Beginner’s guide to continuous integration

Gilles QUERRET
Riverside Software
About the speaker

- Working with Progress and Java since 10 years
- Started Riverside Software 5 years ago
- Based in Lyon, France
- Focused on technical expertise and continuous integration in those environments
- Selling WebClient automation solution
WHEN YOU HEAR THIS:

YOU KNOW YOU'RE IN A SOFTWARE PROJECT

YESTERDAY IT WORKED

JUST IN CASE YOU'RE STILL NOT SURE WHETHER YOU'RE IN A SOFTWARE PROJECT

WAIT UNTIL YOU HEAR THIS:

ON MY MACHINE IT WORKS

http://geekandpoke.typepad.com
Agenda

- Definitions
- Implementing continuous integration in a few steps
- Demo
define: build automation

Source code

Build scripts

Deliverables
define: continuous integration

- Build automation
- Easy access to deliverables
- Automated deployment
- Automated tests

CONTINUOUS INTEGRATION
Agenda

- Definitions
- Implementing continuous integration in a few steps
- Demo
Steps to CI

1. SCM setup
2. Build automation
3. CI tool setup
4. Automated deployment
5. Automated tests
Use the right tools
Step 1: code repositories

- **Client / Server:**
  - CVS / SVN
  - ClearCase
  - Perforce
  - VSS

- **Distributed**
  - Mercurial
  - Git
  - BitKeeper

**Today at 1:15pm**
Version control with Mercurial and Progress
Julian Lyndon-Smith

**Today at 4:00pm**
SCMS & Build automation panel discussion
Julian Lyndon-Smith
Michael Solomon
Gilles Querret
Step 1: code repositories

- If you can generate something, don’t store it in your SCM
  - It will be part of your build script
- Separate requirements and dependencies
  - OpenEdge is a requirement, pdf_include is a dependency
- Don’t forget database versioning
- Commit as much as possible, using branches if necessary
- Associate a bugtracker to your SCM
Step 2: Build automation

- Generate PL
- Compile
- Create DB

Diagram showing the processes in sequence.
Step 2: Build automation

- Many tools involved during this process
  - OE procedures
  - Shell scripts
  - Ant + PCT for OpenEdge tasks
Step 2 : Build automation

```xml
<PCTCreateBase dbName="ged" destDir="$\{db\}"
    codepage="utf" schemaFile="db/schema1.df,db/schema2.df"
    structFile="db/struct.st" blockSize="4"
    dlcHome="$\{DLC\}" />

<PCTRun procedure="src/initDb.p" paramFile="conf/param.pf"
    dlcHome="$\{DLC\}" cpstream="utf-8">
    <PCTConnection dbName="ged" dbDir="$\{db\}" singleUser="yes" />
    <PCTConnection dbName="cust" dbDir="$\{db\}" singleUser="yes" />
</PCTRun>
```
Step 2: Build automation

```xml
<PCTCompile destDir="${build}" graphicalMode="true"
dlcHome="${DLC}" md5="false" minSize="false" cpinternal="iso8859-15"
cpstream="iso8859-15" inputChars="16384" debugListing="true">
  <fileset dir="src/core" includes="**/*.p,**/*.w" />
  <fileset dir="src/module1" includes="**/*.p,**/*.w" />
  <fileset dir="src/oo" includes="**/*.cls" />
  <PCTConnection dbName="ged" dbDir="${db}" />
  <PCTConnection dbName="cust" dbDir="${db}" />
  <propath>
    <pathelement location="src/core" />
    <pathelement location="src/oo" />
  </propath>
</PCTCompile>
```
Step 2: Build automation

- Use common build tools
- Check for requirements
- Download dependencies
- Use variables as much as possible
- Use different folders for build objects
Step 3: CI Servers

- Many products, but same functionalities:
  - Define and trigger jobs
  - Store deliverables (keeping history)…
  - And make them easily available
  - Keep users informed of build results

- Server choice:
  - Free or not (support subscription)
  - Integration with your tools
  - Plugins
Step 3 : CI Servers, some names

- Cruise control
- Hudson / Jenkins
- Teamcity (JetBrains)
- Bamboo (Atlassian)
Step 3 – A few tips

- Always use a clean server
- Use distributed jobs
- Define a job for every product and every environment (branches / clones)
- Keep deliverables only for production jobs. Keep only a dozen for integration
- Only send alerts for failures!
Step 4: Deployment

- Use only what have been generated during the build process
- But remember requirements...

- Always deploy to a clean server

- Every deliverable generated by the CI server should be deployed
Using virtualization

- Available in VMWare ESX
- Clones, snapshots and remote execution are your friends!
- Define clean VM for every target OS
- Define snapshots to be able to improve configuration
- New clone for a new job
- Remote execution to execute deployment
Step 5 : tests

- Unit testing
- User interface testing
- API testing
- Regression testing
- Load testing
- Security testing
Agenda

• Definitions

• Implementing continuous integration in a few steps

• Bonus track

• Demo
Proparse – PCT – Prolint – Jenkins
Proparse – PCT – Prolint – Jenkins

- New PCT task to generate AST tree for every source file
- Use old ProLint procedures or new Java Lint tasks
- Generates XML output
- Result is presented in Jenkins:
  - Drill-down warnings by source file, category, priority, …
  - Display warnings trend
  - View highlighted source code
Agenda

- Definitions
- Implementing continuous integration in a few steps
- Bonus track
- Demo
Questions ?
Questions?

- Ant: http://ant.apache.org
- PCT: http://pct.rssw.eu
- Hudson: http://hudson-ci.org
- Jenkins: http://jenkins-ci.org

- And contact me directly (g.querret@riverside-software.fr) if you want additional informations