Федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский государственный университет

информационных технологий, механики и оптики

Факультет прикладной информатики и компьютерных технологий

Основы Программной Инженерии

Лабораторная работа №3

Вариант 529

Группа: P3218

Студент: Петкевич Константин

Преподаватель: Харитонова А.Е.

г. Санкт-Петербург

Задание

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

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. **Xml** - валидация всех xml-файлов в проекте.
6. **Team** - осуществляет получение из svn-репозитория 3 предыдущих ревизий, их сборку (по аналогии с основной) и упаковку получившихся jar-файлов в zip-архив. Сборку реализовать посредством вызова цели **build**.

Файл build.properties

build.dir=build

build.classes.java=${build.dir}/class/java

build.classes.test=${build.dir}/class/test

classes.dir = ${build.dir}/class/java

src=src

src.java=${src}/Lab3

src.test=${src}/test

junit=${src}/junit-4.12.jar

hamcrest=${src}/hamcrest-core-1.3.jar

build.jar=${build.dir}/dist.jar

mainClass = Work3

blabla=blabla

sdfds=kdsf

dsf=tdsf

Файл build.xml

```
<project name="JavaLab3" default="build" basedir=".">
  <property file="build.properties"/>

  <target name="make.dirs" description="Create required dirs">
    <echo>Creating dirs...</echo>
    <mkdir dir="${build.dir}"/>
    <mkdir dir="${build.classes.java}"/>
  </target>

  <target name="compile" depends="make.dirs" description="Compile the files" >
    <echo>Compiling...</echo>
    <javac includeantruntime="false" destdir="${build.classes.java}">
      <src path="${src.java}"/>
    </javac>
  </target>

  <target name="build" depends="compile" description="Compile the files and pack them in JAR archive" >
    <echo>Building the project...</echo>
    <jar destfile="${build.jar}" basedir="${build.classes.java}">
      <manifest>
        <attribute name="Main-Class" value="Lab3.Work3"/>
        <attribute name="Created-By" value="Konstantin"/>
      </manifest>
    </jar>
  </target>

  <target name="run" depends="build">
    <echo>Running application...</echo>
    <java jar="${build.jar}" fork="true"/>
  </target>
  <target name="clean" description="Deleting excess files">
    <echo>Deleting excess files...</echo>
    <delete dir="${build.classes.java}"/>
  </target>
  <target name="make.test.dirs" description="Create dirs for tests">
    <echo>Creating dirs for tests...</echo>
    <mkdir dir="${build.classes.test}"/>
  </target>

  <target name="compile.tests" depends="build, make.test.dirs">
    <echo>Compiling tests...</echo>
    <javac includeantruntime="false" destdir="${build.classes.test}">
      <src path="${src.test}"/>
      <classpath>
        <pathelement location="${junit}"/>
        <pathelement location="${build.classes.java}"/>
      </classpath>
    </javac>
  </target>

  <target name="test" description="Run tests" depends="build, compile.tests">
    <echo>Running the tests...</echo>
```

```
  <junit printsummary="true" haltonerror="yes" haltonfailure="yes" fork="yes">
    <formatter type="plain" usefile="false"/>
    <test name="test.MyTest"/>
    <classpath>
      <pathelement location="${junit}"/>
      <pathelement location="${build.classes.java}"/>
      <pathelement location="${build.classes.test}"/>
      <pathelement location="${hamcrest}"/>
    </classpath>
  </junit>
</target>

<target name="xml" description="Validate all xml files in project" >
  <echo>Validating xmls...</echo>
  <xmlvalidate failonerror="false" lenient="true">
    <fileset dir="." includes="**/*.xml"/>
    <attribute name="http://xml.org/sax/features/validation" value="true"/>
    <attribute name="http://apache.org/xml/features/validation/schema"  value="true"/>
    <attribute name="http://xml.org/sax/features/namespaces" value="true"/>
  </xmlvalidate>
</target>
<target name="jar" description="Build project">
  <jar destfile="${dfile}" basedir="${bdir}">
    <manifest>
      <attribute name="Main-Class" value="${mainClass}" />
    </manifest>
  </jar>
</target>

<target name="team">
  <echo>Rollback 'till 1st rev if compilation failed...</echo>
  <exec executable="svn">
    <arg value="update"/>
  </exec>
  <exec executable="svn">
    <arg value="update"/>
    <arg value="-r"/>
    <arg value="PREV"/>
  </exec>
  <antcall target="jar">
    <param name = "dfile" value = "${build.dir}/lab3r-1"/>
    <param name = "bdir" value = "${classes.dir}"/>
  </antcall>
  <exec executable="svn">
    <arg value="update"/>
    <arg value="-r"/>
    <arg value="PREV"/>
  </exec>
  <antcall target="jar">
    <param name = "dfile" value = "${build.dir}/lab3r-2"/>
    <param name = "bdir" value = "${classes.dir}"/>
  </antcall>
  <exec executable="svn">
```

```
    <arg value="update"/>
    <arg value="-r"/>
    <arg value="PREV"/>
   </exec>
   <antcall target="jar">
    <param name = "dfile" value = "${build.dir}/lab3r-3"/>
    <param name = "bdir" value = "${classes.dir}"/>
   </antcall>
   <exec executable="svn">
    <arg value="update"/>
   </exec>
  </target>
</project>
```

Java-class с тестами

```
public class MyTest {
   @Test
   public void testPointContaining() {
      Shape shape = new Shape(10.0);
      Circle circle = new Circle(shape.getR());
      Square square = new Square(shape.getR());
      Triangle triangle = new Triangle(shape.getR());
      shape.addShapes(circle, square, triangle);
      Assert.assertTrue(shape.checkVertice(new Vertice(0, 0)));
      Assert.assertFalse(shape.checkVertice(new Vertice(-3, 3)));
      Assert.assertFalse(shape.checkVertice(new Vertice(-1, 1)));
      Assert.assertTrue(shape.checkVertice(new Vertice(3, -3)));
   }

   @Test
   public void testShapeCreatingPositive() {
      Assert.assertNotNull(new Shape(5));
      Assert.assertNotNull(new Shape(100));
   }
}
```

Вывод: в результате выполнения данной лабораторной работы я познакомился с системой автоматической сборки Apache Ant и декларативным подходом к процессу описания сборки проекта. В процессе выполнения мною был создан класс для тестирования моей программы. С помощью такого тестирования достаточно легко отслеживать, чтобы при изменении каких-либо частей кода (добавлении нового функционала) не ломалась старая функциональность. Такое тестирование очень хорошо работает при разработке в команде.