

Java SE: Eclipse

- Sviluppo Java con Eclipse
 - Maven
 - Git
- Jenkins
 - Git – Maven
- Progetto di riferimento
 - <https://github.com/egalli64/mpjp>

Integrated Development Environment

- Semplifica lo sviluppo di applicazioni
 - IntelliJ IDEA
 - Eclipse IDE
 - Installer: <https://www.eclipse.org/downloads/>
 - vedi anche STS – Spring Tool Suite <https://spring.io/tools>
 - Apache NetBeans
 - ...
- Alternative più leggere:
 - Microsoft VS Code (“code editor”)
 - ...

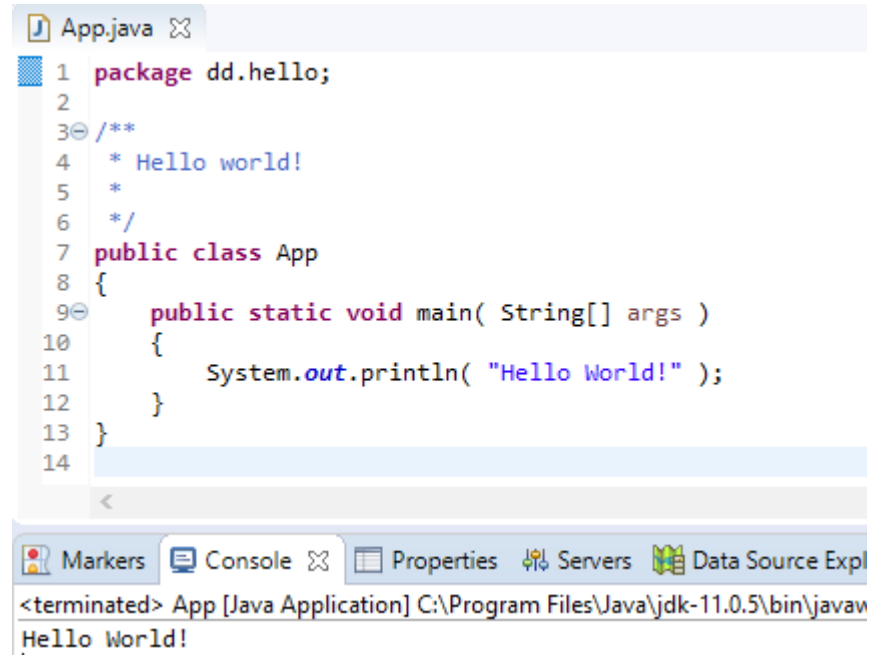
Nuovo progetto Maven in Eclipse

- Creare un progetto Maven
 - File, New, Maven Project
 - È necessario specificare solo `group id` e `artifact id`
- Nel POM specifichiamo le nostre `variazioni`
 - Properties
 - `project.build.sourceEncoding: UTF-8`
 - `maven.compiler.source` e `maven.compiler.target: 11`
 - Dependencies
 - ...
- A volte occorre forzare l'update del progetto dopo aver cambiato il POM
 - Alt-F5 (o right-click sul nome del progetto → Maven, Update project)

Eclipse - alcuni task comuni

- Formattazione del testo: Ctrl+Shift+F
- Salvataggio del file corrente su disco: Ctrl+S
- Salvataggio dei file aperti su disco: Ctrl+Shift+S
- Autosave: Window - Preferences - General - Editors - Auto-save
- Rinominare un elemento: Alt-Shift-R
- Generazione automatica di codice
 - Source - Generate toString()...
- Segnalazione di “lavori in corso” nel codice // TODO
 - Window - Show View - Tasks

Hello World!



The screenshot displays the Eclipse IDE interface. The top editor window, titled 'App.java', contains the following Java code:

```
1 package dd.hello;
2
3 /**
4  * Hello world!
5  *
6  */
7 public class App
8 {
9     public static void main( String[] args )
10    {
11        System.out.println( "Hello World!" );
12    }
13 }
14
```

Below the editor, the 'Console' view is active, showing the output of the program:

```
<terminated> App [Java Application] C:\Program Files\Java\jdk-11.0.5\bin\javaw
Hello World!
```

Import in Eclipse di un progetto Git

- File, Import ..., Git, Projects from Git (with smart import)
 - Clone URI
 - <https://github.com/egalli64/mpjp>
 - Eclipse dovrebbe riconoscerlo come progetto Maven
 - Altrimenti va importato come “General Project” e poi “mavenizzato”
 - Oppure, se il repository è già stato clonato
 - import del progetto come Existing local repository

Nuovo repository Git in Eclipse

- GitHub, creazione di un nuovo repository “xyz”
- Shell di Git, nella directory git locale:

```
git clone <url di xyz.git>
```

(oppure si può clonarlo dalla prospettiva Git di Eclipse)

- Eclipse: creazione di un nuovo progetto
 - Location: directory del repository appena clonato git/xyz
- Il nuovo progetto viene automaticamente collegato da Eclipse al repository Git presente nel folder

Team per Git in Eclipse

- Right click sul nome del progetto, Team
 - Pull (o Pull... per il branch corrente)
 - Commit rimanda alla view “Git staging”
 - Push to upstream (per il branch corrente)
 - Switch To, New branch...
 - Basta specificare il nome del nuovo branch
 - Switch To, per cambiare il branch corrente
 - Merge branch, per fondere due branch

.gitignore in Eclipse

- Per ignorare file o folder
 - Come già visto, file .gitignore
 - Oppure: right-click sulla risorsa, Team, Ignore
- Eclipse annota le icone di file e folder con simboli per mostrare come sono gestiti da Git
 - punto di domanda: risorsa sconosciuta
 - asterisco: risorsa staged per commit
 - più: risorsa aggiunta a Git ma non ancora tracked
 - assenza di annotazioni: risorsa ignorata

Jenkins Maven Git

- Progetto Java Maven su GitHub
- Jenkins con plugin
 - Git: <https://plugins.jenkins.io/git/> e Maven: <https://plugins.jenkins.io/maven-plugin/>
 - Jenkins deve sapere dove sono i tool sulla nostra macchina: `/configureTools/`
- New Jenkins Item “simple” come Maven project
- Configurazione `/job/simple/configure`
 - Source Code Management: <https://github.com/egalli64/simple.git>
 - Additional Behaviours: *Clean before checkout*, ...
 - Build:
 - Root POM: pom.xml
 - Goals and options: package [...]