Primi passi operativi 24

Installazioni su PC

1. Installare *Java JDK 11* (non useremo versioni più recenti) *Windows->Preferences->Java->InstalledJRE*
2. Installare *GIT* PERSONALE PRIVATO e **dare accesso al docente**
3. Compilare il *template* inserendo **fotografia, email e numero di matricola**
4. Clonare *https://github.com/anatali/issLab24* in una directory vuota e.g. **C:/.../issLab24**
5. Installare *Gradle* (8.6 o 7.6) (per *System Properties Advanced* su Windows: **sysdm.cpl**)
6. Installare *Eclipse IDE*: *Eclipse IDE for Java and DSL Developers* (2023 12)
7. Installare *IntelliJ*
8. Installare *Docker*
9. Installare *Node.js*
10. Installare *Python (3.8.8)*

Creazione progetto con Gradle

La costruzione di un sistema software non può oggi prescindere dall'uso di un **IDE** e di strumenti di come *Gradle*, uno strumento open source per automatizzare la costruzione (build) del software.

Gradle è ormai uno standard di fatto in questo settore ed è interessante non solo in quanto strumento, ma anche perchè applica i principi del *Domain Driven Design* (**DDD**) per modellare il suo proprio domain-building software.

Ne consegue che non poco tempo deve essere dedicato allo studio di questi tools e alla preparazione del file **build.gradle** che governa lo sviluppo e il deployment del sistema software.

In questa prima fase il file **build.gradle** necessario viene fornito, lasciando a un momento successivo un approfondimento su come costruirlo.

Anticipiamo però anche che noi useremo un software factory custom che produrrà questo e altri numerosi file in modo automatico, a partire da un descrizione di alto livello del sistema, scritto nel linguaggio custom qak.

Eclipse e Gradle

Per usare Eclipse insieme a Gradle, potrebbe essere necessario eseguire i seguenti passi (Grazie a *Stefano Arasi*):

1. Help->Eclipse MarketPlace
2. Search “Gradle”
3. Go to “BuildShop Gradle Integration 3.0” (the elephant) which is indicated as installed and clic on the grayed installed button.
4. Either click on update or uninstall and reinstall it, this will restart eclipse.

Impostazione di un progetto

1. imposto una directory di lavoro per il corso (ad esempio **issLab24**) e mi posiziono in essa
2. Creo la directory **unibolibs**, che conterrà le librerie (file **.jar**) da noi sviluppate. Per il momento, inserisco le seguenti librerie:
   - **unibo.basicomm23-1.0.jar**
   - **uniboInterfaces.jar**
3. creo directory **project0** e mi posiziono in essa
4. eseguo **gradle init** e rispondo: 1 2 default default
5. importo in Eclipse il progetto gradle
6. aggiorno il file **build.gradle** con il contenuto riportato in *Un primo build.gradle*
7. click mouse destro su **project0** e seleziono **show in LocalTerminal->Terminal** (oppore uso un *CommandPrompt*)

1. eseguo gradlew eclipse (refresh F5 e vedo .classpath o *Project->Properties->Java Bsuild Path->Libraries*)
2. creo un nuovo sourcefolder **src**
3. in **src**, creo il package **main**
4. inserisco il file *HelloWorld.java* nel **package main**
5. eseguo **gradlew build** e/o **gradlew run**

Un primo build.gradle

```
plugins {
        id 'application'
        id 'java'
        id 'eclipse'
}
```

```
version '1.0'

java {
        toolchain.languageVersion.set(JavaLanguageVersion.of(11))
}

repositories {
        mavenCentral()
        flatDir {   dirs '../unibolibs'  }
}

dependencies {
        /* JSON ************************************************** */
        implementation 'com.googlecode.json-simple:json-simple:1.1.1'

        /* UNIBO **************************************************/
        implementation name: 'unibo.basicomm23-1.0'
}

sourceSets {
        main.java.srcDirs += 'src'
        main.java.srcDirs += 'src/main'
}

eclipse {
        classpath {
        sourceSets -= [ sourceSets.main ]
        }
}

application {
        mainClass = 'main.HelloWorld'
}

jar {
        println("building jar")
        from sourceSets.main.allSource
        manifest {
                attributes 'Main-Class': "$mainClassName"
        }
}

task dovesiamo {
        println("projectDir= $projectDir")
        println("buildDir  = $buildDir")
}
```

HelloWorld.java

```
package main;
import unibo.basicomm23.utils.CommUtils;

public class HelloWorld {
        public static void main( String[] args) {
                System.out.println("Hello world from project0");
                CommUtils.outblue("Hello world again from project0");
        }
}
```

Utility CommUtils

La classe **unibo.basicomm23.utils.CommUtils** è una utility che permette la visualizzazione di messaggi colorati (*black, blue, green, magenta, red, yellow*) sullo standard output.

Per abilitare i colori, installare dal *MarketPlace* il plugin **ansi-escape-console**.

Questa utility svolge un ruolo molto più importante coe supporto alle comunicazioni via rete. Si veda: *unibo.basicomm23*.

Uso di GIT

Una volta creato il progetto, è opportuno salvarlo su un nostro repository GIT.

Per un aiuto ad usare GIT può essere utile consultare *Basic Git commands* e/o guardare il video *Video on GIT* di cui riportiamo l'inizio di alcuni punti salienti:

```
0:00  - Introduction
1:31  - Distributed vs Central Version Control
3:17  - Installing Git
3:39  - First Time Setup
6:36  - Getting Started (Local repository)
10:41 - Git File Control
14:55 - Getting Started (Remote repository)
20:37 - Branching
20:50 - Common Workflow
23:03 - Push Branch on remote
27:38 - Faster Example
29:41 - Conclusion
```

Per quanto riguarda il nostro progetto:

1. Mi posiziono sulla directory **issLab24**.
2. Eseguo:

```
git init  //creates the directory .git
git status
```

3. Osservo il contenuto del file generato **.gitignore** :

```
# Ignore Gradle project-specific cache directory
.gradle
# Ignore Gradle build output directory
build

git status --ignored      //see ignored files
```

I files elencati non saranno salvati sul repository.

4. Eseguo i comandi .. code:

```
git add -A
git commit -m "Appl1"
git log   //q to exit
git status
```

Creazione di un repository remoto

1. Supponendo di avere accesso su *github* come user di nome **userxyz**, creiamo un repository personale di nome **issLab24**, selezionando il tipo **private**, con *README* file e **Add .gitignore** (*template Java*). Quindi aggiungiamo il nostro progetto al repository:

```
git remote add origin https://github.com/userxyz/issLab24
git remote -v   //osservo
```

2. Rendo visibile al docente del corso il progetto su *github*

3. Posizionato sulla directory **issLab24**, salvo il progetto corrente sul repository remoto.

```
git push origin master
```