Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №3**

по предмету «Методы и средства программной инженерии»

**Вариант 2189**

Выполнил:

Студент группы P33301

Хакимов Никита Минерахимович

Преподаватели:

Барсуков Илья Александрович

Санкт-Петербург

2023

1. Задание

Написать сценарий для утилиты [Apache Ant](http://ant.apache.org/" \t "_blank), реализующий компиляцию, тестирование и упаковку в jar-архив кода проекта из [лабораторной работы №3](https://se.ifmo.ru/courses/web#lab3) 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.

**Cценарий 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. 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.
7. Выполнение

**Разберем содержание файла build.xml:**

1. target compile

<target name="compile">  
<mkdir dir="${target}/${classes}" />  
<javac srcdir="${src}" destdir="${target}/${classes}" classpathref="classpath"  
 source="${version-source}" target="${version-target}"  
 includeantruntime="false" />  
</target>

1. target build

<target name="build" depends="compile">  
 <jar destfile="${target}/${jars}/${name}.jar"

basedir="${target}/${classes}">  
 <manifest>  
 <attribute name="Main-Class" value="${main-class}"/>  
 <attribute name="Package-Version" value="${package-version}"/>  
 </manifest>  
 </jar>  
 </target>

1. target clean

<target name="clean">  
 <delete dir="${target}" />  
</target>

1. target test

<target name="compile-test" depends="compile">  
 <mkdir dir="${target}/${classes-test}" />  
 <javac srcdir="${test}" destdir="${target}/${classes-test}"   
 source="${version-source}" target="${version-target}"  
 classpath="${target}/${classes}" classpathref="classpath"/>  
</target>

<target name="test" depends="compile-test">  
 <junit printsummary="on" fork="true" haltonfailure="yes">  
 <classpath>  
 <pathelement location="${target}/${classes}" />  
 <pathelement location="${target}/${classes-test}" />  
 <pathelement location="${junit}" />  
 <pathelement location="${junit-harmcrest-core}" />  
 </classpath>  
 <formatter type="plain" usefile="no" />  
 <batchtest>  
 <fileset dir="${test}">  
 <include name="\*\*/Test\*.java" />  
 </fileset>  
 </batchtest>  
 </junit>  
</target>

1. target doc

<target name="doc" depends="build">  
<checksum algorithm="SHA-1" file="${target}/${jars}/${name}.jar" property="sha1.digest" />  
<checksum algorithm="MD5" file="${target}/${jars}/${name}.jar" property="md5.digest" />  
   
<javadoc sourcepath="${src}"   
 destdir="${target}/${docs}" classpathref="classpath" >  
 </javadoc>  
  
<jar update="true" destfile="${target}/${jars}/${name}.jar">  
 <manifest>  
 <attribute name="MD5" value="${sha1.digest}"/>  
 <attribute name="SHA-1" value="${md5.digest}"/>  
 </manifest>  
 <zipfileset dir="${target}/${docs}" prefix="docs"/>  
 </jar>  
</target>

1. target team

<target name="team" depends="compile">  
 <exec executable="svn" outputproperty="svn-log">  
 <arg value="log"/>  
 <arg value="-l"/>  
 <arg value="4"/>  
 <arg value="--xml"/>  
 </exec>  
 <for list="1,2,3,4" param="i" delimiter=",">  
 <sequential>  
 <echo message="@{i}" />  
 <propertyregex property="next" override="true" input="${svn-log}"  
 regexp="(revision=&quot;[0-9]+&quot;)" select="\1" global="true"/>  
 <propertyregex property="svn-log" override="true" input="${svn-log}"  
 regexp="${next}" replace="" global="false"/>  
 <propertyregex property="next" override="true" input="${next}"  
 regexp="revision=&quot;([0-9]+)&quot;" select="\1" global="false"/>  
 <exec executable="svn">  
 <arg value="co"/>  
 <arg value="-r${next}"/>  
 <arg value="${svn-repo}"/>  
 <arg value="${user.dir}/${tmp-team}/"/>  
 </exec>  
 <antcall target="build">  
 <param name="src" value="${user.dir}/${tmp-team}/${src}" />  
 <param name="jars" value="tmp-jars/" />  
 <param name="name" value="revision${next}" />  
 </antcall>  
 <delete dir="${tmp-team}" />  
 </sequential>  
 </for>  
 <zip destfile="${target}/${jars}/team.zip">  
 <zipfileset dir="${target}/tmp-jars/" includes="\*.jar" />  
 </zip>  
 <delete dir="${target}/tmp-jars" />  
</target>

**Дополнительные target’ы вне рамках задания:**

1. target war

<target name="war" depends="compile">  
 <war destfile="${target}/wars/${name}.war" webxml="${webXML}">  
 <fileset dir="${web}"/>  
 <lib dir="${lib}" />  
 <classes dir="${target}/${classes}"/>  
 </war>  
 </target>

1. target deploy

<target name="deploy" depends="war, test">  
 <copy file="${target}/wars/${name}.war" todir="${deploy-local}" overwrite="true"/>  
</target>

1. target deploy-remote

<target name="deploy-remote" depends="war, test">  
 <scp file="${target}/wars/${name}.war" trust="true"  
 todir="${remote-user}@${deploy-remote}"  
 port="${remote-port}" password="${remote-password}" />  
</target>

1. Реализация Unit-тестов

import beans.MainBean;  
import org.junit.Test;  
import org.junit.runner.RunWith;  
  
import static org.junit.Assert.assertTrue;  
import static org.junit.Assert.assertFalse;  
  
public class TestMainBean {  
  
 @Test  
 public void testBorders(){  
 assertTrue(MainBean.check(-1.6,1.2,2));  
 assertTrue(MainBean.check(-1.2,1.6,2));  
 assertTrue(MainBean.check(-4,0,4));  
 assertTrue(MainBean.check(-4,-1,4));  
 assertTrue(MainBean.check(-4,-2,4));  
 assertTrue(MainBean.check(-2,-2,4));  
 assertTrue(MainBean.check(0,-2,4));  
 assertTrue(MainBean.check(0,0,4));  
 assertTrue(MainBean.check(4,0,4));  
 assertTrue(MainBean.check(2,1,4));  
 assertTrue(MainBean.check(0,2,4));  
 assertTrue(MainBean.check(0,4,4));  
 }  
   
 @Test  
 public void testInside(){  
 assertTrue(MainBean.check(-2,2,4));  
 assertTrue(MainBean.check(1,1,4));  
 assertTrue(MainBean.check(-2,-1,4));  
 assertTrue(MainBean.check(0,0,4));  
 }  
   
 @Test  
 public void testOutside(){  
 assertFalse(MainBean.check(-3.442,2.05,4));  
 assertFalse(MainBean.check(-4.0001,0,4));  
 assertFalse(MainBean.check(0.0001,-0.0001,4));  
 assertFalse(MainBean.check(2,-1,4));  
 assertFalse(MainBean.check(-10,10,4));  
 assertFalse(MainBean.check(-10,-10,4));  
 assertFalse(MainBean.check(10,-10,4));  
 assertFalse(MainBean.check(10,10,4));  
 assertFalse(MainBean.check(4.0001,0,4));  
 assertFalse(MainBean.check(4,-0.0001,4));  
 }  
}

1. Вывод

В ходе выполнения данной лабораторной работы я вспомнил, как пользоваться фреймворком Junit, вспомнил классификацию тестов. Познакомился с замечательной системой сборки Ant и ее чудесному преимуществу: написанию итеративного кода прямо в xml!