USING PHING FOR FUN AND PROFIT

PHING: A PHP BUILD SYSTEM

Nic Jansma

nicj.net

@NicJ

PHING

Phing is a cross-platform PHP build tool, similar to Apache Ant.

WHAT IS A BUILD TOOL?

A build tool helps you automate repetitive tasks.

A BUILD TOOL HELPS YOU...

- Build resources (CSS, JavaScript, templates, etc)
- Validate code (lint, sniff, etc)
- Run unit tests
- Build documentation
- Create packages
- Deploy code
- Execute system commands
- ... and anything else you do repetitively ...

WHY USE A BUILD TOOL?

Otherwise you will f*** up.

You should automate **everything** you can.

Automating your processes will save you time.

Automating your processes will save you from your future self.

Maybe not today, maybe not tomorrow, but at some point you will make a mistake if it's not automated.

WHY USE PHING?

It's written in PHP, so if the rest of your project is PHP, you can run Phing.

Besides PHP, no other external dependencies are needed (such as Ruby or Java).

Great community support, with hundreds of different tasks.

INSTALLATION - GITHUB

github.com/phingofficial/phing

```
$> git clone https://github.com/phingofficial/phing.git
$> php phing\bin\phing
```

INSTALLATION - PHAR PACKAGE

www.phing.info/trac/wiki/Users/Download

```
$> wget http://www.phing.info/get/phing-latest.phar -O phing.phar
$> php phing.phar
```

INSTALLATION - PEAR

```
$> pear channel-discover pear.phing.info
$> pear install phing/phing
$> phing
```

INSTALLATION - COMPOSER

composer.json:

```
{
    "require": {
        "phing/phing": "2.6.1"
    }
}
```

```
$> php composer.phar install
Loading composer repositories with package information
Installing dependencies (including require-dev)
  - Installing phing/phing (2.6.1)
    Downloading: 100%

Writing lock file
Generating autoload files
$> php vendor\phing\phing\bin\phing
```

PHING OVERVIEW

Phing is driven by XML files that define your projects, build targets, and individual tasks.

The XML format is very similar to Apache Ant.

PHING - PROJECTS

A project is the root element of your XML.

The `<project>` defines of all of your build targets and the tasks that will execute for the targets.

Only the `default` attribut is required, which specifies the default target to run.

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="my-project" default="build">
    <!-- targets and tasks -->
    <!-- ... -->
</project>
```

PHING - TARGETS

A `<target>` is a logical set of actions you want to take.

A `<target>` can have tasks.

A `<target>` can also have a list of other targets it depends on.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project name="my-project" default="build">
    <target name="build" depends="clean,lint,minify" />

    <target name="clean">
        <echo msg="I'm cleaning your build" />
    </target>
    <!-- ... -->
</project>
```

PHING - TASKS

A task will take an action.

Tasks can be a core task that Phing ships with, or an external plug-in that you write.

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="my-project" default="build">
    <target name="clean">
        <echo msg="I'm cleaning your build" />

        <delete dir="build" />
    </target>
</project>
```

PHING - CORE TASKS

123 built-in tasks.

Some of the most useful:

- File operations: `append`, `copy`, `delete`, `mkdir`
- Conditional logic: `condition`, `foreach`, `if`, `fail`
- Input / output: `echo`, `input`
- System: `exec`, `tstamp`, `taskdef`
- Source control: `git*`, `svn*`, `cvs*`
- Network: `ftpdeploy`, `httpget`, `mail`, `s3put`, `scp`, `ssh`
- External Libraries: `phpcodesniffer`, `phpunit`, `phpdocumentor`, `jslint`, `phpmd`

PHING - PROPERTIES

A property is a variable.

Properties can come from built-in properties (Phing environment variables), a `build.properties` file, or created at runtime in your XML file via the `<property>` element.

Use properties via the `${propertyname}` syntax.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project name="my-project" default="clean">
    <property name="builddir" value="./build" />

    <target name="clean">
        <echo msg="I'm cleaning ${builddir}" />

        <delete dir="${builddir}" />
    </target>
</project>
```

PHING - BUILT-IN PROPERTIES

There are many built-in properties:

`application.startdir`, `env.*`, `host.arch`, `host.domain`, `host.fstype`, `host.name`, `host.os`, `host.os.release`, `host.os.version`, `line.separator`, `os.name`, `phing.file`, `phing.dir`, `phing.home`, `phing.version`, `phing.project.name`, `php.classpath`, `php.version`, `project.basedir`, `user.home`

PHING - .PROPERTIES FILES

Simple `key=value` format

```
# This is a comment in the .properties file
key=value

builddir=build

myapp.name=foo
myapp.url=http://foo.com
```

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="my-project" default="clean">
    <property file="./build.properties" />

    <target name="clean">
        <echo msg="I'm cleaning ${builddir} for ${myapp.name}" />

        <delete dir="${builddir}" />
    </target>
</project>
```

PHING - <PROPERTY>

Define new properties in your XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="my-project" default="build">
    <property name="builddir" value="./build" />
    <property name="cssdir" value="${build}/css" />
    <property name="jsdir" value="${build}/js" />

    <target name="clean">
        <delete dir="${cssdir}" />
        <delete dir="${jsdir}" />
    </target>
</project>
```

EXAMPLES

SIMPLE

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project name="my-project" default="echo">
    <target name="echo">
        <echo msg="Hello" />
    </target>
</project>
```

```
>$ phing echo
Buildfile: .\build.xml

my-project > echo:

     [echo] Hello

BUILD FINISHED

Total time: 0.1780 seconds
```

TWO TASKS

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="my-project" default="first">
    <target name="first" depends="second" />

    <target name="second">
        <fail message="You messed up" />
    </target>
</project>
```

```
>$ phing first
Buildfile: .\build.xml

my-project > second:

Execution of target "second" failed for the following reason:
    .\build.xml:16:22: You messed up

BUILD FAILED
.\build.xml:16:22: You messed up
Total time: 0.1800 seconds
```

SHELL COMMANDS

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project name="my-project" default="deploy">
    <property name="deploy.hostname" value="foo.com" />

    <target name="deploy">
        <exec
            command="rsynz -avz ./ ${deploy.hostname}/"
            dir="${project.basedir}"
            checkreturn="true" />
    </target>
</project>
```

EXTERNAL PHP

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project name="my-project" default="externaltask">
    <target name="externaltask">
        <taskdef
            name="myprojecttask"
            classpath="${project.basedir}"
            classname="MyTask" /> <!-- ./MyTask.php -->

        <myprojecttask message='hi' />
    </target>
</project>
```

```php
<?php
require_once 'phing/Task.php';

class MyTask extends Task {
    protected $message;

    // set from the task's attribute in the XML
    public function setMessage($message) {
        $this->message = $message;
    }

    // executed when task is called
    public function main() {
        echo $this->message;
    }
```

EXTERNAL PHP - OUTPUT

```
$>phing externaltask
Buildfile: .\build.xml
 [property] Loading .\build.properties

my-project > externaltask:

hi
BUILD FINISHED

Total time: 0.2930 seconds
```

CONTINUOUS INTEGRATION / BUILD SERVER

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project name="big-project" default="build">
    <!-- main targets -->
    <target name="build" depends="clean,checkout,lint,
        sniff,phpmd,test,doc,package" />

    <target name="deploy" depends="build,pre-deploy,rsync,release" />

    <target name="clean">
        <!-- clean the build -->
    </target>

    <target name="checkout">
        <!-- checkout / pull latest from source control -->
    </target>

    <target name="lint" depends="phplint,jslint,csslint" />
```

DOWNSITDES

- Learning curve if you're not familiar with Ant.
- Asynchronous operation, so large builds/deploys may be slow.
- Not everything is available on all OSs.

CONCLUSION

1. Use Phing
2. ???
3. Profit!

CONCLUSION

Phing is a great way to automate repetitive tasks.

Phing can be as simple or complex (and powerful) as you make it.

Phing can save time and reduce human error.

More Info:

- phing.info
- Jenkins integration
- Jetbrains PhpStorm integration
- Eclipse integration

Thanks - Nic Jansma - nicj.net - @NicJ