Mobile Programming

Pertemuan 11

Alauddin Maulana Hirzan, S.Kom., M.Kom.
NIDN. 0607069401

Fakultas Teknologi Informasi dan Komunikasi, Universitas Semarang

Konsep Kompilasi

Definisi Kompilasi

Definisi:

Kompilasi adalah proses penerjemahan kode sumber yang dapat dibaca manusia yang ditulis dalam bahasa pemrograman ke dalam instruksi yang dapat dibaca mesin yang dapat dieksekusi oleh komputer.

Proses ini melibatkan transformasi kode yang ditulis oleh programmer ke dalam format yang dapat dimengerti dan dieksekusi oleh prosesor komputer.

Konsep Kompilasi

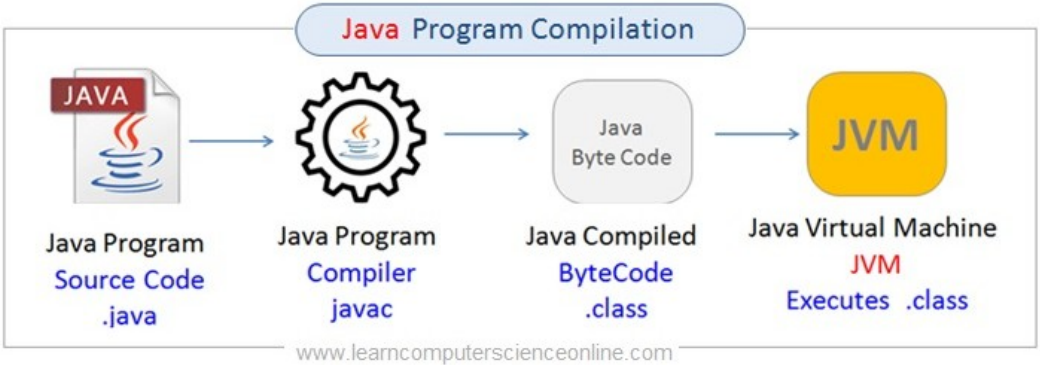
Definisi Kompilasi

```
#include<stdio.h>
main()
{
printf("Hello javaTpoint");
return 0;
}
```

```
01000000000000
01111111111111
01010101101010
00000011111111
00000111111111
00000010101011
```

Konsep Kompilasi

Definisi Kompilasi



Konsep Kompilasi

Alat Kompilasi

Agar kode bisa dikompilasi menjadi program yang bisa dijalankan, maka memerlukan alat-alat yang mengolah semuanya itu:

- **Kompiler**: Alat utama yang bertanggung jawab untuk menerjemahkan kode sumber ke dalam kode mesin.
- **Perakit (Assembler)**: Mengonversi kode bahasa rakitan menjadi kode mesin.
- **Penghubung (Linker)**: Menyelesaikan referensi antara berbagai modul program dan menghasilkan file yang dapat dieksekusi.
- **Debugger**: Membantu pengembang mengidentifikasi dan memperbaiki kesalahan dalam kode

Konsep Kompilasi

Permasalahan Kompilasi

Kompilasi menjadi merepotkan ketika:

1. **Banyak File dan Sumber Daya**
2. **Waktu dan Sumber Daya**
3. **Manajemen Ketergantungan**

Konsep Kompilasi

Permasalahan Kompilasi

1. Banyak File dan Sumber Daya

Ketika mengerjakan proyek besar, programmer sering kali memiliki banyak file kode dan berbagai sumber daya seperti gambar, suara, atau file konfigurasi yang diperlukan agar program dapat bekerja dengan baik.

File-file ini seperti blok bangunan yang disatukan untuk membuat perangkat lunak akhir.

Konsep Kompilasi

Permasalahan Kompilasi

2. Waktu dan Sumber Daya

Satu tantangan besar adalah waktu yang diperlukan untuk mengkompilasi semua file ini. Semakin banyak file yang dimiliki, semakin lama waktu yang dibutuhkan komputer untuk memproses semuanya secara manual.

Programmer mungkin harus menunggu lama sebelum mereka dapat melihat apakah perubahan yang mereka buat berfungsi dengan baik.

Konsep Kompilasi

Permasalahan Kompilasi

3. Manajemen Ketergantungan

Selain itu, ketika programmer memiliki banyak file yang bergantung satu sama lain (artinya satu file bergantung pada file lain untuk bekerja dengan benar), mengelola ketergantungan ini menjadi lebih kompleks.

Jika ada kesalahan atau perubahan pada satu file, hal ini dapat mempengaruhi banyak file lain, yang menyebabkan kesalahan pada program akhir.

Otomatisasi Build

Definisi Otomatisasi Build

Definisi:

Proses pembuatan otomatis dalam pengembangan perangkat lunak, mengacu pada eksekusi otomatis tugas-tugas yang terlibat dalam kompilasi, pengujian, dan pengemasan kode perangkat lunak.

Proses ini menyederhanakan tugas berulang yang diperlukan untuk mengubah kode sumber yang dapat dibaca manusia menjadi program atau pustaka yang dapat dieksekusi.

Otomatisasi Build

Manfaat Otomatisasi Build

Otomatisasi Build ini memiliki manfaat berupa:

1. **Konsistensi**: Proses pembuatan otomatis memastikan bahwa setiap pembuatan dilakukan dengan cara yang sama
2. **Efisiensi**: Dengan mengotomatiskan tugas-tugas seperti kompilasi, pengujian, dan pengemasan, pengembang dapat menghemat waktu dan tenaga
3. **Keandalan**: Pembuatan otomatis tidak terlalu rentan terhadap kesalahan dibandingkan dengan proses manual
4. **Skalabilitas**: Seiring bertambahnya kompleksitas proyek, proses pembuatan manual menjadi semakin rumit dan rentan terhadap kesalahan.

Otomatisasi Build

Tugas Otomatisasi Build

Otomatisasi Build memiliki tugas untuk:

1. **Kompilasi**: Alat bantu otomasi membangun kompilasi kode sumber menjadi binari atau pustaka yang dapat dieksekusi.
2. **Pengujian**: Kerangka kerja pengujian otomatis yang diintegrasikan ke dalam alat bantu build melakukan pengujian unit, pengujian integrasi, dan pengujian regresi.
3. **Pengemasan**: Alat bantu otomatisasi build mengemas kode yang telah dikompilasi beserta sumber daya yang diperlukan
4. **Penyebaran**: Beberapa alat bantu build memfasilitasi penyebaran perangkat lunak ke berbagai lingkungan

Otomatisasi Build

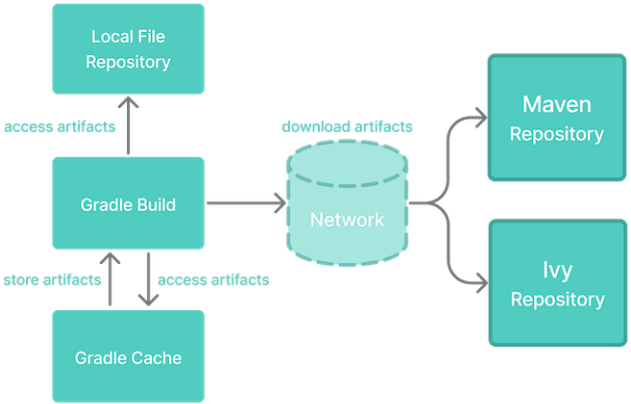
Manajemen Ketergantungan Proses Kompilasi

Alat bantu build otomatis, seperti Maven, Gradle, atau npm, mengotomatiskan proses penyelesaian dan pengelolaan dependensi.

Alat-alat ini secara otomatis mengunduh, mengelola, dan menyertakan dependensi berdasarkan file konfigurasi, seperti pom.xml atau package.json, sehingga mengurangi upaya manual dan memastikan konsistensi.

Otomatisasi Build

Manajemen Ketergantungan Proses Kompilasi



Alat Otomatisasi Build

Maven

Maven adalah alat bantu yang ampuh yang digunakan dalam pengembangan perangkat lunak untuk mengelola ketergantungan proyek, membangun proses, dan struktur proyek

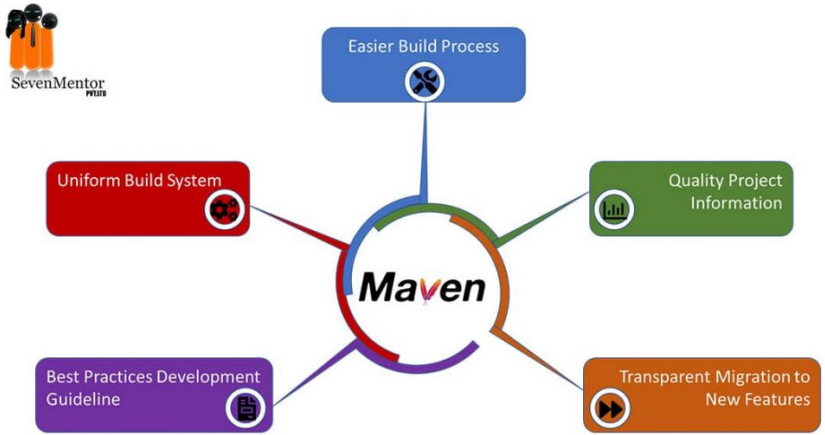
Alat Otomatisasi Build

Maven

Maven adalah alat bantu yang ampuh yang digunakan dalam pengembangan perangkat lunak untuk mengelola ketergantungan proyek, membangun proses, dan struktur proyek

Alat Otomatisasi Build

Maven



Alat Otomatisasi Build

Relasi Maven dengan Projek Android

Dalam konteks pengembangan Android, Maven adalah salah satu alat otomatisasi build paling awal yang digunakan untuk mengelola dependensi dan membangun proyek Android. Namun, dengan diperkenalkannya Android Studio dan Gradle sebagai sistem build resmi untuk Android, penggunaan Maven dalam pengembangan Android telah menurun.

Meskipun Maven masih dapat digunakan untuk proyek Android, Gradle telah menjadi standar de facto karena integrasinya yang erat dengan Android Studio dan dukungan yang lebih baik untuk tugas-tugas khusus Android, seperti manajemen sumber daya dan pembuatan multi-modul.

Alat Otomatisasi Build

Siklus Build Maven

Untuk bisa membangun aplikasi, Maven menggunakan beberapa tahapan berikut:

1. **Clean Lifecycle**: Fase ini bertanggung jawab untuk membersihkan proyek dan menghapus artefak yang dibuat oleh build sebelumnya.
2. **Default Lifecycle**: Fase ini menangani penyebaran proyek dan membangun artefak proyek.
3. **Site Lifecycle**: Fase ini menghasilkan dokumentasi dan laporan proyek.

Alat Otomatisasi Build

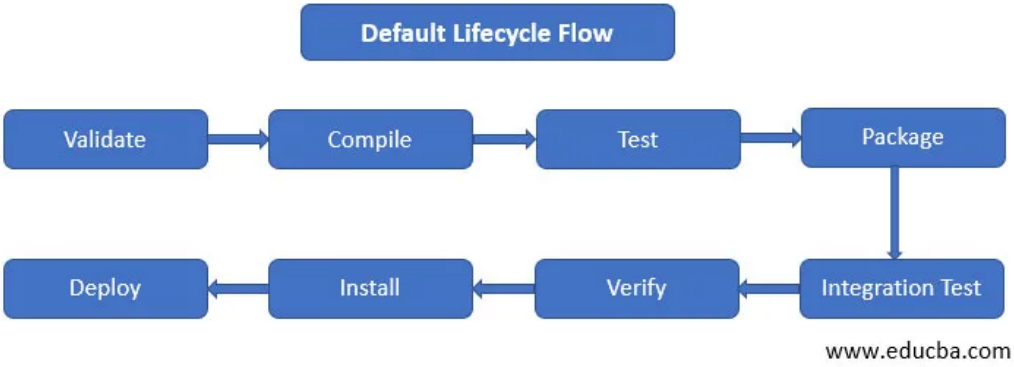
Siklus Build Maven - Default Lifecycle

Default Lifecycle merupakan siklus utama untuk membangun aplikasi yang terdiri dari:

- **validate**: Memvalidasi konfigurasi proyek.
- **compile**: Mengkompilasi kode sumber.
- **test**: Menjalankan pengujian terhadap kode sumber yang telah dikompilasi.
- **package**: Memaketkan kode yang telah dikompilasi
- **install**: Menginstal paket ke dalam repositori Maven lokal.
- **deploy**: Menyalin paket akhir ke repositori jarak jauh

Alat Otomatisasi Build

Siklus Build Maven - Default Lifecycle



Alat Otomatisasi Build

Gradle

Gradle adalah alat otomatisasi pembangunan yang kuat yang digunakan terutama untuk proyek-proyek Java, tetapi juga dapat diterapkan pada bahasa lain seperti C++, Python, dan banyak lagi. Alat ini adalah alat sumber terbuka yang mengotomatiskan proses pembangunan, pengujian, dan penyebaran proyek perangkat lunak.

Gradle menggunakan Domain Specific Language (DSL) berbasis Groovy atau Kotlin untuk skrip, memberikan fleksibilitas dan kemudahan penggunaan.

Alat Otomatisasi Build

Gradle

Alat Otomatisasi Build

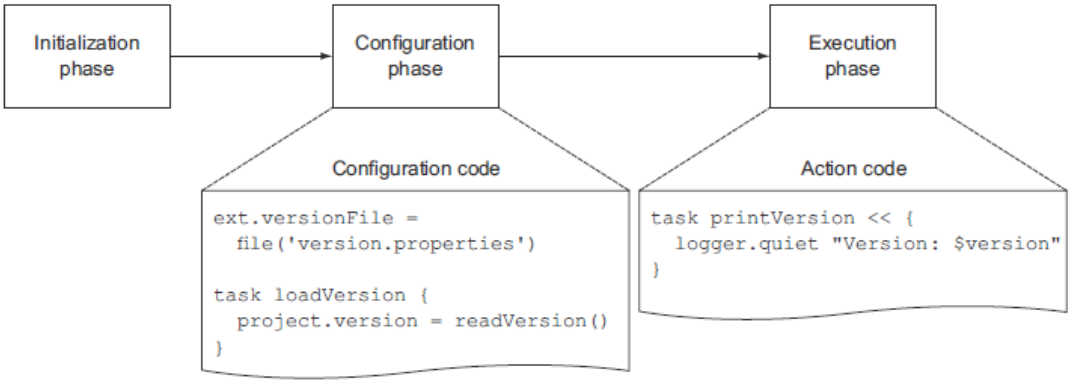
Siklus Hidup Gradle

Sama seperti Maven, Gradle juga memiliki siklus Build seperti berikut:

1. **Fase Inisialisasi**: Selama inisialisasi, Gradle menyiapkan lingkungan proyek, termasuk mengidentifikasi struktur proyek, menentukan pengaturan, dan mempersiapkan diri untuk mengeksekusi build.
2. **Fase Konfigurasi**: Fase ini mengevaluasi konfigurasi proyek dan tugas, menyelesaikan ketergantungan, dan menyiapkan tugas untuk dieksekusi.
3. **Fase Eksekusi**: Fase eksekusi adalah fase di mana tugas-tugas aktual yang didefinisikan dalam fase konfigurasi dieksekusi.

Alat Otomatisasi Build

Siklus Hidup Gradle



Alat Otomatisasi Build

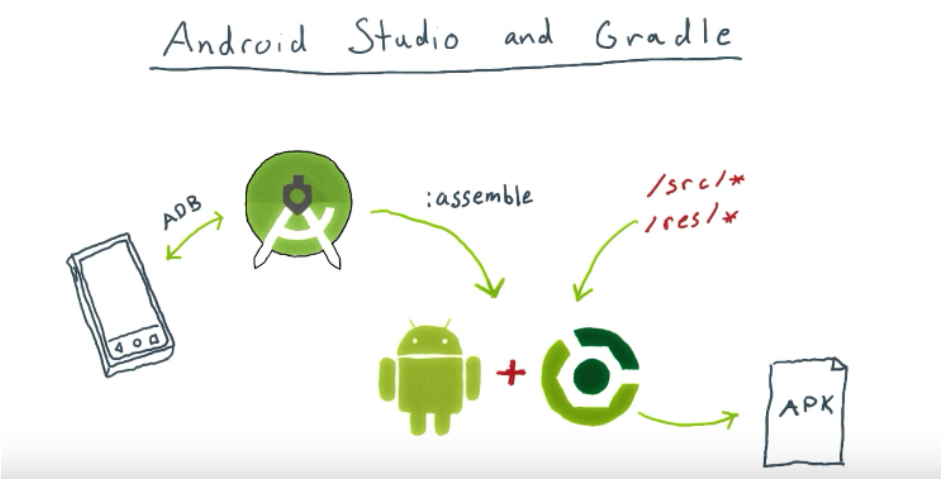
Integrasi Gradle ke Android Studio

Gradle membantu pembuatan aplikasi Android dengan cara:

1. **Menyiapkan Build**: Ketika pengguna memulai proyek baru di Android Studio, Gradle akan membuat file bernama build.gradle.
2. **Ketergantungan**: Di dalam file build.gradle, pengguna dapat menambahkan pustaka dan dependensi lain yang dibutuhkan aplikasi
3. **Tugas**: Gradle memecah proses pembuatan menjadi tugas-tugas yang lebih kecil, seperti mengkompilasi kode, mengemas sumber daya, dan menandatangani aplikasi.
4. **Membangun Aplikasi**: Ketika Pengguna mengklik tombol ”Jalankan” di Android Studio, Gradle mulai membangun aplikasi.

Alat Otomatisasi Build

Integrasi Gradle ke Android Studio



Terima Kasih