



Maven2.apache.org:
**usare un linguaggio formale per
descrivere in modo standard tutte le fasi
del ciclo di vita del software**

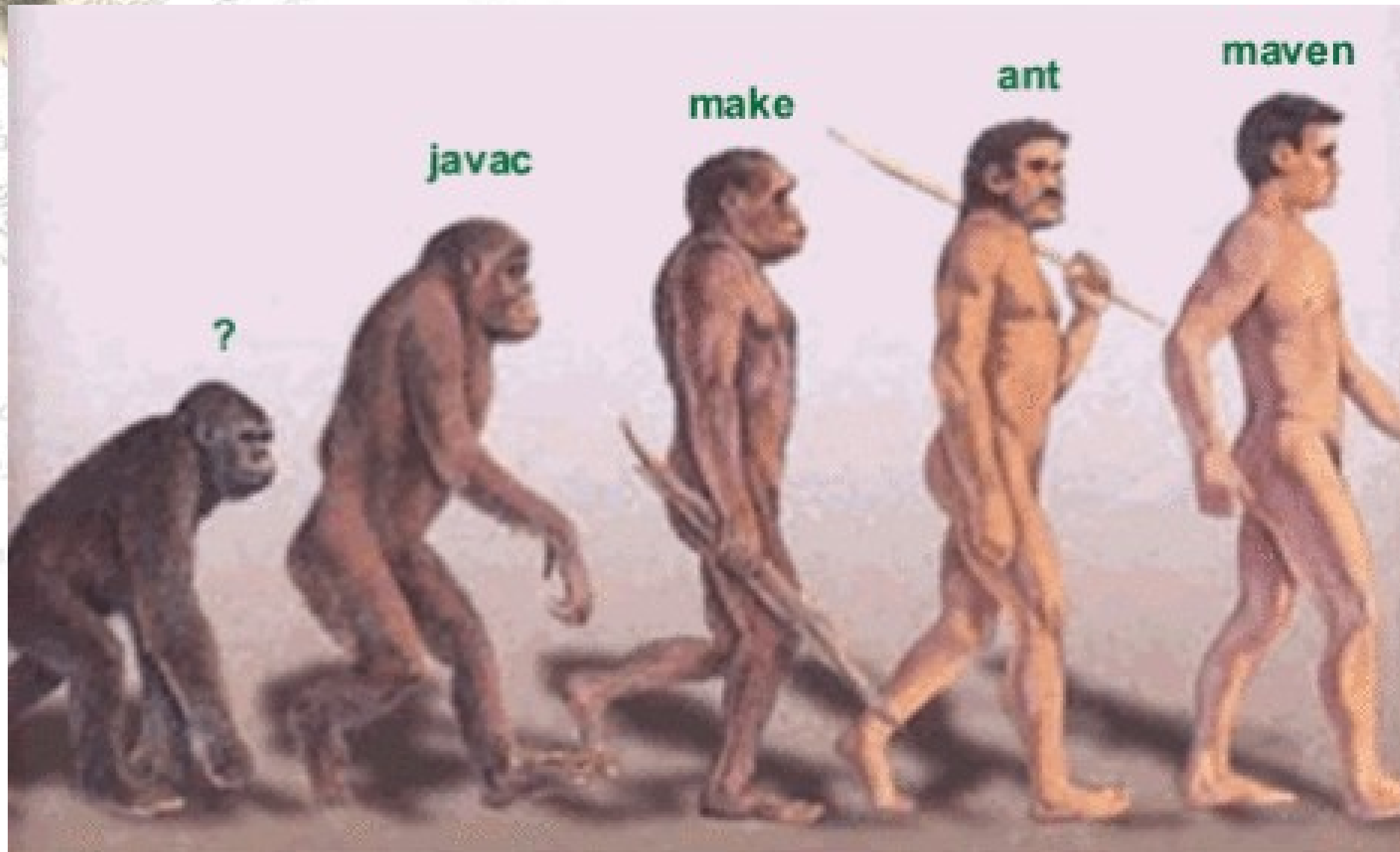
Simone Federici
s.federici@k-tech.it



KeyTECH

The K-Solution Factory

Evoluzione nella gestione dei progetti



Ogni riferimento a volti o persone conosciute è puramente casuale.

Cos'è Maven

- Un modo standard per descrivere progetto
- un build, test, package, deploy tool
- un gestore di dipendenze
- un generatore di report e documentazione
- e molto altro ancora...

Convention over Configuration

- Un default nella configurazione rende tutto più semplice.
- La descrizione del progetto con il suo ciclo di vita è sempre possibile fin nei minimi dettagli ma non è più obbligatoria

Un esempio... vale più di 1000 parole!

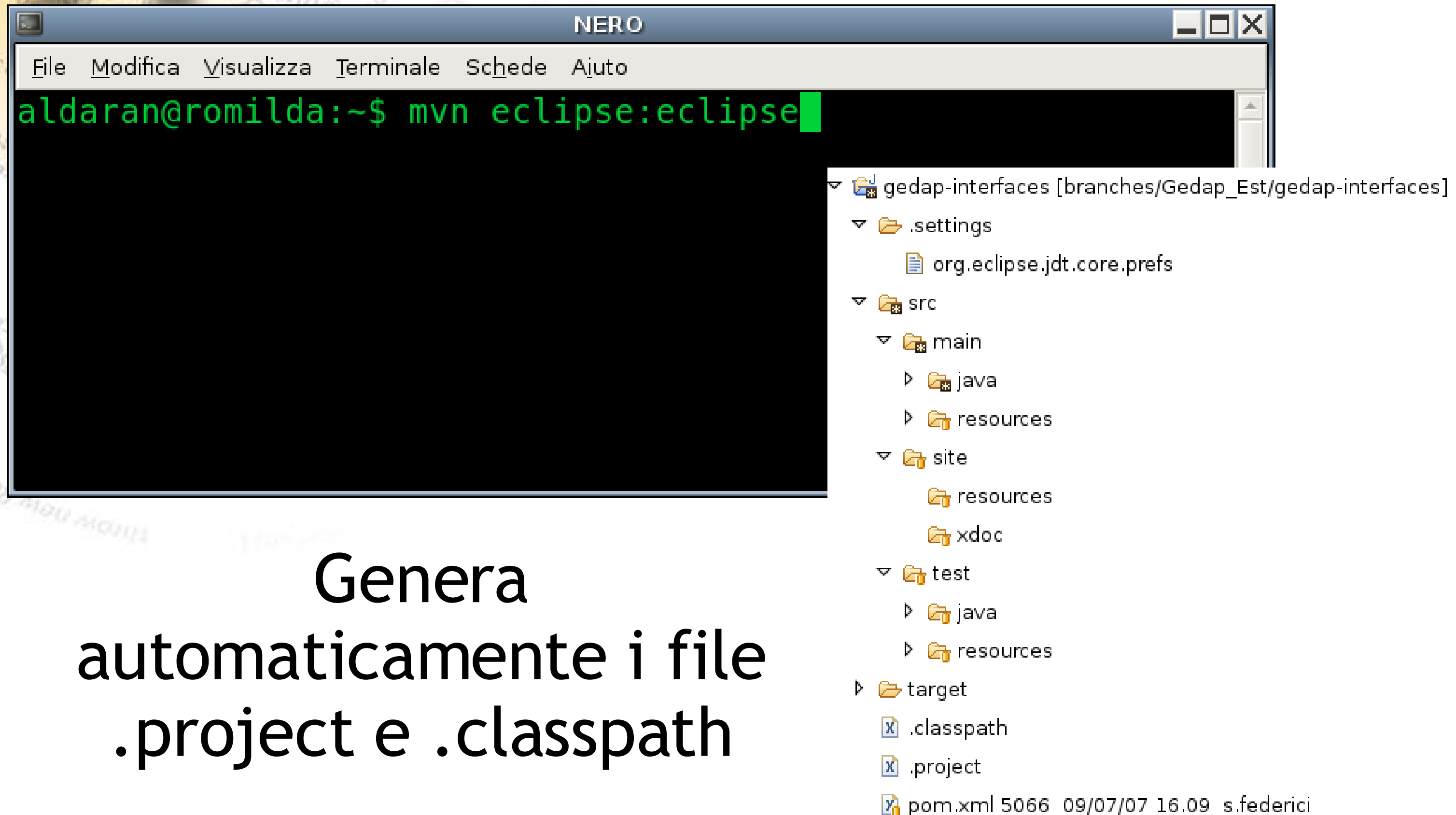
`mvn archetype:create`

`-DgroupId=it.jip.confsl`

`-DartifactId=esempio`



Eclipse- Plug-in



The image shows a screenshot of a terminal window and an Eclipse IDE project view. The terminal window, titled "NERO", shows the command `aldaran@romilda:~$ mvn eclipse:eclipse` being executed. The Eclipse IDE project view shows the following structure:

- gedap-interfaces [branches/Gedap_Est/gedap-interfaces]
 - .settings
 - org.eclipse.jdt.core.prefs
 - src
 - main
 - java
 - resources
 - site
 - resources
 - xdoc
 - test
 - java
 - resources
 - target
 - .classpath
 - .project
 - pom.xml 5066 09/07/07 16.09 s.federici

Genera
automaticamente i file
.project e .classpath

IL POM

Project Object Model

- Il progetto viene descritto interamente tramite un file xml (pom.xml)
- Uno standard che descrive il progetto, il ciclo di build e rilascio, il reporting e molto altro ancora

Semplice 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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0 </modelVersion>
  <groupId>it.jip.confsl</groupId>
  <artifactId>esempio </artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT </version>
  <name>esempio</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1 </version>
      <scope>test </scope>
    </dependency>
  </dependencies>
</project>
```

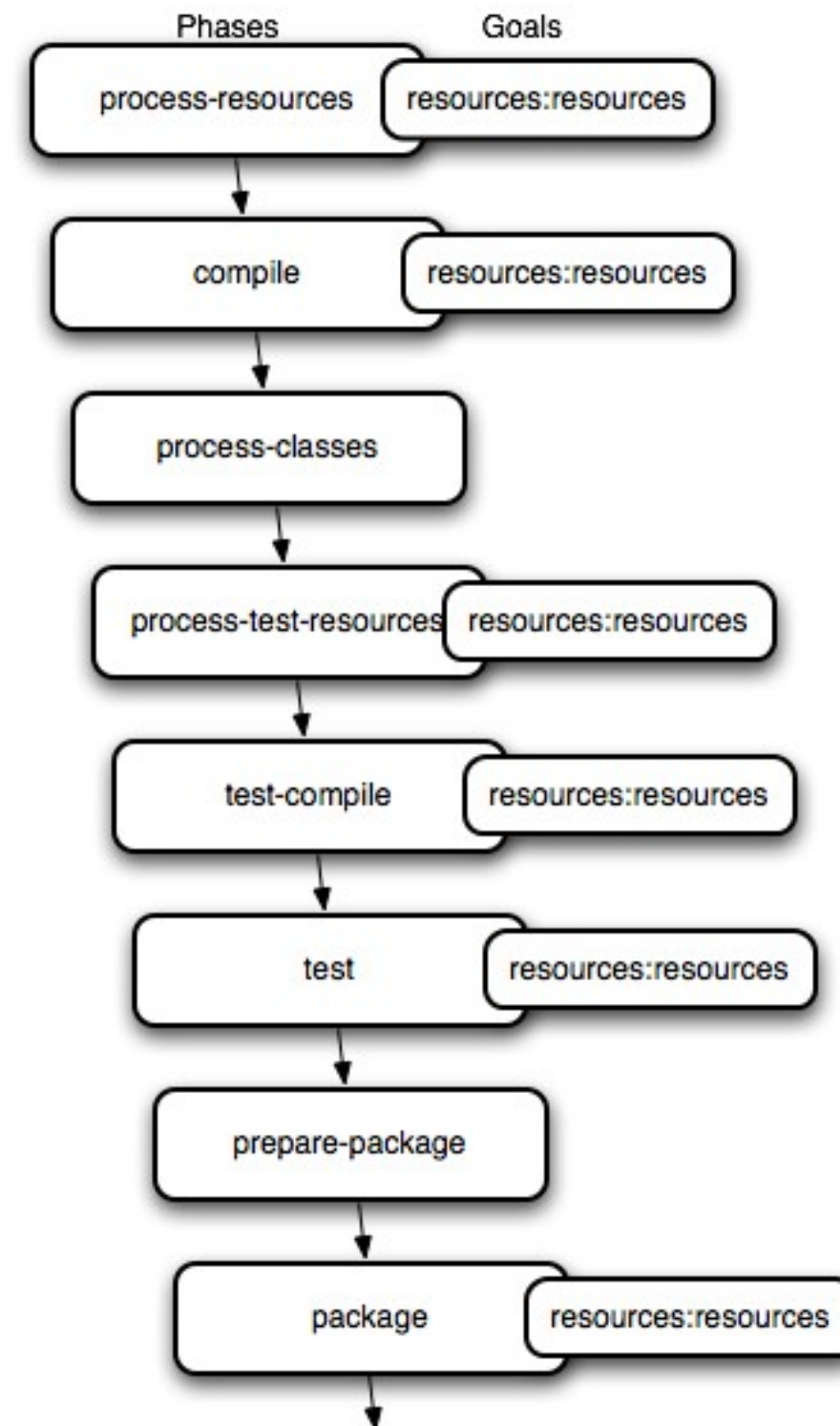
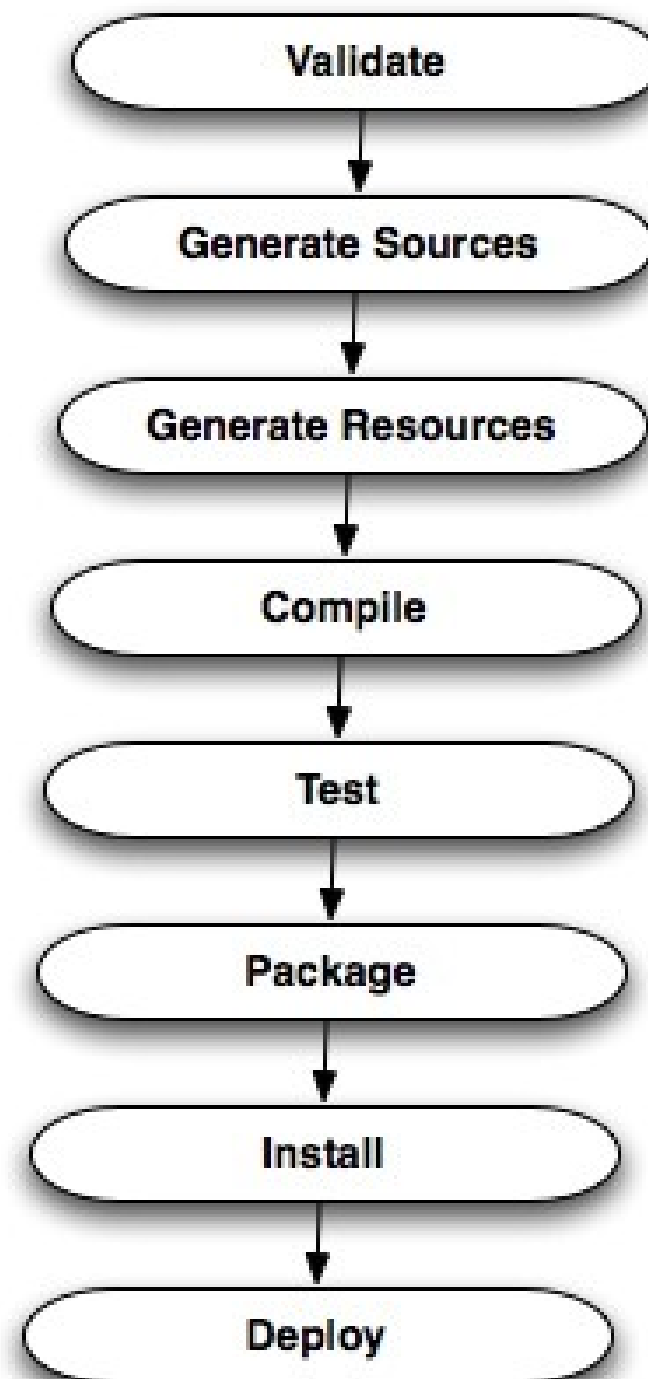
Artefact

- Identificazione precisa di un artefact è data dalla terna:
 - groupId + artifactId + version
- Packaging:
 - POM (Può contenere dei moduli)
 - JAR
 - WAR
 - EAR

Le Dipendenze

- una dipendenza è essa stessa un artefatto e quindi è identificata tramite:
 - groupId + artifactId + version
- Scope:
compile, test, runtime, provided, system
- Repository condiviso (locale e remoto)

I goals e le Phases



In pratica

- mvn compile
- mvn test -Ddbuser=simone -Ddbpas=****
- mvn package
- mvn install
- mvn deploy -Ddbuser=prod -Ddbpas=secret

i file all'interno delle resources saranno filtrati con le properties passate, prima di eseguire i test. `${dbuser}` e `${dbpas}`

Plugin: Sveliamo il trucco!

Un artifact speciale che definisce uno o più goal da “agganciare” alle phases o da invocare in modo indipendente.

Manca quello per il caffè...

In pratica con i profili

- `mvn compile -Pdevelop`
- `mvn test -Psystem`
- `mvn package -P produzione`
- `mvn install -P produzione,nolog`
- `mvn deploy -P remoto`

Plugin HELP

- mvn help:active-profiles
- mvn help:effective-pom

mvn site

Il plugin site, legge il pom e genera un sito documentale del progetto.

Tramite template velocity è possibile personalizzare il sito.

I numerosi plugin di reporting:
metrics, checkstyle, cobertura, findbugs,
pdm, svn-stat, ecc...

Profili

- Un progetto tanti contesti.
- Test unitari, Test funzionali, Collaudo, Produzione...

Plugin per Eclipse grafico!

- Esiste un plugin per eclipse grafico in grado di gestire le dipendenze automaticamente.
- Per esperienza vissuta, lo sconsiglio vivamente.
- La struttura di un progetto Eclipse è molto semplice, quella di maven no :-P

Tutto in una pagina

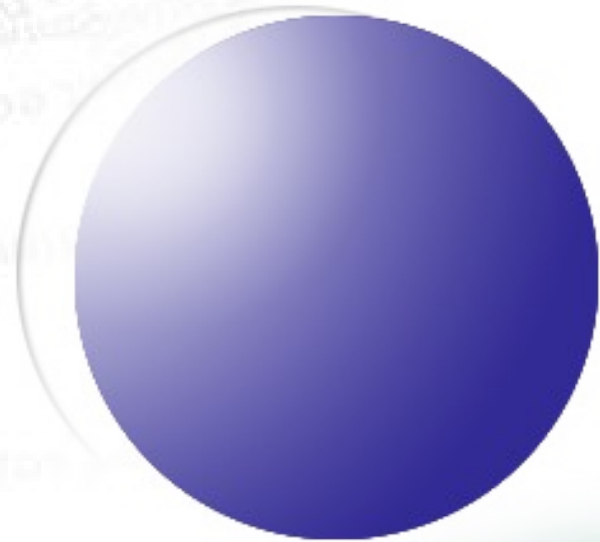
Centralizzazione, Convenzioni, Plugins,
Profili, Configurazioni, Sito Documentale,
Ambiente di sviluppo, Deploy remoto,
Deploy Locale, Continuous Integration,
Release e Snapshot, System Test, Test
Unitari, Test di integrazione, code style,
SVN/ CVS pubblico/ sviluppo, Packaging,
+ ANT, +script, e non entra tutto:-)

Riferimenti

- <http://maven.apache.org>
- <http://docs.codehaus.org/display/MAVENUSER>
- **Maven: The Definitive Guide (1.0 Alpha 1)**
- <http://www.sonatype.com/book/index.html>

Ringraziamenti:

**Giorgio Vinci
Mara Marzocchi**



eclipse
it.2007
4-5 ottobre
Napoli, Italy



**K-Tech
Eclipse-IT**