Федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки:

Системное и Прикладное Программное Обеспечение

(09.03.04 Программная инженерия)

Дисциплина «Основы программной инженерии»

Отчет

По лабораторной работе №3

Вариант №1577

Студент:

Карташев Владимир Сергеевич,
группа P3215

Практик:

Цопа Евгений Алексеевич

г. Санкт-Петербург, 2024 г.

Задание

Вместо Apache Ant - Gradle
Вместо Ant targets - Gradle advanced tasks
Сценарий тот же

Написать сценарий для утилиты Apache Ant, реализующий компиляцию, тестирование и упаковку в jar-архив кода проекта из лабораторной работы №3 по дисциплине "Веб-программирование".

Каждый этап должен быть выделен в отдельный блок сценария; все переменные и константы, используемые в сценарии, должны быть вынесены в отдельный файл параметров; MANIFEST.MF должен содержать информацию о версии и о запускаемом классе.

Сценарий должен реализовывать следующие цели (targets):

1. **compile** -- компиляция исходных кодов проекта.
2. **build** -- компиляция исходных кодов проекта и их упаковка в исполняемый jar-архив. Компиляцию исходных кодов реализовать посредством вызова цели **compile**.
3. **clean** -- удаление скомпилированных классов проекта и всех временных файлов (если они есть).
4. **test** -- запуск junit-тестов проекта. Перед запуском тестов необходимо осуществить сборку проекта (цель **build**).
5. **music** - воспроизведение музыки по завершению сборки (цель **build**).
6. **diff** - осуществляет проверку состояния рабочей копии, и, если изменения не касаются классов, указанных в файле параметров выполняет commit в репозиторий svn.

Выполнение

gradle.properties

```
# dependencies

servletApiVersion=2.5
jakartaWebApiVersion=9.1.0
primefacesVersion=13.0.3:jakarta
postgresqlVersion=42.7.3
eclipsePersistenceJpaVersion=4.0.2
junitJupiterApiVersion=5.11.0-M1
junitJupiterEngineVersion=5.11.0-M1

# project paths

webAppPath=src/main/webapp
webXmlPath=src/main/webapp/WEB-INF/web.xml
classesMainPath=build/classes/java/main
classesTestPath=build/classes/java/test
mainResourcesPath=src/main/resources
mainWebappPath=src/main/webapp

# jar

jarName=LAB3
jarVersion=1.0.0

# trunk working copy path

trunkPath=/home/studs/s373440/LAB3svn/main/trunk
```

build.gradle.kts

```
import java.io.ByteArrayOutputStream
import javax.sound.sampled.AudioSystem
import javax.sound.sampled.Clip
import javax.sound.sampled.DataLine

plugins {
    java
    war
}

repositories {
    mavenCentral()
}

// dependencies
val servletApiVersion: String by properties
val jakartaWebApiVersion: String by properties
val primefacesVersion: String by properties
val postgresqlVersion: String by properties
val eclipsePersistenceJpaVersion: String by properties
```

```
val junitJupiterApiVersion: String by properties
val junitJupiterEngineVersion: String by properties

// project paths
val webAppPath: String by properties
val webXmlPath: String by properties
val classesMainPath: String by properties
val classesTestPath: String by properties
val mainResourcesPath: String by properties
val mainWebappPath: String by properties

// jar
val jarName: String by properties
val jarVersion: String by properties

// trunk working copy path
val trunkPath: String by properties

dependencies {
    providedCompile("javax.servlet:servlet-api:${servletApiVersion}")

implementation("jakarta.platform:jakarta.jakartaee-web-api:${jakartaWebAp
iVersion}")
    implementation("org.primefaces:primefaces:${primefacesVersion}")
    implementation("org.postgresql:postgresql:${postgresqlVersion}")

implementation("org.eclipse.persistence:org.eclipse.persistence.jpa:${ecl
ipsePersistenceJpaVersion}")

testImplementation("org.junit.jupiter:junit-jupiter-api:${junitJupiterApi
Version}")

testImplementation("org.junit.jupiter:junit-jupiter-engine:${junitJupiter
EngineVersion}")
}

tasks.war {
    duplicatesStrategy = DuplicatesStrategy.EXCLUDE
    webAppDirectory = file(webAppPath)
    webXml = file(webXmlPath) // copies a file to WEB-INF/web.xml
}

tasks.register<Test>("lab-test") {
    dependsOn("lab-build")
    useJUnitPlatform()
}

tasks.register<JavaCompile>("lab-compile") {
    source = sourceSets["main"].java
    classpath = sourceSets["main"].compileClasspath
    destinationDirectory.set(file(classesMainPath))
    source = sourceSets["test"].java
    classpath = sourceSets["test"].compileClasspath
    destinationDirectory.set(file(classesTestPath))
}
```

```
tasks.register<Jar>("lab-build") {
    dependsOn("lab-compile")
    from(classesMainPath)
    from(mainResourcesPath)
    from(mainWebappPath)
    archiveBaseName.set(jarName)
    archiveVersion.set(jarVersion)
    manifest {
        attributes["Main-Class"] = "Main"
    }

    doLast {
        project.exec {
            commandLine = "echo 'Project have been build'".split(" ")
        }
    }
}

tasks.register("lab-clean") {
    doLast {
        delete("build")
        delete(".gradle")
        delete(".database")
    }
}

tasks.register("lab-music") {
    dependsOn("lab-build")
    doLast {
        val file = File("anime-ahh.wav")
        val audioInputStream = AudioSystem.getAudioInputStream(file)
        val audioFormat = audioInputStream.format
        val info = DataLine.Info(Clip::class.java, audioFormat)
        val line = AudioSystem.getLine(info) as Clip
        line.open(audioInputStream)
        line.start()
        while (line.frameLength > line.framePosition) {
            Thread.sleep(100)
        }
        line.close()
    }
}


tasks.register("lab-diff") {
    doLast {
        val immutableClasses = File("immutable_classes.txt").readLines()
        println("immutableClasses: $immutableClasses")
        val byteOut = ByteArrayOutputStream()
        project.exec {
            commandLine = "svn status".split(" ")
            standardOutput = byteOut
        }
        val output = byteOut.toByteArray().toString(Charsets.UTF_8)
        val changedFiles = output.lines().filter { it.startsWith("M ") ||
it.startsWith("! ") }
```

```
            .map { it.trim().split("\\s+".toRegex(), 2)[1] }
        println("changedFiles: $changedFiles")
        if (immutableClasses.any { immutableFile -> changedFiles.any {
changedFile -> changedFile.contains(immutableFile) } }) {
            println("Changes found in classes that should not be
changed")
        } else {
            val projectDir = file(trunkPath)
            println("Current directory: ${projectDir.absolutePath}")

            project.exec {
                workingDir = projectDir
                commandLine = "sh svn.sh".split(" ")
            }
        }
    }
}
```

immutable_classes.txt

```
src/main/java/beans/AreaCheckService.java
src/main/java/beans/Points.java
```

svn-init.sh

```
LAB3_dir=$(pwd)

cd ~
mkdir LAB3svn
cd LAB3svn || exit
svnadmin create repo
svn_dir=$(pwd)

mkdir main
cd "$svn_dir/main" || exit

svn mkdir "file://$svn_dir/repo/trunk" -m "Init trunk"
svn checkout "file://$svn_dir/repo/trunk"
cd "$svn_dir/main/trunk" || exit

cp -r /home/studs/s373440/LAB3/. /home/studs/s373440/LAB3svn/main/trunk

./gradlew lab-build

svn add --force .
svn commit -m "Init files of LAB3 project" --username=s373440
cd "$svn_dir/main/trunk" || exit
echo "Subversion initialization have been finished"
```

svn.sh

```
cd /home/studs/s373440/LAB3svn/main/trunk

svn commit -m 'Automated commit'
```

Проверка работоспособности таски diff

```
./gradlew lab-diff
svn log "file:///home/studs/s373440/LAB3svn/repo/trunk"

rm /home/studs/s373440/LAB3svn/main/trunk/src/main/java/beans/AreaCheckService.java

./gradlew lab-diff
svn log "file:///home/studs/s373440/LAB3svn/repo/trunk"
```

Вывод

Лабораторная работа была успешно переписана с использованием системы сборки Gradle вместо Apache Ant. Теперь вместо отдельных Ant targets используются более продвинутые Gradle tasks. Сценарий остался тем же, реализуя компиляцию, тестирование и упаковку кода проекта в jar-архив.