Cut to the Chase Series
More Walk – Less Talk

zenyan Builder

farmer.pl
Release 0.97

zenyan Builder

Table of Contents

About zenyan Builder - farmer.pl

farmer.pl is a build, configure, and deployment tool for the zenyan framework. Apart from the configuration functions farmer is a fairly generic program that can be applied to more than managing zenyan. Given that farmer is open source it also can be extended and modified to suit your specific needs. I modeled it somewhat after ant, though I must confess that my understanding of ant is NOT hand-on. My understanding comes from work I did serving in a product management role.

The general concept behind farmer is to promote repeatable process with a high degree of flexibility. These might seem at first blush like conflicting goals, but they really are not.
First I will take you behind the build management philosophy for zenyan. This is the next logical step in understanding how to utilize farmer.

zenyan Build Management Philosophy

Building and managing system builds is a vital engineering discipline. It certainly is not one of the "sexier" engineering activities, but none the less one that is instrumental in the software life cycle of a product. I would go as far to say that a vast dissatisfaction with software can boil down to how this process is managed. It is a very complex discipline that is often overlooked.

It is also a discipline that requires an enormous degree of flexibility. Also meaning that it requires a great deal of specific domain expertise. We have qualified buildmasters for a reason. And, it is not uncommon to see senior development staff taking on the role as the buildmaster gatekeepers!

General Build Process

The zenyan build process is built with the notion that an implementor will need at least two builds. In reality one needs a great many more than this. If you think about it at the very least you should have a development build, a test build, and a production build. If you are using zenyan to release a product to customers then you would at minimum need a development build, a test build, and a release build. You would need this for each version of your software. If you are using zenyan to build custom applications for your customers then you would also need a build for each released version.

Those were just three scenarios. There are many more than that. Anyone needing to manage software builds will have their own unique set of requirements. While the number of scenarios may indeed be finite, there are still to many to adequately craft a "one size fits all" strategy. Hence, we need a tool that is flexible.

It should also be noted that farmer only deals with the file system side of the build. zenyan is a framework that uses a back end database (elixir). While zenyan manages the schema files and certain types of data files it does not manage the database side of the build process. This is covered under a separate process.

General Build Process Steps - Net New Build

1. Clone a system
2. Build out new system
3. Configure new system
4. Package for deployment
5. Deploy
6. Build
7. Test

I read recently where one of the software guru's in the industry said that the build process should be a single step. The only thing I could think was that we have a different definition as to what a build step is. I am a bit of a zealot when it comes to process automation, but with all the variables involved in a build I just do not see how zenyan could be built in a single step. If there is anyone out there that could do this and still meet the scenarios I listed above I would love to see it. I never like to say something is not possible. But a single step build for zenyan is something well beyond my current skill set and imagination.

Now as a matter of practice, let's go through the each of these steps using farmer.

Clone a system

First we need to clone an existing system. In order to do this the cArch command is used.

Syntax

```
cArch <archive filename> <path to archive> <zipargs>
```

Example

```
>perl farmer.pl cArch zenyanTEST.zip c:\webroot\public_html\zenyan\* a
```

This process can take a few minutes with zenyan as their are hundreds of directories and thousands of files that need to be archived.

Once the process has completed an archive file will be created in the directory
`/webroot/public_html/zenyan_builder/buildout`
The output of the clone function is the creation of an archive file. Once this is done we can build out the system.

Build Out New System

I want to stop here and take a moment to discuss some of the variances in system build. In this example we have clones the latest version of the zenyan build. This system resides on the C drive of a Windows server. If we were going to build this system out onto another servers C drive "as-is", without the need for any configuration modifications then we already have an archive file that is ready to deploy, and we could skip directly to the "Deploy" step. However, in this scenario we need to do some configuration.

For our example we are planning on ultimately deploying this to a remote server. Additionally, there are a number of configuration changes that we need to make. In order to do this we first need to build out the system in our local environment. The following accomplishes this task.

Syntax

```
cBuild <archive file> <root build dir>
```

Example

```
>perl farmer.pl cBuild zenyanTEST.zip zenyanTEST
```

This process can also take a few minutes to complete.

After it is done verify that build was successful to the path webroot/public_html. You can also delete the archive file under the buildout directory once you have verified the build.

Configure New System

The configuration step is the only step in the build management process that is specific to the zenyan framework.

1. Under public_html\zenyan_builder\buildin, make a copy of the file TEMPLATE.seed. Name the file <archive file>.seed where <archive file> is the name of your archive file.
2. Open this file in a test editor.

The .seed file contains all of the configuration parameters. We will go through each of these in the table below.

Parameter	Description
ostyp	w=windows, p=posix (Linux, Unix, etc)
Drvltr	Windows drive letter where you will be installing zenyan. Leave as-is if you are deploying to a posix environment.
rootpath1, rootpath2, rootpath3	This is the relative path to the application root directory. Names only. If deploying to Windows we recommend you use the default settings.
rootdirname	This is the rootname of your application directory.
url	Url to the web application. Can be either a dns name or an

	IP. Typically will be an dns name.
dns	DNS name.
ip and fw_host	IP address
fw_logon	Logon to the dbms (typically this would be a MySQL dbms)
fw_password	Password to dbms
db	Default is elixir. This should not be changed unless you are integrating zenyan into another database.

3. Make changes and save file.
4. Run the farmer alcConf command.

Syntax

```
alcConf alcConf <archive rootname> <config manifestname, do not include
extension>
```

Example

```
>perl farmer.pl alcConf zenyanTEST zenyanTEST
```

This process is pretty fast. Since a great deal is happening during the configuration step we will look at farmers output with some narrative added (in blue, italic)

Example Output

farmer checks to see that the build exists

Checking that build path /webroot/public_html/zenyanTEST exists

```
Appears build was created, continuing...
```

farmer attempts go get the seed file

```
Attempting to retrieve seed file
/webroot/public_html/zenyan_builder/buildin/zenyanTEST.seed
```

farmer processes the seed file checking for any omissions

```
Processing seed file /webroot/public_html/zenyan_builder/buildin/zenyanTEST.seed...
No ommissions in seed file... continuing...
```

farmer processes the main configuration file. This file is a map of all the files in zenyan that require potential reconfiguration. The consumer and user of farmer does not need (and should not) to use this file. But, it is good to know that if the zenyan framework adds additional files that require configuration then this file must be updated!

```
Processing /webroot/public_html/zenyan_builder/buildin/zenyan.cfg
```

farmer creates a source directory to place the files it has touched (whether it actually makes

changes to them or not). The reason for this is to make tracking and management simpler that having to root through a full build to locate these files for verification.

farmer also at the same time copies these files into the actual build, replacing the existing files.

```
mkdir /webroot/public_html/zenyan_builder/buildin/src/zenyanTEST
Finished creating transformed config file
/webroot/public_html/zenyan_builder/buildin/zenyanconfstage/pdfobj_template.php.
Copying file to /webroot/public_html/zenyanTEST/stubs/pdfobj_template.php
Moving source file to
/webroot/public_html/zenyan_builder/buildin/src/zenyanTEST/pdfobj_template.php
Finished creating transformed config file
/webroot/public_html/zenyan_builder/buildin/zenyanconfstage/excelobj_template.php.
Copying file to /webroot/public_html/zenyanTEST/stubs/excelobj_template.php
Moving source file to
/webroot/public_html/zenyan_builder/buildin/src/zenyanTEST/excelobj_template.php
Finished creating transformed config file
/webroot/public_html/zenyan_builder/buildin/zenyanconfstage/eDoccon_f.php.
Copying file to /webroot/public_html/zenyanTEST/eDMS/eDoccon_f.php
Moving source file to
/webroot/public_html/zenyan_builder/buildin/src/zenyanTEST/eDoccon_f.php
Finished creating transformed config file
/webroot/public_html/zenyan_builder/buildin/zenyanconfstage/dynagrid_template.php.
Copying file to /webroot/public_html/zenyanTEST/stubs/dynagrid_template.php
Moving source file to
/webroot/public_html/zenyan_builder/buildin/src/zenyanTEST/dynagrid_template.php
Finished creating transformed config file
/webroot/public_html/zenyan_builder/buildin/zenyanconfstage/dgobj_test_mysql.php.
Copying file to /webroot/public_html/zenyanTEST/reposdg/dgobj_test_mysql.php
Moving source file to
/webroot/public_html/zenyan_builder/buildin/src/zenyanTEST/dgobj_test_mysql.php
Finished creating transformed config file
/webroot/public_html/zenyan_builder/buildin/zenyanconfstage/envref.php.
Copying file to /webroot/public_html/zenyanTEST/eConfig/envref.php
Moving source file to
/webroot/public_html/zenyan_builder/buildin/src/zenyanTEST/envref.php
Finished creating transformed config file
/webroot/public_html/zenyan_builder/buildin/zenyanconfstage/pdfobj_test_mysql.php.
Copying file to /webroot/public_html/zenyanTEST/repospdf/pdfobj_test_mysql.php
Moving source file to
/webroot/public_html/zenyan_builder/buildin/src/zenyanTEST/pdfobj_test_mysql.php
Finished creating transformed config file
/webroot/public_html/zenyan_builder/buildin/zenyanconfstage/envvars.php.
Copying file to /webroot/public_html/zenyanTEST/eLib/envvars.php
Moving source file to
/webroot/public_html/zenyan_builder/buildin/src/zenyanTEST/envvars.php
Finished creating transformed config file
/webroot/public_html/zenyan_builder/buildin/zenyanconfstage/xlsobj_test_mysql.php.
Copying file to /webroot/public_html/zenyanTEST/reposxls/xlsobj_test_mysql.php
Moving source file to
/webroot/public_html/zenyan_builder/buildin/src/zenyanTEST/xlsobj_test_mysql.php
Finished creating transformed config file
/webroot/public_html/zenyan_builder/buildin/zenyanconfstage/loggers.php.
Copying file to /webroot/public_html/zenyanTEST/eLib/loggers.php
```

```
Moving source file to
/webroot/public_html/zenyan_builder/buildin/src/zenyanTEST/loggers.php
Re-configuration of /webroot/public_html/zenyanTEST completed.
```

The configuration process is an algorithmic process. Having said this it is recommended that you still validate that your configuration settings were correctly applied to the files as this acts as a secondary qa check to merely reviewing the .seed file you modified.

Package for Deployment

Packaging for deployment involves using the same framer command that was used to clone a system.

Syntax

```
cArch <archive filename> <path to archive> <zipargs>
```

Example

```
>perl farmer.pl cArch zenyanTESTBLD.zip c:\webroot\public_html\zenyanTEST\*
a
```

Note: You can also delete the existing zip file that was part of the clone and reuse that name.

Deploy

Any FTP tool will do the job, but farmer offers a built-in FTP function. This function assumes that you have created reachable FTP connections.

Syntax

```
rDeploy <package name, include extension> <connection>
```

Example

```
>perl farmer.pl rDeploy zenyanTESTBLD.zip summitdev2
```

If you are not sure what connections are available you can run the connlist command.

Example

```
>perl farmer.pl connlist
```

General Build Process Steps - Incremental Update

There are a plethora of scenarios that fall under this process. Here are a few scenarios.

- You have added new functionality to you application and need to compare it to your Gold build to determine the package you need to create for update.
- You found and fixed a critical bug on a customer build and you now want to move the bug fix back into your standard build.

- You have two separate development builds for custom software you have created that you want to assess and merge into a common build.

In all reality, the incremental build process is the one you will use most often. In general the following are the build steps involved in performing an incremental update

1. Compare two systems (Source to Target, and potentially Target to Source.)
2. Analyze the differences (net new files, same file with different timestamp, etc... .)
3. Tweak the build manifest with desired updates.
4. Create an incremental update package.

Compare two systems

There are two compare functions. The first will compare an entire source and target system performing a two-way compare (source to target, target to source.) The second method does a source to target compare, but only on the directories/files you specify. This function is meant to be used by skilled buildmasters or when you are troubleshooting and want to hone in on a specific area of a build.

In most instances comparing the entire source and target systems is the preferred method. This full compare method does not take that long, and it does provide a two-way compare on the following metrics:

- Files on Source not on Target
- Files on Target not on Source
- Files on Target with Newer Timestamps
- Files on Source with Newer Timestamps

To reiterate from the beginning section of the manual, farmer does not dictate what system you declare as the source or target.

Full System Compare

1. Under \webroot\public_html\zenyan_builder\comparein make a copy of TEMPLATE.pkg. Recommend naming the file the name of your build.
2. Edit your .pkg file and save.

The package file contains the four items.

Item	Description
sourcerootpath	This is the root path to the source system. Unless you have deviated from the zenyan standards you do not need to modify this.

targetrootpath	This is the root path to the target system. Unless you have deviated from the zenyan standards you do not need to modify this.
src	This is the source systems directory name
trg	This is the target systems directory name

3. Run the farmer command compsys

Syntax

```
compsys <package_name, no extension>
```

Example

```
>perl farmer.pl compsys zenyanTEST
```

The following status will be echoed back (narrative in italic)

```
saddle up boys and girls here we go, time to compare a source and target build...
------------------------------------------------------------------------------
...now retrieving runtime paramters for compare package:
/webroot/public_html/zenyan_builder/comparein/zenyanTEST.pkg
------------------------------------------------------------------------------
```

Echoes back the path that it is comparing...

```
sourcerootpath: /webroot/public_html/
targetrootpath: /webroot/public_html/
src: zenyan/
trg: zenyanTEST/
```

...as well as the source and target systems being compared

```
$source_sys_loc= /webroot/public_html/zenyan/
$target_sys_loc= /webroot/public_html/zenyanTEST/
------------------------------------------------------------------------------
...now building our compare files, take a chill
------------------------------------------------------------------------------
```

Then farmer tells you which output files it has created.

```
...created audit file
/webroot/public_html/zenyan_builder/compareout/zenyanTESTRESULTS-
REPORT-1302102995.txt
...created build mainfest input file
/webroot/public_html/zenyan_builder/buildin/zenyanTEST-for_target.csv

...created build mainfest input file
/webroot/public_html/zenyan_builder/buildin/zenyanTEST-on_trg_only.csv
...created audit file /webroot/public_html/zenyan_builder/compareout/zenyanTEST-
src-
RAW-1302102995.txt
```

```
...created audit file /webroot/public_html/zenyan_builder/compareout/zenyanTEST-
trg-
RAW-1302102995.txt
```

4. Review the output files in /webroot/public_html/zenyan_builder/compareout

The following table describes the contents of each output file.

File	Description
<pkg>-RESULTS-REPORT-<ts>.txt	These are the results of the compare, grouped by category. • Files on Source not on Target • Files on Target not on Source • Files on Target with Newer Timestamps • Files on Source with Newer Timestamps
<pkg>-src-RAW-<ts>.txt	This is a complete list of all the files in the source system that were compared. It can be saved in source control if desired.
<pkg>-trg-RAW-<ts>.txt	This is a complete list of all the files in the target system that were compared. It can be saved in source control if desired.

IMPORTANT NOTE: The results of <pkg>-RESULTS-REPORT-<ts>.txt get used by the iBuild (Incremental Build) function. Since you will likely need to manually edit this file before running that command you should check the file into your source management system before doing any editing so you have a complete traceable trail!

Incremental System Compare

This function performs a source to target comparison. This function requires a manifest file that specifically lists all the directories and/or files we want to compare.

Authors Note: I added this function as a convenience. I suspect it has a number of applications for use. But, I find that I mostly use the full compare function.

1. Under \webroot\public_html\zenyan_builder\comparein make a copy of inclusions.man. Recommend naming the file the name of your build name
2. Edit your .man file and save.

In this file you add all the directories and files you wish to compare. REMEMBER: the comparison is only done source to target. You also need to set the sourcerootpath, targetrootpath, src, and trg parameters as well.

Please note that if you have a big directory list to include you can run the farmer command dirlst which is a helper function which will create a file with a list of all the directories in the provided path

that you can manually edit and paste into your .man file.

3. Run the farmer compincr command.

Syntax
```
compincr <manifestname, no extension>
```

Example
```
>perl farmer.pl compincr incl_zenyanTEST
```

The following status will be echoed back (narrative in italic)

```
...now retrieving runtime paramters for compare manifest:
/webroot/public_html/zenyan_builder/comparein/incl_zenyanTEST.man
--------------------------------------------------------------------------------
```

Echoes back the path that it is comparing...

```
sourcerootpath: /webroot/public_html/
targetrootpath: /webroot/public_html/
src: zenyanTEST/
trg: zenyan/
Echoes back the files and directories it will compare...
will compare file: cLib/clvars.php
will compare dir: eLog/
```

Then farmer tells you which output files it has created.

```
...created audit file
/webroot/public_html/zenyan_builder/compareout/incl_zenyanTESTRESULTS-
REPORT-1302107246.txt
...created build mainfest input file
/webroot/public_html/zenyan_builder/buildin/incl_zenyanTEST-for_target.csv
```

4. Review the output files created.

Buildmaster Stuff Related to Infrastructure

If you are tasked with managing the farmer code base and internal setup and configuration this is the section.

7-Zip Implementation

farmer uses 7-Zip command line for some of its functions. I have included 7z920.exe for you to

install. I recommend you install it to /zipper. If you install it elsewhere you will need to modify the variable $zip7loc in farmer.

zenyan.cfg

Located in \webroot\public_html\zenyan_builder\buildin

This file includes a complete (hopefully) list of files that contain specific configurations. These files fit into two general classifications:

1. Configuration Files.
2. Example code or code generated programs that contain site specific configurations.

If you have any specific files that you are adding that fit into these two categories then you need to ensure that they get included into this configuration manifest.

TEMPLATE.seed

This contains a complete (hopefully) list of all of the parameters used in zenyan. If you need to add your own custom parameters this is the file they get added to. Besides adding the parameters here you will also need to ensure that a) any files that includes your parameters are referenced in the zenyan.cfg file; b) you will need to add conditionals and any business logic to the farmer subroutine "processSeedFile."

Template Files

You may need to tweak the following template file code to reflect your environment.

```
sourcerootpath~/webroot/public_html/
targetrootpath~/webroot/public_html/
```

These references are contained in the files

- TEMPLATE.pkg
- inclusions.man

located in \webroot\public_html\zenyan_builder\comparein

Connections for FTP Function

To add any FTP connections to the farmer program you need to do the following. Note: an

example connection is included in the program.

- Add a connection string variable

```
my @exampleftpsite =
('theFTPurl','logonname','passwordgoeshere','/webroot/public_html');
my $exampleftpsite = \@exampleftpsite;
```

- Update the @masterftplist array with your new connection name
- Find the rDeploy conditional and add your new connection logic (you can use the example connection code as a template.)

Why this all may seem a bit dicey and not so elegant it was done this way on purpose. You can use any FTP program to move the build files. What I wanted was to create an FTP function within farmer that limited the ability to move files to offer more control to you. Personally, I use FileZilla to move my files around.

You may also will need to download and install the Net::FTP package from CPAN.

Other Potential Tweaks

Within farmer you may need to review the following variables if you are not using the recommended zenyan paths or settings.

```
$locbldpath
$sysbldr_root
$db
```