Forrestbot - automated building and deploying

Table of contents

1. Overview

Forrestbot lets you automate building and deploying websites. The whole process gets the source docs, builds the site, then deploys it where you want it to go. Forrestbot can also notify you afterwards, and it keeps a log of the build process. To accomplish these tasks, Forrestbot uses four "workstages" (getsrc, build, deploy, notify). Each workstage has various implementations (e.g., getsrc has getsrc.cvs or getsrc.svn or getsrc.local implementations), which have various properties that may be set, depending upon the implementations chosen.

2. Using Forrestbot

You need to create a customized buildfile directing Forrestbot's work and then simply execute:

```
forrest -f mybuildfile.xml
```

This project buildfile is simply an Ant buildfile with specific targets that control Forrestbot's operation. The next section explains how to create such a buildfile. For details on the syntax of Ant buildfiles and the operation of Ant itself consult the Ant documentation.

3. Creating a Forrestbot Project Buildfile

Within the new buildfile, you need to first set properties needed by the workstages you are going to use and then specify which implementations will be used by each workstage. Note that the properties need to be set at the global scope (as children of `<project>`, i.e., outside of `<target>` elements) in order for your settings to override the defaults in the Forrestbot implementation. Other than that, the property definitions can appear anywhere before the `<import>` task.

This sample buildfile can be used as a base from which to customize your own buildfile. The file starts with the project name and default target, then sets the specific properties we need to get the source, indicate the deployment location, and set up notification. It then specifies which implementations we will use and finishes up with importing the `forrestbot.xml` file. The 'main' target, which is specified as the default here, is a convenience target (defined in `forrestbot.xml`) that executes the four workstages (getsrc, build, deploy, notify) sequentially.

```
<project name="mysampleproject" default="main">
  <property name="notify.email.host" value="smtp.myhost.com"/>
  <property name="notify.email.to" value="me@domain.com"/>
  <property name="notify.administrator" value="Your Name"/>
  <property name="getsrc.cvs.user" value="anoncvs"/>
  <property name="getsrc.cvs.password" value="anoncvs"/>
  <property name="getsrc.cvs.root" value="/home/cvspublic"/>
  <property name="getsrc.cvs.host" value="cvs.myhost.com"/>
  <property name="getsrc.cvs.module" value="myproject"/>
  <property name="deploy.scp.dest"
      value="username@myhost.com:/var/www/mydomain/htdocs"/>

  <!-- here we declare the ssh keyfile and passphrase in an external file -->
  <import file="../deploy-settings.xml" optional="true"/>

  <!-- here we specify to use two notification implementations -->
  <target name="notify" depends="notify.local, notify.email"/>

  <!-- here we specify to deploy with the scp implementation -->
  <target name="deploy" depends="deploy.scp"/>

  <!-- the default implementation for getsrc is getsrc.cvs,
       which is what we want -->
```

```
  <!-- assumes FORREST_HOME has been set as an environment variable -->
  <property environment="env"/>
  <import file="${env.FORREST_HOME}/tools/forrestbot/core/forrestbot.xml"/>
</project>
```

3.1. Workstages

It is only necessary to include specific target implementations in the buildfile if we want to override the default implementations. The following table shows the various workstages, which implementations may be used for each, and which one is the default.

Workstage	Implementations
getsrc	• getsrc.local • getsrc.cvs (default) • getsrc.svn
build	• build.forrest
deploy	• deploy.local (default) • deploy.scp • deploy.cvs • deploy.svn • deploy.ftp
notify	• notify.local (default) • notify.email

If you want to do more advanced processing for your project, you can override the 'main' target, which by default is `<target name="main" depends="getsrc, build, deploy, notify"/>`, create your own implementation of a workstage, or use any other ant tasks to do additional work. In order to create your own workstage implementation, define the workstage target in question in your `mybuildfile.xml` anywhere before the `<import>` task. This will override the default implementation provided by Forrestbot.

Also, you can choose a different target as the default by changing the `default` attribute of `<project>`. For example, you will much more frequently do a 'build' without a 'deploy' during the development of your website, and only at the end do an actual 'deploy', so you might want to choose 'build' as your default target.

3.1.1. Deploying only modified files

Use the checkums feature of the Cocoon CLI. This enables Forrest to keep track of which generated files have actually been changed. The Ant tasks used by Forrestbot will then deploy only the modified files. Ant keeps a `cache.properties` file. If you need to deploy all files then remove this file and let it be re-generated.

Note that the deplo.svn and deploy.cvs workstages handle modified files with their own mechanism.

3.1.2. Correct Use of getsrc.local

There is a wrinkle when using the 'getsrc.local' implementation of the 'getsrc' workstage. If you define your own 'getsrc.local' target, make sure it starts with the `<property>` task given here:

```
<target name="getsrc.local">
  <property name="build.home-dir" location="${getsrc.local.root-dir}"/>
  [...]
```

```
</target>
```

Alternatively (and preferably), define your 'getsrc' target like this:

```
<target name="getsrc" depends="getsrc.clean-workdir, getsrc.get, getsrc.local"/>
```

and then implement the actual fetching of the sources in the 'getsrc.get' target. This latter approach is safer since it is more likely to be forward-compatible with future versions of Forrestbot.

3.2. Workstage Properties

Each workstage implementation is configurable with properties. The following tables describe each property and whether or not you are required to set it in your buildfile.

Many workstage properties use usernames and passwords. You may want to keep them out of your project's Ant buildfile (especially if you store that file in CVS or SVN). A nice way to do this is to create a separate properties file (e.g., `auth.properties`) that just sets those properties (don't include it in CVS/SVN). Then, at the top of your project buildfile, have `<property file="auth.properties"/>`.

3.2.1. Misc. Properties

Property	Description	Default Value	Required?
ant.project.name (you specify this by <project name="____"> in your buildfile)	This must be unique for each project.		Yes

3.2.2. getsrc.clean-workdir

This should be executed before a getsrc implementation is executed, e.g., `<target name="getsrc" depends="getsrc.clean-workdir, getsrc.svn"/>`.

3.2.3. getsrc.local

Property	Description	Default Value	Required?
getsrc.local.root-dir	Absolute path to the project's root directory on the local computer. Use **location=** instead of **value=** for this <property>		Yes

3.2.4. getsrc.cvs

Property	Description	Default Value	Required?
getsrc.cvs.user	CVS username		Yes
getsrc.cvs.password	CVS password		Yes
getsrc.cvs.root	CVS root directory	/home/cvsroot	Yes
getsrc.cvs.host	CVS host	cvs.apache.org	Yes

getsrc.cvs.module	CVS module name (an alias for or the full path to the directory that contains forrest.properties)	${ant.project.name}	Yes
getsrc.cvs.tag	CVS tag or branch name		No

3.2.5. getsrc.svn

Property	Description	Default Value	Required?
getsrc.svn.url	Full repository URL for project (this directory must contain forrest.properties)		Yes
getsrc.svn.revision	Revision number to fetch	HEAD	No

3.2.6. build.forrest

Property	Description	Default Value	Required?
build.work-dir	Directory to temporarily hold working files	work/ ${ant.project.name}	No
build.log-dir	Directory to hold log files	logs	No

3.2.7. deploy.local

Property	Description	Default Value	Required?
deploy.local.dir	Path to deploy site to - the dir that would be the equivalant of build/site dir. Relative paths are relative to ${basedir}, which defaults to the dir containing the Forrestbot project buildfile (mybuildfile.xml).	sites/${ant.project.name}	No

3.2.8. deploy.scp

The `${user.home}/.ssh/known_hosts` must properly recognize the host, so you should manually make an ssh connection to the host if you never have before.

If `${deploy.scp.keyfile}`is defined, then it will use key-based authentication in preference. Otherwise it will use `${deploy.scp.password}`

Property	Description	Default Value	Required?

deploy.scp.dest	Full destination reference in the format user@host:/directory/path		Yes
deploy.scp.keyfile	Location of the local file holding the private key. Usually /home/me/.ssh/id_dsa or /home/me/.ssh/id_rsa Note that the deploy.scp.passphrase must also be supplied.		No. However, if this is not supplied then scp will fallback to use the less secure deploy.scp.password
deploy.scp.passphrase	Local passphrase for your private key.		No. You will be prompted if it is not set.
deploy.scp.password	Password for user@host		No. You will be prompted if it is not set. Not needed if using the preferred deploy.scp.keyfile/deploy.scp.passphrase

3.2.9. deploy.cvs

This is only available on *nix operating systems.

Property	Description	Default Value	Required?
deploy.cvs.user	CVS username to use when committing changes		Yes
deploy.cvs.password	CVS password		Yes
deploy.cvs.root	CVS root	/home/cvs	Yes
deploy.cvs.host	CVS host	cvs.apache.org	Yes
deploy.cvs.module	CVS module	${ant.project.name}	Yes
deploy.cvs.commit-message	Message to use when committing. You probably want to put a machine name or person's name here.	Automatic publish from forrestbot	No

3.2.10. deploy.svn

Property	Description	Default Value	Required?
deploy.svn.user	SVN username to use when committing changes		Yes
deploy.svn.password	SVN password		Yes
deploy.svn.url	Full repository URL		Yes
deploy.svn.commit-message	Message to use when committing. You	Automatic publish from forrestbot	No

	probably want to put a machine name or person's name here.		

3.2.11. deploy.ftp

Property	Description	Default Value	Required?
deploy.ftp.server	FTP server to upload files to	localhost	No
deploy.ftp.user	FTP username to use for authenticating with the server	anonymous	No
deploy.ftp.password	Password for the FTP user	forrestbot@	No
deploy.ftp.remotedir	The directory to upload to (this can be an absolute path or relative to the FTP user's default directory)	incoming	No

3.2.12. notify

These settings are used by all notify implementations.

Property	Description	Default Value	Required?
notify.administrator	Name and email address of the forrestbot administrator		Yes
notify.on.failure	On a build failure, notification will happen if this is true.	true	No
notify.on.success	On a succesful build, notification will happen if this is true.	true	No
notify.log	Log file		No. Set by other workstage(s).
notify.deploy-location	Deployed location		No. Set by other workstage(s).
notify.completion-status	Result of the build		No. Set by other workstage(s).

3.2.13. notify.local

No properties.

3.2.14. notify.email

Property	Description	Default Value	Required?

notify.email.host	SMTP host through which the email will be sent.	localhost	Yes
notify.email.to	Email address to send notification to.	${user.name}@localhost	Yes
notify.email.from	From: address in the email	Forrestbot	No, but some mailers may require a valid email address.

4. Forrestbot Design

Forrest and Forrestbot use Ant buildfiles extensively. Ant 1.6's `<import>` task is used to import multiple buildfiles into a single build. The following is the flow of control when running Forrestbot:

- Your project buildfile (`mybuildfile.xml`)
  - `$FORREST_HOME/tools/forrestbot/core/forrestbot.xml`
    - Workstage buildfiles (`$FORREST_HOME/tools/forrestbot/core/{getsrc,build,deploy,notify}.xml`)
    - `$FORREST_HOME/main/forrest.build.xml`

The workstage buildfiles define the default workstage implementations and set up the properties and files so that targets in the main Forrest buildfile (`forrest.build.xml`) will run. After those targets are executed, the targets in the workstage buildfiles can perform reporting, deployment, or other post-build activities.

Your project buildfile specifies the workstages you want to use, sets properties for them, and does any additional pre- and post-processing. In addition, you can override the default workstage implementations by defining the relevant targets in your project buildfile before the `<import>` task (see the example above).