

Continuous Integration Practice

<http://morningspace.51.net/>
<mailto:morningspace@126.com>

Agenda

- ★ Overview
- ★ Build Scripts
- ★ Guidelines
- ★ Reference

Overview

Concept

- ✦ A fully automated, reproducible build
- ✦ Including compiling, testing, ...
- ✦ A term used as one of the primary practices of XP
- ✦ The practice has been used from a long time ago.
 - ✦ Daily build, recommend by McConnell in his Rapid Software Development, is known as a feature of the Microsoft development approach
- ✦ It's anecdotal evidence, without rigorous scientific study

How it works?

- ✦ Retrieve the repository snapshot and put into the working directory.
- ✦ Call the build scripts
- ✦ Create build log, report, etc.
- ✦ Send email after build completed

How to make it work?

- ✦ At the very beginning, keep a single source repository
 - ✦ CVS, SVN, VSS, etc.
- ✦ Automate the build process
- ✦ Automate the testing, BVT(build verification tests)
- ✦ Anyone can get the newest well-executable result
- ✦ Take a certain amount of discipline. When it's stable, there is not too much effort to keep it running.

What's the benefit?

- ✦ For team development, let the problem occur as early as possible
- ✦ No need to take a long time to find integration bugs
- ✦ Be helpful to small releases, and have the members more confident about their working

My own experience

- ✦ Gather code pieces from other people. Ensure the integration result by hand. Professional guys use tools such as Win Diff.
- ✦ Long term integration, which is called *integration hell*, over time working, leads to discourage.
- ✦ Be in fear of making demonstration for the customer

Other things

- ★ The key points are
 - ✿ Automation: Get the code, compile, test, report, deploy..., at last, failure or success.
 - There're also many cool things such as starting and stopping server automatically. But that's not the main idea.
 - ✿ Do it frequently
- ★ Just getting a program to compile is not enough (catch compile-time error).
 - ✿ To make the C.I. more significantly, We should automate the test as much as possible to catch the run-time bugs.

The more often the better

- ✦ The effort of integration is exponentially proportional to the amount of time between integrations
- ✦ Microsoft does daily builds on projects with tens of millions of lines of code.
- ✦ Run BVT several times a day. As the scope of problem is quite small, correcting the bug is easy.
- ✦ The extreme practice: build automatically whenever check in

Continuous Integration Systems

- ★ Cruise Control, *ThoughtWorks, open source*
- ★ Damage Control, *ThoughtWorks, open source, written by ruby*
- ★ Anthill, *open source*
- ★ Gump, *open source*
- ★ LuntBuild, *open source*
- ★ Anthill professional
- ★ ...

Build Scripts

The importance of build scripts

- ✦ A fully C.I. System is not only including C.I. Platform itself
- ✦ A good set of build scripts should allow you to build alternative targets for different cases
- ✦ Take ant into account:
 - ✦ Target dependency
 - ✦ Immutable property
 - ✦ ...

Immutable Properties

- ✦ Put the user specific properties into `{user.home}`, or use `-D` in IDE. Never commit them into repository.

High Priority



Low Priority

- ✦ Read build-specific properties
- ✦ Read user-specific properties
- ✦ Read project-specific properties
- ✦ Default property values

Token replace

- ✦ Write a template, then replace token during build.

- `<target name="token-replace">`
- `<copy todir="${build.dir}">`
- `<fileset dir="${conf.dir}">`
- `<include name="*.properties" />`
- `</fileset>`
- `<filterset begintoken="%" endtoken="%">`
- `<filter token="a" value="b" />`
- `</filterset>`
- `</copy>`
- `</target>`

- ✦ *Be careful when deal with testing*

Cross platform

- ✦ `<condition property="os.windows">`
- ✦ `<os family="windows"/>`
- ✦ `</condition>`
- ✦ `<condition property="os.unix">`
- ✦ `<os family="unix"/>`
- ✦ `</condition>`
- ✦ `<target name="..." if="os.windows">`
- ✦ `... ..`
- ✦ `</target>`
- ✦ `<target name="..." if="os.unix">`
- ✦ `... ..`
- ✦ `</target>`

New features in ant 1.6

- ✦ `<MacroDef>`, user defined task, no need to write code.
- ✦ `<import>`, similar to xml entity import but more powerful.
- ✦ `<subant>`, call a given target for all defined sub-builds
 - ✦ `<subant target="compile">`
 - ✦ `<property name="build.dir" value="subant1.build"/>`
 - ✦ `<fileset dir="." includes="*/build.xml"/>`
 - ✦ `</subant>`

Large scale project(1)

★ master build and child build

- `<target name="do-tools">`
- `<ant dir="tools" target="${target}" inheritAll="false"/>`
- `</target>`
- `<target name="do-client" depends="do-tools">`
- `<ant dir="client" target="${target}" inheritAll="false"/>`
- `</target>`
- `<target name="do-all" depends="do-tools,do-client"/>`
- `<target name="all" description="build everything">`
- `<antcall target="do-all">`
- `<param name="target" value="all"/>`
- `</antcall>`
- `</target>`

Large scale project(2)

- * `<?xml version="1.0"?>`
- * `<!DOCTYPE project [`
- * `<!ENTITY targets SYSTEM "file:../targets.xml">`
- * `]>`
- * `<project name="webapp" default="default" basedir=".">`
- * `&targets;`
- * `<target name="clean" description="clean everything" />`
- * `<target name="all" depends="dist"`
- * `description="build everything"/>`
- * `<target name="all" depends="clean, dist"`
- * `description="build everything"/>`
- * `</project>`

Dependency Loop

- ☀ Be careful about dependency loop when dealing with large scale project.
 - ☀ Draw a dependency graph before writing build scripts
 - ☀ Make the packages well-structured(R. C. Martin, Agile Software Development)
 - ☀ Metrics(<http://metrics.sourceforge.net>)
- ☀ How to avoid package dependency loop?
 - ☀ Merge sub-project
 - ☀ Extract common sub-project
 - ☀ Programming to interface

Some other things should be mentioned

- ★ Don't forget to remove duplication
 - ★ Use “target library”
 - ★ ...
- ★ How to deal with library dependency
 - ★ Does dependent lib need to be committed into repository?

Guidelines

About Repository

- ✦ Put all the source into repository: code, scripts, properties, dependent lib (optional), etc.
- ✦ Use one source tree. Let the build scripts do all the things for you
- ✦ To Integrate frequently, check in frequently. For daily build, check in once a day at least.
- ✦ Synchronize with repository frequently
- ✦ ...

About Build

- ✦ Use different configuration between C.I. platform and IDE to make the build more flexible
- ✦ Use different build environment while developing, testing and deploying
- ✦ Use incremental build carefully in IDE. Clean build is necessary sometimes
- ✦ The more you depend on specific IDE, the harder auto build can be run.
- ✦ When build failure on C.I. platform, make master build on your local machine to find the problem
- ✦ ...

Misc.

- ✦ Use mail client such as outlook☺
- ✦ Some people should be responsible for track the build result. (one of my awful experience)
- ✦ ...

Reference

- ✦ <http://martinfowler.com/articles/continuousIntegration.html>
- ✦ <http://www.objectmentor.com/resources/articles/oodmetric.pdf>
- ✦ <http://cruisecontrol.sourceforge.net>
- ✦ <http://www.urbancode.com/products/anthillpro/default.jsp>
- ✦ <http://ant.apache.org/manual/>
- ✦ Erik Hatcher, Steve Loughran, Java development with Ant
- ✦ <http://docs.codehaus.org/display/DAMAGECONTROL/Continuous+Integration+Server+Feature+Matrix>
- ✦ ...

**The End
Thanks!**