat-embed-logging-juli:${tomcatVersion}"  tomcat("org.apache.tomcat.embed:tomcat-embed-jasper:${tomcatVersion}") {  exclude group: 'org.eclipse.jdt.core.compiler', module: 'ecj'  }  }  // context where tomcat is deployed, by defautl localhost:8080/  tomcatRun.contextPath = '/'  tomcatRunWar.contextPath = '/' |
| C:\Users\Andres\workspaceLuna\GradleWarPlugin>gradle war  :GradleWarPlugin:compileJava UP-TO-DATE  :GradleWarPlugin:processResources UP-TO-DATE  :GradleWarPlugin:classes UP-TO-DATE  :GradleWarPlugin:war UP-TO-DATE  BUILD SUCCESSFUL  Total time: 5.838 secs  C:\Users\Andres\workspaceLuna\GradleWarPlugin> |
| C:\Users\Andres\workspaceLuna\GradleWarPlugin>gradle tomcatRun  :GradleWarPlugin:compileJava UP-TO-DATE  :GradleWarPlugin:processResources UP-TO-DATE  :GradleWarPlugin:classes UP-TO-DATE  :GradleWarPlugin:tomcatRun  Started Tomcat Server  The Server is running at http://localhost:8080  > Building 75% > :GradleWarPlugin:tomcatRun |
| war {  from 'src/rootContent' // 웹 애플리케이션 루트에 지정된 경로 이하의 파일 셋 추가, into에 웹 애플리케이션 루트에 상대적인 경로로 복사될 목표 디렉토리 지정 가능.  webInf { from 'src/additionalWebInf' } // WEB-INF에 지정된 파일셋 추가, into에 WEB-INF에 상대적인 경로로 복사될 목표 디렉토리 지정 가능.  classpath fileTree('additionalLibs') // WEB-INF/lib 에 지정된 파일셋 추가  classpath configurations.moreLibs // WEB-INF/lib 에 지정된 구성 추가  webXml = file('src/someWeb.xml') // WEB-INF/web.xml 로 복사  } |

buildscript

|  |
| --- |
| buildscript {  repositories {  jcenter()  mavenCentral()  }  dependencies {  classpath 'org.ajoberstar:grgit:1.1.0'  } } |

ext

|  |
| --- |
| ext{  //변수선언  } |

gradle git사용하기

|  |
| --- |
| buildscript {  repositories {  jcenter()  mavenCentral()  }  dependencies {  classpath 'org.ajoberstar:grgit:1.1.0'  }  ext {  name = "cms"  } }  import org.ajoberstar.grgit.\* task pull {  def repo = Grgit.*open*(project.file('..'))  print "project pull :${name} head : ${repo.head().id}";  repo.pull(rebase: false) } |

시스템명령어 치기 cmd commandline

|  |
| --- |
| task pull\_egov<<{  println '----pull\_egov---start-'  exec{  workingDir omnicns\_lib\_home+'/basic\_egov'  commandLine 'cmd','/c','git', 'pull', 'origin', 'master'  } }  /\*git 비밀번호 저장 방법  \* git config --global credential.helper store  \* git fetch origin master  \* 한뒤 비밀번호 저장되면 다음부터 묻지 않음.  \*/ task pull(dependsOn: [pull\_java,pull\_jsp,pull\_spring,pull\_egov]){  println '----pull---start-' } |
| exec {  executable "sh"  args "-c","source ${root}/etc/localBuildPorts"  }  println System.getenv()['FINAL\_LOCALTEST\_TOMCAT\_HTTP\_PORT'] |
| exec{  workingDir webAppDir  executable 'node' args ['node\_modules/gulp/bin/gulp.js','oneshot']  } |
| exec{  workingDir webAppDir  commandLine 'node','node\_modules/gulp/bin/gulp.js','oneshot' } |

의존성 제거

|  |
| --- |
| // commons-logging, log4j, jul 의존성 제거 configurations.all {  exclude group: 'commons-logging', module: 'commons-logging'  exclude group: 'log4j', module: 'log4j'  exclude group: 'log4j', module: 'log4j-api'  exclude group: 'org.apache.logging.log4j' //// exclude group: 'org.slf4j', module: 'slf4j-jcl' // exclude group: 'org.slf4j', module: 'slf4j-jdk14' // exclude group: 'org.slf4j', module: 'slf4j-log4j12' // exclude group: 'org.slf4j', module: 'slf4j-log4j2' // exclude group: 'org.slf4j', module: 'slf4j-log4j' // exclude group: 'log4j', module: 'log4j' } |

### 동적 버전(Dynamic Version) 결정과 변하는 모듈(Changing Module)

* 때로는 특정 의존 라이브러리에 대해 항상 최신 버전을 사용하거나, 혹은 특전 버전대(2.x 버전대중에서) 최신을 사용하고 싶을 경우가 있다. 동적 버전을 통해 가능하다.
* 특정 버전대 : 2.+
* 사용 가능한 최신 버전 : latest.integration
* 변하는 모듈 : 어떤 경우에는 동일 버전이라도 모듈이 변경되는 때가 있다(Maven의 SNAPSHOT, 특히 사내 프로젝트의 경우 이런게 많음).
* 동적 버전은 실제 버전이 변경되고, 변하는 모듈은 버전은 그대로이지만 그 내용물(jar)이 계속해서 변경될 수 있다.
* 기본적으로 동적 버전과 변하는 모듈은 24시간 캐시된다. 설정을 통해 바꿀 수 있다.
* 특정 라이브러리의 변경을 매번 검사해야 한다면 changing = true 옵션 추가

compile ('com.some:library:0.1') { changing = [true](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20true) }

* [\*-SNAPSHOT 버전은 기본으로 changing=true 설정이 된다](http://forums.gradle.org/gradle/topics/how_to_get_gradle_to_download_newer_snapshots_to_gradle_cache_when_using_an_ivy_repository#reply_8068039)
  + 단, Maven Repository 일 때만 그렇다(Maven 자체의 기본정책).
  + Ivy Repository는 SNAPSHOT이라도 changing = true 설정이 필요하다.

org.gradle.java.home in the gradle.properties

java

# Gradle Java Plugin

* [Java Plugin](http://www.gradle.org/docs/current/userguide/java_plugin.html)

apply plugin: 'java'

* [Gradle Dependencies](http://kwonnam.pe.kr/wiki/gradle/dependencies)

## Java Project 기본 build.gradle

* 기본적인 자바 프로젝트 구성을 도와주는 build.gradle로 Eclipse 프로젝트를 가정하고 있다.

apply plugin: 'java'

apply plugin: 'eclipse'

ext {

javaVersion='1.6'

}

buildDir = 'build'

repositories {

mavenCentral()

}

dependencies {

compile '원하는 모듈'

testCompile group: 'junit', name: 'junit', version: '4.+'

}

task initSrc << {

project.sourceSets\*.allSource.srcDirTrees.[flatten](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20flatten)().dir.[each](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20each) { dir ->

dir.mkdirs()

}

}

sourceCompatibility = javaVersion

targetCompatibility = javaVersion

// 소스 인코딩 지정방법 1

[compileJava, compileTestJava]\*.options\*.encoding = 'UTF-8'

// 소스 인코딩 지정밥법 2

tasks.withType(JavaCompile) {

options.encoding = 'UTF-8'

}

/\* Eclipse 관련 설정들 \*/

tasks.eclipse.dependsOn cleanEclipse

tasks.eclipse.dependsOn initSrc

eclipse {

classpath {

downloadSources = [true](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20true)

defaultOutputDir = file("${buildDir}/classes/main")

}

}

## 기본 디렉토리 구조

* Maven과 동일하다.
* 소스 : src/main/java, src/main/resources
* 테스트 : src/test/java, src/test/resources
* 빌드 결과 : build/\*
  + buildDir 를 프로젝트에 상대적인 경로로 바꿔줄 수 있다.

buildDir = 'target'

## 기본 태스크

* clean : build 디렉토리 삭제
* compileJava, compileTestJava : 소스 컴파일
* assemble : 컴파일 하고 jar 생성. 웹 프로젝트일 경우에는 war 생성. 단위 테스트 실행 안함.
* check : 컴파일하고 단위테스트 실행.
* 자세한 것은 [Java Plugin](http://www.gradle.org/docs/current/userguide/java_plugin.html)의 Tasks 참조.

## 기본 의존성 설정

repositories {

mavenCentral()

}

dependencies {

compile group: 'commons-collections', name: 'commons-collections', version: '3.2'

testCompile group: 'junit', name: 'junit', version: '4.+'

}

## Publishing Jar

* [Gradle maven plugin](https://docs.gradle.org/current/userguide/maven_plugin.html) 참조

// 저장소를 지정하고,

uploadArchives {

repositories {

flatDir {

dirs 'repos'

}

}

}

gradle uploadArchives 명령으로 퍼블리싱한다.

## Source Sets

* Java 플러그인에는 Source Set이라는 개념이 들어가 있으며, 이는 함께 컴파일과 실행되는 소스 파일들의 그룹을 뜻한다.
* 소스 셋에는 자바 소스 파일과 리소스 파일들이 들어간다.
* 다른 플러그인들이 그루비나 스칼라 소스를 추가할 수 있다.
* 소스 셋은 컴파일 클래스패스와 런타임 클래스패스와 관련된다.
* 소스 셋의 목적은 소스들를 논리적 그룹으로 묶고 그 목적을 설명하는 데 있다. 통합 테스트용 소스셋, API 인터페이스 클래스들, 구현체 클래스들 형태로 구분 가능하다.
* 기본 Java Source Set
  + main : 실제 작동 소스코드. 컴파일해서 JAR 파일로 들어간다.
  + test : 단위 테스트 소스코드. 컴파일해서 JUnit이나 TestNG로 실행한다.
* 아래에서 자세히 다룸.

## 프로젝트 레이아웃

자바 프로젝트의 기본 프로젝트 레이아웃은 다음과 같다.

* src/main/java : 실행 자바 소스
* src/main/resources : 실행 리소스
* src/test/java : 테스트 자바 소스
* src/test/resources : 테스트 리소스
* src/소스셋/java : 특정 소스 셋의 Java 소스
* src/소스셋/resources : '특정 소스 셋의 리소스

## 의존성 관리

다음과 같은 의존성 설정(configurations)이 추가되다. 이 의존성 설정은 자바 플러그인의 태스크들에 의해 참조 된다.

| **이름** | **부모** | **사용하는 태스크** | **의미** |
| --- | --- | --- | --- |
| compile | - | compileJava | 컴파일 시점 의존성 |
| runtime | compile | - | 실행시 의존성 |
| testCompile | compile | compileTestJava | 테스트를 컴파일할 때 필요한 추가적인 의존성 |
| testRuntime | runtime, testCompile | test | 테스트만 실행할 때 필요한 추가적인 의존성 |
| archives | - | uploadArchives | 해당 프로젝트가 생성한 Artifact(jar 등) |
| default | runtime | - | 이 프로젝트에서 기본으로 사용되는 의존성. 이 프로젝트의 실행시에 필요한 Artifact들과 의존성을 포함한다. |

소스 셋을 추가하면 다음과 같은 의존성 설정도 함께 추가된다.

| **이름** | **부모** | **사용하는 태스크** | **의미** |
| --- | --- | --- | --- |
| 소스셋Compile | - | compile소스셋Java | 특정 소스셋의 컴파일시점 의존성 |
| 소스셋Runtime | 소스셋Compile | - | 특정 소스셋의 실행시 의존성 |

## 관례 프라퍼티들

Gradle 문서 참조. [JavaPluginConvention](http://www.gradle.org/docs/current/dsl/org.gradle.api.plugins.JavaPluginConvention.html), [BasePluginConvention](http://www.gradle.org/docs/current/dsl/org.gradle.api.plugins.BasePluginConvention.html) 참조.

## Source Set 다루기

* 프로젝트의 소스 셋은 sourceSets 프라퍼티로 접근할 수 있다. [SourceSetContainer](http://www.gradle.org/docs/current/javadoc/org/gradle/api/tasks/SourceSetContainer.html) 타입.
* [println](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20println) sourceSets.main.output.classesDir
* [println](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20println) sourceSets['main'].output.classesDir
* sourceSets {
* [println](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20println) "Sources Sets " + main.output.classesDir
* }
* sourceSets {
* main {
* [println](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20println) "Source Sets main " + output.classesDir
* }
* }
* sourceSets.all {
* [println](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20println) "Iterate all Source Sets : " + name

}

* 위에서 사용한 sourceSets 접근 방식 중 하나로 소스셋에 접근하고 변경할 수 있다.
* // main java 소스셋의 디렉토리 추가
* sourceSets {
* main {
* java {
* srcDir 'src/java'
* }
* resources {
* srcDir 'src/resources'
* }
* }

}

저기서 main 은 [SourceSet](http://www.gradle.org/docs/current/javadoc/org/gradle/api/tasks/SourceSet.html)이다.

저기서 'java'와 'resources'는 [SourceDirectorySet](http://www.gradle.org/docs/current/javadoc/org/gradle/api/file/SourceDirectorySet.html)이며 srcDir()메소드는 소스 디렉토리를 **추가**하는 것이다.

### Source Set 프라퍼티들

* [SourceSet](http://www.gradle.org/docs/current/dsl/org.gradle.api.tasks.SourceSet.html) 참조.
* 기본적인 프라퍼티들

| **프라퍼티 이름** | **Type** | **기본값** | **설명** |
| --- | --- | --- | --- |
| name | String(read-only) | not null | 소스셋 이름 |
| output | [SourceSetOutput](http://www.gradle.org/docs/current/dsl/org.gradle.api.tasks.SourceSetOutput.html) | not null | 소스셋의 출력 디렉토리. 컴파일된 클래스와 리소스를 저장할 곳 |
| output.classesDir | File | buildDir/classes/name | 소스셋의 클래스를 생성할 디렉토리 |
| output.resourcesDir | File | buildDir/resources/name | 소스셋의 리소스를 생성할 디렉토리 |
| compileClasspath | [FileCollection](http://www.gradle.org/docs/current/javadoc/org/gradle/api/file/FileCollection.html) | compileSourceSet configuration | 소스를 컴파일할 때 사용할 클래스패스 |
| runtimeClasspath | [FileCollection](http://www.gradle.org/docs/current/javadoc/org/gradle/api/file/FileCollection.html) | output + runtimeSourceSet configuration | 클래스 실행시의 클래스패스 |
| java | [SourceDirectorySet](http://www.gradle.org/docs/current/javadoc/org/gradle/api/file/SourceDirectorySet.html) (read-only) | not null | 소스셋의 자바 소스파일들. \*.java 파일만 포함호며 그 외 파일은 제외된다. |
| java.srcDirs | Set<File>, Project.files()에서 사용할 수 있는 모든 값 | [projectDir/src/name/java] | 자바 소스 파일을 포함하고 있는 소스 디렉토리들, srcDir()로 추가가능 |
| resources | [SourceDirectorySet](http://www.gradle.org/docs/current/javadoc/org/gradle/api/file/SourceDirectorySet.html) (read-only) | not null | 소스셋의 리소스들. \*.java 파일은 제외된다. 플러그인에 때라 제외되는 목록이 추가된다. |
| resources.srcDirs | Set<File>, Project.files()에서 사용할 수 있는 모든 값 | [projectDir/src/name/resources] | 리소스를 포함하고 있는 소스 디렉토리들 |
| allJava | [SourceDirectorySet](http://www.gradle.org/docs/current/javadoc/org/gradle/api/file/SourceDirectorySet.html) (read-only) | java | 모든 \*.java 파일들. 플러그인에 따라 더 추가 될 수 있음 |
| allSource | [SourceDirectorySet](http://www.gradle.org/docs/current/javadoc/org/gradle/api/file/SourceDirectorySet.html) (read-only) | resources + java | 모든 소스 파일들(리소스 + \*.java). 플러그인에 따라 더 추가 될 수 있음. |

### main/resources에 java 디렉토리를 추가하면서 \*.java 파일은 제외하기

때로는 src/main/java 를 리소스에 포함시켜야 할 경우가 있다. 개발자들이 resources에 리소스를 넣지 않고 java에 넣고서 클래스와 리소스간의 결합성을 쉽게 파악할 수 있고자 할 경우가 있기 때문이다(iBATIS SqlMapper 사용하는 경우 등).

sourceSets {

main {

resources {

srcDir "${project.projectDir}/src/main/java"

// exclude "\*\*/\*.java" : 하지 말것. Gradle에서 문제 없지만 Eclipse에서 문제를 일으킴.

}

}

}

### 새로운 소스셋 만들기

* sourceSets { } 블럭 안에 정의하면 된다.
* 소스셋이 추가되면 소스셋 이름으로 시작하는 의존성 설정도 추가된다. 이를 사용해 원하는 대로 의존성 설정을 할 수 있다.
* sourceSets {
* intTest
* }
* // 의존성 설정
* dependencies {
* intTestCompile 'junit:junit:4.8.2'
* intTestRuntime 'org.ow2.asm.asm-all:4.0'

}

* 소스셋이 추가되면 소스셋 태스크도 추가된다.
  + gradle intTestClasses 형태로 실행.
  + compile소스셋Java : 소스셋의 소스 컴파일
  + process소스셋Resources : 소스셋의 리소스를 클래스 디렉토리로 복사한다.
  + 소스셋Classes : compile소스셋Java와 process소스셋Resources 실행.

### 소스셋 태스크 예제

* Jar 묶기 태스크
* task intTestJar(type: Jar) {
* from sourceSets.intTest.output

}

* Javadoc 생성 태스크
* task intTestJavadoc(type: Javadoc) {
* source sourceSets.intTest.allJava

}

* 단위 테스트 실행 태스크
* task intTest(type: Test) {
* testClassesDir = sourceSets.intTest.output.classesDir
* classpath = sourceSets.intTest.runtimeClasspath

}

* java/test 소스셋 디렉토리 자동 생성
* // eclipse 플러그인 실행시 자동으로 기본 Java 디렉토리 구조를 생성하도록 한다.
* task baseDirs << {
* sourceSets.[each](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20each) { set ->
* set.java.srcDirs.[each](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20each) { dir ->
* [if](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20if) (!dir.exists()) dir.mkdirs()
* }
* set.resources.srcDirs.[each](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20each) { dir ->
* [if](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20if) (!dir.exists()) dir.mkdirs()
* }
* }
* [if](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20if) (!buildDir.exists()) buildDir.mkdirs()
* }

tasks.eclipse.dependsOn baseDirs

## Javadoc

* [Javadoc](http://www.gradle.org/docs/current/dsl/org.gradle.api.tasks.javadoc.Javadoc.html) Task의 인스턴스
* [Javadoc Reference Guide](http://docs.oracle.com/javase/1.5.0/docs/tooldocs/windows/javadoc.html#referenceguide)
* [SpringDoclet](http://kwonnam.pe.kr/wiki/springframework/springdoclet) 참조.
* Javadoc의 프라퍼티들

| **태스크 프라퍼티** | **타입** | **기본값** |
| --- | --- | --- |
| classpath | [FileCollection](http://www.gradle.org/docs/current/javadoc/org/gradle/api/file/FileCollection.html) | sourceSets.main.output + sourceSets.main.compileClasspath |
| source | [FileTree](http://www.gradle.org/docs/current/javadoc/org/gradle/api/file/FileTree.html) | sourceSets.main.allJava]] |
| destincationDir | File |  |
| title | String | 프로젝트 이름과 버전 |

\* [StandardJavadocDocletOptions (Gradle API 1.10)](http://www.gradle.org/docs/current/javadoc/org/gradle/external/javadoc/StandardJavadocDocletOptions.html) 참조하여 javadoc 옵션들 지정.

### 멀티 모듈의 소스를 합쳐 Javadoc 생성

* [aggregating javadocs](http://gradle.1045684.n5.nabble.com/aggregating-javadocs-td1433469.html)

task javadoc(type: Javadoc) {

source subprojects.[collect](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20collect) {project ->

project.sourceSets.main.allJava

}

destinationDir = [new](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20new) [File](http://www.google.de/search?as_q=File&num=100&hl=en&as_occt=url&as_sitesearch=java.sun.com%2Fj2se%2F1%2E5%2E0%2Fdocs%2Fapi%2F)(buildDir, 'javadoc')

// Might need a classpath

classpath = files(subprojects.[collect](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20collect) {project ->

project.sourceSets.main.compileClasspath})

}

}

## Delombok

* [Lombok](http://kwonnam.pe.kr/wiki/java/lombok) 사용시 delombok 후에 javadoc 생성을 해야 올바르게 나온다. [참조](https://github.com/maydesk/modules/blob/master/master/build.gradle)
* ext {
* lombokVersion = '1.12.4'
* }
* configurations {
* lombok
* }
* dependencies {
* lombok "org.projectlombok:lombok:${lombokVersion}"
* }
* task delombok {
* ext.srcJava = 'src/main/java'
* ext.srcDelomboked = "${buildDir}/src-delomboked"
* inputs.files file(srcJava)
* outputs.dir file(srcDelomboked)
* doLast {
* // 보통은 configurations.runtime만으로 충분하지만, 가끔 provided 등의 사용자정의 configuration이
* // 존재 할 경우 classpath에 존재하지 않는 라이브러리라서 경고를 보여줄 수 있으므로
* // 모든 configurations의 의존성을 하나로 모아서 classpath로 지정한다.
* [def](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20def) allDependencies = configurations.lombok.asFileTree
* configurations.all { configuration -> allDependencies = allDependencies + configuration.asFileTree }
* // 구버전 Task class : lombok.delombok.ant.DelombokTask
* // 최신버전 Task class: lombok.delombok.ant.Tasks$Delombok
* ant.taskdef(name: 'delombok', classname: 'lombok.delombok.ant.Tasks$Delombok',
* classpath: configurations.lombok.asPath)
* ant.delombok(from: srcJava, to: srcDelomboked, verbose: [true](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20true),
* encoding: 'UTF-8', classpath: allDependencies.asPath)
* }

}

* delombok 후 javadoc 생성
* javadoc {
* dependsOn delombok
* source = fileTree(dir: delombok.srcDelomboked, includes: ['\*\*/\*.java', '\*\*/\*.html'])
* options.encoding = 'utf-8'
* }

// 불필요한 리소스가 복사되는 경우에 대비해 java/html 파일만 include

## Clean

* [Delete](http://www.gradle.org/docs/current/dsl/org.gradle.api.tasks.Delete.html) Task의 인스턴스
* Clean의 프라퍼티

| **태스크 프라퍼티** | **타입** | **기본값** |
| --- | --- | --- |
| dir | File | buildDir |

## Resources

* ProcessResources 태스크([Copy](http://www.gradle.org/docs/current/dsl/org.gradle.api.tasks.Copy.html) 상속)
* ProcessResources 태스크의 프라퍼티들

| **태스크 프라퍼티** | **타입** | **기본값** |
| --- | --- | --- |
| srcDirs | files()가 받을 수 있는 모든 값 | sourceSet.resources |
| destinationDir | File | sourceSet.output.resourcesDir |

## CompileJava

* [JavaCompile](http://www.gradle.org/docs/current/dsl/org.gradle.api.tasks.compile.JavaCompile.html) Task 인스턴스
* compileTestJava에도 공통 적용된다.
* 기본적인 옵션들

| **태스크 프라퍼티** | **타입** | **기본값** |
| --- | --- | --- |
| classpath | [FileCollection](http://www.gradle.org/docs/current/javadoc/org/gradle/api/file/FileCollection.html) | sourceSet.compileClasspath |
| source | [FileTree](http://www.gradle.org/docs/current/javadoc/org/gradle/api/file/FileTree.html) files()가 받을 수 있는 모든 인자]] | sourceSet.java |
| sourceCompatibility | String | Java 소스의 Java 언어 레벨 (…, 1.4,1.5,1.6,1.7 …) project.sourceCompatibilty |
| targetCompatibility | Strign | Java 클래스의 Java 언어레벨 project.targetCompatibility |
| destinationDir | File | sourceSet.output.classesDir |
| options | [CompileOptions](http://www.gradle.org/docs/current/javadoc/org/gradle/api/tasks/compile/CompileOptions.html) | 컴파일 관련 각종 옵션 설정 |

* compile 태스크는 Ant의 javac 태스크를 호출한다. options.useAnt=false로 바꾸면 Ant를 건너뛰고 Gradle 기본 컴파일러로 수행한다.
* 기본적으로 Gradle 프로세스 안에서 컴파일러를 호출하지만 options.fork=true로 설정하면 독립 컴파일러 프로세스가 뜨게 된다. 성능이 떨어질 수 있다.
* options에서 소스 인코딩 등을 지정할 수 있다.

## 컴파일 옵션들

아래 값들은 Java 플러그인을 적용한 뒤에 설정해야 한다. 그렇지 않으면 Java 플러그인이 값을 초기화 해 버릴 수도 있다. 아래 값들은 ext 블럭으로 만들면 “안” 된다.

* 자바 소스 인코딩 지정
* compileJava.options.encoding = 'UTF-8'
* // 혹은

[compileJava, compileTestJava]\*.options\*.encoding = 'UTF-8'

* 자바 언어 레벨 지정
* // project 단위
* sourceCompatibility = '1.6'
* targetCompatibility = '1.6'

* // compileJava 단위
* compileJava {
* sourceCompatibility = '1.6'
* targetCompatibility = '1.6'

}

* 컴파일러 옵션 지정
* compileJava.options.compilerArgs = ["-Xlint:unchecked", "-Xlint:deprecation", ...]
* // 혹은
* tasks.withType(JavaCompile) {
* options.compilerArgs << "-Xlint:unchecked" << "-Xlint:deprecation"

}

* define 사용하여 한번에 여러 [CompileOptions](http://gradle.org/docs/current/javadoc/org/gradle/api/tasks/compile/CompileOptions.html) options 지정
* compileJava.options.define(
* 옵션1: 값,
* 옵션2: 값, ...

)

## Test

* test 태스크는 [Test](http://www.gradle.org/docs/current/dsl/org.gradle.api.tasks.testing.Test.html)의 인스턴스이다.
* 테스트 소스셋에서 모든 단위 테스트를 자동으로 감지하여 실행한다.
* 테스트 수행이 끝나면 보고서를 생성한다.
* JUnit, TestNG를 지원한다.

### 테스트 실행

* 테스트는 독립 JVM에서 격리상태로 실행된다.
* test.debug 프라퍼티를 true로 설정하면 디버그모드로 실행되며 5005 포트로 디버깅할 수 있다.
* 병렬 테스트
  + 여러 테스트 프로세스를 동시에 실행한 수 있다.
  + maxParallelForks 프라퍼티로 테스트 프로세스 갯수를 설정할 수 있다. 기본값은 1이다.(병렬 테스트 안함)
  + 테스트 프로세스는 org.gradle.test.worker 시스템 프라퍼티를 설정한다.
  + forkEvery 프라퍼티로 몇개의 테스트를 수행한뒤에 프로세스를 재시작 할지 정할 수 있다. 단위 테스트가 JVM Heap을 너무 많이 소모할 경우 이 값을 작게준다. 기본은 재시작 안함.
  + [Gradle Goodness: Running Tests in Parallel](http://mrhaki.blogspot.kr/2010/11/gradle-goodness-running-tests-in.html)
  + test {
  + [if](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20if) (project.hasProperty('maxParallelForks'))
  + maxParallelForks = project.maxParallelForks [as](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20as) [int](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20int)
  + [if](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20if) (project.hasProperty('forkEvery'))
  + forkEvery = project.forkEvery [as](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20as) [int](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20int)

}

* ignoreFailures 프라퍼티는 테스트 실패시 행위를 정의한다. 기본값은 false이며 테스트가 실패하면 즉시 멈춘다. true일 경우 테스트가 실패해도 멈추지 않고 다음으로 넘어간다.
* testLogging 프라퍼티는 테스트의 로깅 레벨을 설정한다. 기본적으로 모든 실패한 테스트에 대한 요약 메시지를 보여준다. [TestLoggingContainer](http://www.gradle.org/docs/current/dsl/org.gradle.api.tasks.testing.logging.TestLoggingContainer.html) 참조.
  + 표준 출력/에러를 화면에 표시하려면 testLogging.showStandardStreams = true 설정 필요.

### 성공한 Test 강제 실행(force run test)

* 성공한 테스트는 다시 테스트를 실행하면 실행이 안된다.
* 이 때, cleanTest 태스크를 먼저 실행하고 테스트를 하면 된다.

gradlew cleanTest test

### System Properties

* 시스템 프라퍼티는 gradle -D프라퍼티이름=값 형태로 지정한다.
* taskName.single=testNamePattern 형태를 지정하면 testNamePattern에 일치하는 테스트만 실행된다.
* taskName은 멀티프로젝트 패스 형태(:sub1:sub2:test)로 기술하거나 그냥 태스크 이름만 기술해도 된다.
* testNamePattern은 \*\*/testNamePattern\*.class형태로 기술한다.
* 패턴은 각 서버 프로젝트에 적용된다. 특정 서브 프로젝트에서 패턴에 매칭되는 테스트가 없으면 예외가 발생한다. 이 경우 패턴에 서브프로젝트를 명시할 수 있다.

gradle -Dtest.single=ThisUniquelyNamedTest test

gradle -Dtest.single=a/b/ test

gradle -DintegTest.single=\*IntegrationTest integTest

gradle -Dtest.single=:proj1:test:Customer build

gradle -DintegTest.single=c/d/ :proj1:integTest

### 테스트 감지

* Test 태스크는 컴파일된 클래스를 분석하여 테스트 클래스를 감지한다. 기본적으로 모든 \*.class 파일을 분석한다.
* 추상클래스는 실행 안한다.
* 상위 클래스까지 모두 분석한다.
* scanForTestClasses를 false로 하면 자동감지를 수행하지 않는다. 이 경우 명시적으로 포함/제외 시킨 클래스만 실행한다.
* scanForTestClasses=false이면서 포함/제외 클래스를 명시하지 않으면 기본적으로 \*\*/\*Tests.class와 \*\*/\*Test.class를 실행하고, \*\*/Abstract\*.class는 제외한다.

#### JUnit

* JUnit 3, 4 테스트 클래스
* TestCase, GroovyTestCase 상속
* @RunWith 어노테이션 적용
* @Test 어노테이션을 가진 메소드가 있는 클래스

#### TestNG

* @Test 어노테이션을 가진 메소드가 있는 클래스

### 테스트의 분리

* \*Test 와 \*IntegrationTest를 분리해서 실행하고자 하는 경우가 있을 수 있다.
* [Gradle goodness - Running Single Test 참조](http://java.dzone.com/articles/gradle-goodness-running-single)
* include와 exclude를 사용하고 [Test](http://www.gradle.org/docs/current/dsl/org.gradle.api.tasks.testing.Test.html)를 상속하는 또 다른 태스크를 만들어 지정한다.

test {

exclude '\*\*/\*IntegrationTest.class'

}

task integrationTest(type: Test, dependsOn: testClasses) {

description = 'Integration test'

group = 'verification'

include '\*\*/\*IntegrationTest.class'

testReportDir file("${buildDir}/reports/integration-test")

}

tasks.withType(Test) {

// Test 들의 공통 설정

useJUnit()

maxHeapSize '2048m'

jvmArgs '-XX:MaxPermSize=256m'

testLogging {

events 'started', 'passed'

}

}

### Convention Values

| **태스크 프라퍼티** | **타입** | **기본값** |
| --- | --- | --- |
| testClassesDir | File | sourceSets.test.output.classesDir |
| classpath | [FileCollection](http://www.gradle.org/docs/current/javadoc/org/gradle/api/file/FileCollection.html) | sourceSets.test.runtimeClasspath |
| testResultsDir | File | testResultsDir |
| testReportDir | File | testReportDir |
| testSrcDirs | List<File> | sourceSets.test.java.srcDirs |
| jvmArgs | List<String> | [], 문자열 배열로 JVM 옵션을 지정한다. |
| maxHeapSize | String | null, '256m' 형태 |
| systemProperty | 키, 값 | 키, 값 쌍을 인자로 테스트 수행 JVM의 시스템 프라퍼티 지정 |
| include | String[] | [], '\*\*/\*IntgrationTest.class', 'org/foo/\*\*', … |
| exclude | String[] | [], '\*\*/\*IntgrationTest.class', 'org/foo/\*\*', … |

### TestLogging

* [TestLoggingContainer](http://www.gradle.org/docs/current/dsl/org.gradle.api.tasks.testing.logging.TestLoggingContainer.html)
* test {
* testLogging {
* // set options for log level LIFECYCLE
* events "failed"
* exceptionFormat "short"
* showStandardStreams [true](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20true)
* // set options for log level DEBUG
* debug {
* events "started", "skipped", "failed"
* exceptionFormat "full"
* }
* }

}

### test 제외하기

* build 등을 할 때 테스트를 실행하지 않으려면 gradle build -x test 처럼 -x test 옵션을 준다.

### multi project 테스트 결과 모아보기

[Creating a unit test report for subprojects](http://www.gradle.org/docs/current/userguide/java_plugin.html#subProjectsTestReport)

subprojects {

apply plugin: 'java'

// Disable the test report for the individual test task

test {

reports.html.enabled = [false](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20false)

}

}

task testReport(type: TestReport) {

destinationDir = file("$buildDir/reports/allTests")

// Include the results from the `test` task in all subprojects

reportOn subprojects\*.test

}

## Jar

* [jar](http://gradle.org/docs/current/dsl/org.gradle.api.tasks.bundling.Jar.html) 태스크는 프로젝트의 클래스와 리소스를 포함하는 JAR 파일을 생성한다.
* 파일명 지정
* jar {
* archiveName = 'nameWhatIWant.jar'

}

### Manifest

* 모든 jar 혹은 war 객체는 [Manifest](http://www.gradle.org/docs/current/javadoc/org/gradle/api/java/archives/Manifest.html)manifest 프라퍼티가 있다.
* 압축 파일이 생성될 때 MANIFEST.MF 파일이 함께 저장된다.
* MANIFEST.MF 커스터마이징
* jar {
* manifest {
* attributes("Implementation-Title": "Gradle", "Implementation-Version": version)
* }

}

* 여러 jar에서 manifest 정보 공유
* ext.sharedManifest = manifest {
* attributes("Implementation-Title": "Gradle", "Implementation-Version": version)
* }
* task fooJar(type: Jar) {
* manifest = project.manifest {
* from sharedManifest
* }

}

* Manifest를 병합할 수 있다. 병합할 대상은 다른 Manifest 객체나 파일이 될 수 있다.
* task barJar(type: Jar) {
* manifest {
* attributes key1: 'value1'
* from sharedManifest, 'src/config/basemanifest.txt'
* from('src/config/javabasemanifest.txt', 'src/config/libbasemanifest.txt') {
* eachEntry { details ->
* [if](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20if) (details.baseValue != details.mergeValue) {
* details.value = baseValue
* }
* [if](http://www.google.de/search?q=site%3Agroovy.codehaus.org/%20if) (details.key == 'foo') {
* details.exclude()
* }
* }
* }
* }

}

* + Manifest는 from절에 기술한 순서에 따라 병합된다.
  + 동일 키가 존재할 경우 이후에 병합된 manifest것이 우선한다.
  + eachEntry 를 통해 [ManifestMergeDetails](http://www.gradle.org/docs/current/javadoc/org/gradle/api/java/archives/ManifestMergeDetails.html) 객체를 받아서 조정 가능하다.
  + 병합은 jar 생성시점 혹은 writeTo 아니면 effectiveManifest가 호출되는 시점에 늦은 초기화 방식으로 수행된다.
* manifest 기록

jar.manifest.writeTo("$buildDir/mymanifest.mf")

## JavaExec

Java 클래스를 실행할 때 [Ant를 사용하는 방법](http://kwonnam.pe.kr/wiki/gradle/ant)과 [JavaExec](http://gradle.org/docs/current/dsl/org.gradle.api.tasks.JavaExec.html)를 사용하는 방법이 있다. project.javaexec() 메소드를 호출해도 된다. javaexec 메소드는 Closure를 인자로 받는데 거기 들어가는 내용은 JavaExec 설정과 같다. [How to use in gradle javaexec with classpath dependency](https://discuss.gradle.org/t/how-to-use-in-gradle-javaexec-with-classpath-dependency/7479)

[Gradle Application Plugin](http://kwonnam.pe.kr/wiki/gradle/application)도 참조한다.

// 외부 의존성 지정 필요시

configurations {

newConfForJavaexec

}

dependencies {

newConfForJavaexec "xxxx:xxx:1.1"

// ...

}

task someTask(type: JavaExec) {

main = 'xxx.yyy.MainClass'

// 외부 의존성의 Java 클래스 실행시

classpath = configurations.newConfForJavaexec

// 외부 jar가 아닌 현재 프로젝트의 Java 클래스 실행시 - 1번 방식

classpath = configurations.runtime

classpath += sourceSets.main.output

// 외부 jar가 아닌 현재 프로젝트의 Java 클래스 실행시 - 2번 방식

classpath = sourceSets.main.runtimeClasspath

args '인자1', '인자2', .....

systemProperty 'simple.message', 'Hello '

}

* classpath는 [FileCollection](http://gradle.org/docs/current/javadoc/org/gradle/api/file/FileCollection.html) 객체이다.

### System Properties 전달

Gradle을 통해 실행되는 Java Application에 Gradle의 System Properties를그대로 전달하기 - [Gradle Goodness: Pass Java System Properties To Java Tasks](http://mrhaki.blogspot.kr/2015/09/gradle-goodness-pass-java-system.html)

// The run task added by the application plugin

// is also of type JavaExec.

tasks.withType(JavaExec) {

// Assign all Java system properties from

// the command line to the JavaExec task.

systemProperties [System](http://www.google.de/search?as_q=System&num=100&hl=en&as_occt=url&as_sitesearch=java.sun.com%2Fj2se%2F1%2E5%2E0%2Fdocs%2Fapi%2F).properties

}

풀버전

|  |
| --- |
| apply plugin: 'java' apply plugin: 'war' apply plugin: 'maven' apply plugin: 'eclipse' apply plugin: 'eclipse-wtp' apply plugin: 'idea'  group = 'omnifit\_cms' version = '3.5.0'   sourceCompatibility = 1.8 targetCompatibility = 1.8 // 소스 인코딩 지정방법 1 [compileJava, compileTestJava]\*.options\*.encoding = 'UTF-8' // 소스 인코딩 지정밥법 2 tasks.withType(JavaCompile) {  options.encoding = 'UTF-8' } compileJava {  options.fork = true  options.forkOptions.executable = "javac" // assumes that javac is on PATH  options.compilerArgs << "-XDignore.symbol.file"// does not exist 오류나면.. } war{  //archiveName = "${baseName}.${extension}"   archiveName = "cms.war"   } task explodedWar(type: Copy) {  description = 'Generate exploded war'  group = 'build'  dependsOn war.dependsOn    into "${buildDir}/exploded"  with war } war.doLast {  ant.unzip(src: war.archivePath, dest: "${buildDir}/exploded") } repositories {  maven { url "https://code.lds.org/nexus/content/groups/main-repo" }  maven { url "http://repo1.maven.org/maven2/" }  maven { url "http://www.egovframe.go.kr/maven/" }  maven { url "http://repo.maven.apache.org/maven2" } } dependencies {  compile(group: 'egovframework.rte', name: 'egovframework.rte.ptl.mvc', version:'3.5.0') { exclude(module: 'commons-logging')  }  compile group: 'egovframework.rte', name: 'egovframework.rte.psl.dataaccess', version:'3.5.0'  compile group: 'egovframework.rte', name: 'egovframework.rte.fdl.idgnr', version:'3.5.0'  compile group: 'egovframework.rte', name: 'egovframework.rte.fdl.property', version:'3.5.0'  compile group: 'javax.servlet', name: 'jstl', version:'1.2'  compile group: 'taglibs', name: 'standard', version:'1.1.2'  compile group: 'org.antlr', name: 'antlr', version:'3.5'  compile group: 'org.hsqldb', name: 'hsqldb', version:'2.3.2'  compile group: 'commons-dbcp', name: 'commons-dbcp', version:'1.4'  compile group: 'com.oracle', name: 'ojdbc6', version: '11.2.0.3'  compile group: 'net.sourceforge.jexcelapi', name: 'jxl', version: '2.6.12'  compile group: 'org.aspectj', name: 'aspectjweaver', version: '1.8.0'  /\*commons-io\*/  compile group: 'commons-io', name: 'commons-io', version: '2.4'  /\*poi\*/  compile group: 'org.apache.poi', name: 'poi', version: '3.14'   /\*groovy\*/  compile group: 'org.codehaus.groovy', name: 'groovy-all', version: '2.4.6'  /\*hibernate\*/  compile group : 'org.hibernate', name:'hibernate-core', version : '4.3.6.Final'  /\* lombok\*/  compile group: 'org.projectlombok', name: 'lombok', version: '1.16.8'  /\* jdbc log \*/  compile group: 'org.lazyluke', name: 'log4jdbc-remix', version: '0.2.7'    /\* httpClient \*/  compile group: 'org.apache.httpcomponents', name: 'httpclient', version: '4.5.2'  compile group: 'commons-httpclient', name: 'commons-httpclient', version: '3.1'    /\* junit \*/  compile group: 'junit', name: 'junit', version: '4.12'  compile group: 'org.hamcrest', name: 'hamcrest-library', version: '1.3'    /\* json \*/  compile group: 'com.bluelinelabs', name: 'logansquare', version: '1.1.0'  compile group: 'com.google.code.gson', name: 'gson', version: '2.6.2'   /\* tiles3 \*/  compile group: 'org.apache.tiles', name: 'tiles-core', version: '3.0.5'  compile group: 'org.apache.tiles', name: 'tiles-servlet', version: '3.0.5'  compile group: 'org.apache.tiles', name: 'tiles-jsp', version: '3.0.5'    /\* email \*/  compile group: 'javax.mail', name: 'mail', version: '1.4.7'    /\* fileUpLoad \*/  compile group: 'commons-fileupload', name: 'commons-fileupload', version: '1.2.2'  compile group: 'commons-net', name: 'commons-net', version: '3.3'    /\* spring-security \*/  compile group: 'org.springframework.security', name: 'spring-security-core', version: '4.0.4.RELEASE'  compile group: 'org.springframework.security', name: 'spring-security-config', version: '4.0.4.RELEASE'  compile group: 'org.springframework.security', name: 'spring-security-taglibs', version: '4.0.4.RELEASE'  compile group: 'org.springframework.security', name: 'spring-security-aspects', version: '4.0.4.RELEASE'    /\* server validation \*/  compile group: 'javax.validation', name: 'validation-api', version: '1.1.0.Final'  compile group: 'org.hibernate' , name: 'hibernate-validator', version: '5.2.4.Final'    /\* PDF LIB \*/  compile group: 'com.itextpdf', name: 'itextpdf', version: '5.0.6'  //json path  compile group: 'com.jayway.jsonpath', name: 'json-path', version: '2.2.0'   /\*spring dev toll\*/  //compile group: 'org.springframework', name: 'springloaded', version: '1.2.6.RELEASE'   providedCompile group: 'javax.servlet', name: 'javax.servlet-api', version:'3.1.0'  providedCompile 'org.apache.tomcat:tomcat-catalina:8.0.32'  compile project(':basic\_java')  compile project(':basic\_jsp')  compile project(':basic\_spring')  compile project(':omnifit\_common') }  //project(":basic\_java"){ //}   // 공통 library LinkSoruce //type은 1은 파일 2는 디렉토리이다. //http://kwonnam.pe.kr/wiki/gradle/eclipse eclipse {  project.name = 'cms'  String omnicns\_lib\_home = System.*getenv*().OMNICNS\_LIB.replaceAll('\\\\','/');  String omnicns\_omnifit\_home = System.*getenv*().OMNICNS\_OMNIFIT.replaceAll('\\\\','/');  println omnicns\_lib\_home  println omnicns\_omnifit\_home   project {  linkedResource name: 'basic\_java', type: '2', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_java/common/src/main/java'  linkedResource name: 'basic\_jsp', type: '2', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_jsp/common/src/main/java'  linkedResource name: 'basic\_spring', type: '2', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_spring/common/src/main/java'  linkedResource name: 'omnifit\_common', type: '2', locationUri: 'file://'+omnicns\_omnifit\_home+'/omnifit\_common/common/src/main/java'  linkedResource name: 'omnifit\_common\_resources', type: '2', locationUri: 'file://'+omnicns\_omnifit\_home+'/omnifit\_common/common/src/main/resources'  linkedResource name: 'src/main/webapp/js/sizzle.js', type: '1', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_jsp/common/src/main/webapp/js/sizzle.js'  linkedResource name: 'src/main/webapp/js/util.js', type: '1', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_jsp/common/src/main/webapp/js/util.js'  linkedResource name: 'src/main/webapp/js/jquery.extends.js', type: '1', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_jsp/common/src/main/webapp/js/jquery.extends.js'  linkedResource name: 'src/main/webapp/WEB-INF/tld/functions.tld', type: '1', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_jsp/common/src/main/webapp/WEB-INF/tld/functions.tld'  linkedResource name: 'src/main/webapp/WEB-INF/tld/tag.tld', type: '1', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_jsp/common/src/main/webapp/WEB-INF/tld/tag.tld'  }  classpath {  file {  withXml {  def node = it.asNode()  node.appendNode('classpathentry', [kind: 'src', path: 'basic\_java'])  node.appendNode('classpathentry', [kind: 'src', path: 'basic\_jsp'])  node.appendNode('classpathentry', [kind: 'src', path: 'basic\_spring'])  node.appendNode('classpathentry', [kind: 'src', path: 'omnifit\_common'])  node.appendNode('classpathentry', [kind: 'src', path: 'omnifit\_common\_resources'])  }  }  } }  allprojects { } //subprojects 는 공통이다 subprojects { } idea {  //module {  //print(configurations.integrationTestCompile);  //scopes.TEST.plus += [configurations.integrationTestCompile]  //scopes.TEST.plus += [configurations.integrationTestRuntime]  //testSourceDirs += file('some-extra-test-dir')  //sourceDirs += file('D:\\\\omnicns\\\\project\\\\lib\\\\basic\_java\\\\common\\\\src')  //generatedSourceDirs += file('D:\\omnicns\\project\\lib\\basic\_java\\common\\src')  //} } |
| setting.gradle  rootProject.name = 'omnifit' def modulesMap = [:] modulesMap['basic\_java'] = "${System.*getenv*().OMNICNS\_LIB}/basic\_java/common"; modulesMap['basic\_jsp'] = "${System.*getenv*().OMNICNS\_LIB}/basic\_jsp/common"; modulesMap['basic\_spring'] = "${System.*getenv*().OMNICNS\_LIB}/basic\_spring/common"; modulesMap['omnifit\_common']= "${System.*getenv*().OMNICNS\_OMNIFIT}/omnifit\_common/common"; modulesMap.each{ key, value->  println (rootProject.name+' '+key+"-------"+value)  include key  def projectDir = new File(rootDir, key)  if( !projectDir.exists() ) {  projectDir.mkdirs()  }  if (value?.trim()) {  project(":${key}").projectDir = new File(value)  } } |
| apply plugin: 'java' apply plugin: 'maven' apply plugin: 'war' apply plugin: 'eclipse' apply plugin: 'eclipse-wtp' apply plugin: 'idea' sourceCompatibility = 1.8 targetCompatibility = 1.8 // 소스 인코딩 지정방법 1 [compileJava, compileTestJava]\*.options\*.encoding = 'UTF-8' // 소스 인코딩 지정밥법 2 tasks.withType(JavaCompile) {  options.encoding = 'UTF-8' }   sourceSets {  main{  resources{  srcDirs = ['src/main/webapp']  exclude '\*\*/\*.js'  }  } }   jar {  archiveName='basic\_jsp.jar'  includeEmptyDirs = false    filesMatching('\*\*/\*.tld'){details ->  details.path = "META-INF/$details.name"  println(details.name)  };     /\* eachFile { details ->  println(details.name)  def filePattern = ~/${fileSubStr}/  if (filePattern.matcher(it.name).find()) {  details.path = "META-INF/$details.path"  }  }\*/ }     // In this section you declare where to find the dependencies of your project repositories {  maven { url "http://mesir.googlecode.com/svn/trunk/mavenrepo" }  maven { url "http://repo1.maven.org/maven2/" }  maven { url "http://repo.maven.apache.org/maven2" }   }  dependencies {  /\* jdbc log \*/  compile group: 'org.lazyluke', name: 'log4jdbc-remix', version: '0.2.7'  compile group: 'commons-codec', name: 'commons-codec', version:'1.10' //보안 base64를 이용하기 위해 추가함  compile group: 'com.google.code.gson', name: 'gson', version: '2.6.2'  compile group: 'net.sourceforge.jexcelapi', name: 'jxl', version: '2.6.12'  //compile group: 'javax.comm', name: 'comm', version: '2.0.3'  compile group: 'org.aspectj', name: 'aspectjweaver', version: '1.8.0'  providedCompile group: 'javax.servlet', name: 'javax.servlet-api', version:'3.1.0'  providedCompile 'org.apache.tomcat:tomcat-catalina:8.0.32'  /\*commons-io\*/  compile group: 'commons-io', name: 'commons-io', version: '2.4'  /\*poi\*/  compile group: 'org.apache.poi', name: 'poi', version: '3.14'  /\* httpClient \*/  compile group: 'org.apache.httpcomponents', name: 'httpcore', version: '4.4.5'  compile group: 'org.apache.httpcomponents', name: 'httpmime', version: '4.5.2'  compile group: 'org.apache.httpcomponents', name: 'httpclient', version: '4.5.2'   compile group: 'commons-httpclient', name: 'commons-httpclient', version: '3.1'    /\* quartz \*/ // compile group: 'org.quartz-scheduler', name: 'quartz', version: '2.2.3'  compile group: 'org.quartz-scheduler', name: 'quartz', version: '1.8.6'  // https://mvnrepository.com/artifact/javax.transaction/jta  compile group: 'javax.transaction', name: 'jta', version: '1.1'   /\*hibernate\*/  compile 'org.hibernate:hibernate-core:4.3.6.Final'  /\* email \*/  compile group: 'javax.mail', name: 'mail', version: '1.4.7'  //json path  compile group: 'com.jayway.jsonpath', name: 'json-path', version: '2.2.0'  compile project(':basic\_java') } eclipse {  project.name = 'basic\_jsp'  String omnicns\_lib\_home = System.getenv().OMNICNS\_LIB.replaceAll('\\\\','/');  String omnicns\_omnifit\_home = System.getenv().OMNICNS\_OMNIFIT.replaceAll('\\\\','/');  println omnicns\_lib\_home  println omnicns\_omnifit\_home    project {  linkedResource name: 'basic\_java\_common', type: '2', locationUri: 'file://'+omnicns\_lib\_home+'/basic\_java/common/src/main/java'  }    classpath {  file {  withXml {  def node = it.asNode()  node.appendNode('classpathentry', [kind: 'src', path: 'basic\_java\_common'])  }  }  } }  allprojects { } //subprojects 는 공통이다 subprojects { } idea{ }  // //project(":basic\_java") {} |

gulp 처리후 war묶기

|  |
| --- |
| war{  //archiveName = "${baseName}.${extension}"  exec{  workingDir webAppDir  commandLine 'node','node\_modules/gulp/bin/gulp.js','oneshot'  }  archiveName = "nhis.war"  } |
| gulpfile.js  **var** es = require('event-stream'); **var** gulp = require('gulp'); **var** concat = require('gulp-concat'); **var** connect = require('gulp-connect'); **var** templateCache = require('gulp-angular-templatecache'); **var** ngAnnotate = require('gulp-ng-annotate'); **var** uglify = require('gulp-uglify'); **var** minify = require('gulp-minify');   **var** scripts = require('./app.scripts.json');  **var** source = {  js: {  main: 'app/main.js',  src: [  // application config  'app.config.js',   // application bootstrap file  'app/main.js',   // main module  'app/app.js',   // module files   'app/\*\*/module.js',   // other js files [controllers, services, etc.]  'app/\*\*/!(module)\*.js',  ],  tpl: 'app/\*\*/\*.tpl.html'  },  css: {   src: [   'styles/css/bootstrap.min.css',   //'styles/css/dataTables.bootstrap.min.css',   'styles/css/font-awesome.min.css',   'styles/css/smartadmin-production-plugins.min.css',   'styles/css/smartadmin-production.min.css',   'styles/css/smartadmin-skins.min.css',  'styles/css/smartadmin-angular.css',  'plugin/datatables-extensions/datatables.min.css',  //'styles/css/main.css'  'styles/css/fullcalendar.min.css',  //'styles/css/main.css'  'styles/css/GGIJA200.css',  'styles/css/GGIJA300.css',  'styles/css/GGIJA400.css',  'styles/css/GGIJA600.css',  'styles/css/GGIJA700.css',  'styles/css/GGIJA800.css',  ]  },  login\_css: {   src: [   'styles/css/bootstrap.min.css',   'styles/css/font-awesome.min.css',   'styles/css/smartadmin-production-plugins.min.css',   'styles/css/smartadmin-production.min.css',   //'styles/css/main.css'  ]  },  vendor: {  src: [  'plugin/util.js'  ]  },  dest: {  src: [  'build/app.js'  ]  } };  **var** destinations = {  js: 'build',  css: 'build',  login\_css: 'front/css' };  gulp.task('build', **function**(){  **return** es.merge(gulp.src(source.js.src) , getTemplateStream())  //.pipe(depsOrder())  //.pipe(ngAnnotate())  //.pipe(uglify())  .pipe(concat('app.js'))  // .pipe(uglify({mangle: false}))  .pipe(gulp.dest(destinations.js))  //.pipe(rename('app.min.js'))    //.pipe(gulp.dest(destinations.js))  ; });  // min 파일을 생성하려면 gulp 를 다시 실행 필요 gulp.task('minify', **function**(){   **return** gulp.src(source.dest.src)  .pipe(minify({ext:{src:'.debug.js', min:'.min.js'}}))  .pipe(gulp.dest(destinations.js)); });  gulp.task('css', **function**(){  **return** gulp.src(source.css.src)  // .pipe(ngAnnotate())  //.pipe(uglify({mangle: false}))  .pipe(concat('cbis.min.css'))  .pipe(gulp.dest(destinations.css)); });  gulp.task('login\_css', **function**(){  **return** gulp.src(source.login\_css.src)  // .pipe(ngAnnotate())  //.pipe(uglify({mangle: false}))  .pipe(concat('login.min.css'))  .pipe(gulp.dest(destinations.login\_css)); });  gulp.task('vendor', **function**(){  **var** paths = [];  scripts.prebuild.forEach(**function**(script){  paths.push(scripts.paths[script]);  });  gulp.src(paths)  // .pipe(uglify())  .pipe(concat('vendor.js'))  .pipe(uglify({mangle: **false**}))  //.on('error', swallowError)  .pipe(gulp.dest(destinations.js)) });  gulp.task('watch', **function**(){  gulp.watch(source.js.src, ['build']);  gulp.watch(source.js.tpl, ['build']);  gulp.watch(source.css.src, ['css']);  gulp.watch(source.login\_css.src, ['login\_css']);  gulp.watch(source.vendor.src, ['vendor']);   //gulp.watch(source.dest.src, ['minify']);  });   gulp.task('connect', **function**() {  connect.server({  port: 18888  }); });  gulp.task('default', ['vendor', 'build', 'css', 'login\_css', 'watch', 'connect', 'minify'], **function** () {  **return** gulp.src('js/\*.js')  .pipe(uglify().on('error', **function**(e){  console.log(e);  }))  .pipe(gulp.dest('minjs')); });  **var** swallowError = **function**(error){  console.log(error.toString());  **this**.emit('end') };  **var** getTemplateStream = **function** () {  **return** gulp.src(source.js.tpl)  .pipe(templateCache({  root: 'app/',  module: 'app'  })) };  **var** getCssStream = **function** () {  **return** gulp.src(source.css.src); };   gulp.task('oneshot', ['vendor', 'build', 'css', 'login\_css'], **function** () {  setTimeout(**function**(){  console.log("--------");  process.exit(0);  }, 1000, 'shutdown')  **return** gulp.src('js/\*.js')  .pipe(uglify().on('error', **function**(e){  console.log(e);  }))  .pipe(gulp.dest('minjs')); }); |

명령어

|  |  |
| --- | --- |
| gradle clean | build path 클린하게 지움 |

시스템변수 넣기

${System.getenv().OMNICNS\_OMNIFIT}/omnifit\_common/common";

변수

<https://docs.gradle.org/current/userguide/war_plugin.html>

https://docs.gradle.org/current/userguide/java\_plugin.html

|  |  |
| --- | --- |
| $projectDir  또는  new File(".").absolutePath | 프로젝트 디렉토리 |
| archiveName = "${baseName}.${extension}" | war{안쪽에..}  nhis.1.0.2.war |
| ${buildDir} | 빌드 디렉토리 |
| ${project.sourceSets}  ${project.sourceSets.main}/webapp/ | 프로젝트 소스셋 main,test..등등  sourceSets {  main {  java {  srcDirs = ['src/java']  }  resources {  srcDirs = ['src/resources']  }  }  } |
| ${System.*getProperty*("user.dir") | user dir |
| ${webAppDir} | webAppDir 경로 |

실행시 아규먼트넣기

gradle run -Dexec.args="arg1 arg2 arg3"

execArgs = System.getProperty('exec.args')

if(execArgs)

args = execArgs.split()

Edited by Question Asker: using if does work but I had to change the syntax slightly.

if(System.getProperty("exec.args") != null) {

args System.getProperty("exec.args").split()

}

gradle build -DOMNIFIT\_COMMON=aa

if(System.getProperty('OMNIFIT\_COMMON')){

modulesMap['omnifit-common']= System.getProperty('OMNIFIT\_COMMON');

}else{

modulesMap['omnifit-common']= "${System.getenv().OMNICNS\_OMNIFIT}/omnifit\_common/common";

}