generate-forrest-docs

Command Reference

Table of contents

1. Description

Generate forrest docs from type.xml files.

This command generates HTML/PDF documentation for the target set of modules. Each type.xml file is converted to a set of forrest-xml documents which are used by Forrest to produce the structured website and PDF files. The website is packaged into a zip file and stored at the targetdir location.

The resulting site has a "User Reference" and a "Developer Reference" section. These two sections have the purposes of explaining how to use the commands found in the module source, and how to develop using the Type with the class-based modelling system of ControlTier, respectively.

The Developer Reference section contains a page for each Type module, and contains necessary reference information about Constraints, Attributes, related Types, etc. as defined in the "type.xml" files.

1.1. Running the command

The first time you run this command for a set of modules, it will create a directory at `${opts.basedir}/doc` by copying a set of templates in place. This framework for your forrest site can be stored in a source repository if desired. In particular, you may want to modify the index.xml file located at `${opts.basedir}/doc/src/documentation/content/xdocs/index.xml`.

The command will look for `${opts.basedir}/modules/TYPENAME/type.xml` files, for every TYPENAME directory found in the basedir, then generate the forrest documentation XML files to the `${opts.basedir}/doc` directory. These sources will be used to render the HTML/PDF files, which are then packaged into a file called "site.zip" and stored at `${opts.targetdir}/site.zip`.

For each type module in the basedir that is processed, a `type.pdf` file will be copied to the `${opts.basedir}/modules/TYPENAME/doc/type.pdf` location, which is the PDF rendering of the Developer Reference page for the type.

1.2. Using the DOC tag to write documentation

The forrest docs have a standard structure, but any element in the source type.xml file can have additional documentation by including a <doc> tag as an immediate child.

The User Reference will contain an Overview page for each Type module, which will

include any `<doc>` content under the `<commands>` section of the type.xml. The User Reference also contains a list of all commands and types found in the module source, and for each command produces a reference page. This page will contain the `<doc>` content for the command definition from the type.xml. It will also contain reference information for the options for each command, and an example of how to execute the command. If a `<doc id="examples">` section is found in the `<command>` section, that content is placed an a section titled Examples, which is meant to contain example executions of the command with discussion of the purpose and result of those command executions.

<doc> tags in the type.xml files can contain valid forrest xml fragments, with the addition of two special tags:

- <typelink>TYPENAME</typelink> - *produces a link to a type if valid, or the type name otherwise.*
- <commandlink>COMMANDNAME</commandlink> - *produces a link to the command of the same name of for the current type.*

For example, "<typelink>ProjectBuilder</typelink>" produces this: ProjectBuilder, and "<commandlink>generate-forrest-docs</commandlink>" produces this: generate-forrest-docs.

An invalid type link "<typelink>NotAType</typelink>", produces this: NotAType, similarly "<commandlink>invalid-command-name</commandlink>" produces invalid-command-name.

If you're reading this, then this command was used to generate this documentation. This text is included as a <doc> tag inside the <command> tag:

```
            <command name="generate-forrest-docs" description="generate
forrest docs from type.xml files"
            command-type="AntCommand">
            <doc>
            ...
            </doc>
            ...
            </command>
```

The <doc> tag must contain only valid forrest document content xml. (See the DTD.) Any elements/tags that are not recognized by forrest will cause an error when generating documentation from the xml file.

The xml inside the <doc> tag will be inserted into the forrest output document in one of two locations: directly underneath a `<section>` tag or a `<td>` tag (in some cases).

This means that for the most part you will need to include `<p>...</p>` tags around your

documentation content. Also note that tags like `<ol>, <ul>, <table>` cannot be placed inside <p> tags, as that is not valid forrest content.

The XSL transformation used by this command tries to be smart about the content of the <doc>tag. If it looks like it is just textual content with normal text markup, then the content is wrapped in `<p>...</p>` automatically. Otherwise it is included as-is in the forrest xml output, so you must be sure it is valid.

Example: this is invalid:

```
<doc>
This is some text, and some <em>emphasis</em>.
<p>
Here is a "&lt;p&gt;" embedded.
</p>
</doc>
```

It must be changed to this:

```
<doc>
<p>
This is some text, and some <em>emphasis</em>.
</p>
<p>
Here is a "&lt;p&gt;" embedded.
</p>
</doc>
```

Whereas this:

```
<doc>
This is some text, and some <em>emphasis</em>.
</doc>
```

Will be wrapped as a valid paragraph in the output:

```
<document>
...
<p>
This is some text, and some <em>emphasis</em>.
</p>
...
</document>
```

static: This command can be run outside of an object context.

2. Usage

```
ctl -m ProjectBuilder -c generate-forrest-docs [-basedir <>]
[-docbase <>] [-forresthome <${env.FORREST_HOME}>] [-name <>]
[-overwrite] [-targetdir <>]
```

2.1. Options

Option	Description	Type	Default

`basedir`	*Directory containing a 'modules' directory, which contains module types.*
	The basedir is expected to contain:
	• ${opts.basedir}/modul
	  • Type1/
	  • type.xml
	  • ...
	  • Type2/
	  • type.xml
	  • ...
		string	`${entity.attribute.basedir}`

`docbase`	*Path to documentation base directory.* This will default to the value `${opts.basedir}/do`	string	
`forresthome`	*Path to FORREST_HOME* Specify the FORREST_HOME directory with this option, or set the environment variable `FORREST_HOME` prior to running this command.	string	`${env.FORREST_HOME}` / `${entity.attribute.forresthome`
`name`	*Name of the library or site for documentation.*	string	`${entity.attribute.projectName`

`overwrite`	*Whether or not to overwrite existing site definition files.*	boolean	
`targetdir`	*destination dir for forrest docs* A file called "site.zip" will be created and stored at `${opts.targetdir}/`	string	`${entity.attribute.targetdir}`

3. Examples

To generate a forrest site for a set of modules stored in the `/home/ctier/src/mylib` directory, and give the site the title "mylib":

```
ctl -p demo -m ProjectBuilder -c generate-forrest-docs -- -basedir
/home/ctier/src/mylib -name mylib \
-targetdir /home/ctier/target
```

The command will look for `/home/ctier/src/mylib/modules/TYPENAME/type.xml` files, for every TYPENAME directory found in the basedir, then generate the forrest documentation XML files to the `/home/ctier/src/mylib/doc` directory. These sources will be used to render the HTML/PDF files, which are then packaged into a file called "site.zip" and stored at `/home/ctier/target/site.zip`.