Лабораторная работа. VisualUseCase

Цель работы

Получение опыта работы в группе:

- Выбор партнёров (группы по 3 — 6 человек), распределение ролей;
- Выбор и формулирование задания;
- Проведение мозгового штурма и оформление его результатов;
- Составление User Stories и Use Cases;
- Работа в среде VisualUseCase.

План работ

1) Поделиться на группы по 3 - 6 человек каждая.
2) Предложить для проектирования некоторую программную систему (далее система) или выбрать из предложенных.
3) Провести мозговой штурм с целью формулирования требований предъявляемых к разрабатываемой системе, оформить его результаты аналогично тому, как это было показано на лекции.
4) Записать User Stories для выбранной системы. При оформлении записей следует пользоваться рекомендациями, приведёнными в лекциях.
5) Описать Use Cases для выбранной системы. При оформлении описаний следует использовать рекомендации, приведённые в лекциях.
6) Ознакомится со средой VisualUseCase.
7) Ввести в среду VisualUseCase написанные ранее User Stories и User Cases.
8) Сформировать в среде VisualUseCase отчёты «Use Case Flows Tabular Style» и « Use Case Flows with Extension Names below each step». Продемонстрировать их в электронном виде преподавателю (в общем отчёте по лабораторной работе).
9) Все действия пп.1 — 8 сопровождать при документировании краткими комментариями и скриншотами — результатами выполнения данных пунктов.
10) Итогом выполнения работы должен стать отчёт о выполнении всех пунктов данного плана. Отчёт оформляется в электронном виде и демонстрируется преподавателю в конце занятия. Преподаватель оставляет за собой право потребовать выполнить какой-либо пункт данного плана повторно, для того, чтобы воочию убедится в правильности выполнения того или иного пункта и/или для установления действительного понимания студентом того, что было им выполнено в ходе лабораторной работы.

Примеры вариантов проектируемых систем

1) БД + GUI (а) − web либо б) − desktop) справочник организаций.
2) БД + GUI (а) − web либо б) − desktop) сотрудники организации.
3) БД + GUI (а) − web либо б) − desktop) регистрации входящих документов.
4) БД + GUI (а) − web либо б) − desktop) регистрации исходящих документов.
5) БД + GUI (а) − web либо б) − desktop) регистрации внутренних документов.
6) БД + GUI (а) − web либо б) − desktop) регистрации приказов и распоряжений.
7) БД + GUI (а) − web либо б) − desktop) регистрации договоров.
8) БД + GUI (а) − web либо б) − desktop) регистрации обращений граждан.
9) Записная книжка, «напоминалка», для телефона, использовать БД.
10) Записная книжка, «напоминалка», для ПК, использовать БД + GUI (а) − web либо б) −

desktop).

11) Программа для учёта личных расходов, для ПК, использовать БД + GUI (а) − web либо б) − desktop).
12) Программа для учёта личных расходов, для телефона, использовать БД.

Лабораторная работа. Apache Ant

Цель работы

Знакомство с автоматизированной системой сборки ПО Apache Ant.

План работ

1) Повторить материалы лекции посвящённой обзору автоматизированной системы сборки ПО Apache Ant.
2) Изучить пример использования Apache Ant, который приводится в приложении к лабораторной работе.
3) Повторить действия, которые описываются в данном в приложении примере.
4) Продемонстрировать преподавателю результат проделанной работы.

Приложение к лабораторной работе

Далее приводится Tutorial: Hello World with Apache Ant.

Дано:

- Расположение исходных файлов: d:\Temp\ant\ant_projects\src\ HelloWorld.java
- Расположение файла build.xml: d:\Temp\ant\ant_projects\build.xml

Необходимо:

- Требуется скомпилировать файл HelloWorld.java в директорию d:\Temp\ant\ant_projects\build\classes\
- Требуется создать jar-файл из файла с байт-кодом полученного после компиляции файла HelloWorld.java.

Листинг файла HelloWorld.java:

```
package Example;
class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello World!");
   }
}
```

Далее приводится перечень команд для выполнения необходимых требований «вручную», из терминала (консоль, cmd, в ОС Windows):

```
cd d:\Temp\ant\ant_projects\
md build\classes
javac -sourcepath src -d build\classes src\Example\HelloWorld.java
java -cp build\classes Example.HelloWorld

echo Main-Class: Example.HelloWorld>myManifest
```

```
md build\jar
jar cfm build\jar\HelloWorld.jar myManifest -C build\classes .
java -jar build\jar\HelloWorld.jar
```

Теперь выполнение всех приведённых выше команд автоматизируем с помощью Ant. Опишем файл build.xml.

```
<project>
   <target name="clean">
      <delete dir="build"/>
   </target>
   <target name="compile">
      <mkdir dir="build/classes"/>
      <javac srcdir="src" destdir="build/classes"/>
   </target>
   <target name="jar">
      <mkdir dir="build/jar"/>
      <jar destfile="build/jar/HelloWorld.jar" basedir="build/classes">
         <manifest>
            <attribute name="Main-Class" value="oata.HelloWorld"/>
         </manifest>
      </jar>
   </target>
   <target name="run">
      <java jar="build/jar/HelloWorld.jar" fork="true"/>
   </target>
</project>
```

Теперь мы можем скомпилировать, запаковать и выполнить наше простое приложение:

```
ant compile
ant jar
ant run
```

В данном примере запуск осуществляется из директории d:\Temp\ant\ant_projects\ с помощью команд вида:

```
d:\Temp\ant\apache-ant-1.9.7\bin\ant.bat compile
d:\Temp\ant\apache-ant-1.9.7\bin\ant.bat jar
d:\Temp\ant\apache-ant-1.9.7\bin\ant.bat run
```

Или сразу всех вместе:

```
d:\Temp\ant\apache-ant-1.9.7\bin\ant.bat compile jar run
```

В результате получиться нечто следующее:

```
Buildfile: D:\Temp\ant\ant_projects\build.xml

compile:
```

```
    [javac] D:\Temp\ant\ant_projects\build.xml:7: warning: 'includeantruntime' was not
set, defaulting to build.sysclasspath=last; set to false for repeatable builds

jar:

run:

    [java] Hello World!

BUILD SUCCESSFUL
Total time: 0 seconds
```

На практике можно существенно улучшить описание файла build.xml путём применения зависимостей при выполнении задач, свойств и атрибутов. Усовершенствованный с учётом этого файл build.xml приводится ниже:

```
<project name="HelloWorld" basedir="." default="main">
  <property name="src.dir"     value="src"/>
  <property name="build.dir"   value="build"/>
  <property name="classes.dir" value="${build.dir}/classes"/>
  <property name="jar.dir"     value="${build.dir}/jar"/>
  <property name="main-class"  value="Example.HelloWorld"/>
  <target name="clean">
    <delete dir="${build.dir}"/>
  </target>
  <target name="compile">
    <mkdir dir="${classes.dir}"/>
    <javac srcdir="${src.dir}" destdir="${classes.dir}"/>
  </target>
  <target name="jar" depends="compile">
    <mkdir dir="${jar.dir}"/>
    <jar destfile="${jar.dir}/${ant.project.name}.jar" basedir="${classes.dir}">
      <manifest>
        <attribute name="Main-Class" value="${main-class}"/>
      </manifest>
    </jar>
  </target>
  <target name="run" depends="jar">
    <java jar="${jar.dir}/${ant.project.name}.jar" fork="true"/>
  </target>
  <target name="clean-build" depends="clean,jar"/>
  <target name="main" depends="clean,run"/>
</project>
```

В результате именование директорий будет идти от той директории, где лежит файл build.xml, main - задача, которая выполняется по умолчанию. Доступны задачи: main, clean, compile, jar, run, clean-build. Определены и добавлены зависимости в выполнении задач.

После выполнения команды:

```
d:\Temp\ant\apache-ant-1.9.7\bin\ant.bat
```

Мы получим нечто следующее:

```
Buildfile: D:\Temp\ant\ant_projects\build.xml

clean:
   [delete] Deleting directory D:\Temp\ant\ant_projects\build

compile:
    [mkdir] Created dir: D:\Temp\ant\ant_projects\build\classes
    [javac] D:\Temp\ant\ant_projects\build.xml:12: warning: 'includeantruntime'
was not set, defaulting to build.sysclasspath=last; set to false for repeatable
builds
    [javac] Compiling 1 source file to D:\Temp\ant\ant_projects\build\classes

jar:
    [mkdir] Created dir: D:\Temp\ant\ant_projects\build\jar
      [jar] Building jar: D:\Temp\ant\ant_projects\build\jar\HelloWorld.jar

run:
     [java] Hello World!

main:

BUILD SUCCESSFUL
Total time: 1 second
```