Sistemi informativi per la gestione d'azienda 2013

LABORATORIO
ESERCITAZIONE 1

Info

- Contacts
  - sandro.labruzzo@isti.cnr.it
  - claudio.atzori@isti.cnr.it
- WEB sites
  - http://www.nmis.isti.cnr.it/casarosa/SIA/

Development Environment

- Java
  - JDK >= 1.5
  - Download and install from: http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Eclipse
  - Eclipse >= 3.6
  - Download and install from http://www.eclipse.org

Eclipse basics

- The Eclipse workbench
- Project Explorer
- Editors
  - Java
  - Text
  - More plugins (XML, C/C++, etc)
- Output and error tabs

Creating a new project

- File → New → Java Project
- Choose the project folder and name
- .project automatically created
- Create a new package:
  - Logical way to organize your classes by functionalities
  - Usually hierarchical packages:
    - it.unipi.ing.sia

Hello world on Eclipse

- File → New → Java Class
  - With static main method

```
public static void main(String[] args){

     System.out.println(Hello world!”);

}
```

Compiling

- Shell:
  - `javac <pathToClass>.java`
  - Class files in sources folder
  - To change class file destination: `-d <directory>`
- From eclipse:
  - Automatic compilation:
    check Project → Build Automatically
  - Class files in output folder:
    right click on project → BuildPath → Configure BuildPath → Sources

Running Hello World

- From the shell:
  - `cd bin`
  - `java <packagename>.<mainClassName>`
- From eclipse:
  - Run → Run as…
  - The green “play” button
  - Output on the console tab

Debugging on Eclipse

- Breakpoints
  - Instruction to the JVM to stop at a certain point of execution
- Add a breakpoint:
  - Double click on the left bar
- Start debugging:
  - Run → Debug As…
  - Bug button

Advanced project compilation

Automatizing compilation and dependency management:
ANT and Maven

Introduction to Ant and Maven

- Apache (Java) projects for build process automation:
  - ANT: http://ant.apache.org/
  - MVN: http://maven.apache.org/
- Compilation
- Packaging
- Execution
- Other

Setting up your environment

- Check:
  - java -version
  - ant -version
  - mvn -version
- Linux, MacOS: You should be ok! ☺

Setting up your environment variables (Windows)

- Set the following variables
  - via Control Panel → System
  - Risorse del computer →Proprieta' →Avanzate → Variabili d'ambiente
- **ANT_HOME**: set ANT_HOME=c:\ant
- **M2_HOME**: set M2_HOME=c:\maven
- **JAVA_HOME**: set JAVA_HOME=c:\jdk_1.6.12
- **PATH**: set PATH=%PATH%;%JAVA_HOME%\bin;%ANT_HOME%\bin;%M2_HOME%\bin

ANT

- Portability:
  - Platform independent commands
  - Back/forward slash directory
  - Class path separators: “:” or “;”
- build.xml
  - **Tasks** (commands) to execute by ANT to obtain a given **target**
- Execution:
  - `ant`: executes default target
  - `ant target1`: executes the specified target
  - `ant target1 target2 target3`: executes the specified target sequentially

Ant: some tasks

- `ant`
  Esegue un altro build file
- `antcall`
  Chiama un altro target
- `copy`
  Copia file o directory
- `delete`
  Cancella file o directory
- `mkdir`
  Crea una directory
- `exec`
  Esegue un comando di sistema
- `get`
  Recupera un file da una URL specificata
- `java`
  Esegue una classe Java
- `zip` Produce un file zip
- `jar` Produce un file jar

Ant: project structure

```
Project dir
|-- build.xml
|-- lib/
|    |-- …
|-- src/
|    |-- …
\-- build/
     |-- classes/
     |    |-- …
     \-- jar/
          |-- project.jar
```

Ant: the build.xml file

```
<project name="HelloWorld" basedir=".">
    …
    <target name="…">…</target>
    <target name="…">…</target>
    …
</project>
```

Using properties in the build.xml

```
<project name="HelloWorld" basedir=".”>
      <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="test.HelloWorld"/>
      <property name="lib.dir" value="lib"/>
</project>
```

Classpath definition

```
<project name="HelloWorld" basedir=".”>
      …
      <path id="classpath”>
            <fileset dir="${lib.dir}“
                  includes="**/*.jar"/>
      </path>
      …
</project>
```

The clean target

```
<project name="HelloWorld" basedir=".">
    …
    <target name="clean">
        <delete dir="${build.dir}"/>
    </target>
    …
</project>
```

The compile target

```
<project name="HelloWorld" basedir=".”>
      …

      <target name="compile”>
            <mkdir dir="${classes.dir}"/>
            <javac srcdir="${src.dir}”
                  destdir="${classes.dir}”
                  classpathref="classpath"/>
      </target>
      …

</project>
```

Jar target with depends

```
<project name="HelloWorld" basedir=".”>
...
<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>
...
</project>
```

Run target

```
<project name="HelloWorld" basedir=".”>
…

…
</project>
```

Main target

```
<project name="HelloWorld" basedir="."
default="main">

      …

      <target name="main"
depends="clean,run"/>

      <target name="clean-build”
              depends="clean,jar"/>



</project>
```

MAVEN

- Automatic dependency resolution
- Main actions (targets) standardized in a set of ready-to-use plugins
- Project information in a **pom**.xml (**P**roject **O**bject **M**odel)
- A project is identified by a triple:
  - groupId: identifies a group of projects (e.g., it.unipi.ing)
  - artifactId: identifies the project inside the group
  - version: the version of the project (e.g., 1.0)

Create a new project: mvn archetypes

- Archetypes: project templates
  - Define project structure and jar dependencies based on the project typology
  - Default archetype: : maven-archetype-quickstart
- New project: `mvn archetype:generate`
- Proprieta' del nuovo progetto:
  - groupId=it.unipi.ing.sia
  - artifactId=helloworld
  - version=1.0
  - package=it.unipi.ing.sia

The project structure

```
helloworld
|-- pom.xml
\-- src
    |-- main
    |   \-- java
    |       \-- it
    |           \-- corsows
    |               \-- App.java
    \-- test
        \-- java
            \-- it
                \-- corsows
                    \-- AppTest.java
```

The generated pom

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation=http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd>
<modelVersion>4.0.0</modelVersion>
  <groupId>it.unipi.ing.sia</groupId>
  <artifactId>helloworld</artifactId>
  <version>1.0</version>
  <packaging>jar</packaging>
  <name>helloworld</name>
  <url>http://maven.apache.org</url>
  <properties>
     <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>3.8.1</version>
        <scope>test</scope>
    </dependency></dependencies></project>
```

Packaging

- `mvn package`
  - Creates the directory (target) containing the compiled classes and the project jar
- `java  -cp target/helloworld-1.0.jar it.unipi.ing.sia.App`

Mvn goals

- `mvn clean`
- `mvn dependency:copy-dependencies`
- `mvn site` generates a web site for the project
- `mvn site:run` launches the site on port 8080
- `mvn javadoc:javadoc`
- `mvn eclipse:eclipse`
- More goals on: http://maven.apache.org/plugins/index.html