Demonstration of linking

Table of contents

1 Overview....................2
2 Building and maintaining consistent URI space....................2
3 Mapping the local resource space to the final URI space....................2
3.1 How Plugins May Affect The URI Space....................3
4 Basic link to internal generated pages....................3
5 Basic link to raw un-processed content....................3
5.1 Serving (X)HTML content without Skinning....................4
6 Full URL to external documents....................5
7 Using site.xml to manage the links....................5
7.1 Basic link to internal generated pages....................5
7.2 Group some items....................6
7.3 Fragment identifiers....................6
7.4 Define items for raw content....................6
7.5 External links....................6

1. Overview

Forrest has many powerful techniques for linking between documents and for managing the site navigation. This document demonstrates those techniques. The document "Menus and Linking" has the full details.

2. Building and maintaining consistent URI space

When Forrest builds your site, it starts from the front page. Like a robot, it traverses all of the links that it finds in the documents and builds the corresponding pages. Any new links are further traversed.

Sometimes those links lead to documents that are generated directly from xml source files, sometimes they are generated from other source via an intermediate xml format. Other times the links lead to raw un-processed content.

The site navigation configuration file "`site.xml`" provides a way to manage this URI space. In the future, when documents are re-arranged and renamed, the site.xml configuration will enable this smoothly.

3. Mapping the local resource space to the final URI space

For both generated and raw (un-processed) files, the top-level of the URI space corresponds to the "`content/xdocs/`" directory, i.e. the location of the "`site.xml`" configuration file.

Note:
In versions prior to 0.7 raw un-processed content was stored in the "`content/`" directory. In 0.7 onwards, raw un-processed data is stored alongside the xdocs. In addition, in 0.6 and earlier, HTML documents could be stored in the xdocs directory and served without processing. If you you wish to emulate the behaviour of 0.6 and earlier see the next section.

A diagram will help.

```
The resource space          ===============>    The final URI space
------------------                              -------------------
Generated content ...
 content/xdocs/index.xml                        index.html
 content/xdocs/samples/index.xml                samples/index.html
 content/xdocs/samples/faq.xml                  samples/faq.html
 content/xdocs/test1.html                       test1.html
 content/xdocs/samples/test3.html               samples/test3.html
 content/xdocs/samples/subdir/test4.html        samples/subdir/test4.html

Raw un-processed content ...
```

```
content/xdocs/hello.pdf                       hello.pdf
content/xdocs/hello.sxw                       hello.sxw
content/xdocs/subdir/hello.sxw                subdir/hello.sxw
```

3.1. How Plugins May Affect The URI Space

By using Forrest Input Plugins you can process some file formats, such as OpenOffice.org documents and produce processed content from them. For example, the file `content/xdocs/hello.sxw` can be used to produce a skinned version of the document at with the name `hello.html`. Similarly, you can use Forrest Output Plugins to create different output formats such as PDF, in this case `content/xdocs/hello.sxw` can produce `hello.pdf`.

However, this does not affect the handling of raw content. That is, you can still retrieve the raw un-processed version with, for example, `hello.sxw`. If you want to prevent the user retrieving the un-processed version you will have to create matchers that intercept these requests within your project sitemap.

4. Basic link to internal generated pages

When this type of link is encountered, Forrest will look for a corresponding xml file, relative to this document (i.e. in `content/xdocs/samples/`).

A generated document in the current directory, which corresponds to `content/xdocs/samples/sample.html` ...

```
<a href="sample.html">sample.html</a>
```

In a sub-directory, which corresponds to `content/xdocs/samples/subdir/index.html` ...

```
<a href="subdir/index.html">subdir/index.html</a>
```

5. Basic link to raw un-processed content

Raw content files are not intended for any processing, they are just linked to (e.g. pre-prepared PDFs, zip archives). These files are placed alongside your normal content in the "`content/xdocs`" directory.

A raw document in the current directory, which corresponds to `content/xdocs/samples/helloAgain.pdf` ...

```
<a href="helloAgain.pdf">helloAgain.pdf</a>
```

A raw document in a sub-directory, which corresponds to `content/xdocs/samples/subdir/hello.zip` ...

```
<a href="subdir/hello.zip">subdir/hello.zip</a>
```

A raw document at the next level up, which corresponds to `content/hello.pdf` ...

```
<a href="../hello.pdf">../hello.pdf</a>
```

5.1. Serving (X)HTML content without Skinning

Prior to version 0.7, the raw un-processed content was stored in the "`content/`" directory. In 0.7 onwards, raw un-processed data is stored alongside the xdocs. In addition in 0.6 and earlier, HTML files could be stored in the xdocs directory and they would be served without further processing. As described above, this is not the case in 0.7 where HTML files are, by default, skinned by Forrest.

If you you wish to emulate the behaviour of 0.6 and earlier then you must add the following to your project sitemap.

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

The above allows us to create links to un-processed skinned files stored in the `{project:content}` or `{project:content.xdocs}` directory. For example: <a href="/test1.html">HTML content</a>. However, it will break the 0.7 behaviour of skinning HTML content. For this reason the old ".ehtml" extension can be used to embed HTML content in a Forrest skinned site

Note that you can change the matchers above to selectively serve some content as raw un-processed content, whilst still serving other content as skinned documents. For example, the following snippet would allow you to serve the content of an old, deprecated site without processing from Forrest, whilst still allowing all other content to be processed by Forrest in the normal way:

```
<map:match pattern="old_site/**.html">
 <map:select type="exists">
  <map:when test="{project:content}{1}.html">
    <map:read src="{project:content}/{1}.html" mime-type="text/html"/>
    <!--
      Use this instead if you want JTidy to clean up your HTML
      <map:generate type="html" src="{project:content}/{0}" />
      <map:serialize type="html"/>
    -->
  </map:when>
</map:match>
```

For example, HTML content.

6. Full URL to external documents

A full URL ...

```
<a href="http://forrest.apache.org/">http://forrest.apache.org/</a>
```

A full URL with a fragment identifier ...

```
<a
href="http://forrest.apache.org/faq.html#link_raw">http://forrest.apache.org/faq.html#l
```

Note that Forrest does not traverse external links to look for other links.

7. Using site.xml to manage the links

As you will have discovered, using pathnames with ../../ etc. will get very nasty. Real problems occur when you use a smart text editor that tries to manage the links for you. For example, it will have trouble linking to the raw content files which are not yet in their final location.

Links and filenames are bound to change and re-arrange. It is essential to only change those links in one central place, not in every document.

The "`site.xml`" configuration file to the rescue. It maps symbolic names to actual resources.

7.1. Basic link to internal generated pages

This single entry ...

```
<index label="Index" href="index.html"/>
```

enables a simple link to a generated document, which corresponds to `content/xdocs/index.xml` ...

```
<a href="site:index">site:index</a>
```

7.2. Group some items

This compound entry ...

```
  <samples label="Samples" href="samples/" tab="samples">
    <faq label="FAQ" href="faq.html"/>
    ...
  </samples>
```

enables a link to a generated document, which corresponds to `content/xdocs/samples/index.xml` ...

```
<a href="site:samples">site:samples</a>
```

and a link to a generated document, which corresponds to `content/xdocs/samples/faq.xml` ...

```
<a href="site:faq">site:faq</a>
which can also be a complete reference
<a href="site:samples/faq">site:samples/faq</a>
```

7.3. Fragment identifiers

This compound entry ...

```
  <samples label="Samples" href="samples/" tab="samples">
    <sample label="Apache document" href="sample.html">
      <top href="#top"/>
      <section href="#section"/>
    </sample>
    ...
  </samples>
```

enables a link to a fragment identifier within the `samples/sample.html` document ...

```
<a href="site:samples/sample/section">site:samples/sample/section</a>
```

7.4. Define items for raw content

This entry ...

```
<hello_print href="hello.pdf"/>
```

enables a link to a raw document, which corresponds to `content/hello.pdf` ...

```
<a href="site:hello_print">site:hello_print</a>
```

7.5. External links

This compound entry ...

```
  <external-refs>
    <forrest href="http://forrest.apache.org/">
      <linking href="docs/linking.html"/>
      <webapp href="docs/your-project.html#webapp"/>
    </forrest>
  </external-refs>
```

enables a link to an external URL ...

```
<a href="ext:forrest">ext:forrest</a>
```

and a link to another external URL ...

```
<a href="ext:linking">ext:linking</a>
which can also be a complete reference
<a href="ext:forrest/linking">ext:forrest/linking</a>
```

and a link to another external URL with a fragment identifier ...

```
<a href="ext:webapp">ext:webapp</a>
which can also be a complete reference
<a href="ext:forrest/webapp">ext:forrest/webapp</a>
```