

Extreme Programming dengan Ant

Bagi Anda yang menerapkan *extreme programming*, berbagai *tools* siap pakai sudah dapat Anda gunakan. Salah satunya adalah dengan Ant. Dengan extreme programming dan Ant, pembuatan aplikasi besar menjadi lebih menyenangkan.

Semenjak Kent Beck mengarang buku *Extreme Programming Explained* yang menerangkan tentang bagaimana perasaan berinteraksi saat membuat program. Extreme Programming (XP) menjadikan pembuatan aplikasi lebih berstruktur, dengan pendekatan penyelesaian berdasarkan dari masalah yang datang. Saat ini situs mengenai XP dapat dilihat melalui <http://www.xprogramming.com> menerangkan banyak informasi mengenai XP.

Kent Beck mendefinisikan, bahwa XP memiliki 4 acuan dalam pengembangan, yaitu:

- Communication
- Simplicity
- Feedback
- Courage

Berdasarkan empat acuan di atas, Kent Beck membuat 5 prinsip lagi, yaitu:

- Rapid Feedback
- Assume Simplicity
- Incremental Changes
- Embrace Change
- Quality Work

Kent Beck juga menerangkan mengenai 4 dasar cara praktis pengembangan, yaitu:

- Coding
- Testing
- Listening
- Designing

Di mana semua cara praktis tersebut mengekspresikan 12 area, yaitu:

- Planning Game
- Small Releases
- Simple Design
- Testing
- Continuous Integration

- Refactoring
- Pair Programming
- Collective Ownership
- 40-hour week
- On-site customer
- Metaphor
- Coding Standard

Dari semua teori XP yang ada, yang menarik dari semua ini adalah adanya kedekatan antara open source dengan XP. Keduanya merupakan ideologi yang kuat sekali yang memungkinkan terjadinya kolaborasi berdasarkan sesuatu yang sering terjadi yang disebabkan oleh kelemahan manusia, meliputi ketidakmampuan untuk mengembangkan sebuah kode yang sempurna, asistensi antar pengembang untuk mencari dan menyelesaikan masalah dalam *coding*.

Dari semua area XP, penulis akan membahas dan menerangkan 2 area saja yaitu *Automated Testing* dan *Continuous Integration* untuk artikel ini, dan membahas 1 Java Open Source project yang terkenal yang dapat digunakan untuk implementasi XP, yaitu Ant. Ant membuat Java menjadi sangat bernilai. Ingin tahu nilainya, cobalah gunakan Ant untuk pekerjaan sehari-hari.

Automated Testing

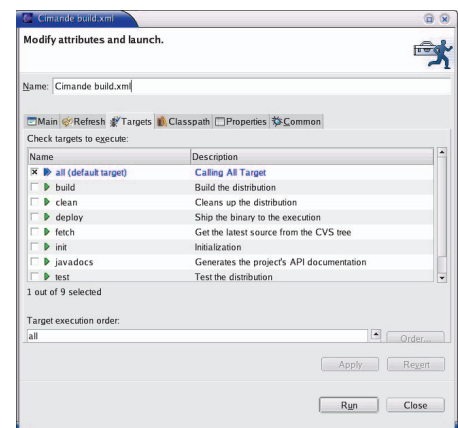
Malah dengan menggunakan *automated testing* memungkinkan untuk melakukan testing code, *step* ini dirasakan perlu terutama saat hendak melakukan integrasi. Automated Testing ini akan secara tidak langsung melakukan validasi terhadap code-code yang dikembangkan. Tanpa testing, biasanya tim akan mengira-ngira code yang dikerjakan adalah sesuai. Ada yang menarik dari XP ini, yaitu bila kita



memiliki 2 buah desain dan setelah dilakukan testing, kita hendak memin-dahkan salah satu desain ke desain yang lain, proses ini disebut dengan *refactoring*. Umumnya bila automated testing telah berjalan dengan baik, code dari desain yang satu ke desain yang lain akan dapat dipindahkan dengan mudah.

Untuk melakukan automated testing ini, banyak cara yang dapat dilakukan, penulis umumnya menggunakan 2 metode pendekatan yaitu *unit testing* dan *unit acceptance testing*. Dua metode ini dilihat dari sisi *management level*, padahal sesungguhnya yang harus dilakukan dalam testing itu adalah *integration testing*, *functional testing*, dan *auxiliary testing* (*performance testing*, *regression testing*), di mana semua testing ini akan meyakinkan bahwa code bekerja dengan baik.

Saat ini ada beberapa tools pendukung yang membantu automated testing ini, yaitu Junit (<http://www.junit.org>) untuk Unit Testing, Cactus (<http://jakarta.apache.org/cactus>) untuk Integration Testing, HttpUnit (<http://www.httpunit.org>) untuk Acceptance Testing, JUnitPerf



▲ Eclipse dan Ant.

dan JMeter (<http://jakarta.apache.org/jmeter>) untuk Performance Testing.

Continuous Integration

Continuous Integration memungkinkan dilakukan pengulangan testing secara berkesinambungan, sehingga tim dapat melihat *progress* dari sebuah pengembangan sistem, dan semua tim yakin bahwa hasil kerjanya adalah sesuai. Continuous Integration ini memungkinkan terjadinya penekanan error karena dilakukannya testing secara rutin.

Continuous Integration ini akan terasa sekali bila dilakukan pengembangan yang menggunakan aplikasi yang saling tergantung. Apalagi para programmer Java Open Source yang mana setiap komponen yang digunakan cenderung terus berubah dan bertambah, Continuous Integration ini sangat diperlukan.

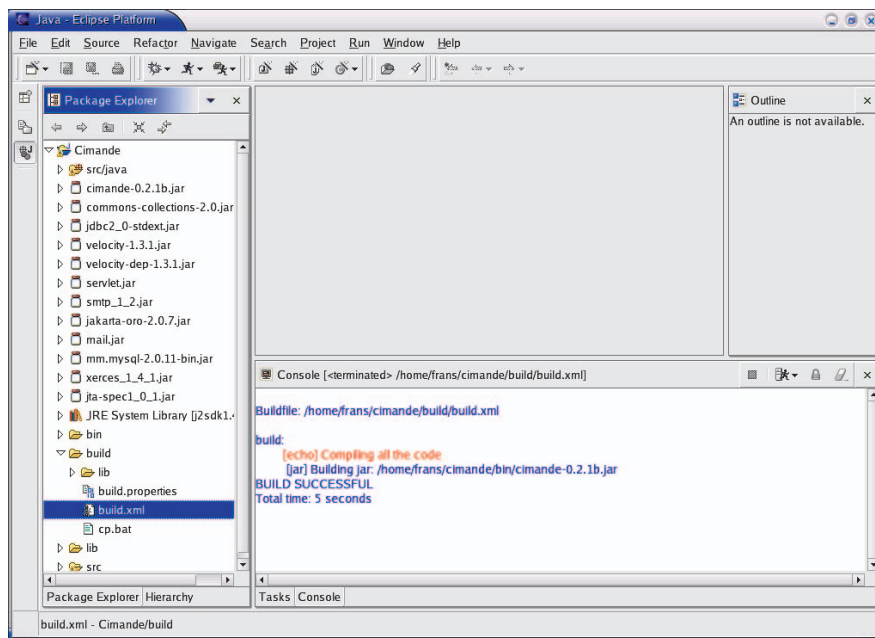
Saat ini didunia Java Open Source ada 2 tools yang sangat bermanfaat yang dapat di-download secara cuma-cuma yaitu Ant (<http://ant.apache.org>) dan Maven (<http://maven.apache.org>). Maven ini berjalan diatas Ant. Maven asal mulanya merupakan sebuah project didalam Turbine, sebuah *project framework* populer di Apache Jakarta.

Ant memungkinkan kita membuat semua aktivitas testing hanya dalam sebuah file XML, umumnya build.xml. Integrasinya dengan beberapa tools automated testing, memungkinkan kita melakukan continuous integration secara otomatis, karena file script Ant ini akan secara berkelanjutan akan melakukan testing secara terus menerus. Malah kasus update component terbaru, dapat dilakukan dengan menggunakan Maven.

Bekerja dengan Ant

Ant adalah sebuah project Java Open Source yang paling bernilai dalam dunia pengembangan Java. Project ini dikembangkan oleh **James Duncan Davidson**, semenjak 1.5, Ant tidak hanya mendukung Java, tetapi juga Python dan Perl. Untuk mendapatkan Ant edisi terbaru, dapat mengunjungi situsnya <http://ant.apache.org>.

Untuk menginstal Ant, caranya mudah sekali. Dapatkanlah *binary distribution*-nya,



▲ Proses kompilasi dengan eclipse.

kemudian extractlah file tersebut. Anggap saja direktorinya di `/usr/java/jakarta-ant-1.5.3`. Setelah itu dilanjutkan dengan memodifikasi `.bash_profile`. Tambahkan 2 baris di bawah `JAVA_HOME`.

```
JAVA_HOME=/usr/java/j2sdk1.4.2 (bila menggunakan JSDK 1.4.2)
```

```
export $JAVA_HOME
```

```
ANT_HOME=/usr/java/jakarta-ant-1.5.3
```

```
export $ANT_HOME
```

```
PATH=$PATH:$ANT_HOME/
bin:$JAVA_HOME/java
```

```
export $PATH
```

setelah selesai coba lakukan relogin terhadap account yang dimodifikasi, ketik Ant di console. Maka akan keluar sebuah pesan kesalahan seperti di bawah ini:

```
Buildfile: build.xml does not exist!
```

```
Build failed
```

Ini artinya Ant sudah terinstal, dan siap digunakan untuk pengembangan.

Ant Console Development

Memulai sebuah project dengan dengan Ant untuk pemula terkadang sulit, karena harus membaca tag-tag XML yang harus didefinisikan dahulu, tetapi tidak perlu

khawatir, karena penulis telah membuat sebuah `ant-init.zip`, sebuah archive file siap pakai sehingga bisa digunakan untuk project Java baru. Ant-init ini sudah siap digunakan project Java standalone ataupun menggunakan Eclipse IDE.

Bila file ini telah di-extract, seorang programmer tinggal memindahkan package aplikasi Javanya ke direktori `/src/java`. Misalnya kalau package tersebut adalah `org.blueoxygen.cimande`, maka di bawah direktori java akan ada file `*.java` di `/java/org/blueoxygen/cimande`.

Setelah itu lakukan modifikasi `build.xml` yang disertakan (terdapat di directory `build`) editlah tag di bawah ini:

```
<property name="project.fullname"
value="Cimande Project"/>
```

```
<property name="project.name"
value="cimande"/>
```

```
<property name="project.version"
value="0.2b"/>
```

Tag ini yang nantinya akan menghasilkan sebuah file `cimande-0.2b.jar`.

Kemudian modifikasi tag di arena `javadoc`, tag ini akan membuat `javadoc` secara otomatis.

```
<javadoc
sourcepath="${src.java.dir}"
destdir="${javadocs.destdir}"
packagenames="org.blueoxygen.
```

```

cimande.*"
author = "true"
private = "false"
version = "true"
use = "true"
windowtitle = "${project.fullname}
API"
doctitle = "${project.fullname}"
bottom = "Copyright &#169; ${year},
BlueOxygen Foundation. All Rights
Reserved."
>

```

Yang diubah hanya packagenames menjadi package project kamu, kasus di atas adalah package untuk org.blueoxygen.cimande. Kehebatan tag ini adalah, semua code Java yang bukan termasuk package org.blueoxygen.cimande tidak akan di-generate menjadi javadoc.

Setelah itu cobalah buat sebuah code di bawah /src/java dan lakukan execution dengan cara mengetik ant di bawah directory build.

```

> cd /home/frans/cimande/build
> ant

```

Maka secara otomatis akan terjadi sebuah proses kompilasi source code, terus akan dilakukan *archiving bytecode* menjadi sebuah file .jar, setelah itu akan dilakukan generate javadoc.

Menggunakan Ant-Init dengan Eclipse

Berterimakasihlah dengan tim IBM yang telah membuat Ant terintegrasi penuh dengan Eclipse, sehingga kita tidak perlu report menginstall plug-ins-nya. Ant-ini yang penulis buat ini dapat juga bekerja dengan baik di Eclipse IDE versi 2.x dan 3.x. Hanya cara pendekatannya tidak sama dengan membuat sebuah project Java di Eclipse.

Untuk melakukannya cobalah buat sebuah project Java, tetapi setelah directorinya bukan default tetapi mengarah ke direktori di mana ant-init ini di-extract.

Setelah itu klik *Next*, dan Eclipse secara otomatis akan mencari semua setting untuk memulai project. Direktori lib akan dibaca, sehingga semua .jar yang ada, langsung terinclude. File source code dan binary secara otomatis akan dipisah, untuk

source akan diarahkan ke directory src. Sedangkan hasil compile akan ditaruh di direktori /bin/classes.

Cobalah masuk ke directory build, dan arahkan mouse kamu ke file build.xml (build.xml akan diberi icon semut), dan lakukan click kanan, terus pilihlah Run Ant. Secara otomatis akan keluar menu, coba pilihlah build, build akan meng-compile source code, dan meng-generate sebuah java archive sesuai dengan tag modifikasi diatas. Bila berhasil di-console Eclipse akan keluar lokasi di mana file .jar itu dibuat.

Untuk membuat javadoc, dapat dilakukan cara di atas, dengan memilih target javadoc, tetapi Eclipse memiliki kemampuan untuk membuat Javadoc dengan Javadoc Wizard-nya.

Penggunaan Ant di dunia nyata

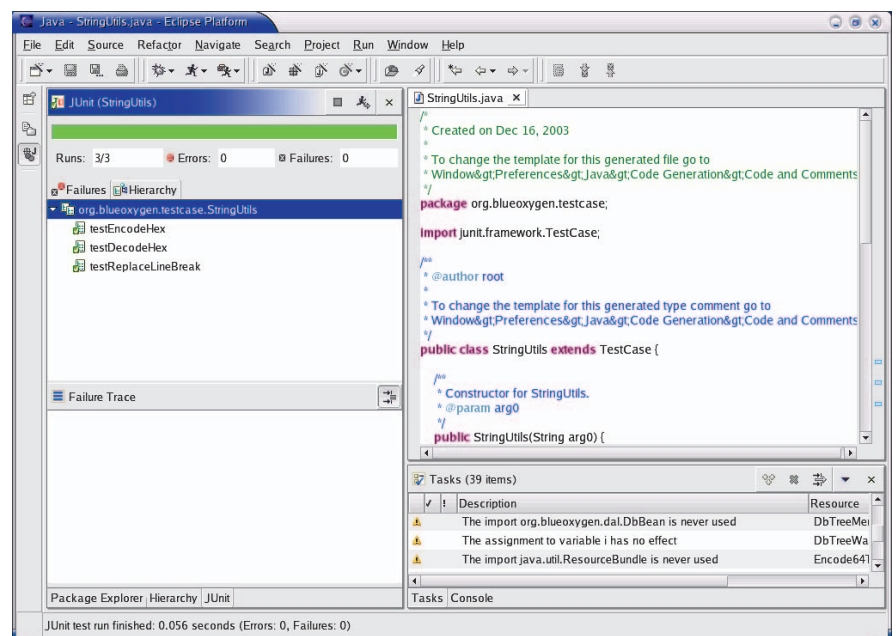
Dalam pengembangan sesungguhnya, Ant ini bekerja baik sekali dengan CVS, sayang sekali module Subversion (CVS versi terbaru) belum ada. CVS memiliki beberapa kekurangan yang telah diperbaiki oleh Subversion. Saat ini, walaupun modul untuk subversion belum tersedia, dalam waktu singkat, harapannya akan segera tersedia. Ant ini adalah solusi yang efektif untuk pengembangan secara tim.

Umumnya setiap anggota tim, akan melakukan *checkout intial project structure* dari server baik itu CVS maupun Subversion, kemudian tim akan bekerja secara lokal di komputer masing-masing. Setelah selesai, programmer tersebut menjalankan Ant, setelah semua sukses, baru dikirim ke server CVS. Dengan demikian, pembuatan suatu aplikasi dapat dibagi-bagi sehingga waktu pengerjaan menjadi lebih singkat. Setelah update selesai, tim member lainnya tinggal update repository lokal mereka, dan secara otomatis code rekan tim yang lain pindah ke repository-nya di PC-nya itu.

Jangan mengirim code yang belum dicoba ke CVS server. Hal ini merupakan suatu etika yang perlu dipahami dalam pengerjaan software secara teamwork. Setiap programmer memiliki tanggung jawab masing-masing, termasuk tanggung jawab agar kerja sama tim tetap baik. Apabila seorang programmer mengirimkan code yang belum teruji, maka ketika terjadi kesalahan, pihak yang melakukan merger akan mengalami kesulitan.

Ant dapat diintegrasikan dengan Junit untuk melakukan unit testing, umumnya sebelum kode terbaru dikirim ke server CVS, Junit akan menghasilkan sebuah mekanisme unit testing.

Frans Thamura (frans@intercitra.com)



▲ Eclipse menjalankan unit testing.