Frequently Asked Questions

Table of contents

Questions

1. Getting Started and Building Forrest

1.1. How to use these FAQs?

There is no particular order to these FAQs. Use your browser's "Find in this page" facility to search for keywords.

1.2. Where can I read an overview about how to work with Forrest?

See the Using Forrest guide.

1.3. Where is all of the documentation?

You have a local copy of the main documentation with your version of Forrest. Do 'cd site-author; forrest run' and visit http://localhost:8888/ in your browser. The most recent documentation is in SVN trunk which creates the forrest.apache.org website.

Each plugin has its own documentation and working examples of its techniques.

The example seed site has other documentation and working examples of various techniques. Do 'cd my-new-directory; forrest seed-sample; forrest run'. Every hour the forrestbot generates a static version of this documentation on our testing zone.

1.4. What are the system requirements for Forrest?

Forrest includes everything necessary to build and run, except of course for Java. In addition to all the Cocoon JARs, Forrest includes and uses its own version of Apache Ant.

Java 1.4 (or newer) is required. If you are only going to use Forrest as-is then you need only the Java Runtime Environment (JRE). If you intend to enhance and rebuild Forrest (or use the Forrest sources with Subversion or use a source snapshot) then you need the full JDK.

1.5. The old xml-forrest CVS code repository seems to be stale. What happened?

Forrest switched from a CVS code repository to SVN (Subversion) code repository. The old CVS repository is closed and not kept current.

1.6. How can I use SVN to keep up to date with the latest codebase?

Follow these Building Forrest notes.

The Using Forrest guide provides further step-by-step assistance in getting started with Forrest for your project.

1.7. How to use older versions of specific plugins?

Sometimes one does not want to use the most recent functionality of a plugin and instead need to use

an older version. Information about changes to each plugin can be found in its documentation.

In the forrest.properties file, specify the version of the plugin that you require, e.g.

```
project.required.plugins=org.apache.forrest.plugin.input.PhotoGallery-0.1,...
```

Users of Forrest-0.7 will need to do this for the projectInfo plugin if you get the following error ...

```
Could not find component for role:
 [org.apache.cocoon.components.modules.input.InputModule/lm]
 (Key='org.apache.cocoon.components.modules.input.InputModule/
```

... then sorry, we mistakenly added new "locationmap" functionality (due in version 0.8). So do this ...

```
project.required.plugins=org.apache.forrest.plugin.input.projectInfo-0.1,...
```

1.8. What is the best way to generate "standalone documents" using Forrest?

There is a trick that can cut down your turnaround time with building. In forrest.properties ...

```
# The URL to start crawling from
#project.start-uri=linkmap.html
```

Uncomment that and set it to the specific page that you want. Forrest will build that single document, then of course it will keep crawling links from there. It might be confined to a sub-directory, but depending on links could end up generating the whole site. The main thing is that your page of interest is built first.

It is probably easiest to make this change temporarily as a command-line parameter, e.g.

```
forrest -Dproject.start-uri=live-sites.html
```

You can terminate forrest with 'kill' or Ctrl-C after it has built your pages of interest.

Cocoon can be instructed via the Cocoon cli.xconf file to not follow links (see its "follow-links" parameter). So this will build only the document that was specified. Be careful, if you also usually build PDF pages, then they will not be built.

Cocoon can also be instructed to not process certain URIs if you need to temporarily exclude then.

Another useful technique is to use 'wget' or Apache Ant's Get task to retrieve individual files, e.g. Do 'forrest run' and then 'wget http://localhost:8888/index.pdf'.

1.9. When running ./build.sh in cygwin, I get an error: cygpath.exe: *** can't create title mutex, Win32 error 6.

This appears to be a bug in cygwin. Please use the .bat script instead.

1.10. How can I specify the amount of memory to be used by Java?

There are two ways to control this. If you get an OutOfMemoryError when Cocoon is generating pages, see the first paragraph. If you get an OutOfMemoryError when outside of Cocoon (e.g., copying raw files), see the second paragraph.

The `maxmemory` property in the `forrest.properties` file controls how much memory Cocoon uses. Like many other properties you can copy them from the default configuration at `main/fresh-site/forrest.properties`

Set the `ANT_OPTS` environment variable before you run forrest. The exact value you set it to is

dependant on your JVM, but something like `ANT_OPTS=-Xmx500M` will probably work.

1.11. How can I start forrest in Java debug mode?

The `forrest.jvmargs` property in the `forrest.properties` file can be used to start forrest in debug mode on a specific port. `forrest.jvmargs=-Xdebug -Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n`

2. Content Creation

2.1. What tools can be used to edit the content?

If you are using the Apache Forrest XML document format or DocBook or other XML document types, then you can use any text editor or even a dedicated XML editor. You must ensure valid XML. See our configuration notes for various editors.

There are content management systems like Apache Lenya.

Remember that Forrest can also use other source formats, such as OpenOffice.org docs or JSPWiki. Use the appropriate editor for those document types and ensure that the document stucture is consistent. Forrest can also use "html" as the source format, in which case you can use text editors or "html editors" such as the one provided with the Mozilla web browser.

2.2. How to use the site.xml configuration file for menus and linking.

The `site.xml` configuration file is used for two different purposes: defining the navigation menus, and as a method for defining references to be used when linking between documents. This file is fully explained in Menus and Linking. Here is a precis:

The labels can be whatever text you want.

```
<faq label="FAQs" href="faq.html">
  <tech label="Technical" href="faq-tech.html">
    <docbook href="#docbook"/>
    <ignoring_javadocs href="#ignoring_javadocs"/>
  </tech>
  <user label="User" href="faq-user.html">
</faq>
```

That will create a menu like this with three links:

```
FAQs
   Technical
   User
```

These documents can be linked to from other documents, like this:

```
<a href="site:faq/tech"> link to the top of the Tech FAQs
<a href="site:faq/tech/docbook"> link to the DocBook FAQ in the Tech FAQs
```

If that "docbook" entry was a unique name in your site.xml then you can shorten that latter link:

```
<a href="site:docbook"> link to the DocBook FAQ in the Tech FAQs
```

2.3. Where are examples of documents and site.xml and tabs.xml files?

There are examples in the 'forrest seed site' and also the Forrest website documents are included with the distribution (`cd forrest/site-author; forrest run`).

2.4. Help, one of my documents is not being rendered.

Did you make a link to it? Forrest does not find documents by scanning the filesystem to find the source documents. Rather it starts at one document and crawls the links to find other documents to process.

There are essentially two ways to create links. Via a site.xml file to define the navigation and menu structure, or via direct relative linking. See the to previous FAQs.

Normally the source material will be local. The Forrest crawler does not follow and process off-site links. The new locationmap (0.8+) enables content to be drawn from remote sources.

2.5. How can I generate one pdf-file out of the whole site or selected pages of the site?

Add the following entries to your site.xml file:

```
<about tab="home" label="About" href="">
      ...
  <all_site label="Full HTML" href="wholesite.html"/>
  <all_sitePDF label="Full PDF" href="wholesite.pdf"/>
   ...
</about>
```

In this case the menu labeled "About" will have 2 new items: "Full PDF" and "Full HTML". (See also How to create a PDF document for each tab.)

This assumes that you use the site.xml method for your site structure and navigation, rather than the old book.xml method.

2.6. How do I insert page breaks into documents?

Page breaks do not make a great deal of sense in HTML documents intended for display on a screen. However, PDF documents are intended for printing and therefore page breaks can be important.

To insert a page break in a PDF document simply add *pageBreakBefore* and/or *pageBreakAfter* to the class attribute of the block you wish to force a page break on. All the common block grouping elements support this class, for example, note, warning, p and so on.

If you want these classes to be processed in your HTML documents as well you should add the following to the `extra-css` element in your projects `skinconf.xml`

```
.pageBreakBefore {
  margin-bottom: 0;
  page-break-before: always;
}
.pageBreakAfter {
  margin-bottom: 0;
  page-break-after: always;
  }
```

2.7. How can I generate html-pages to show a 'clickable' email-address (of the author-element)?

You would override `$FORREST_HOME/main/webapp/skins/common/xslt/html/document-to-html.xsl` and edit the "headers/authors" template.

2.8. How do I link to raw files such as config.txt and brochure.pdf?

Handling of raw files was significantly changed in Forrest 0.7. See Upgrading to Apache Forrest 0.7 for all the details.

2.9. Images don't display in PDFs. How do I fix this?

Forrest uses Apache FOP for rendering PDFs. FOP cannot handle all image types natively, and requires third-party jars to be added. FOP natively handles BMP, GIF, JPG, TIFF and EPS (with a few limitations). FOP can also handle SVG (via Batik!and PNG (see below). For details, see FOP Graphics formats

To get PNGs working in PDFs with Jimi:

1. Download Jimi from http://java.sun.com/products/jimi/
2. Unpack the Jimi distribution and copy JimiProClasses.zip to `$FORREST/lib/optional/jimi-1.0.jar`.

Alternatively you can use JAI (Java Advanced Imaging API at `http://java.sun.com/products/java-media/jai`). For more info, see FOP Graphics Packages

Note:
Due to Sun's licensing, we cannot redistribute Jimi or JAI with Forrest.

2.10. The tab link in my site incorrectly assumes that 'index.html' is present in the linked-to directory. How do I fix this?

In `tabs.xml`, use @href instead of @dir, and omit the trailing '/'. Which file to serve is then a concern of the sitemap. For example, if the "User Manual" tab should link to `manual/Introduction.html` then `tabs.xml` should contain:

```
<tab label="User Manual" href="manual"/>
```

and add this rule to the sitemap:

```
<map:match pattern="manual">
  <map:redirect-to uri="manual/Introduction.html"/>
</map:match>
```

2.11. I need help with the interaction between tabs.xml and site.xml

See the tips.

2.12. How can I change the default file name that Forrest will look for when I request a URL like http://myserver or http://myserver/mydir/ ?

To change the default file name from 'index.html' (default) to 'overview.html' you need to make the following changes:

1. Create a 'cli.xconf' file for your project
2. Edit that file to replace 'index.html' in <default-filename>index.html</default-filename> with 'overview.html'.

3. Edit your project's sitemap.xmap file.
4. Add the following code just before the end of the pipelines-element:

```
<map:pipeline>
  <map:match type="regexp" pattern="^.+/$">
     <map:redirect-to uri="overview.html"/>
  </map:match>
</map:pipeline>
```

2.13. How can I use a start-up-page other than index.html?

Forrest by default assumes that the first page (home page) of your site is named index.html. Which is good because most web servers are configured to look for index.html when you call a url like http://myserver

Like most settings in Forrest however this can be changed, for example when you want your start-up-page for a CD-based documentation project to be named 'start.html'.

To change the start page of a site:

1. Edit your project's sitemap.xmap file.
2. Add the following code just before the end of the pipelines-element:

```
<map:pipeline>
  <map:match pattern="">
    <map:redirect-to uri="start.html" />
  </map:match>
</map:pipeline>
```

3. Name the uri-attribute whatever you'd like your start page to be.
4. Don't forget to create that page and refer to it in your site.xml

2.14. How to use special characters in the labels of the site.xml file?

Use the numeric values for character entities. For example, rather than using `&ouml;` use `&#246;`

See the XHTML Character Entities and see more discussion at Issue FOR-244.

2.15. Does Forrest handle accents for non-English languages?

Yes, Forrest can process text in any language, so you can include:

- accents: á é í ó ú
- diereses: ä ë ï ö ü
- tildes: ã ñ # õ #

This is because sources for Forrest docs are XML documents, which can include any of these, provided the encoding declared by the XML doc matches the actual encoding used in the file. For example if you declare the default encoding:

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

but the file content is actually using ISO-8859-1 then you will receive validation errors, especially if you include some non-ASCII characters.

This situation is commonly encountered when you edit the templates created by `forrest seed` with your favorite (probably localized) editor without paying attention to the encoding, or when you create a new file and simply copy the headers from another file.

Although UTF-8 is an encoding well-suited for most languages, it is not usually the default in popular editors or systems. In UNIX-like systems, most popular editors can handle different encodings to write the file in disk. With some editors the encoding of the file is preserved, while with others the default is used regardless of the original encoding. In most cases the encoding used to write files can be controlled by setting the environment variable `LANG` to an appropriate value, for instance:

```
[localhost]$ export LANG=en_US.UTF-8
```

Of course the *appropriate* way to set the encoding depends on the editor/OS, but ultimately relys on the user preferences. So you can use the encoding you prefer, as long as the `encoding` attribute of the XML declaration matches the actual encoding of the file. This means that if you are not willing to abandon ISO-8859-1 you can always use the following declaration instead:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Another option is to use "character entities" such as `&ouml;` (ö) or the numeric form `&#246;` (ö).

Another related issue is that your webserver needs to send http headers with the matching charset definitions to the html page.

Here are some references which explain further: GT2004 presentation by Torsten Schlabach and Alan Wood's Unicode resources.

2.16. How to use XML entities, for example string replacement?

A set of symbols is available. See the demonstration in a fresh 'forrest seed' site (at samples/xml-entities.html). For example, use "`&myp-t;`" to represent the project name together with trademark symbol "My Project Name™". Avoid lengthy typing and potential spelling errors.

2.17. How to make Forrest clean up the project build directories?

By default Forrest does not clean its build directories in the project workspaces. This enables Cocoon to use its disk cache to speed up successive runs of forrest.

Doing 'forrest clean-site' will remove the contents of the project's generated documents directory. Doing 'forrest clean-work' will remove the project's work directories (usually build/tmp and build/webapp which include the Cocoon cache and the Cocoon logs). Doing 'forrest clean' will remove both sections.

2.18. How can I internationalise (i18n) my content?

The i18n features of Forrest are still under development (as of 0.7) however there are some features available. For example, navigation menus can be i18n'd (see fresh-site for an example). Currently, work is underway to i18n skins

All internationalistation of tokens in, for example, the skins and the menus, is carried out by the Cocoon i18n Transformer. You can see an example of how it works in the above linked issue.

2.19. How can I include HTML content that is not to be skinned by Forrest?

To serve, for example a legacy HTML site, add something like the following to your project's sitemap and place the source content at the `src/documentation/content/xdocs/old_site/` directory.

```
<map:match pattern="old_site/**.html">
```

```
 <map:select type="exists">
  <map:when test="{properties:content}{0}">
    <map:read src="{properties:content}/{0}" mime-type="text/html"/>
    <!--
      Use this instead if you want JTidy to clean up your HTML
      <map:generate type="html" src="{properties:content}/{0}" />
      <map:serialize type="html"/>
    -->
  </map:when>
 </map:select>
</map:match>
```

Exactly what the match should be is dependant on your content structure. It is outside the scope of this FAQ to provide full details, but new users may like to refer to the Cocoon sitemap docs.

There is a more detailed discussion of this topic in the samples of a freshly seeded site. To see this documentation do the following:

1. mkdir seed
2. cd seed
3. forrest seed-sample
4. forrest
5. `http://localhost:8888/samples/linking.html#no-decoration`

2.20. How to include additional Javascript and CSS files?

Place various resources (e.g. javascript, css) into the "project skins" directory. The default forrest.properties has this at src/documentation/skins/$skin-name/ Javascript files would go in a "scripts" subdirectory. CSS files would go in a "css" subdirectory.

Then refer to those from your source documents with URIs like /skin/blah.js and /skin/foo.css

See how this is handled in the core sitemap called forrest/main/webapp/resources.xmap Search for "javascript" then follow to the <map:resource name="skin-read"> section.

2.21. How to show a Table Of Contents for the whole site?

Every site has an automatically generated document at `/linkmap.html` which is produced from the site.xml navigation configuration. It uses the @label and absolutized @href and element name and @description attribute for each node.

For example, the Forrest project's Site Linkmap Table of Contents.

The document is also useful when developing your documentation and linking to other docs. The element names (column #2) e.g. href="site:**mail-lists**" or href="site:**howto/overview**"

This is also the document that 'forrest site' uses to kick-start the Cocoon crawler which then follows links to build each page. See the project.start-uri in the forrest.properties file.

3. Technical

3.1. Where is the Java code?

Because we are based on Apache Cocoon, a lot of the functionality is provided behind-the-scenes, i.e. we use Cocoon's sitemaps and sitemap components such as XSLT transformers. So there is not much need for Java code in Forrest.

For Forrest developers who want to explore or enhance that code, see the Apache Cocoon SVN trunk. From time-to-time we update Forrest's packaged version of Cocoon and so can include your contributions.

That said, you will find some Java code in Forrest at main/java/... for Cocoon components that have been developed at Forrest, e.g. Locationmap and Dispatcher. There is also Java code for some plugins with specialised purpose, e.g. PhotoGallery.

3.2. How to enhance the responsiveness of the cache?

Apache Cocoon has a sophisticated cache. When running Forrest in dynamic mode, the initial visitor will receive slower response. The very first page served will cause Cocoon to cache the pipelines. Later requests will re-use those cached components and add others to the cache. A good technique is to warm up the cache after the forrest webapp has been re-started. Requesting the front page alone will populate the cache with the common items used for other pages. Using a spider such as wget, will warm up everything.

The Cocoon cache and sitemaps can be tuned. See Cocoon Performance Tips and CocoonPerformance and the "Object Stores" section of main/webapp/WEB-INF/forrest-core.xconf

Responsiveness can be further enhanced by utilising a transparent proxy server, e.g. Apache HTTP Server as a frontend. See CocoonAndApacheModProxy.

3.3. I'm behind a proxy and it's preventing Plugins from being downloaded, what should I do?

You can configure the proxy in the `forrest.properties` file. Set the `proxy.host` and `proxy.port` accordingly.

You can also cross an authenticated proxy by setting the `proxy.user` and `proxy.password` accordingly.

Generalise the proxy configuration

You certainly need to cross your proxy for every Forrest projects you have. To avoid to edit every project `forrest.properties` files, you can do once in your `${user.home}/forrest.properties` !

3.4. How can I generate html-pages to show the Revision tag of CVS or SVN?

If you have:`<version>$Revision: 1.30 $</version>`The '1.30' will be extracted and displayed at the bottom of the page as "version 1.30". See for example the bottom of the Using Forrest document.

This technique could also be used for a modification date with $Date: 2004/01/15 08:52:47 $

When using Subversion, remember to set the relevant svn:keywords properties.

3.5. How to control the processing of URIs by Cocoon, e.g. exclude certain URIs, include other additional ones.

Forrest uses a configuration file to control the processing done by the Apache Cocoon command-line called cli.xconf

Your project can supply its own `cli.xconf` and define patterns for URIs to exclude. There are also other powerful configuration features.

This means creating a directory `src/documentation/conf` (or wherever `${forrest.conf-dir}` points) and copying `$FORREST_HOME/main/webapp/WEB-INF/cli.xconf` to it. Declare the location of this file in the forrest.properties configuration, e.g. `project.configfile=${project.home}/src/documentation/conf/cli.xconf`

Then edit cli.xconf, and add any exclude sections that you require. The default cli.xconf ignores directory links and links containing 'apidocs' or starting with 'api/':

```
   ....
   <!-- Includes and excludes can be used to limit which URLs are rendered -->

   <exclude pattern="**/"/>
   <exclude pattern="**apidocs**"/>
   <exclude pattern="api/**"/>

   <uri src="favicon.ico"/>
</cocoon>
```

This is just an example, and you should modify it appropriately for your site.

Note:
Wildcards may be used. These are a powerful feature of Cocoon's sitemap. For example, **foo/*** would match `foo/bar`, but not `foo/bar/baz` — use **foo/**** to match that.

3.6. How do I stop Forrest breaking on links to external files that may not exist, like javadocs?

This can be done by overriding the `cli.xconf` Cocoon config file, and defining patterns for URLs to exclude.

3.7. Some of my files are not being processed because they use common filenames.

Certain patterns are claimed by the default sitemaps for special processing. These reserved words include: `site, changes, todo, faq, images, my-images, skinconf, community, howto`

Sometimes there are workarounds, e.g. faq.html or faq-interview.html would fail, but interview-faq.html would be fine. In future versions of Forrest we will attempt to deal with this issue (FOR-217).

3.8. What do the symbols and numbers mean when Forrest lists each document that it has built?

Each time that Cocoon processes a link, it will report the status messages ...

```
...
* [212/166] [0/0]  1.16s  62.4Kb  docs_0_60/your-project.pdf
X [0]         /docs_0_80/upgrading_08.html  BROKEN: No pipeline matched...
* [213/164] [0/0]  0.391s 29.2Kb  docs_0_70/howto/howto-buildPlugin.pdf
^                            apidocs/index.html
* [214/170] [7/66] 1.476s 45.5Kb  docs_0_60/sitemap-ref.html
...
```

- Column 1 is the page build status (*=okay X=brokenLink ^=pageSkipped).
- Column 2 is the page count (pagesComplete/pagesRemaining). The latter will change because during processing one page, Cocoon will discover more.
- Column 3 is the number of links that were gathered from that page (newLinksInPage/linksInPage).
- Column 4 is the time taken.

- Column 5 is the page size.

3.9. When generating PNG images from SVG, I get an error: Can't connect to X11 window server using ':0.0' as the value of the DISPLAY variable.

If you are using JDK 1.4.0 or newer, you can enable *headless* operation by running Forrest with the `forrest.jvmarg` parameter set to `-Djava.awt.headless=true`, like this:

```
forrest -Dforrest.jvmargs=-Djava.awt.headless=true site
```

See also Cocoon FAQ.

3.10. The project logo that is generated from SVG is truncating my project name.

In a 'forrest seed site' the project and the group logo are generated from a Scalable Vector Graphics (SVG) file, using the text from the `<project-name>` and `<group-name>` elements of the `skinconf.xml` file. If you have a long project-name then you may need to adjust the width of the image. Perhaps you want to change the colours too. Edit the file at `src/documentation/content/xdocs/images/project.svg` and adjust the "width" attribute of the <svg> element. For further details see SVG resources.

3.11. How do i configure my favourite XML editor or parser to find the local Forrest DTDs?

Notes are provided for various tools at Using Catalog Entity Resolver for local DTDs.

3.12. How to configure the Catalog Entity Resolver to use my own local project DTDs?

See Using Forrest for configuration guidance.

3.13. We need an additional system-wide catalog to share DTDs between projects

See Using Forrest for configuration guidance.

3.14. How to debug the Catalog Entity Resolver and local DTDs?

See XML validation.

3.15. How to make the site look better and change its skin?

There are default skins provided, which are configurable and so should meet the needs of most projects. The aim is to provide many capabilities so that extra skins are not needed.

See notes about configuration of the skins. Some projects may have special needs and can define their own skin.

3.16. How do I enable XSP processing?

First consider whether your needs would be better met by Cocoon itself, rather than Forrest.

That said, there are valid reasons for wanting programmatically generated content, so here is how to enable XSP:

1. Download jdtcore-*.jar from Cocoons SVN tree, and copy it to the

$FORREST_HOME/main/webapp/WEB-INF/lib directory (or lib/core/ directory in the source distribution).

2. Add the following generator definition in the map:generators section of your project sitemap

```
  <map:generator name="serverpages"
     pool-grow="2" pool-max="32" pool-min="4"
     src="org.apache.cocoon.generation.ServerPagesGenerator"/>
```

3. Decide how you want to use XSP. For single files, you could just define a *.xml matcher:

```
<map:match pattern="dynamic.xml">
  <map:generate src="content/xdocs/dynamic.xsp" type="serverpages"/>
  ...
  <map:serialize type="xml"/>
</map:match>
```

   You may instead wish to override forrest.xmap to define a general mapping for XSPs.

See also the AddingXSPToForrest Wiki page.

3.17. How do breadcrumbs work? Why don't they work locally?

Breadcrumbs begin with up to three URLs specified in `skinconf.xml`. Here is what the Forrest site uses:

```
<trail>
  <link1 name="apache" href="http://www.apache.org/"/>
  <link2 name="xml.apache" href="http://xml.apache.org/"/>
  <link3 name="" href=""/>
</trail>
```

If any links are blank, they are not used. After these first links, JavaScript looks at the URL for the current page and makes a link for each directory after the domain. If you are viewing the site locally, there is no domain and so there will be no extra breadcrumbs, only the ones that are specified in `skinconf.xml`.

3.18. How do I make forrest run listen on a different port?

```
forrest run -Dforrest.jvmargs="-Djetty.port=80"
```

Or copy Forrest's main/webapp/jettyconf.xml file to your project's src/documentation directory and set the port number in that file. Then do `forrest run`

3.19. Can I run Forrest with Java debugging turned on?

If you use an IDE like Eclipse and want to debug java code in Forrest you need to start Forrest with debugging mode turned on. To do this you need to add `-Xdebug -Xrunjdwp:transport=dt_socket,address=8000,server=y,susp end=n` to the `forrest.jvmargs` property in the `forrest.properties` file. Don't forget to ensure the property is uncommented in that file.

3.20. How do I enable Cocoon's document checksum feature?

Why might you want to do this? There is really no effect on Cocoon processing, but a little time can be saved on filesystem writes, which will accumulate to a big savings for a site with thousands of files.

Some tools depend on the "date-last-modified" timestamp of the generated files. For example, the

Forrestbot will then deploy only the modified files.

There was some discussion about this on the Forrest developer mailing list: Cocoon Checksum Specifically note that this feature only stops Cocoon from writing to disk if the new file is the same as the existing file. Cocoon still spends the same amount of time generating the content as it would if checksums were not enabled.

Locate the `checksums-uri` tag within cli.xconf and replace the contents with an absolute path and filename for the checksums file. Projects can supply their own (see FAQ: Cocoon cli.xconf) or use the default installation-wide cli.xconf file.

4. Older version: 0.6

4.1. Some of my files are not being processed because they use common filenames.

Certain patterns are claimed by the default sitemaps for special processing. These include: `site, changes, todo, faq, images, my-images, skinconf, community, howto`

Sometimes there are workarounds, e.g. faq.html or faq-interview.html would fail, but interview-faq.html would be fine. In future versions of Forrest we will attempt to deal with this issue (FOR-217).

5. General

5.1. What is the relationship between site.xml and book.xml?

One `site.xml` file in your project root can replace all the book.xml files (one per directory) in your site. Internally, Forrest uses `site.xml` to dynamically generate book.xml files. However, Forrest first checks for the existence of a book.xml file, so backwards-compatibility is preserved. If a directory has a book.xml file, the book.xml will be used to generate the menu. This supplement is useful in situations where `site.xml`-generated menus aren't appropriate. See Menus and Linking.

5.2. How do I use DocBook as the XML documentation format?

There are two ways. Forrest has a `simplifiedDocbook` plugin which can transform the DocBook format into the Forrest "xdocs" format on-the-fly and then render that as normal Forrest documents. Be aware that the stylesheet that does this transformation is deliberately very limited and does not attempt to deal with all DocBook elements.

The other way is to use the full DocBook stylesheets directly. The DocBook DTDs are shipped with Forrest and automatically handled. However, you will need to have the DocBook stylesheets on your system (they are too massive to ship with Forrest) and configure Forrest accordingly. You will need to create a project sitemap as explained in Using Forrest and add matches to handle your DocBook documents. Here is an example. Note that you need to change it to suit your situation. The match must be very specific so that only the DocBook documents are matched. The rest of the documents will be handled by Forrest core. Powerful regex capabilities are available.

```
<?xml version="1.0"?>
<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
 <map:pipelines>
  <map:pipeline>
   <map:match pattern="resolver-*.html">
    <map:generate src="{properties:content.xdocs}resolver-{1}.xml"/>
```

```
    <map:transform
      src="file:///usr/share/sgml/docbook/xsl-stylesheets/xhtml/docbook.xsl"/>
    <map:serialize type="xhtml"/>
   </map:match>
  </map:pipeline>
 </map:pipelines>
</map:sitemap>
```

You need to define the xhtml serializer used in <map:serialize type="xhtml"/> in the components section of the sitemap. See the Cocoon docs for the elements you need to add to define this component. You can see examples of other components being added in the `FORREST_HOME/main/webapp/sitemap.xmap` file. Alternatively use the "html" DocBook stylesheets and the default Cocoon serializer, i.e. <map:serialize type="html"/>

The output of the above sitemap will be plain html not adorned with a Forrest theme and navigation. If instead you need the latter, then use the following technique instead. This transforms DocBook xml to html, then uses a Forrest core stylesheet to transform and serialize to the internal xml format, then the normal machinery takes over and does the output transformation. This use the Content Aware Pipelines (SourceTypeAction) to peek at the source xml. If it is DocBook-4.2 then this sitemap match is triggered, if not then it falls through to the core of Forrest.

```
<?xml version="1.0"?>
<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
 <map:components>
  <map:actions>
   <map:action logger="sitemap.action.sourcetype"
      name="sourcetype"src="org.apache.forrest.sourcetype.SourceTypeAction">
    <sourcetype name="docbook-v4.2">
     <document-declaration public-id="-//OASIS//DTD DocBook XML V4.2//EN"/>
    </sourcetype>
   </map:action>
  </map:actions>
  <map:selectors default="parameter">
   <map:selector logger="sitemap.selector.parameter"
      name="parameter" src="org.apache.cocoon.selection.ParameterSelector"/>
  </map:selectors>
 </map:components>
 <map:pipelines>
  <map:pipeline>
   <map:match pattern="**.xml">
    <map:act type="sourcetype" src="{properties:content.xdocs}{1}.xml">
     <map:select type="parameter">
      <map:parameter name="parameter-selector-test" value="{sourcetype}"/>
      <map:when test="docbook-v4.2">
       <map:generate src="{properties:content.xdocs}{../1}.xml"/>
       <map:transform
          src="file:///usr/share/sgml/docbook/xsl-stylesheets/html/docbook.xsl"/>
       <map:transform src="{forrest:forrest.stylesheets}/html-to-document.xsl"/>
       <map:transform type="idgen"/>
       <map:serialize type="xml-document"/>
      </map:when>
     </map:select>
    </map:act>
   </map:match>
  </map:pipeline>
 </map:pipelines>
</map:sitemap>
```

You can also use a mixture of the methods, some handled automatically by Forrest and some directly using DocBook stylesheets. You can also have a mixture of source files as "document-v*" DTD and DocBook.

Ensure that the document type declaration in your XML instance is well specified. Use a public identifier. The DTD will then be properly resolved by Forrest. If you need to use different DTDs, then

see Using Forrest for configuration guidance.

5.3. How to report which version of Forrest is being used and the properties that are set?

Do `'forrest -projecthelp'` or `'./build.sh'` to find the version number.

To list the properties, add "forrest.echo=true" to your forrest.properties file and watch the build messages. Doing `'forrest -v'` will provide verbose build messages.

5.4. Where are the log files to find more infomation about errors?

The logfiles are at `build/webapp/WEB-INF/logs/`

The log level can be raised with the `logkit.xconf` configuration. If you are using Forrest in the interactive webapp mode (which is generally easiest for debugging errors) then see the `main/webapp/WEB-INF/logkit.xconf` file. If you are generating a static site (with command-line 'forrest') then copy `$FORREST_HOME/main/webapp/WEB-INF/logkit.xconf` to your project at `src/documentation/conf/logkit.xconf` and modify it. See more information and efficiency tips with Cocoon logging.

Doing `'forrest -v'` will provide verbose build messages to the standard output.

5.5. How to help?

Join one of the Forrest project mailing lists and tell us what you would like to see improved. We regard all feedback as valuable, particularly from newcomers—often, close proximity blinds software developers to faults that are obvious to everyone else. Don't be shy!

5.6. How to contribute a patch?

Please send all contributions via our issue tracker. Here are notes about making patches.

More info about contributing can be found at the Contributing to Forrest page. It is always a good idea to check the Forrest issue tracker before diving in.

5.7. How can job positions be advertised?

Employers can send notices about employment opportunities. There is a special jobs<AT>apache.org mailing list. You can also send these notices to the project mailing lists, e.g. dev list at Forrest or Cocoon (add [jobs] to the subject line). You can also approach particular developers off-list. However only genuine jobs, not pleas for free support (see mailing lists).

Some enlightened employers enable their employees to contribute material which was created during work time using work-related resources. Please note the need to file a Corporate Contributor License Agreement (CCLA) with The Apache Software Foundation.