**Το Gradle** είναι ένα σύστημα [αυτοματισμού](https://en.wikipedia.org/wiki/Build_automation) ανοιχτού κώδικα που βασίζεται στις έννοιες [Apache Ant](https://en.wikipedia.org/wiki/Apache_Ant) και [Apache Maven](https://en.wikipedia.org/wiki/Apache_Maven) και εισάγει μια [γλώσσα που](https://en.wikipedia.org/wiki/Domain-specific_language) βασίζεται σε [Groovy](https://en.wikipedia.org/wiki/Groovy_(programming_language)) (DSL), αντί για τη μορφή [XML που](https://en.wikipedia.org/wiki/XML) χρησιμοποιεί το [Apache Maven](https://en.wikipedia.org/wiki/Apache_Maven) για να δηλώσει τη διαμόρφωση του έργου. [[2] Το](https://en.wikipedia.org/wiki/Gradle" \l "cite_note-2) Gradle χρησιμοποιεί ένα [κατευθυνόμενο ακυκλικό γράφημα](https://en.wikipedia.org/wiki/Directed_acyclic_graph) ("DAG") για να καθορίσει τη σειρά με την οποία μπορούν να εκτελεστούν οι εργασίες.

Το Gradle σχεδιάστηκε για κατασκευές πολλαπλών έργων, οι οποίες μπορούν να αυξηθούν αρκετά. Υποστηρίζει τις αυξομειώσεις δημιουργώντας με έξυπνο τρόπο ποια μέρη του δομημένου δένδρου είναι ενημερωμένα. κάθε εργασία που εξαρτάται μόνο από αυτά τα μέρη δεν χρειάζεται να εκτελείται εκ νέου.

Από το 2016 οι αρχικές plugins επικεντρώθηκαν σε [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) , [[3]](https://en.wikipedia.org/wiki/Gradle#cite_note-3) [Groovy](https://en.wikipedia.org/wiki/Groovy_(programming_language)) και [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)) ανάπτυξη και εγκατάσταση, αλλά περισσότερες γλώσσες και ροές εργασίας του έργου είναι σχετικά με τον χάρτη πορείας.

Το Gradle έχει μια πολύ σφιχτή ενσωμάτωση με το Ant, και μάλιστα αντιμετωπίζει τα Ant build αρχεία ως σενάρια που θα μπορούσαν να εισαχθούν απευθείας κατά την οικοδόμηση. Το παρακάτω παράδειγμα δείχνει έναν απλοποιημένο στόχο Ant που ενσωματώνεται ως έργο Gradle.

**build.xml**

**<project>**

**<target** name="ant.target"**>**

**<echo** message="Running ant.target!"**/>**

**</target>**

**</project>**

Βιβλιογραφία

*Berglund, Tim; McCullough, Matthew (Ιούλιος 2011).*[*Δομή και δοκιμή με Gradle*](http://oreilly.com/catalog/0636920019909/)*. Πρόλογος του Hans Dockter (πρώτη έκδοση). [O'Reilly Media](https://en.wikipedia.org/wiki/O%27Reilly_Media" \o "O'Reilly Media) . Π. 116.*[*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number)[*978-1-4493-0463-8*](https://en.wikipedia.org/wiki/Special:BookSources/978-1-4493-0463-8)*.*

* *Berglund, Tim (Αύγουστος 2013). [Gradle Beyond the Basics](http://shop.oreilly.com/product/0636920019923.do) (Πρώτη έκδοση).*[*O'Reilly Media*](https://en.wikipedia.org/wiki/O%27Reilly_Media)*. Π. 69.*[*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number)[*978-1-449-30467-6*](https://en.wikipedia.org/wiki/Special:BookSources/978-1-449-30467-6)*.*
* *Ikkink, Hubert (Νοέμβριος 2012).*[*Οδηγός εφαρμογής αποτελεσματικής εφαρμογής*](http://www.packtpub.com/gradle-effective-implementation-guide/book)*(First ed.). [Packt Publishing](https://en.wikipedia.org/wiki/Packt_Publishing" \o "Packt Publishing) . Π. 382.*[*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number)[*978-1849518109*](https://en.wikipedia.org/wiki/Special:BookSources/978-1849518109)*.*
* *Berglund, Tim; McCullough, Matthew (Μάιος 2013). [Gradle DSLs](http://oreilly.com/catalog/0636920019923/) (πρώτη έκδοση).*[*O'Reilly Media*](https://en.wikipedia.org/wiki/O%27Reilly_Media)*. σελ. 50 εκ.*[*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number)[*978-1-4493-0467-6*](https://en.wikipedia.org/wiki/Special:BookSources/978-1-4493-0467-6)*.*
* *Muschko, Benjamin (πτώση 2013). Gradle σε δράση (πρώτη έκδοση). [Εκπαιδευτικές εκδόσεις](https://en.wikipedia.org/wiki/Manning_Publications" \o "Εκπαιδευτικές εκδόσεις) . Π. 390.*[*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number)[*9781617291302*](https://en.wikipedia.org/wiki/Special:BookSources/9781617291302)*.*

**Το Maven** είναι ένα εργαλείο [αυτοματοποίησης κατασκευής](https://en.wikipedia.org/wiki/Build_automation) που χρησιμοποιείται κυρίως για έργα [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) .

Maven αντιμετωπίζει δύο πτυχές του λογισμικού κτιρίου: το πρώτο, περιγράφει πώς το λογισμικό είναι [χτισμένο](https://en.wikipedia.org/wiki/Software_build) , [*[έπρεπε να διευκρινιστεί](https://en.wikipedia.org/wiki/Wikipedia:Please_clarify" \o "Wikipedia: Παρακαλώ διευκρινίστε)*] και η δεύτερη, περιγράφει τις εξαρτήσεις του. Σε αντίθεση με προηγούμενα εργαλεία όπως το [Apache Ant](https://en.wikipedia.org/wiki/Apache_Ant) , χρησιμοποιεί συμβάσεις για τη διαδικασία δημιουργίας και μόνο οι εξαιρέσεις πρέπει να καταγραφούν. Ένα αρχείο [XML](https://en.wikipedia.org/wiki/XML) περιγράφει το έργο λογισμικού που κατασκευάζεται, τις εξαρτήσεις του σε άλλες εξωτερικές μονάδες και εξαρτήματα, την τάξη κατασκευής, τους καταλόγους και τις απαραίτητες [προσθήκες](https://en.wikipedia.org/wiki/Plug-in_(computing)) . Έρχεται με προκαθορισμένους στόχους για την εκτέλεση ορισμένων σαφώς καθορισμένων εργασιών όπως η σύνταξη κώδικα και η συσκευασία του.

Η Maven μεταφορτώνει δυναμικά τις βιβλιοθήκες [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) και τις προσθήκες Maven από έναν ή περισσότερους χώρους αποθήκευσης όπως το Maven 2 Central Repository και τις αποθηκεύει σε μια τοπική μνήμη cache. [[2]](https://en.wikipedia.org/wiki/Apache_Maven" \l "cite_note-maven2repo-2) Αυτή η τοπική μνήμη cache των ληφθέντων [αντικειμένων](https://en.wikipedia.org/wiki/Artifact_(software_development)) μπορεί επίσης να ενημερωθεί με αντικείμενα που δημιουργούνται από τοπικά έργα. Τα δημόσια αποθετήρια μπορούν επίσης να ενημερωθούν.

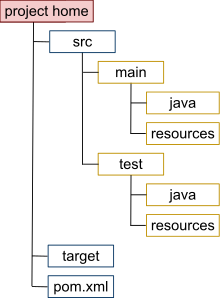
Το Maven μπορεί επίσης να χρησιμοποιηθεί για την κατασκευή και διαχείριση έργων γραμμένα σε [C #](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)) , [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)) , [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)) και σε άλλες γλώσσες. Το πρόγραμμα Maven φιλοξενείται από το [Apache Software Foundation](https://en.wikipedia.org/wiki/Apache_Software_Foundation) , στο οποίο ανήκε στο [έργο](https://en.wikipedia.org/wiki/Jakarta_Project) της [Τζακάρτα](https://en.wikipedia.org/wiki/Jakarta_Project) .

Το Maven είναι κατασκευασμένο χρησιμοποιώντας μια αρχιτεκτονική που βασίζεται σε προσθήκες, η οποία του επιτρέπει να κάνει χρήση οποιασδήποτε εφαρμογής που ελέγχεται μέσω πρότυπης εισόδου. Θεωρητικά, αυτό θα επέτρεπε σε οποιονδήποτε να γράψει plugins για να διασυνδέσει με εργαλεία δημιουργίας (μεταγλωττιστές, εργαλεία δοκιμής μονάδας κ.λπ.) για οποιαδήποτε άλλη γλώσσα. Στην πραγματικότητα, η υποστήριξη και η χρήση για άλλες γλώσσες εκτός από την Java ήταν ελάχιστες. Μια πρόσθετη εφαρμογή για το .NET Framework υπάρχει και διατηρείται [[3]](https://en.wikipedia.org/wiki/Apache_Maven#cite_note-3) και ένα εγγενές plugin [C](https://en.wikipedia.org/wiki/C_(programming_language)) / [C ++](https://en.wikipedia.org/wiki/C%2B%2B) διατηρείται για το Maven 2. [[4]](https://en.wikipedia.org/wiki/Apache_Maven" \l "cite_note-4)

Οι εναλλακτικές τεχνολογίες όπως το [Gradle](https://en.wikipedia.org/wiki/Gradle" \o "Gradle) και το [sbt](https://en.wikipedia.org/wiki/Sbt" \o "Sbt) ως εργαλεία δημιουργίας δεν βασίζονται σε [XML](https://en.wikipedia.org/wiki/XML) , αλλά διατηρούν τις βασικές έννοιες που εισήγαγε η Maven. Με τον [Apache Ivy](https://en.wikipedia.org/wiki/Apache_Ivy) , αναπτύχθηκε επίσης ένας εξειδικευμένος διαχειριστής εξάρτησης, ο οποίος υποστηρίζει επίσης αποθετήρια Maven. [[5]](https://en.wikipedia.org/wiki/Apache_Maven" \l "cite_note-5)

το POM ορίζει μόνο ένα μοναδικό αναγνωριστικό για το έργο ( *συντεταγμένες* ) και την εξάρτησή του από το πλαίσιο [JUnit](https://en.wikipedia.org/wiki/JUnit" \o "JUnit) . Ωστόσο, αυτό είναι ήδη αρκετό για την κατασκευή του έργου και τη διεξαγωγή των [δοκιμών μονάδας που](https://en.wikipedia.org/wiki/Unit_testing) σχετίζονται με το έργο. Η Maven το επιτυγχάνει ακολουθώντας την ιδέα της [Συνέλευσης μέσω Διαμόρφωσης](https://en.wikipedia.org/wiki/Convention_over_Configuration) , δηλαδή, το Maven παρέχει προεπιλεγμένες τιμές για τη διαμόρφωση του έργου.

Η δομή καταλόγου ενός κανονικού [ιδιωματικού](https://en.wikipedia.org/wiki/Programming_idiom) έργου Maven έχει τις ακόλουθες καταχωρήσεις καταλόγου:

[](https://en.wikipedia.org/wiki/File:Maven_CoC.svg)

Το εργαλείο λογισμικού Maven δημιούργησε αυτόματα αυτή τη δομή καταλόγου για ένα έργο Java.

|  |  |
| --- | --- |
| **Όνομα καταλόγου** | **Σκοπός** |
| σπίτι έργου | Περιέχει το αρχείο pom.xml και όλους τους υποκαταλόγους. |
| src / main / java | Περιέχει τον παραδοτέο πηγαίο κώδικα Java για το έργο. |
| src / main / resources | Περιέχει τους παραδοτέους πόρους για το έργο, όπως τα αρχεία ιδιοκτησίας. |
| src / test / java | Περιέχει το πηγαίο κώδικα δοκιμής Java (παραδείγματα δοκιμών JUnit ή TestNG, για παράδειγμα) για το έργο. |
| src / test / πόροι | Περιέχει τους πόρους που απαιτούνται για τη δοκιμή. |

Στη συνέχεια, η εντολή

πακέτο mvn

θα συντάξει όλα τα αρχεία Java, θα εκτελέσει τυχόν δοκιμές και θα πακετάρει τον παραδοτέο κώδικα και τους πόρους target/my-app-1.0.jar(υποθέτοντας ότι το artifactId είναι η εφαρμογή μου και η έκδοση είναι 1.0).

Χρησιμοποιώντας το Maven, ο χρήστης παρέχει μόνο διαμόρφωση για το έργο, ενώ οι διαμορφώσιμες προσθήκες πραγματοποιούν την πραγματική εργασία της σύνταξης του έργου, τον καθαρισμό των καταλόγων στόχων, τις δοκιμές μονάδων λειτουργίας, τη δημιουργία τεκμηρίωσης API κ.ο.κ. Σε γενικές γραμμές, οι χρήστες δεν θα πρέπει να γράψουν τα ίδια τα plugins. Αντιστοιχίστε αυτό με τον [Ant](https://en.wikipedia.org/wiki/Apache_Ant) και [κάνετε](https://en.wikipedia.org/wiki/Make_(software)) , στον οποίο γράφετε επιτακτικές διαδικασίες για την εκτέλεση των προαναφερθέντων εργασιών.

Ένα μοντέλο αντικειμένου έργου (POM) παρέχει όλες τις ρυθμίσεις για ένα μόνο έργο. Η γενική διαμόρφωση καλύπτει το όνομα του έργου, τον ιδιοκτήτη του και τις εξαρτήσεις του σε άλλα έργα. Μπορούμε επίσης να ρυθμίσουμε μεμονωμένες φάσεις της διαδικασίας δημιουργίας, οι οποίες υλοποιούνται ως [plugins](https://en.wikipedia.org/wiki/Plug-in_(computing)) . Για παράδειγμα, μπορείτε να διαμορφώσετε το μεταγλωττιστή-plugin για να χρησιμοποιήσετε την έκδοση 1.5 της Java για τη σύνταξη ή να καθορίσετε τη συσκευασία του έργου ακόμα και αν αποτύχουν ορισμένες δοκιμές μονάδας.

Τα μεγαλύτερα έργα θα πρέπει να χωριστούν σε διάφορες ενότητες ή υπο-έργα, το καθένα με το δικό του POM. Κάποιος μπορεί στη συνέχεια να γράψει ένα root POM μέσω του οποίου μπορεί κανείς να μεταγλωττίσει όλες τις μονάδες με μία μόνο εντολή. Τα POM μπορούν επίσης να κληρονομήσουν διαμόρφωση από άλλα POM. Όλα τα POM κληρονομούνται από το Super POM [[8]](https://en.wikipedia.org/wiki/Apache_Maven#cite_note-8) από προεπιλογή. Το Super POM παρέχει προεπιλεγμένες ρυθμίσεις, όπως προεπιλεγμένους καταλόγους πηγών, προεπιλεγμένες προσθήκες κ.λπ.

**Plugins**

Οι περισσότερες από τις λειτουργίες του Maven είναι σε [plugins](https://en.wikipedia.org/wiki/Plug-in_(computing)) . Ένα plugin παρέχει ένα σύνολο στόχων που μπορούν να εκτελεστούν χρησιμοποιώντας την ακόλουθη σύνταξη:

mvn [plugin-name]: [όνομα-στόχου]

Παράδειγμα

Τα έργα Maven έχουν ρυθμιστεί χρησιμοποιώντας ένα [μοντέλο αντικειμένου έργου](https://en.wikipedia.org/wiki/Project_Object_Model) , το οποίο αποθηκεύεται σε ένα pom.xmlαρχείο. Ακολουθεί ένα ελάχιστο παράδειγμα:

**<project>**

*<!-- model version is always 4.0.0 for Maven 2.x POMs -->*

**<modelVersion>**4.0.0**</modelVersion>**

*<!-- project coordinates, i.e. a group of values which*

*uniquely identify this project -->*

**<groupId>**com.mycompany.app**</groupId>**

**<artifactId>**my-app**</artifactId>**

**<version>**1.0**</version>**

*<!-- library dependencies -->*

**<dependencies>**

**<dependency>**

*<!-- coordinates of the required library -->*

**<groupId>**junit**</groupId>**

**<artifactId>**junit**</artifactId>**

**<version>**3.8.1**</version>**

*<!-- this dependency is only used for running and compiling tests -->*

**<scope>**test**</scope>**

**</dependency>**

**</dependencies>**

**</project>**