Drupal Automated Staging Toolkit

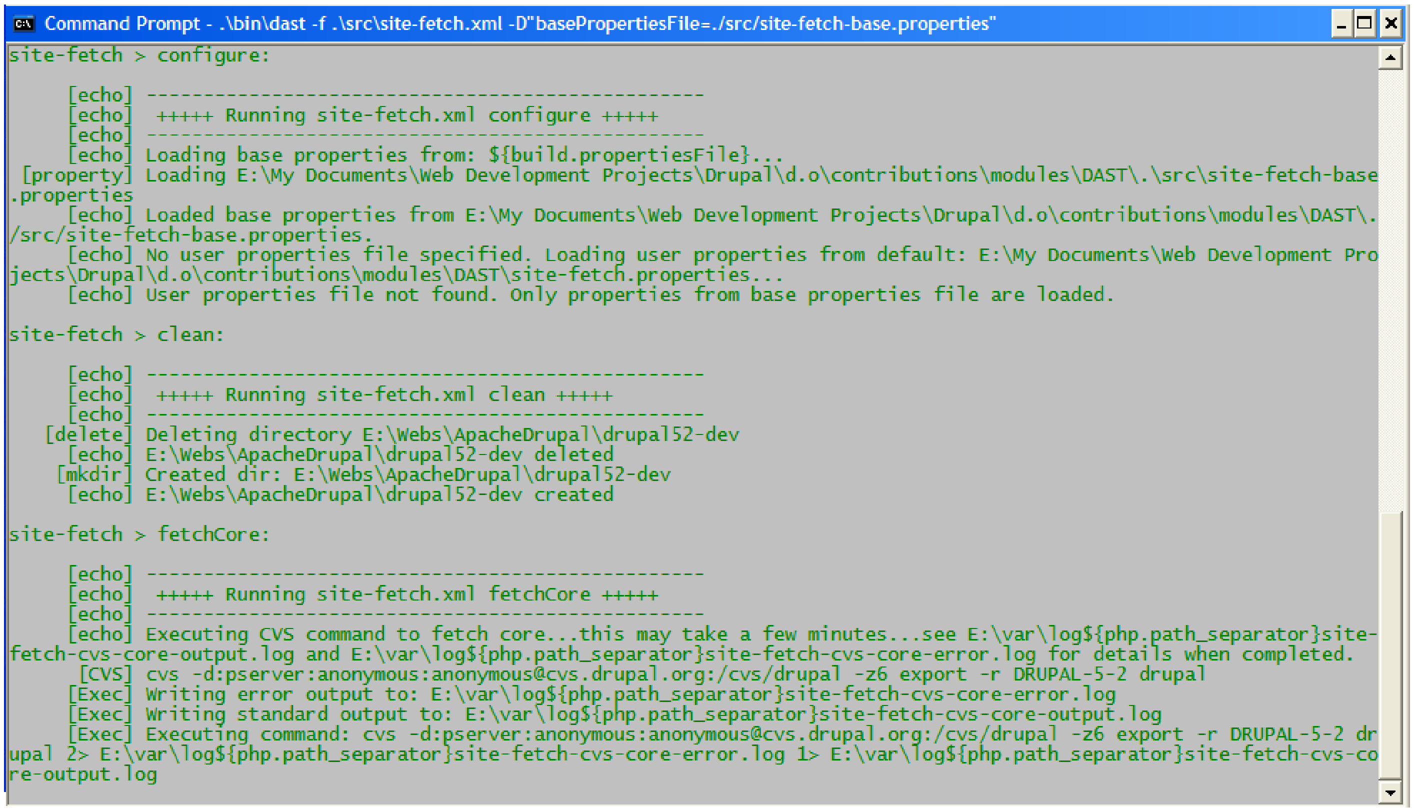
In the name of all Drupal developers testers and administrators we do solemnly swear. Amen.

This page will be the central location for all and sundry documentation for the Drupal Automated Staging Toolkit project - tutorial, user's guide, developer's guide, use cases, code examples, quick-starts, making money from home guides, and everything else related to DAST. Feel free to add or edit anything you find useful or in need of correction.

Existing documentation from my posts on the SoC-2007 group and DAST wiki page will be incorporated here with links to the original articles. For now each section of documentation will reside on the central page but in future as the corpus of docs grows they may be broken into separate pages.

Contents

1. Introduction and Overview


Command Prompt - .\bin\dast -f .\src\site-fetch.xml -D"basePropertiesFile=./src/site-fetch-base.properties"

```
site-fetch > configure:

     [echo] --------------------------------------------------
     [echo]  +++++ Running site-fetch.xml configure +++++
     [echo] --------------------------------------------------
     [echo] Loading base properties from: ${build.propertiesFile}...
 [property] Loading E:\My Documents\Web Development Projects\Drupal\d.o\contributions\modules\DAST\.\src\site-fetch-base
.properties
     [echo] Loaded base properties from E:\My Documents\Web Development Projects\Drupal\d.o\contributions\modules\DAST\.
/src/site-fetch-base.properties.
     [echo] No user properties file specified. Loading user properties from default: E:\My Documents\Web Development Pro
jects\Drupal\d.o\contributions\modules\DAST\site-fetch.properties...
     [echo] User properties file not found. Only properties from base properties file are loaded.

site-fetch > clean:

     [echo] --------------------------------------------------
     [echo]  +++++ Running site-fetch.xml clean +++++
     [echo] --------------------------------------------------
   [delete] Deleting directory E:\Webs\ApacheDrupal\drupal52-dev
     [echo] E:\Webs\ApacheDrupal\drupal52-dev deleted
    [mkdir] Created dir: E:\Webs\ApacheDrupal\drupal52-dev
     [echo] E:\Webs\ApacheDrupal\drupal52-dev created

site-fetch > fetchCore:

     [echo] --------------------------------------------------
     [echo]  +++++ Running site-fetch.xml fetchCore +++++
     [echo] --------------------------------------------------
     [echo] Executing CVS command to fetch core...this may take a few minutes...see E:\var\log${php.path_separator}site-
fetch-cvs-core-output.log and E:\var\log${php.path_separator}site-fetch-cvs-core-error.log for details when completed.
      [CVS] cvs -d:pserver:anonymous:anonymous@cvs.drupal.org:/cvs/drupal -z6 export -r DRUPAL-5-2 drupal
     [Exec] Writing error output to: E:\var\log${php.path_separator}site-fetch-cvs-core-error.log
     [Exec] Writing standard output to: E:\var\log${php.path_separator}site-fetch-cvs-core-output.log
     [Exec] Executing command: cvs -d:pserver:anonymous:anonymous@cvs.drupal.org:/cvs/drupal -z6 export -r DRUPAL-5-2 dr
upal 2> E:\var\log${php.path_separator}site-fetch-cvs-core-error.log 1> E:\var\log${php.path_separator}site-fetch-cvs-co
re-output.log
```

For Unix...

and Windows.

see more sexy screen shots here)

1.1 Introduction

DAST is a PHP 5 CLI application for *nix/Windows for automating the installation tasks necessary to create an instance of the Drupal CMS on a server so that it is ready to be used for a particular purpose. We're not going to fight over synonyms - if staging == deploying == installing ok - what we mean is you have a web server with PHP and a database server somewhere, anywhere, you have *nix shell account on or Windows command line access to the server, and you need to create a Drupal site. Maybe you're a developer and you need to create a site to develop a Drupal module that will make you rich and famous. Maybe you're a tester and you need to create a site to test the constant stream of patches from the Drupal development hives; or you need to duplicate an issue reported on a project you maintain. Maybe you're an administrator and you need to move an existing website from development to testing to production. Maybe you're just an end-user who wants to try-out a Drupal distro made with installation profiles like DrupalEd. My you're an infrastructure or high performance guru who wants to benchmark Drupal on different web server PHP and database server configuration.

There is a great deal of documentation on how to go about the process of installing Drupal. Normally you start from the installation guide and work from there. There is a great deal of know-how on what you you need to do - get the tarballs for the Drupal version you want to install or pull the code code directly from CVS; do the same for the modules you need and the installation profiles you want to use to tailor your site,

create a MySQL or PostgreSQL database; create an Apache virtual host or IIS virtual directory... What doesn't exist is a way to specify the process of installing Drupal in a formal language which can be used by a tool to create an **automated**, **self-documenting** and **reusasble**, process,
and automate executing this process as many times as needed

From the project wiki:

> Right now developing for Drupal requires a lot of manual tasks. If you're a developer or tester looking to run or test code on a particular Drupal site configuration i.e. specific versions of core or module or patches, specific server configurations, and a specific server location where you want to host the site - creating this site requires a lot of manual steps. Drupal developers have several scripts to automate part of these manual steps. DAST's primary objective is to standardize automating the Drupal build process, much like Ant does for the Java build process, or make for C/C++ projects.
>
> DAST is a build tool. It's like Ant for Drupal. Actually it's a build process which will use an automated build framework like Ant to enable creation of a fully functional Drupal site, given the site's description and configuration specified in an XML build file. DAST is for developers and testers who need the ability to quickly build a Drupal site with specific versions of code and enabled modules, a specific app/database/web server configuration, staged to a specific server location. Right now, building a Drupal site is like building a C++ project, source file by source file, tool-by-tool from the command line. With DAST, developers can build and stage a site to a target physical or virtual server the same way they use make or Ant to build C++ or Java software.

1.2 Overview

DAST falls in the category of build utilities or build management tools (and hopefully manages to land on its feet.) It treats the steps required for creating a Drupal site like the steps for building a software project. You have set of components in different physical/conceptual locations that you need to pull together to produce a complete Drupal site. You don't want to keep doing it by hand - you want a script to automate the process of:

1. **Fetching the Drupal core code and module**, - maybe you just want to install the latest stable release or you maybe you need the latest CVS HEAD. You wish to specify the version of core to fetch and a list of modules to fetch.
2. **Configuring the web server**, **PHP and database server to host the Drupal site** - This means at the very least creating a MySQL or PostrgeSQL database. You may also wish to create a new Apache virtual host and also enable or disable certain extensions in PHP
3. **Installing Drupal** - Copy the core and module code to your new web site's physical directory, point the Drupal site database url to your newly created database. Then run the Drupal web installer to launch locale and profile installation.
4. **Patching Drupal** - you may wish to apply a patch to resolve a known issue or as part of your testing activities.
5. **Testing Drupal** - Unit tests are to cool for words. If you're testing patches or developing modules you might want to have a suite of unit tests to run to see if anything breaks functionality that's know to be working

You want a script to automate this and you want the script to be configured through variables so if you want to install Drupal-5.1 instead of Drupal 5.2 or HEAD you only need to change the variable that controls the revision, not the script or the process. You can repeat the process as many times as you like knowing that you will never forget any steps that can lead to a lot of time-wasting and head-scratching.

1.3 Motivation

2. Quick-Start (The Great Use Case of '07)

This article is to get you up-and-running with the motivation, concepts, implementation, and use of DAST through a step-by-step procedure for implementing a set of important use cases. I posted the first use case for DAST in the SoC 2007 group - http://groups.drupal.org/node/4477 on June 8th 2007:

So I volunteered to help test the MSSQL database driver for 5.x and 6.x-dev. Right now Souvent22 is trying to stabilize the install script. So the routine for everybody is:

while (mssql install path != stable) {
1. Checkout DRUPAL-5-1 HEAD
2. Apply mssql patch
3. Delete all the stuff in the test vdir and put in the fresh HEAD
4. drop everything in the SQL Server test database
5. Go to Drupal install page
}

Now after a few iterations this is going to become tedious. So this is what will eventually replace it:

```
<
Licd /media
```

And that's all there is. Just run `phing -Dpatchfile=whatever` A completely automated and error-free repeatable way to test the patch. This is just the first set of tasks that could be automated. The remaining testing tasks:
6. go to site
7. select 'mssql' as database
8. set db options (db name, username, pass, port)
9. click install
10. See what happens next...

could also be automated using Phing's SQL and PHPExec taks. Even cooler, how about automatically running unit tests on the patch, and putting the results showing which tests passed and failed and why, in an HTML file. Implementing this use case is going to be one of the first major milestones in the DAST project.

This was the genesis of the core concepts in DAST - the whole project coalesces around 3 simple core concepts:

- **Document the process steps and workflow needed to create a Drupal site for a particular purpose.**
- **Identify parameters within the process which can be changed without the need to change the entire script**
- **Write a Phing build script which implements the process steps and workflow for creating the site using properties and properties for variable parameters**

Lets jump into this use case. So you're part of the Drupal community and you would like to help out by testing core patches from the Drupal issue queue. In order to do so you need to setup a test environment to review patches. The nature of testing means you may have to setup and tear-down this test environment several times a day. You want to automate creating this patch test-bed.

The following HowTo was first posted to SoC-2007 on July 14th: http://groups.drupal.org/node/5124. The original HowTo is for for the **0.1.1 release only** (**DAST-5.x-1.1** in the Drupal project release nomenclature) but is not forward compatible with newer versions. Unless you have DAST-5.x-1.1 already installed you should skip to the next HowTo.

The dast-patch build script automates the process of creating and patching a fresh Drupal site fetched from CVS. This is a step-by-step guide on using the build script to:
0. Clean the existing directory and database location you want to run the test site from;
1. Fetch core HEAD from CVS;
2. Drop and recreate a MySQL database and user for the site;
3. Fetch a patch file from a URL and apply it to the Drupal site code-tree

Step 1: Install DAST. Download the d.o release package at http://ftp.drupal.org/files/projects/DAST-5.x-1.1.tar.gz. Download the dependencies package from http://www.abeharry.info/stuff/dast-deps.tar.gz or http://www.abeharry.info/stuff/dast-deps.zip. and unpack them to the same dir - /usr/share/DAST for example. There are no additional dependencies besides PHP 5.2.x, cvs, wget, and patch.

Step 2: Set your properties. Each build script comes with one example properties file, in both a *nix and Windows flavour. The *nix sample properties file for dast-patch is dast-patch-nix-sample.properties and for Windows dast-patch-windows-sample.properties. Copy this to dast-patch.properties and edit this file to configure the build with the following properties (comments delimited with #):

patch.method = HTTP #Set to HTTP to fetch the patch from a HTTP URL, dfile to fetch the patch from a local directory

patch.file = #The name of the patch file that will be applied, e.g. update-core-94154-108.patch

patch.Url = #The full absolute Url to the patch file e.g. http://drupal.org/files/issues/update-core-94154-108.patch

drupal.dir = #The directory which will contain the fresh Drupal site, e.g /var/www/drupal6

```
drupal.database.CreateUrl = #The Url to connect to when creating the Drupal user
and db, e.g. mysql://root:root@localhost/mysql
#The account specified must have rights to create and drop the Drupal user and
database. Currently only mysql supported

#The script to run to (re)create the Drupal database and user and privileges. The user
specified in the drupal.database.Createurl must have sufficient privileges to perform
this task. Leave this as is unless you have your own script you want to run.
drupal.database.CreateScript = ${dast.home}/ext/MYSQL5-CREATE-DRUPAL-DB.sql

#The database connection url in the format driver://user:pass@host/dbname of the
database created in the previous step. Leave this as is if using ext/MYSQL5-CREATE-
DRUPAL-DB.sql.
drupal.database.url = mysql://joe:cool@localhost/DRUPAL6

drupal.core.method = CVS #Set this to directory if instead you want to fetch the core
from a local dir

#The ABSOLUTE path to the directory containing the local Drupal 6 core code tree to
be fetched;
#this will be used if drupal.core.method = directory, e.g /usr/share/src/drupal/HEAD:
drupal.core.sourcedir = /usr/share/src/drupal/HEAD

#drupal,core,Cvs.command, drupal.core.CvsRoot...All the drupal.core.Cvs.* defaults
should be ok but modify these properties if you need something different.

#File to write core CVS operations output to, == stdout. Currently this file path
CANNOT CONTAIN SPACES
#on *nix or Windows.
drupal.core.Cvs.Output = /var/log/dast-patch-cvs-core-output.log

#File to write core CVS operations error to == stderror. Currently this file path,
CANNOT CONTAIN SPACES
#on *nix or Windows.
drupal.core.Cvs.Error = var/log/dast-patch-cvs-core-error.log
```

```
#If this is yes, then a default-settings template will be modified to set $db_url =
drupal.database.url and saved as sites/drupal/settings.php
drupal.settings.addDbUrl = yes

#The path to the settings file to use as a template, relative to drupal.dir. Leave this as
is unless you want to use a different default-settings
drupal.settings.source =
sites${php.directory_separator}default${php.directory_separator}default.settings.php

#The path to place the newly configures settings.php file relative to drupal.dir. Leave
this as is unless you want to place the settings somewhere else
drupal.settings.location =
sites${php.directory_separator}default${php.directory_separator}settings.php
```

Step 3: Test the build script From the root directory where you uncompressed the dast package:

```
./bin/dast -f dast-patch.xml configure
```

This will run the configure task which will verify that Phing can be bootstrapped and that all required properties for the dast-patch build are present. (This isn't like autoconf configure - nothing is written to disk.)

Step 4: Run the build script From the root DAST directory:

```
./bin/dast -f dast-patch.xml
```

If all goes well you should have a fresh Drupal site patched and ready for testing. All properties defined in a file can be overridden using the -D switch at the command-line, e.g.

```
./bin/dast -f dast-patch.xml -D"patch.method=skip"
```

To use a local patch file at patch.file.path You can also specify a logfile to store all output using the switch -f

You can also call most tasks individually -e.g `./bin/dast -f dast-patch.xml clean`

For all the properties and tasks you can use to tailor the build see the - *-sample.properties and check README.txt for usage and any general questions.

Addendum to the original - If you specify different database credentials then you must change the user and password in the MYSQL-CREATE-DRUPAL-DB.sql

The updated HowTo is for the 0.2 release and above.

The dast-patch build script automates the process of creating and patching a fresh Drupal site fetched from CVS. This is a step-by-step guide on using the build script to:
0. Clean the existing directory and database location you want to run the test site from;

1. Create the Drupal site database with the user and password you specify

2. Fetch core from CVS;

3. Fetch modules from CVS;

4. Create a MySQL database and user for the site;
5. Fetch a patch file from a URL and apply it to the Drupal site code-tree

Step 1: Install DAST. Download the d.o release package at http://ftp.drupal.org/files/projects/DAST-5.x-2.0.tar.gz. Download the dependencies package from http://www.abeharry.info/stuff/dast-deps.tar.gz or http://www.abeharry.info/stuff/dast-deps.zip. and unpack them to the same dir - /usr/share/DAST for example. There are no additional dependencies besides PHP 5.2.x, cvs, wget, and patch.

Step 2: Set your properties. Each build script comes with one example properties file, in both a *nix and Windows flavour. The *nix sample properties file for dast-patch is dast-patch-nix-sample.properties and for Windows dast-patch-windows-sample.properties. Copy this to **dast-patch-base.properties** and edit this file to configure the build with the following properties (comments delimited with #, replace ? with your values):

```
#Name of the patch file to apply (REQUIRED):
patch.file = ? #e.g 151394-2.patch

#The full HTTP Url of the patch file, used if patch.method = HTTP:
patch.Url = ? #e.g. http://drupal.org/files/issues/151394-2.patch

#The directory which will contain the Drupal code to be patched:
drupal.dir = ? #e.g. /var/www/drupal51-dev/dast/test

#The Url of the newly created Drupal site  (must include trailing slash if no file
specified)
drupal.Url = ? #e.g. http://localhost/drupal51-dev/dast/test/

#The RDBMS which will host the Drupal site database
drupal.database.driver = ? #e.g. mysql

#The name of the server which will host the database - this name is RDBMS specific,
e.g MySQL uses simple
#hostnames while MSSQL mau use aliases defined in the MSSQL client setup, Oracle
may use TNSNames...
drupal.database.host = ? #e.g. localhost

#The name of the database to create which will be used by the Drupal site
drupal.database.name =? #e.g. DRUPAL51
```

```
#The database user which will be used by the Drupal site
drupal.database.user = ? #e.g.joe

#The host that the user will be connecting from
drupal.database.user.host = ? #e.g. localhost

#The password for the database user which will be used by the Drupal site
drupal.database.user.pass = ? #e.g. cool

#The name of the user which will be used to run the DROP/CREATE script to
(re)create the Drupal database and user
#The user account must have sufficient rights to perfom the operations in this script
drupal.database.CreateScript.user = ? #e.g. root

#The password for the drupal.database.CreateScript.user
drupal.database.CreateScript.user.pass = ? #e.g.root

#The database where the ceate script will be executed, e.g. mysql in MySQL or master
in MSSQL
drupal.database.CreateScript.name = mysql

#CVS Branch/Tag revision of core to retrieve
drupal.core.Cvs.Revision = ? #e.g. DRUPAL-5-1

#The list of modules to retrieve from the directory/CVS contrib- repository - comma
delimited.
drupal.modules.list=? #e.g. devel,simpletest,ulink

#CVS Branch/Tag revision of modules to retrieve
drupal.modules.Cvs.Revision = ? #e.g. DRUPAL-5
```

Step 3: Test the build script From the root directory where you uncompressed the dast package:

```
./bin/dast -f dast-patch.xml configure
```

This will run the configure task which will verify that Phing can be bootstrapped and that all required properties for the dast-patch build are present. (This isn't like autoconf configure - nothing is written to disk.)

Step 4: Run the build script From the root DAST directory:

```
./bin/dast -f dast-patch.xml
```

If all goes well you should have a fresh Drupal site patched and ready for testing. All properties defined in a file can be overridden using the -D switch at the command-line, e.g.

```
./bin/dast -f dast-patch.xml -D"patch.method=skip"
```

To use a local patch file at patch.file.path You can also specify a logfile to store all output using the switch -f
You can also call most tasks individually -e.g `./bin/dast -f dast-patch.xml clean`

For all the properties and tasks you can use to tailor the build see the - *-sample.properties and check README.txt for usage and any general questions.

These are the required properties, the rest can be left at the default for the first run. And that's it, the script will go out and fetch the required components and assemble them into a working Drupal site ready for patching. You can change the `patch.*` properties to download and apply another patch and change the `drupal.core.Cvs.Revision and drupal.modules.Cvs.Revision` to fetch different versions of Drupal. You can opt to install Drupal using a local source dir by changing drupal.core.method and drupal.modules.method to directory. You can also run individual sections of the script, for example run the clean target by invoking: `/bin/dast -f dast-patch.xml clean` to drop and recreate the `Drupal database and directory.`

3.2 The base build scripts

---

1. site-fetch.xml

***Description**: This script fetches Drupal core and contrib-module code. It supports CVS and local source directory locations. You specify the CVS revision of the core you wish to download (other CVS parameters like CVSROOT are also configurable but usually the default values will suffice for the vast majority of the cases,) a list of modules you want, and their CVS revisions, and the script will connect to the Drupal CVS repository and export or checkout the requested code to your local dir. You can also use as a source for the Drupal code a local directory to avoid the overhead of fetching from CVS every time.

***Functions** Use this script to:

- Export or checkout core and module code from CVS and copy it into your local Drupal web dir
- Copy Drupal code from a local source dir to your Drupal web dir
- Copy Drupal code from tar ball release to your Drupal wedb dir
- Clean your Drupal web dir of all files before creating a fresh site

***Targets**: configure, clean, fetchCore, fetchModules, main, dump-config .
*Behaviour:

1. Execute public target **configure:**
    1. Load base properties file ${basePropertiesFiles} (default=site-fetch-base.properties} and user properties file ${propertiesFile} (default=site-fetch.properties)
2. Write logs to ${build.log.defaultdir} (default=/var/log/DAST or E:\var\log\DAST)
3. If ${build.run.clean} == yes (default) execute public target **clean**:
    1. Clean target deletes all files and dirs in ${drupal.dir}

4. Execute target **fetchCore**:
    1. If ${drupal.core.method} == CVS (default)
        1. Verify required drupal.core.Cvs.* properties are present and fill in default values for optional properties
        2. Checkout/export Drupal core from CVS into ${drupal.dir}
    2. If ${drupal.core.method} == directory) then copy files from ${drupal.core.sourcedir} into ${drupal.dir}
    3. If
5. If ${patch.method == file then copy ${patch.file.path} to ${dast.ext.patch}
6. If ${patch.method} == file then HTTP GET ${path.url} and save to ${dast.ext.patch}
7. exec patch -d "${drupal.dir}" -${patch.options} < "${dast.ext.patch}${php.directory_separator}${patch.file.path}" -b in directory ${drupal.dir} with <stdout> = ${build.log.defaultdir}${php.directory_separator}dast-patch-patch-output.log and <stderror>=${build.log.defaultdir}${php.directory_separator}dast-patch-patch-error.log
8. The rejects will be found in the same dir as the files to be patched i.e. drupal.dir. You may wand to do a recursive search for all .rej files to verify your patch was accepted

.

2. site-configure.xml

***Description**: This script configures the web server, database server ,PHP and application server on which the Drupal site will run.

***Functions** Use this script to:

- Create a database for the Drupal site on the different types of database servers Drupal supports and (currently only the MySQL driver is supported, support for more drivers will be added very soon.) and configure the database server through directives specified in configuration files. For example in MySQL you can change [mysqld].port and table_cache, replication master/slave settings, anything you want to enable for your site. You can change existing configuration directives, write new ones, and select either existing configuration files or new configuration files to write to. The task can be run with a command to restart the automatically database server after writing the configuration files so your changes can be applied.

- Configure web servers like Apache Httpd to host the Drupal site through, for example, creating an new virtual host or virtual directory. You can write directives to enable modules or specific security directives (like masking .txt files in the Drupal dir,) or any (currently only Apache Httpd; IIS support planned next) web server coonfiguration directive you require for your site. Like the database configuration function, you may write your directives to existing or new configuration files and restart the server so that these files are (re)loaded and your configuration is applied.
- Configure PHP with directives necessary to run Drupal, including enabling required extensions and their configuration (like myysql), changing date and locale settings, enabling error logging and reporting and so on. You can also enable special extensions like memcache and use standard Phing file and environment manipulation functions to complete external configuration of these extensions. You can choose to run a script after you've made your configuration changes to restart all affected servers

***Targets**: configure, configure-create-database, create-database, configure-create-apachehttpdsite, create-apachehttpdsite, configure-create-php-configuration, create-php-configuration, configure-createappserver-sonfiguration, create-appserver-configuration.

*Behaviour: Unlike site-fetch.xml, site-configure does not rely exclusively on properties for its behaviour - the user must write their own actions depending on what configuration changes they wish to make. Properties are used for the values of the configuration directives but what how and where are these configuration directives is determined by the user. An example: consider the site-configure script below:

```
<project name="site-configure" default="main" basedir="../>
```

```
...

  <!-- =======================================
         target: create-apachehttpd-site
        ======================================= -->
  <target name="create-apachehttpd-site" depends="configure-create-apachehttpd-
site"
    description="--> This task will create a new Apache httpd site; modify the files and
directives written here to suit your httpd setup ">

  <!--Add the port for the our site vhost to ports.conf -->
  <ApacheHttpdConf File="${apachehttpd.ports.conf}">
    <ApacheHttpdConfDirective DirectiveName="Listen"
DirectiveContent="${apachehttpd.site.port}" DirectiveOverwrite="no"/>
  </ApacheHttpdConf>

  <!--Make sure the vhost configs in sites-enabled dirs is being pulled in in
apache2.conf -->
  <ApacheHttpdConf File="${apachehttpd.apache2.conf}">
    <ApacheHttpdConfDirective DirectiveName="include"
DirectiveContent="${apachehttpd.sites-enabled.dir}" DirectiveOverwrite="no"/>
  </ApacheHttpdConf>

  <!--Create the vhost in a conf file in "${apachehttpd.site.vhost.conf} -->
  <ApacheHttpdConf File="${apachehttpd.site.vhost.conf}">
    <ApacheHttpdConfSection SectionType="VirtualHost"
SectionName="*:${apachehttpd.site.port}" File="${>
      <ApacheHttpdConfDirective DirectiveName="ServerAdmin"
DirectiveContent="${apachehttpd.site.vhost.ServerAdmin}" />
      <ApacheHttpdConfDirective DirectiveName="DocumentRoot"
DirectiveContent="${apachehttpd.site.vhost.DocumentRoot}" />
      <ApacheHttpdConfDirective DirectiveName="ServerName"
DirectiveContent="${apachehttpd.site.vhost.ServerName}" />
      <ApacheHttpdConfDirective DirectiveName="ErrorLog"
```

```
DirectiveContent="${apachehttpd.site.vhost.ErrorLog}" />
      <ApacheHttpdConfDirective DirectiveName="CustomLog"
DirectiveContent="${apachehttpd.site.vhost.CustomLog}" />
    </ApacheHttpdConfSection>
  </ApacheHttpdConf>

    </target>
```

In this script we are creating a new Apache virtual host. We first add a port for httpd to listen on which is where we'll put our vhost for our Drupal site. We then add misc. directives for the vhost, followed by configuration of the actual Drupal site directory.

```
#The method to fetch the patch file - if HTTP then the patch will be downloaded from
patch.Url; if file
#then patch.file will be used. If skip, then Drupal won't be patched  (REQUIRED)
patch.method = HTTP
#Name of the patch file to apply (REQUIRED):
patch.file = 151394-2.patch
#Command-line options passed to the patch command. (REQUIRED)
patch.options = p0
#The full HTTP Url of the patch file, used if patch.method = HTTP:
patch.Url = http://drupal.org/files/issues/151394-2.patch
#The full absolute location of the patch file on the filesystem, used if
patch.method=file:
patch.file.path = E:\My Documents\Web Development
Projects\Drupal\Patches\151394-2.patch
#The directory which will contain the Drupal code to be patched: (REQUIRED)
drupal.dir = E:\Webs\drupal-510\dev\dast\test
```

3. site-patch.xml

***Description**: This script applies a patch file using the patch command to a Drupal site code-tree. It saves the output chunks of the patch process in the #{dast.ext}/patch directory. You specify the patch file, the url or directory to fetch it from, and the options you want the patch command to use: the default is -p0
***Targets**: configure, fetch-patch, exec-patch, main
***Behaviour**:

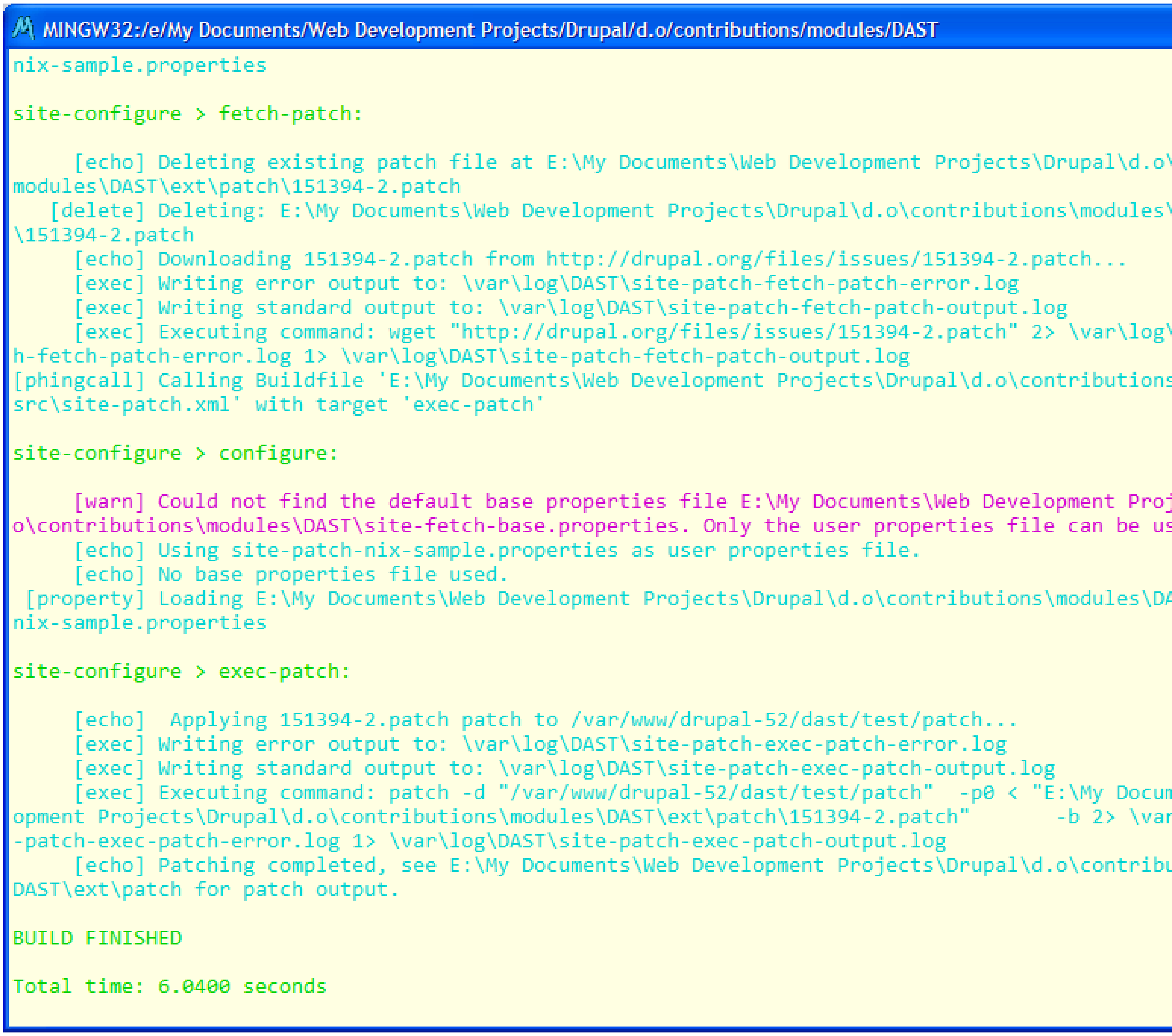
1. Loads base properties file ${basePropertiesFiles} (default=site-patch-base.properties} and user properties file ${propertiesFile} (default=site-patch-user.properties)
2. Defines ${dast.ext.patch} = ${dast.home}${php.directory_separator}ext
3. Writes logs to ${build.log.defaultdir} (default=/var/log or E:\var\log)
4. If ${patch.method} == HTTP (default) then wget ${path.url} and save to ${dast.ext.patch}
5. If ${patch.method == file then copy ${patch.file.path} to ${dast.ext.patch}
6. If ${patch.method} == file then HTTP GET ${path.url} and save to ${dast.ext.patch}
7. exec patch -d "${drupal.dir}" -${patch.options} < "${dast.ext.patch}${php.directory_separator}${patch.file.path}" -b in directory ${drupal.dir} with <stdout> = ${build.log.defaultdir}${php.directory_separator}dast-patch-patch-output.log and <stderror>=${build.log.defaultdir}${php.directory_separator}dast-patch-patch-error.log
8. The output of the patch execution will be found in ${dast.ext}/patch. The originals and rejects will be found in the same dir as the files to be patched i.e. drupal.dir. You may wand to do a recursive search for all .rej and orij files to verify your patch was accepted

***Properties**

```
#dir for DAST log files (REQUIRED)
build.log.defaultdir = E:\var\log\DAST
```

```
#The method to fetch the patch file - if HTTP then the patch will be downloaded from
patch.Url; if file
#then patch.file will be used. If skip, then Drupal won't be patched  (REQUIRED)
patch.method = HTTP

#Name of the patch file to apply (REQUIRED):
patch.file = 151394-2.patch

#Command-line options passed to the patch command. (REQUIRED)
patch.options = p0

#The full HTTP Url of the patch file, used if patch.method = HTTP:
patch.Url = http://drupal.org/files/issues/151394-2.patch

#The full absolute location of the patch file on the filesystem, used if
patch.method=file:
patch.file.path = E:\My Documents\Web Development
Projects\Drupal\Patches\151394-2.patch

#The directory which will contain the Drupal code to be patched: (REQUIRED)
drupal.dir = E:\Webs\drupal\dev\dast\test
```

***Example**


MINGW32:/e/My Documents/Web Development Projects/Drupal/d.o/contributions/modules/DAST

```
nix-sample.properties

site-configure > fetch-patch:

     [echo] Deleting existing patch file at E:\My Documents\Web Development Projects\Drupal\d.o\
modules\DAST\ext\patch\151394-2.patch
   [delete] Deleting: E:\My Documents\Web Development Projects\Drupal\d.o\contributions\modules\
\151394-2.patch
     [echo] Downloading 151394-2.patch from http://drupal.org/files/issues/151394-2.patch...
     [exec] Writing error output to: \var\log\DAST\site-patch-fetch-patch-error.log
     [exec] Writing standard output to: \var\log\DAST\site-patch-fetch-patch-output.log
     [exec] Executing command: wget "http://drupal.org/files/issues/151394-2.patch" 2> \var\log\
h-fetch-patch-error.log 1> \var\log\DAST\site-patch-fetch-patch-output.log
[phingcall] Calling Buildfile 'E:\My Documents\Web Development Projects\Drupal\d.o\contributions
src\site-patch.xml' with target 'exec-patch'

site-configure > configure:

     [warn] Could not find the default base properties file E:\My Documents\Web Development Proj
o\contributions\modules\DAST\site-fetch-base.properties. Only the user properties file can be us
     [echo] Using site-patch-nix-sample.properties as user properties file.
     [echo] No base properties file used.
 [property] Loading E:\My Documents\Web Development Projects\Drupal\d.o\contributions\modules\DA
nix-sample.properties

site-configure > exec-patch:

     [echo]  Applying 151394-2.patch patch to /var/www/drupal-52/dast/test/patch...
     [exec] Writing error output to: \var\log\DAST\site-patch-exec-patch-error.log
     [exec] Writing standard output to: \var\log\DAST\site-patch-exec-patch-output.log
     [exec] Executing command: patch -d "/var/www/drupal-52/dast/test/patch"  -p0 < "E:\My Docum
opment Projects\Drupal\d.o\contributions\modules\DAST\ext\patch\151394-2.patch"       -b 2> \var
-patch-exec-patch-error.log 1> \var\log\DAST\site-patch-exec-patch-output.log
     [echo] Patching completed, see E:\My Documents\Web Development Projects\Drupal\d.o\contribu
DAST\ext\patch for patch output.

BUILD FINISHED

Total time: 6.0400 seconds
```

In this case we're downloading the 151304-2.patch from the Url and applying it to the site at /var/www/drupal-52.**...**

3. site-install.xml

***Description**: This script runs the final installation steps after the Drupal site-code is in place and patched if needed, and the server hosting the Drupal site has been configured. In a typical build project this would be the fourth of the base build scripts used. site-install.xml sets the $db_url variable to point to the database to be used by Drupal, downloads if required the profile

***Targets**: configure, fetch-patch, exec-patch, main

***Behaviour**: