Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Основы программной инженерии

Лабораторная работа №3

Вариант № 33454123

Выполнили:

Кобик Никита Алексеевич

Маликов Глеб Игоревич

Группа № P3224

Преподаватель:

Коновалов Арсений Антонович

г. Санкт-Петербург
2024

1 Задание

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. **doc** – добавление в MANIFEST.MF MD5 и SHA-1 файлов проекта, а также генерация и добавление в архив javadoc по всем классам проекта.
6. **native2ascii** – преобразование native2ascii для копий файлов локализации (для тестирования сценария все строковые параметры необходимо вынести из классов в файлы локализации).
7. **xml** – валидация всех xml-файлов в проекте.
8. **music** – воспроизведение музыки по завершению сборки (цель **build**).
9. **scp** – перемещение собранного проекта по scp на выбранный сервер по завершению сборки. Предварительно необходимо выполнить сборку проекта (цель **build**)
10. **alt** – создаёт альтернативную версию программы с измененными именами переменных и классов (используя задание replace / replaceregexp в файлах параметров) и упаковывает её в jar-архив. Для создания jar-архива использует цель **build**.
11. **team** – осуществляет получение из svn-репозитория 4 предыдущих ревизий, их сборку (по аналогии с основной) и упаковку получившихся jar-файлов в zip-архив. Сборку реализовать посредством вызова цели **build**.
12. **history** – если проект не удаётся скомпилировать (цель **compile**), загружается предыдущая версия из репозитория git. Операция повторяется до тех пор, пока проект не удастся собрать, либо не будет получена самая первая ревизия из репозитория. Если такая ревизия найдена, то формируется файл, содержащий

результат операции diff для всех файлов, изменённых в ревизии, следующей непосредственно за последней работающей.

13. **diff** – осуществляет проверку состояния рабочей копии, и, если изменения касаются классов, указанных в файле параметров выполняет commit в репозиторий git.

14. **env** – осуществляет сборку и запуск программы в альтернативных окружениях; окружение задается версией java и набором аргументов виртуальной машины в файле параметров.

15. **report** – в случае успешного прохождения тестов сохраняет отчет junit в формате xml, добавляет его в репозиторий svn и выполняет commit.

2 Отчет

Файл build.xml

```
<?xml version="1.0"?>
<project name="web-lab3" basedir=".">
    <property file="build.properties"/>

    <path id="classpath">
        <fileset dir="${lib.dir}">
            <include name="**/*.jar"/>
        </fileset>
    </path>

    <taskdef resource="net/sf/antcontrib/antlib.xml">
        <classpath>
            <pathelement location="${lib.dir}/ant-contrib-1.0b3.jar"/>
        </classpath>
    </taskdef>

    <property name="message" value="" />

    <target name="init">
        <exec executable="git">
            <arg line="init"/>
        </exec>
        <exec executable="svnadmin">
            <arg line="create --fs-type fsfs ${svn.repo.dir}"/>
        </exec>

        <path id="repo.path">
            <pathelement location="${basedir}/${svn.repo.dir}"/>
        </path>
        <pathconvert property="repo.url" refid="repo.path">
            <map from="\" to="/"/>
            <map from=":" to=""/>
        </pathconvert>
        <exec executable="svn">
            <arg line="checkout file:///${repo.url} ${svn.checkout.dir}"/>
        </exec>

    </target>

    <target name="compile">
        <mkdir dir="${build.dir}"/>
        <javac srcdir="${src.dir}" destdir="${build.dir}"
            classpathref="classpath" includeantruntime="false"/>
    </target>

    <target name="build" depends="compile">
        <jar destfile="${dist.jar}" basedir="${build.dir}" compress="false"
            index="true" manifest="${manifest.file}"/>
```

```
    <war destfile="${dist.war}" webxml="${web.dir}/WEB-INF/web.xml">
        <fileset dir="${web.dir}">
            <include name="**/*.*"/>
        </fileset>
        <classes dir="${build.dir}"/>
        <lib dir="${lib.dir}"/>
    </war>
    <if>
        <not>
            <equals arg1="${message}" arg2="" />
        </not>

        <then>
            <echo message="${message}" />
        </then>
    </if>
</target>

<target name="clean">
    <echo message="Cleaning project..."/>
    <delete dir="${build.dir}"/>
    <delete dir="${dist.dir}"/>
    <delete dir="${checksums.dir}"/>
    <delete dir="${team.dir}"/>
    <delete dir="${i18n.ascii.dir}"/>
    <delete dir="${test.results.dir}"/>
    <echo message="Project cleaned successfully!"/>
</target>

<target name="test" depends="build">
    <tstamp>
        <format property="timestamp" pattern="MM-dd-yyyy_hh-mm-ss_aa"/>
    </tstamp>

    <echo message="Running tests..."/>

    <mkdir dir="${test.results.dir}/${timestamp}"/>
    <junitlauncher printsummary="yes">
        <classpath>
            <path refid="classpath"/>
            <pathelement location="${build.dir}"/>
        </classpath>
        <testclasses outputdir="${test.results.dir}/${timestamp}">
            <fileset dir="${build.dir}">
                <include name="**/*Test.class"/>
            </fileset>
            <listener type="legacy-brief" sendSysOut="true"
                sendSysErr="true"/>
            <listener type="legacy-xml" sendSysOut="true"
                sendSysErr="true"/>
        </testclasses>
```

```
    </junitlauncher>
    <echo message="Tests run successfully!"/>
</target>

<target name="javadoc" depends="compile">
    <javadoc destdir="${dist.dir}/javadoc">
        <fileset dir="${src.dir}" includes="**/*.java"/>
        <doctitle>${ant.project.name}</doctitle>
        <classpath refid="classpath"/>
    </javadoc>
    <echo message="Javadoc generated successfully!"/>
</target>

<target name="md5">
    <mkdir dir="${checksums.dir}"/>
    <checksum todir="${checksums.dir}" algorithm="MD5">
        <fileset dir="${src.dir}">
            <include name="**/*.java"/>
        </fileset>
    </checksum>
</target>

<target name="sha1">
    <mkdir dir="${checksums.dir}"/>
    <checksum todir="${checksums.dir}" algorithm="SHA-1">
        <fileset dir="${src.dir}">
            <include name="**/*.java"/>
        </fileset>
    </checksum>
</target>

<target name="checksums">
    <echo message="Generating checksums..."/>
    <trycatch>
        <try>
            <antcall target="md5" />

            <concat destfile="${checksums.md5.file}">
                <fileset dir="${checksums.dir}" includes="**/*.MD5"/>
            </concat>

            <checksum file="${checksums.md5.file}" property="md5"/>

            <echo message="MD5 checksum generated successfully!"/>
        </try>
        <catch>
            <echo message="Error generating MD5 checksum"/>
        </catch>
    </trycatch>
    <trycatch>
        <try>
```

```
            <antcall target="sha1" />
            <concat destfile="${checksums.sha1.file}">
                <fileset dir="${checksums.dir}" includes="**/*.SHA-1"/>
            </concat>
            <checksum file="${checksums.sha1.file}" property="sha1"/>
            <echo message="SHA-1 checksum generated successfully!"/>
        </try>
        <catch>
            <echo message="Error generating SHA1 checksum"/>
        </catch>
    </trycatch>
</target>

<target name="native2ascii">
    <native2ascii encoding="UTF8" src="${i18n.dir}"
        dest="${i18n.ascii.dir}" includes="*.properties"/>
</target>

<target name="xml">
    <xmlvalidate failonerror="no" lenient="yes">
        <fileset dir="${src.dir}">
            <include name="**/*.xml"/>
        </fileset>
    </xmlvalidate>
</target>

<target name="music" depends="build">
    <sound>
        <success source="Niik_Time_Again_mastered.wav"
            duration="30000"/>
        <fail source="Niik_Time_Again_mastered.wav"
            duration="100000"/>
    </sound>
</target>

<target name="scp" depends="build">
    <sshexec
            host="${scp.host}"
            port="${scp.port}"
            username="${scp.user}"
            password="${scp.password}"
            command="mkdir -p ${scp.dir}"
            trust="true"
    />
    <scp
            file="${dist.jar}"
            todir="${scp.user}@${scp.host}:${scp.dir}"
            password="${scp.password}"
            port="${scp.port}"
            trust="true"
    />
```

```
</target>

<target name="alt">
    <copy todir="${src.temp.dir}">
        <fileset dir="${src.dir}"/>
    </copy>

    <replace dir="${src.dir}" >
        <exclude name="**/*.jar"/>
        <replacefilter token="${src.user.naming}"
                       value="${alt.user.naming}"/>
    </replace>

    <replace dir="${src.dir}">
        <exclude name="**/*.jar"/>
        <replacefilter token="${src.area.naming}"
                       value="${alt.area.naming}"/>
    </replace>

    <replace dir="${src.dir}" >
        <exclude name="**/*.jar"/>
        <replacefilter token="${src.bean.naming}"
                       value="${alt.bean.naming}"/>
    </replace>


    <move todir="${src.dir}" includeemptydirs="false">
        <fileset dir="${src.dir}">
            <exclude name="**/*.jar"/>
        </fileset>
        <filtermapper >
            <replacestring from="${src.user.naming}"
                to="${alt.user.naming}" />
            <replacestring from="${src.area.naming}"
                to="${alt.area.naming}" />
            <replacestring from="${src.bean.naming}"
                to="${alt.bean.naming}" />
        </filtermapper>
    </move>

    <antcall target="clean"/>
    <antcall target="build"/>

    <delete dir="${src.dir}"/>

    <copy todir="${src.dir}">
        <fileset dir="${src.temp.dir}"/>
    </copy>

    <delete dir="${src.temp.dir}"/>
</target>
```

```
<target name="svn-commit">
    <copy todir="${svn.checkout.dir}">
        <fileset dir="${src.dir}"/>
    </copy>
    <exec executable="svn" dir="${svn.checkout.dir}">
        <arg line="add *"/>
    </exec>
    <exec executable="svn" dir="${svn.checkout.dir}">
        <arg line="commit -m 'Commit from Ant'"/>
    </exec>
</target>

<target name="team">
    <mkdir dir="${team.dir}"/>
    <property name="revision" value="HEAD"/>
    <for list="1,2,3,4" param="i">
        <sequential>
            <exec executable="svn" dir="${svn.checkout.dir}">
                <arg line="update -r ${revision}"/>
            </exec>
            <antcall target="build"/>
            <move file="${dist.jar}" tofile="${dist.dir}/build-@{i}.jar"/>
            <move file="${dist.dir}/build-@{i}.jar" todir="${team.dir}"/>
            <property name="revision" value="PREV"/>
        </sequential>
    </for>
    <echo message="Last revision: ${revision}"/>
    <zip destfile="${team.zip}" basedir="${team.dir}" />
</target>


<target name="history">
    <exec executable="git" outputproperty="git.head_revision">
        <arg line="rev-parse HEAD"/>
    </exec>
    <exec executable="git" outputproperty="git.tail_revision">
        <arg line="rev-list --max-parents=0 HEAD"/>
    </exec>

    <echo message="Head revision: ${git.head_revision}"/>
    <echo message="Tail revision: ${git.tail_revision}"/>

    <var name="git.new_revision" unset="true"/>
    <exec executable="git" outputproperty="git.new_revision">
        <arg line="rev-parse HEAD~1"/>
    </exec>

    <trycatch>
        <try>
```

```
            <antcall target="compile"/>
            <var name="compile.successful" unset="true"/>
            <property name="compile.successful" value="true"/>
            <echo message="Project␣compiled␣successfully!"/>

            <sequential>
                <exec executable="git" outputproperty="diff_output">
                    <arg line=
                        "diff␣${git.head_revision}␣${git.new_revision}"/>
                </exec>
                <echo file="diff_output.txt" message="${diff_output}"/>
            </sequential>

        </try>
        <catch>
            <echo message="Project␣compilation␣failed!"/>
            <echo message="Checkout␣revision:␣${git.new_revision}"/>
            <exec executable="git" dir="${project.dir}">
                <arg line="reset␣--hard␣${git.new_revision}"/>
            </exec>
            <var name="compile.successful" unset="true"/>
            <property name="compile.successful" value="false"/>
            <if>
                <equals arg1="${git.new_revision}"
                    arg2="${git.tail_revision}"/>
                <then>
                    <echo message="No␣more␣revisions␣to␣checkout"/>
                </then>
                <else>
                    <runtarget target="history"/>
                </else>
            </if>
        </catch>
    </trycatch>

    <exec executable="git" dir="${project.dir}">
        <arg line="checkout␣${git.head_revision}"/>
    </exec>
</target>


<target name="diff">
    <exec executable="git" outputproperty="vcs.git.diff_output">
        <arg value="diff" />
        <arg value="--name-only" />
    </exec>

    <condition property="vcs.git.should_commit" value="true">
        <resourcecount count="0">
            <intersect>
                <filelist id="vcs.git.files_to_commit"
```

```
                    files="${vcs.git.diff_output}"/>
                <filelist files="${vcs.git.diff_classes}" />
            </intersect>
        </resourcecount>
    </condition>

    <echoproperties prefix="vcs" />

    <pathconvert property="vcs.git.files_to_commit_separated"
        refid="vcs.git.files_to_commit" pathsep=" " />

    <if>
        <isset property="vcs.git.should_commit" />

        <then>
            <tstamp>
                <format property="timestamp"
                    pattern="MM.dd.yyyy hh:mm:ss aa"/>
            </tstamp>

            <exec executable="git">
                <arg value="add"/>
                <arg line="${vcs.git.files_to_commit_separated}" />
            </exec>

            <exec executable="git">
                <arg value="commit" />
                <arg value="-m" />
                <arg value="Commit changes from ${timestamp}" />
            </exec>

            <echo message="Committing changes"/>
        </then>

        <else>
            <echo message="No changes to commit"/>
        </else>
    </if>
</target>


<target name="env" depends="build">
    <java jar="${dist.war}" fork="true">
        <jvmarg line="${jvm.params}"/>
    </java>
</target>


<target name="report" depends="test, svn-commit">
    <tstamp>
        <format property="timestamp" pattern="MM.dd.yyyy_hh:mm:ss_aa"/>
```

```
        </tstamp>

        <mkdir dir="${test.results.dir}/report_${timestamp}"/>

        <junitreport todir="${test.results.dir}/report_${timestamp}">
            <fileset dir="${test.results.dir}">
                <include name="TEST-*.xml"/>
            </fileset>
        </junitreport>
        <copy todir="${svn.checkout.dir}/test/report">
            <fileset dir="${test.results.dir}"/>
        </copy>
        <exec executable="svn" dir="${svn.checkout.dir}">
            <arg line="add test/report"/>
        </exec>
        <exec executable="svn" dir="${svn.checkout.dir}">
            <arg line="commit -m 'Commit test report from Ant'"/>
        </exec>
    </target>
</project>
```

Файл build.properties

```
project.dir = .
build.dir = build

src.dir = src
src.temp.dir = tempsrc

src.user.naming = User
alt.user.naming = Enjoyer

src.area.naming = Area
alt.area.naming = Place

src.bean.naming = Bean
alt.bean.naming = Bob


lib.dir = lib
dist.dir = dist
dist.jar = ${dist.dir}/${ant.project.name}.jar
dist.war = ${dist.dir}/${ant.project.name}.war
checksums.dir = checksums
checksums.md5.file = ${checksums.dir}/md5.txt
checksums.sha1.file = ${checksums.dir}/sha1.txt

test.dir = ${src.dir}/test
test.results.dir = ${test.dir}/results

music.wav = Niik_Time_Again_mastered.wav
```

```
web.dir = ${src.dir}/main/webapp

i18n.dir = ${src.dir}/main/resources/i18n
i18n.ascii.dir = ${i18n.dir}/ascii

manifest.file = ${src.dir}/main/resources/META-INF/MANIFEST.MF

scp.host = se.ifmo.ru
scp.port = 2222
scp.user =
scp.password =
scp.dir = ~/OPI_LAB3

jvm.params = -Xmx256m

svn.repo.dir = svnRepo
svn.checkout.dir = ${svn.repo.dir}/mainDir

team.dir = team_folder
team.jar = ${team.dir}/${ant.project.name}.war
team.zip = ${team.dir}/here4versions.zip

vcs.git.diff_classes = ${src.dir}/main/java
```

3 Заключение

В ходе выполнения лабораторной работы был создан сценарий для утилиты Apache Ant, который реализует компиляцию, тестирование и упаковку в jar-архив кода проекта. Этот сценарий включает в себя множество целей, каждая из которых отвечает за определенный этап процесса сборки проекта.