Using Forrest

A tutorial on how to use Forrest in your own projects

$Revision: 1.30 $

by Jeff Turner

1. Introduction

This tutorial will lead you through the process of installing Forrest, and using it to create a new project, or add Forrest-based docs to an existing project.

2. Getting the Forrest source

Currently, the primary way to get Forrest is through CVS. Please consult The Forrest Primer for a description of what this involves.

Here is how to get Forrest with a shell-based CVS client:

```
~/apache$ export CVSROOT=:pserver:anoncvs@cvs.apache.org:/home/cvspublic
~/apache$ cvs login
Logging in to :pserver:anoncvs@cvs.apache.org:2401/home/cvspublic
CVS password:
~/apache$ cvs checkout xml-forrest
U xml-forrest/.cvsignore
U xml-forrest/appendcp.bat
U xml-forrest/build-old.xml
U xml-forrest/build.bat
....
```

3. Building and Installing Forrest

To build Forrest, type `build` on Windows, or `./build.sh` on Unix. If everything is successful, you should see a message like this:

```
 *-----------------------------------------------------------------
 | installation notice
 *-----------------------------------------------------------------
 | You have succesfully built the shell-bat distribution of Forrest.
 | Please find it at: ./build/dist/shbat
 | Please copy the contents to the install directory of your choice
```

```
  | Please have the environment variable FORREST_HOME point to it.
  | It is recommended to add
  |    unix: $FORREST_HOME/bin: to your $PATH
  |    win: %FORREST_HOME%\bin; to your %PATH%
  | Calling
  |    unix: $FORREST_HOME/bin/forrest -projecthelp
  |    win: %FORREST_HOME%\bin\forrest -projecthelp
  | will list options for the 'forrest' command
  | More help at http://xml.apache.org/forrest/ and forrest-dev@xml.apache.org
  *-----------------------------------------------------------------
```

You now have the binary distribution built in `build/dist/shbat`. Copy it somewhere more sensible if you like (e.g. `/usr/local/forrest` on Unix), though if you intend to update your Forrest from CVS, leaving it where it is would be best.

As the message says, you need to add the distribution's `bin/` ("binary") directory to your PATH variable, so the `'forrest'` command is available everywhere:

```
~/apache/xml-forrest$ export FORREST_HOME=`pwd`/build/dist/shbat
~/apache/xml-forrest$ export PATH=$PATH:$FORREST_HOME/bin
```

To see what the forrest command can do, type 'forrest -projecthelp'

```
Apache Forrest.  Run 'forrest -projecthelp' to list options

ANT_OPTS is
Buildfile: /home/jeff/apache/xml/xml-forrest/build/dist/shbat/bin/../forrest.build.xml

    *=======================================================*
    |                  Forrest Site Builder                 |
    |                         0.5-dev                       |
    |             $Date: 2003/09/07 04:58:15 $              |
    *=======================================================*

              Call this through the 'forrest' command

Main targets:

 backcopy   If anything has been edited in build/webapps, copies them back to src/docum
 overrides  Prints a summary of which files a project is overriding
 run        Run Jetty with configuration set by the jetty.run property
 seed       Seeds a directory with a template project doc structure
 site       Generates a static HTML website for this project
 validate   Validates XML doc files in the project
 war        Generates a dynamic servlet-based website (an packaged .war file)
 webapp     Generates a dynamic servlet-based website (an unpackaged webapp)

Default target: site
```

As 'site' is the default target, just running 'forrest' without options will generate a "static

HTML website". For example, typing 'forrest' in the xml-forrest directory would build Forrest's own website. But we're going to be building a new site, so read on.

4. Seeding a new project

'Seeding' a project is our own arborial term for adding a template documentation set to your project, which you can then customize.

To try this out, create a completely new directory, and type 'forrest seed':

```
~/apache/xml-mysite$ forrest seed
Apache Forrest.  Run 'forrest -projecthelp' to list options

Buildfile: /home/..../xml-forrest/build/dist/shbat/bin/../forrest.build.xml

init-props:
Loading project specific properties from
/home/jeff/apache/xml-mysite/forrest.properties

echo-settings:

check-contentdir:

ensure-nocontent:

seed:
Expanding: /home/jeff/apache/xml/xml-forrest/build/dist/shbat/fresh-site.zip
           into /home/jeff/apache/xml-mysite

-------------------------------
~~ Template project created! ~~

Here is an outline of the generated files:

/                         # /home/jeff/apache/xml-mysite
/status.xml               # List of project developers, todo list and change log
/forrest.properties       # Optional file describing your site layout
/src/documentation/       # Doc-specific files
/src/documentation/skinconf.xml    # Info about your project used by the skin
/src/documentation/content/        # Site content.
/src/documentation/content/xdocs   # XML content.
/src/documentation/content/xdocs/index.xml # Home page
/src/documentation/content/xdocs/site.xml  # Navigation file for site structure
/src/documentation/content/xdocs/tabs.xml  # Skin-specific 'tabs' file.
/src/documentation/content/*.html,pdf      # Static content files
/src/documentation/resources/images        # Project images (logos, etc)


What to do now?

- Try rendering this template to HTML by typing 'forrest'. View the generated
```

```
  HTML in a browser to make sure everything works.
- Edit status.xml and src/documentation/skinconf.xml and customize for your
  project.
- Start adding content in xdocs/, remembering to add new files to site.xml
- Provide any feedback to forrest-dev@xml.apache.org

Thanks for using Apache Forrest
------------------------------


BUILD SUCCESSFUL
Total time: 8 seconds
```

Note:

As you've probably begun to notice, we like to document things right in the script, on the theory that people only read online docs when desperate :)

You now have a template documentation structure all set up:

```
jeff@expresso:~/apache/xml-mysite$ tree
.
|-- forrest-targets.ent
|-- forrest.properties
|-- src
|   `-- documentation
|       |-- README.txt
|       |-- content
|       |   |-- hello.pdf
|       |   |-- test1.html
|       |   |-- test2.html
|       |   `-- xdocs
|       |       |-- index.xml
|       |       |-- samples
|       |       |   |-- ehtml-sample.ehtml
|       |       |   |-- faq.xml
|       |       |   |-- ihtml-sample.ihtml
|       |       |   |-- index.xml
|       |       |   |-- sample.xml
|       |       |   |-- sample2.xml
|       |       |   |-- sdocbook.xml
|       |       |   |-- subdir
|       |       |   |   |-- book-sample.xml
|       |       |   |   `-- index.xml
|       |       |   `-- wiki-sample.cwiki
|       |       |-- site.xml
|       |       `-- tabs.xml
|       |-- resources
|       |   `-- images
|       |       |-- group-logo.gif
|       |       |-- group.svg
|       |       |-- icon.png
```

```
|       |       |-- project-logo.gif
|       |       `-- project.svg
|       `-- skinconf.xml
`-- status.xml
```

To render this to HTML, type 'forrest'. You should have a HTML site rendered into the `build/site` directory:

New project

5. Seeding an existing project

In the section above, we have run 'forrest seed' in an empty directory. If you have an existing codebase which you wish to add Forrest docs to, run 'forrest seed' in your project base directory, and the Forrest doc structure will be grafted onto your project.

If your project already has XML documentation, it may be easier to tell Forrest where the XML lives, rather than rearrange your project directories to accommodate Forrest. This can be done by editing `forrest.properties` (consult the Changing the layout section for more details).

6. Customizing your project

Having seeded a project with template docs, you will now want to customize it to your project's needs. Here we will deal with configuring the skin, and changing the project layout.

6.1. Configuring the Forrest skin: skinconf.xml

Most Forrest skins can be customized through a single XML file, `src/documentation/skinconf.xml`, which (minus its DTD) looks like this:

```
<skinconfig>
  <!-- Do we want to disable the Google search box? -->
  <disable-search>false</disable-search>
  <!-- Do we want to disable the print link? If enabled, valid HTML 4.0.1 -->
  <disable-print-link>true</disable-print-link>
  <!-- Do we want to disable the PDF link? -->
  <disable-pdf-link>false</disable-pdf-link>
  <!-- Do we want to disable the xml source link? -->
  <disable-xml-link>true</disable-xml-link>
  <!-- Do we want to disable w3c compliance links? -->
  <disable-compliance-links>false</disable-compliance-links>

  <searchsite-domain>mydomain</searchsite-domain>
  <searchsite-name>MyProject</searchsite-name>

  <!-- mandatory project logo
```

```
      skin: forrest-site renders it at the top -->
<project-name>MyProject</project-name>
<project-description>MyProject Description</project-description>
<project-url>http://myproj.mygroup.org/</project-url>
<project-logo>images/project.png</project-logo>
<!-- Alternative static image:
<project-logo>images/project-logo.gif</project-logo> -->

<!-- optional group logo
     skin: forrest-site renders it at the top-left corner -->
<group-name>MyGroup</group-name>
<group-description>MyGroup Description</group-description>
<group-url>http://mygroup.org</group-url>
<group-logo>images/group.png</group-logo>
<!-- Alternative static image:
<group-logo>images/group-logo.gif</group-logo> -->

<!-- optional host logo (e.g. sourceforge logo)
     skin: forrest-site renders it at the bottom-left corner -->
<host-url></host-url>
<host-logo></host-logo>

<!-- The following are used to construct a copyright statement -->
<year>2003</year>
<vendor>The Acme Software Foundation.</vendor>

<!-- Some skins use this to form a 'breadcrumb trail' of links. If you don't
want these, set the attributes to blank. The DTD purposefully requires them.
-->
<trail>
  <link1 name="myGroup" href="http://www.apache.org/"/>
  <link2 name="myProject" href="http://xml.apache.org/"/>
  <link3 name="" href=""/>
</trail>

<!-- Configure how many "section" levels need to be included in the
generated Table of Contents (TOC). By default, if no toc element is provided
below, then 2 levels are included. Level 0 does not generate any TOC at all.
-->
<toc level="2"/>

<!-- Credits are typically rendered as a set of small clickable images in the
page footer -->
<credits>
  <credit>
    <name>Built with Apache Forrest</name>
    <url>http://xml.apache.org/forrest/</url>
    <image>images/built-with-forrest-button.png</image>
    <width>88</width>
    <height>31</height>
  </credit>
  <!-- A credit with @role='pdf' will have its name and url displayed in the
  PDF page's footer. -->
</credits>
```

```
</skinconfig>
```

Customise this file for your project. The `images/` directory mentioned in 'project-logo' and 'group-logo' elements correspond to the `src/documentation/resources/images` directory (this mapping is done in the sitemap).

Having edited this file (and ensured it is valid XML!), re-run the 'forrest' command in the site root, and the site should be updated.

6.2. Changing the layout: forrest.properties

For a simple site, Forrest's default directory layout may seem rather cumbersome. Forrest allows you to place files anywhere you want in your project, so long as you tell Forrest where you have placed the major file types.

The `forrest.properties` file is what maps from your directory layout to Forrest's. If you generated your site with 'forrest seed', you should have one pre-written, with all the entries commented out. The relevant `forrest.properties` entries (with default values) are:

```
# Properties that must be set to override the default locations
#
# Parent properties must be set. This usually means uncommenting
# project.content-dir if any other property using it is uncommented

#project.status=status.xml
#project.content-dir=src/documentation
#project.conf-dir=${project.content-dir}/conf
#project.sitemap-dir=${project.content-dir}
#project.xdocs-dir=${project.content-dir}/content/xdocs
#project.resources-dir=${project.content-dir}/resources
#project.stylesheets-dir=${project.resources-dir}/stylesheets
#project.images-dir=${project.resources-dir}/images
#project.schema-dir=${project.resources-dir}/schema
#project.skins-dir=${project.content-dir}/skins
#project.skinconf=${project.content-dir}/skinconf.xml
#project.lib-dir=${project.content-dir}/lib
#project.classes-dir=${project.content-dir}/classes
```

For example, if you wish to keep XML documentation in src/xdocs rather than `src/documentation/content/xdocs` simply change the 'project.xdocs-dir' definition:

```
        project.xdocs-dir=src/xdocs
```

Say we wish to emulate the nice and simple Maven format:

/xdocs /xdocs/images /xdocs/stylesheets

Here are the required property definitions:

```
project.content-dir=xdocs
project.sitemap-dir=${project.content-dir}
project.xdocs-dir=${project.content-dir}
project.stylesheets-dir=${project.content-dir}/stylesheets
project.images-dir=${project.content-dir}/images
project.skinconf=${project.content-dir}/skinconf.xml
```

Note:

Internally, Forrest rearranges the specified directory into the default `src/documentation/content/xdocs` structure. In the layout above, we have overlapping directories, so you will end up with duplicate files. This small glitch doesn't usually cause problems; just always remember that all links are resolved through the sitemap.

7. Adding content

Now you can start adding content of your own, in `src/documentation/content/xdocs`

7.1. site.xml

Whenever adding a new file, you should add an entry to the project's `site.xml` file. site.xml is like a site index, and is rendered as the vertical column of links in the default skin. Have a look at Forrest's own xdocs for an example. More detailed info about site.xml is provided in Menus and Linking.

7.2. tabs.xml

The `tabs.xml` file is used to produce the 'tabs' in the top-left corner of the default skin.

Tabs

Tabs allow users to quickly jump to sections of your site. See the menu generation documentation for more details, and again, consult Forrest's own docs for a usage example.

7.3. Images

Images usually go in `src/documentation/resources/images/` The default sitemap maps this directory to `images/`, so image tags will typically look like <figure src="images/project-logo.png" alt="Project Logo"/>

8. Advanced customizations: sitemap.xmap

The Cocoon sitemap is a set of rules for generating content (HTML, PDFs etc) from XML sources. Forrest has a default sitemap, which is adequate for everyday sites (like the Forrest site itself).

Sometimes, one needs to go beyond the default set of rules. This is where Forrest really shines, because its Cocoon backend allows virtually any processing pipeline to be defined. For example, one can:

- Transform custom XML content types with XSLT stylesheets
- Generate PNG or JPEG images from SVG XML files. (**Update:** Forrest's sitemap now does this natively).
- Integrate external XML feeds (eg RSS) into your site's content (**Update:** See issues.xmap for an example.
- Merge XML sources via aggregation, or make content from large XML sources available as "virtual" files. (**Update:**: Forrest's default sitemap defines a whole-site HTML and PDF, available as `site.html` and `site.pdf`.
- Read content from exotic sources like XML databases
- Integrate any of Cocoon's vast array of capabilities. The possibilities are best appreciated by downloading the latest Cocoon distribution and playing with the samples.

If your site defines its own sitemap, it must perform all the operations of the Forrest default. To get started, simply copy the Forrest sitemaps from `xml-forrest/src/resources/conf/*.xmap` into your `src/documentation` directory (or wherever `${project.sitemap-dir}` points to).

The sitemap syntax is described in the Cocoon sitemap docs. The Forrest sitemap is broken into multiple files. The main one is **sitemap.xmap**, which delegates to others. See the Sitemap Reference for a tour of the default sitemap.

8.1. Example: Adding a new content type

Say that `download.xml` lists downloads for a certain package. It would be best to represent download information in a custom XML format:

```
<!DOCTYPE document
          PUBLIC "-//Acme//DTD Download Documentation V1.0//EN" "downloads.dtd">
<document>
  <header>
    <title>Downloading Binaries</title>
  </header>
  <body>
    <section>
      <title>Downloading binaries</title>
```

```
      <p>
        Here are the binaries for FooProject
      </p>
      <release version="0.9.13" date="2002-10-11">
        <downloads>
          <file
            url="http://prdownloads.sf.net/aft/fooproj-0.9.13-bin.zip?download"
            name="fooproj-0.9.13-bin.zip"
            size="5738322"/>
          <file
            url="http://prdownloads.sf.net/aft/fooproj-0.9.13-src.zip?download"
            name="fooproj-0.9.13-src.zip"
            size="10239777"/>
        </downloads>
      </release>
      <release version="0.9.12" date="2002-10-08">
        <downloads>
          <file
            url="http://prdownloads.sf.net/aft/fooproj-0.9.12-src.zip?download"
            name="fooproj-0.9.12-src.zip"
            size="10022737"/>
         </downloads>
       </release>
    </section>
    <section>
      <title>Getting FooProject from CVS</title>
      <p>....</p>
    </section>
  </body>
</document>
```

This should be placed in your content directory, typically `src/documentation/content/xdocs`, and an entry added to site.xml.

To handle these special tags, one would write a stylesheet to convert them to regular documentv12 format. Here is such a stylesheet, download2document.xsl:

```
<?xml version="1.0" encoding="utf-8"?>

<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="release">
    <section>
      <title>Version <xsl:value-of select="@version"/> (released
          <xsl:value-of select="@date"/>)</title>
      <table>
        <tr><th>File</th><th>Size</th></tr>
        <xsl:apply-templates select="downloads/*"/>
      </table>
    </section>
  </xsl:template>
```

```
  <xsl:template match="file">
    <tr>
      <td><link href="{@url}"><xsl:value-of select="@name"/></link></td>
      <td><xsl:value-of
           select="format-number(@size div (1024*1024), '##.##')"/> MB</td>
    </tr>
  </xsl:template>


  <xsl:template match="@* | node() | comment()">
    <xsl:copy>
      <xsl:apply-templates select="@*"/>
      <xsl:apply-templates/>
    </xsl:copy>
  </xsl:template>

</xsl:stylesheet>
```

Place this in the default stylesheets directory, `src/documentation/resources/stylesheets` (or wherever ${project.stylesheets-dir} points).

Now the sitemap has to be modified to transform our custom format into doc-v12. The Sitemap Reference provides details on how the sitemap works, and how it can be customized for specific projects. Specifically, the part to read is the forrest.xmap section. We have to register our new DTD and associate a transformation with it.

1. Override `forrest.xmap` in your project by copying $FORREST_HOME/context/forrest.xmap to your project's src/documentation/ directory.
2. Edit `forrest.xmap`, locate the `sourcetype` action, and register the new document type:

   ```
   <sourcetype name="download">
     <document-declaration public-id="-//Acme//DTD Download Documentation V1.0//EN" />
   </sourcetype>
   ```

3. Locate where the sourcetype action is used, and add a `when` clause to handle the conversion for our document type:

   ```
   <map:when test="download">
     <map:transform
       src="resources/stylesheets/download2document.xsl" />
   </map:when>
   ```

8.1.1. Registering a new DTD

By default, Forrest requires that all XML files be valid: i.e. they must have a DOCTYPE

declaration and associated DTD, and validate against it. Our new 'downloads' document type is no exception. The XML Validation section continues this example, showing how to register a new document type. Briefly, this involves:

- Creating a new DTD or (in our case) extending an existing one
- Putting the new DTD in `${project.schema-dir}/dtd`
- Adding a mapping from the DOCTYPE public id to the DTD location, in the catalog file, `${project.schema-dir}/catalog.xcat`. Eg: `PUBLIC "-//Acme//DTD Download Documentation V1.0//EN" "dtd/download-v11.dtd"`

Please read XML Validation for the full story.

8.2. Example: integrating external RSS content

Similar to the previous example, we can integrate RSS into our site by overriding and editing the sitemap. As described in the 'source pipelines' section of sitemap reference, Forrest's `sitemap.xmap` delegates source handling to various subsitemaps in a `**.xml` block. We can add another *.xml matcher in this section, just before the catch-all subsitemap:

```
<map:match pattern="site.xml">
    <map:mount uri-prefix="" src="aggregate.xmap" check-reload="yes" />
  </map:match>

<map:match pattern="weblog.xml">
  <map:generate src="http://blogs.cocoondev.org/stevenn/index.rss"/>
  <map:transform src="resources/stylesheets/rssissues2document.xsl"/>
  <map:call resource="skinit">
    <map:parameter name="type" value="document2html"/>
    <map:parameter name="path" value="weblog.xml"/>
  </map:call>
</map:match>

  <!-- Default source types -->
  <map:mount uri-prefix="" src="forrest.xmap" check-reload="yes" />

</map:match>
```

(You will want to rename and customize `rssissues2document.xsl` to your needs)

9. Forrest skins

As Forrest separates content from presentation, we can plug in different "skins" to instantly change a site's look & feel. Forrest provides one primary skin, `forrest-site`, and a handful of others in various states of maintenance.

To change the skin, edit the `forrest.properties` file, and change the following entry:

```
project.skin=forrest-site
```

9.1. Defining a new skin

Projects can define their own skin, in the `src/documentation/skins` directory (or wherever `${project.skins-dir}` points). The default sitemap assumes a certain skin layout, so the easiest way to create a new skin is by copying an existing Forrest skin. For example, copy `xml-forrest/src/resources/skins/template` to `src/documentation/skins/myskin`, and add `project.skin=myskin` to forrest.properties.

In addition, when using a project-specific skin it is a good idea to also use a project-specific sitemap. This is to protect your skin from changes in the Forrest default sitemap. While the sitemap-skin contract (expressed as XSLT parameters) is now fairly stable, this should not be relied on.

The two most interesting XSLT stylesheets involved are:

xslt/html/document2html.xsl
This stylesheet is applied to individual documentv11 XML files, and converts them to HTML suitable for embedding in a larger HTML page.

xslt/html/site2xhtml.xsl
This stylesheet generates the final HTML file from an intermediate 'site' format produced by the other stylesheets. It defines the general layout, and adds the header and footer.

Typically there is a lot of commonality between skins. XSLT provides an 'import' mechanism whereby one XSLT can extend another. Forrest XSLTs typically 'import' from a common base:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:import href="../../../common/xslt/html/document2html.xsl"/>

  ... overrides of default templates ...
</xsl:stylesheet>
```

This is particularly relevant for menu rendering (book2menu.xsl), where the common stylesheet does the 'logic' of which item is selected, and overriding stylesheets define the presentation.

10. Interactive Forrest: developing docs faster

In comparison to simpler tools like Anakia, Cocoon's command-line mode (and hence Forrest) is painfully slow. As the dream list notes, Forrest was originally intended to be used

for dynamic sites, and the Cocoon crawler used only to create static snapshots for mirroring. This section describes how, by developing with a "live" Forrest webapp instance, Forrest-based doc development can be faster and easier than comparable tools.

10.1. Running as a webapp

Type `forrest run` in your project root to start Forrest's built-in Jetty web server. Once it has started, point your browser at http://localhost:8888, which should show your website, rendered on demand as each page is clicked.

Alternatively if you wish to run Forrest from within an existing servlet container, type `forrest webapp` to build an open webapp in `build/webapp/`.

10.1.1. Using the webapp

With the setup above, you can edit the XML files in `build/webapp/content/xdocs` and see the changes immediately in the browser.

To get the edited content back to its home directory, either copy it once you have finished editing (with the `forrest backcopy` command), or symlink the `src/documentation/content/xdocs` directory to `build/webapp/content/xdocs`.

Note:

In the future, we are hoping that Forrest will be able to work with *in-place* content, eliminating the step of copying everything into the `build/` directory. There are also suggestions for making webapp-based content generation the primary technique. Future directions like these are debated on the forrest-dev mail list. Please join if you have any suggestions.

11. Invoking Forrest from Ant

Established Java projects which use Ant will typically wish to subsume Forrest's build targets ('site', 'webapp', 'validate' etc) into their own build system, to provide a unified interface for users. This section describes how to invoke Forrest operations from an external Ant build file.

Here are the targets that can be included in a project's build.xml file to invoke Forrest:

```
<target name="site" description="Generates static HTML documentation">
  <ant antfile="${forrest.home}/forrest.antproxy.xml" target="site"/>
</target>

<target name="webapp"
  description="Generates an unpackaged webapp of the website">
  <ant antfile="${forrest.home}/forrest.antproxy.xml" target="webapp"/>
```

```
</target>

<target name="war"
  description="Generates a .war file containing the website">
  <ant antfile="${forrest.home}/forrest.antproxy.xml" target="war"/>
</target>

<target name="validate" description="Validates XML documentation files">
  <ant antfile="${forrest.home}/forrest.antproxy.xml" target="validate"/>
</target>
```

Note:

Always use `forrest.antproxy.xml`, not `forrest.build.xml`. The `forrest.antproxy.xml` script invokes `forrest.build.xml` using Forrest's own bundled version of Ant, which has non-standard support for catalog files.

You'll notice that these targets require `${forrest.home}` to be set. `${forrest.home}` must point to the user's shbat distribution of Forrest. Thus we need a mechanism for the user to inform the build script of their Forrest's location. Typically this is done by setting a property in a properties file like `build.properties`, but can also be done by looking for the `FORREST_HOME` environment variable.

Forrest comes with an Ant snippet file, `forrest-targets.ent`, that supplies the targets listed above, as well as searching for a `${forrest.home}` definition in a number of likely places:

- In the `FORREST_HOME` environment variable.
- In the `build.properties` file.
- In the `project.properties` file.
- In the `ant.properties` file.
- In the `.ant.properties` file.

`forrest-targets.ent` also prints out an informative error message if `${forrest.home}` cannot be resolved.

`forrest-targets.ent` is supplied as part of the template Forrest project (`'forrest seed'`). The comments at the top describe how to use it in a project's build.xml. Typical usage is:

```
<!DOCTYPE project [
<!ENTITY forrest-targets SYSTEM "file:./forrest-targets.ent">
]>

<project .... >

  ....

  <!-- Include Forrest targets -->
```

```
  &forrest-targets;

</project>
```

Having done this, the Forrest targets (`site`, `webapp`, `war`, `validate`) are automatically added to your project.