Using Forrest

A tutorial on how to use Forrest in your own projects

Table of contents

1 Introduction....2
2 Installing Forrest....2
3 Seeding a new project....3
4 Seeding an existing project....6
5 Customizing your project....7
5.1 Configuring the Forrest skin: skinconf.xml....7
5.2 Changing the layout: forrest.properties....10
6 Adding content....11
6.1 site.xml....12
6.2 tabs.xml....12
6.3 Images....12
7 Advanced customizations: sitemap.xmap....13
7.1 Example: Adding a new content type....13
7.2 Example: integrating external RSS content....16
8 Forrest skins....17
8.1 Defining a new skin....17
9 Interactive Forrest: developing docs faster....18
9.1 Running as a webapp....18
10 Invoking Forrest from Ant....19

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. Installing Forrest

Download (http://forrest.apache.org/mirrors.cgi) the latest release of Forrest, or if you want to try the development version, build Forrest (http://forrest.apache.org/build.html) from source.

After downloading and extracting forrest, you need to add environment variables.

In Unix/Linux:

```
~/apache-forrest-0.6$ export FORREST_HOME=`pwd`
~/apache-forrest-0.6$ export PATH=$PATH:$FORREST_HOME/bin
```

In Windows:

```
Go to "My Computer", "Properties", "Advanced", "Environment Variables"
and add:
FORREST_HOME as C:\full\path\to\apache-forrest-0.6
PATH as %PATH%;%FORREST_HOME%\bin
```

To see what the 'forrest' command can do, type 'forrest -projecthelp'. The build targets that are marked with * are the commonly used ones.

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

Buildfile: /usr/local/svn/forrest/src/core/bin/../forrest.build.xml

    *=========================================================*
    |                  Forrest Site Builder                   |
    |                        0.6-dev                          |
    *=========================================================*

              Call this through the 'forrest' command

Main targets:

 available-skins       What skins are available?
 clean                 * Clean all directories and files generated during
                        the build
 install-skin          Install the needed skin from the remote repository
 package-skin          Make a package of an existing skin
```

```
 run                    * Run Jetty (instant live webapp)
 run_custom_jetty       Run Jetty with configuration file found in the
project
 run_default_jetty      Run Jetty with configuration file found in Forrest
 seed                   * Seeds a directory with a template project doc
structure
 site                   * Generates a static HTML website for this project
 validate               Validate all: xdocs, skins, sitemap, etc
 validate-sitemap       Validate the project sitemaps
 validate-skinchoice    Validate skin choice
 validate-skinconf      Validate skinconf
 validate-skins         Validate skins
 validate-stylesheets   Validate XSL files
 validate-xdocs         Validate the project xdocs
 war                    * Generates a dynamic servlet-based website
                          (a packaged .war file)
 webapp                 Generates a dynamic servlet-based website
                          (an unpackaged webapp).
 webapp-local           Generates a dynamic servlet-based website
                          (an unpackaged webapp). Note this webapp is
suitable
                          for local execution only, use the 'war' or
'webapp'
                          target if you wish to deploy remotely.
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 top-level "forrest" directory would build Forrest's own website. But we're going to be building a new site for your project, so read on.

3. 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, change directory to it, and do 'forrest seed':

```
[/home/me/forrest/my-test]$ forrest seed

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

Buildfile: /usr/local/svn/forrest/src/core/bin/../forrest.build.xml

init-props:
Loading project specific properties from
 /home/me/forrest/my-test/forrest.properties
...
echo-settings:

check-contentdir:
```

```
ensure-nocontent:

seed:
Copying 41 files to /home/me/forrest/my-test

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

Here is an outline of the generated files:

/                        # /home/me/forrest/my-test
/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, may have
subdirs
/src/documentation/resources/images   # Project images (logos, etc)
# you can create other directories as needed (see forrest.properties)


What to do now?

- Render this template to static HTML by typing 'forrest'.
  View the generated HTML in a browser to make sure everything works.
- Alternatively 'forrest run' and browse to http://localhost:8888/ live
demo.
- Edit status.xml and src/documentation/skinconf.xml
  to customize for your project.
- Start adding content in xdocs/ remembering to declare new files in
site.xml
- Follow the document http://forrest.apache.org/docs/your-project.html
- Provide any feedback to dev@forrest.apache.org

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

BUILD SUCCESSFUL
Total time: 5 seconds
```

Note:

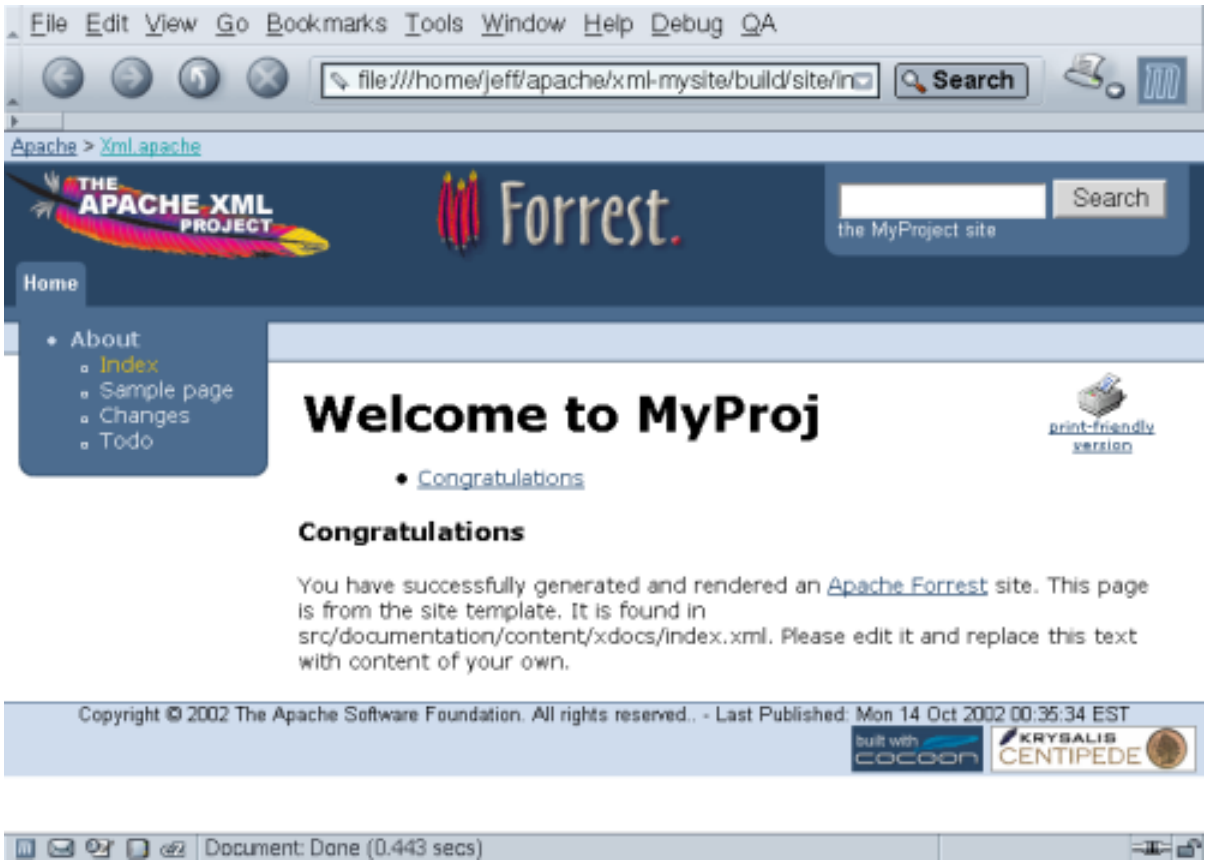
As you have probably noticed, we like to document things right in the script, on the theory that people only read online docs when desperate :)

You now have a template documentation structure all set up:

```
[/home/me/forrest/my-test]$ tree
.
|-- build
|   `-- tmp
|       `-- projfilters.properties
|-- forrest.properties
|-- src
|   `-- documentation
|       |-- README.txt
|       |-- classes
|       |   `-- CatalogManager.properties
|       |-- content
|       |   |-- hello.pdf
|       |   |-- test1.html
|       |   |-- test2.html
|       |   `-- xdocs
|       |       |-- images
|       |       |   |-- group-logo.gif
|       |       |   |-- group.svg
|       |       |   |-- icon.png
|       |       |   |-- project-logo.gif
|       |       |   `-- project.svg
|       |       |-- index.xml
|       |       |-- samples
|       |       |   |-- ascii-art.xml
|       |       |   |-- cocoon-pyramid.aart
|       |       |   |-- faq.xml
|       |       |   |-- ihtml-sample.ihtml
|       |       |   |-- index.xml
|       |       |   |-- openoffice-writer.sxw
|       |       |   |-- sample.xml
|       |       |   |-- sample2.xml
|       |       |   |-- sdocbook.xml
|       |       |   |-- subdir
|       |       |   |   |-- book-sample.xml
|       |       |   |   `-- index.xml
|       |       |   `-- wiki-sample.cwiki
|       |       |-- site.xml
|       |       `-- tabs.xml
|       |-- skinconf.xml
|       `-- translations
|           |-- langcode.xml
|           |-- languages_en.xml
|           |-- languages_es.xml
|           |-- menu.xml
|           |-- menu_af.xml
|           |-- menu_de.xml
|           |-- menu_es.xml
|           |-- menu_it.xml
|           |-- menu_no.xml
|           |-- menu_ru.xml
```

```
|           |-- menu_sk.xml
|           |-- tabs.xml
|           `-- tabs_es.xml
|-- status.xml
```

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



New project

Practise with adding new content. Change to the directory `src/documentation/content/xdocs` and copy the file `index.xml` to create `my-new-file.xml` as a new document. Edit it to change some text. Add an entry to `site.xml` by copying one of the other entries and changing it to suit. Now do 'forrest' to see the result.

4. Seeding an existing project

In the section above, we have run 'forrest seed' in an empty directory to create a new project. If you have an existing codebase to which you want to add Forrest docs, then run 'forrest seed' in your project base directory, and the Forrest doc structure will be grafted onto your project. This procedure only needs to be done once.

If your project already has XML documentation, it may be easier to tell Forrest where the XML sources are, 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).

5. 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.

5.1. Configuring the Forrest skin: skinconf.xml

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

```
<!--
Skin configuration file. This file contains details of your project,
which will be used to configure the chosen Forrest skin.
-->

 <!DOCTYPE skinconfig PUBLIC
        "-//APACHE//DTD Skin Configuration V0.6-2//EN"
        "skinconfig-v06-2.dtd">

<skinconfig>
  <!-- To enable lucene search add provider="lucene"
    Add box-location="alt" to move the search box to an alternate location
    (if the skin supports it) and box-location="all" to show it in all
    available locations on the page.  Remove the <search> element to show
    no search box.
  -->
  <search name="MyProject" domain="mydomain"/>

  <!-- Disable the print link? If enabled, invalid HTML 4.0.1 -->
  <disable-print-link>true</disable-print-link>
  <!-- Disable the PDF link? -->
  <disable-pdf-link>false</disable-pdf-link>
  <!-- Disable the xml source link? -->
  <!-- The xml source link makes it possible to access the xml rendition
    of the source frim the html page, and to have it generated statically.
    This can be used to enable other sites and services to reuse the
    xml format for their uses. Keep this disabled if you don't want other
    sites to easily reuse your pages.-->
```

```
<disable-xml-link>true</disable-xml-link>

<!-- Disable navigation icons on all external links? -->
<disable-external-link-image>false</disable-external-link-image>

<!-- Disable w3c compliance links? -->
<disable-compliance-links>false</disable-compliance-links>
<!-- Render mailto: links unrecognisable by spam harvesters? -->
<obfuscate-mail-links>true</obfuscate-mail-links>

<!-- 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>

<!-- relative url of a favicon file, normally favicon.ico -->
<favicon-url></favicon-url>

<!-- The following are used to construct a copyright statement -->
<year>2004</year>
<vendor>The Acme Software Foundation.</vendor>
<!-- The optional copyright-link URL will used as a link in the
  copyright statement
<copyright-link>http://www.apache.org/licenses/</copyright-link>
-->

<!-- Some skins use this to form a 'breadcrumb trail' of links.
  If you don't want these, then set the attributes to blank.
  The DTD purposefully requires them.
  Use location="alt" to move the trail to an alternate location
  (if the skin supports it).
-->
<trail>
  <link1 name="myGroup" href="http://www.apache.org/"/>
  <link2 name="myProject" href="http://forrest.apache.org/"/>
  <link3 name="" href=""/>
```

```
</trail>

<!-- Configure the TOC, i.e. the Table of Contents.
@max-depth
 how many "section" levels need to be included in the
 generated Table of Contents (TOC).
@min-sections
 Minimum required to create a TOC.
@location ("page","menu","page,menu")
 Where to show the TOC.
-->
<toc max-depth="2" min-sections="1" location="page"/>

<!-- Heading types can be clean|underlined|boxed  -->
<headings type="boxed"/>

<extra-css>
  <!-- A sample to show how the class attribute can be used -->
  p.quote {
    margin-left: 2em;
    padding: .5em;
    background-color: #f0f0f0;
    font-family: monospace;
  }
</extra-css>

<colors>
<!-- CSS coloring examples omitted for brevity -->
</colors>

<!-- Settings specific to PDF output.  -->
<pdf>
  <!--
     Supported page sizes are a0, a1, a2, a3, a4, a5, executive,
     folio, legal, ledger, letter, quarto, tabloid (default letter).
     Supported page orientations are portrait, landscape (default
     portrait).
     Supported text alignments are left, right, justify (default left).
  -->
  <page size="letter" orientation="portrait" text-align="left"/>

  <!--
     Margins can be specified for top, bottom, inner, and outer
     edges. If double-sided="false", the inner edge is always left
     and the outer is always right. If double-sided="true", the
     inner edge will be left on odd pages, right on even pages,
     the outer edge vice versa.
     Specified below are the default settings.
  -->
  <margins double-sided="false">
    <top>1in</top>
    <bottom>1in</bottom>
    <inner>1.25in</inner>
    <outer>1in</outer>
```

```
    </margins>

    <!--
      Print the URL text next to all links going outside the file
    -->
    <show-external-urls>false</show-external-urls>
  </pdf>

  <!-- 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://forrest.apache.org/</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 corresponds to the `src/documentation/resources/images` directory (this mapping is done automatically by 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.

5.2. Changing the layout: forrest.properties

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 maps from your directory layout to Forrest's. If you generated your site with 'forrest seed', you will have one pre-written, with all the entries commented out.

> **Note:**
> You only need to un-comment entries if you are going to change them to something different. If you keep in synchronisation with the 'forrest seed' defaults, then it is easy to diff each time that you update.

The main 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
```

For example, to emulate the simple Maven (http://maven.apache.org/) 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.

6. Adding content

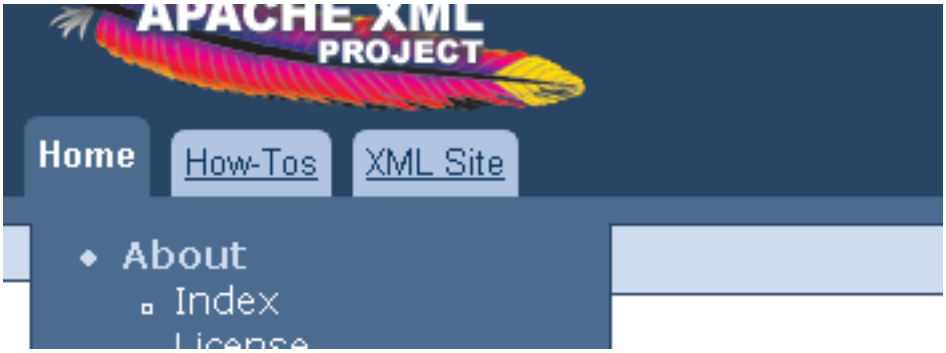
Now you can start adding content of your own, in `src/documentation/content/xdocs`

6.1. site.xml

When adding a new xml document, you would add an entry to the project's `site.xml` file. This site.xml is like a site index, and is rendered as the vertical column of links in the default skin. Look at Forrest's own xdocs for an example. More detailed info about site.xml is provided in the document Menus and Linking (http://forrest.apache.org/docs/linking.html) .

6.2. tabs.xml

The `tabs.xml` file is used to produce the 'tabs'. which enable users to quickly jump to sections of your site. See the menu generation (http://forrest.apache.org/docs/linking.html#menu_generation) documentation for more details, and again, consult Forrest's own docs for a usage example.



Tabs

You can have one or two levels of tabs. The images above show a single level. However, you can create a second level that will only be displayed when its parent tab is selected. For example, the `tabs.xml` snippet below will display either one or two rows of tabs, depending on which of the top level tabs is selected. The first row will have two tabs: one labelled `How-Tos` and the other labelled `Apache XML Projects`. When the `How-Tos` tab is selected there will be no second row of tabs. However, when the `Apache XML Projects` tab is selected, a second row of tabs will be displayed below the first.

```
<tab label="How-Tos" dir="community/howto/"/>
<tab label="Apache XML Projects" href="http://xml.apache.org">
  <tab label="Forrest" href="http://forrest.apache.org/"/>
  <tab label="Xerces" href="http://xml.apache.org/xerces"/>
</tab>
```

6.3. 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"/>

7. 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 (http://forrest.apache.org/) 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 (http://www.w3.org/TR/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 (http://www.rpbourret.com/xml/XMLDBLinks.htm)
- Integrate any of Cocoon's (http://cocoon.apache.org/2.1/) 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. Simply copy the relevant sitemaps that you wish to over-ride, from Forrest sitemaps at `forrest/src/core/context/*.xmap` into your `src/documentation` directory (or wherever `${project.sitemap-dir}` points to).

The sitemap syntax is described in the Cocoon sitemap docs (http://cocoon.apache.org/2.1/userdocs/concepts/sitemap.html) . The Forrest sitemap is broken into multiple files. The main one is **sitemap.xmap**, which delegates to others. See the Sitemap Reference (http://forrest.apache.org/docs/sitemap-ref.html) for a tour of the default sitemap.

7.1. 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:

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

```
<!DOCTYPE document PUBLIC "-//Acme//DTD Download Documentation V1.0//EN"
  "dtd/download-v10.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 file 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 (http://forrest.apache.org/docs/sitemap-ref.html) 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 (http://forrest.apache.org/docs/sitemap-ref.html#forrest_xmap) . We have to register our new DTD and associate a transformation with it.

1. Override `forrest.xmap` in your project by copying $FORREST_HOME/src/core/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>
```

7.1.1. 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 (http://forrest.apache.org/docs/validation.html) 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-v10.dtd"`

Please read XML Validation (http://forrest.apache.org/docs/validation.html) for the full story.

7.2. 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 (http://forrest.apache.org/docs/sitemap-ref.html#source_pipelines) , 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)

8. 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
```

8.1. 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 `forrest/src/core/context/skins/tigris-style` 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>
```

In order to use this feature in your custom skins you must copy the common skin from the forrest distribution into your custom skins directory (see `forrest/src/core/context/skins/common`). This will protect your skin from changes in the Forrest common skin, but you must remember to update this skin in order to take advantage of new features added by the Forrest team.

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.

9. Interactive Forrest: developing docs faster

In comparison to simpler tools like Anakia (http://jakarta.apache.org/velocity/anakia.html) , Cocoon's command-line mode (and hence Forrest) is painfully slow. As the dream list (http://forrest.apache.org/docs/dreams.html) 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.

9.1. 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/`.

9.1.1. 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 (http://forrest.apache.org/mail-lists.html) . Please join if you have any suggestions.

10. Invoking Forrest from Ant

Ant has an <import> (http://ant.apache.org/manual/CoreTasks/import.html) task which can be used to invoke Forrest from Ant. All targets and properties are imported and can be used in your project. Here's a simple example:

```
<project name="myproject" default="hello">
     <!-- FORREST_HOME must be set as an environment variable -->
     <property environment="env"/>
     <import file="${env.FORREST_HOME}/forrest.build.xml"/>

     <!-- 'site' is a target imported from forrest.build.xml -->
     <target name="post-build" depends="site">
       <echo>something here</echo>
     </target>
</project>
```

Note:

If you do not use Ant 1.6+, the <import> task will not be available for you to use. Forrest includes the latest version of Ant so you can invoke your project like this: `forrest -f myproject.xml`. This will not run forrest; it will just use Forrest's Ant to execute your buildfile.

Another option is to use the Forrest Antlet from the Krysalis Project's Antworks Importer (http://antworks.sourceforge.net/importer/) .

The Forrestbot (http://forrest.apache.org/docs/forrestbot.html) provides workstages to get source, build, deploy, and notify. This is very useful for automating builds; you may want to consider using the Forrestbot if your Ant project does those things.