

# ***DITA Open Toolkit Docs Process***

---

*How the DITA Open Toolkit project builds documentation*

*infotexture*  
Information Architecture & Content Strategy

**Roger W. Fienhold Sheen**

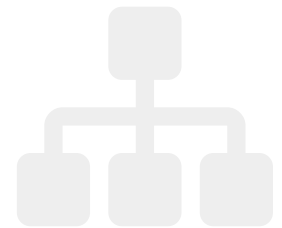
# Agenda

---



*The DITA Open Toolkit project uses DITA features and other open-source tools like Git, Gradle, Jekyll, and Netlify to build the toolkit documentation included in distribution builds and published on the project website at [dita-ot.org](https://dita-ot.org).*

- 1. Content management & collaboration**
- 2. DITA and DITA-OT features**
- 3. Automatically generating topics**
- 4. Continuous integration**
- 5. Building the project website**
- 6. Edit This Page!**



# Git is our Content Management System

---

We use Git, a distributed version control system, to track changes to the documentation source files.



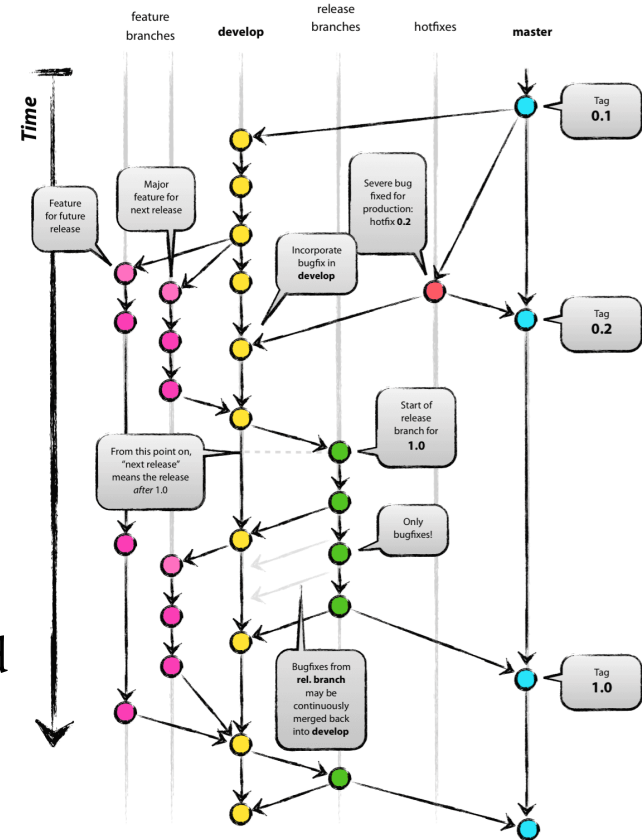
Git provides many of the basic features of content management systems:

- version control
- file comparison via diff tools
- file history via the `git log`
- change attribution via `git blame`
- variant management via branching
- release management via tagging

# GitFlow branching strategy

**GitFlow** uses a series of branches and procedural conventions to organize the software development process and assist in release management.

- The `master` branch reflects a production-ready state; contains only merge commits.
- The `develop` branch is the main integration branch with latest changes for the next release; automatic builds for each commit.
- Feature branches (or “topic” branches) isolate related changes during development of the next release; branch from & merge to `develop`.
- Release branches are used to prepare a new production release; branch from `develop`, merge to `develop` & `master`. Hotfix branches track bugfixes for production releases; branch from `master`, merge to `develop` & `master`.



# GitHub collaboration

GitHub provides free code hosting, project management and collaboration tools for open-source projects.



- Track issues with labels & milestones
- Plan releases with project Kanban boards
- Discuss & approve proposed changes in pull requests
- Comment on commits or individual lines with code review tools

A screenshot of a GitHub Kanban board with three columns: Backlog (3 items), ToDo (6 items), and In Progress (2 items). Each item is a card with a title, description, issue number, and author. Labels like 'feature', 'backlog', 'enhancement', and 'process' are visible on some cards.

Column	Item	Issue #	Author	Labels
Backlog (3)	Expand migration info		infotexture	
	Frequently asked questions	#94	raducoravu	enhancement
ToDo (6)	Revise `dita` command description w/ new argument syntax	#106	infotexture	feature
	Add topic(s) on DITA features used in docs	#50	infotexture	backlog
In Progress (2)	Group extensions by plug-in		infotexture	
	Add DCO requirement to contribution info	#107	infotexture	enhancement, process

# DITA and DITA-OT features

---

The DITA Open Toolkit project uses various recent DITA features in the documentation builds, including:

- subjectScheme classification for controlling available attributes
- profiling and (branch) filtering (*novice/expert content*)
- extending topics with conref push
- keys and key references
- XML mention domain



## ✨ *New in 4.0*

Recent versions of the documentation make use of new features in DITA-OT:

- The distribution documentation is now built with a custom **PDF theme**
- The **Markdown DITA syntax reference** is built from a GitHub wiki

# DITA features — subject schemes

---

Various topics, sections and elements in the docs are profiled by audience:

```
<li id="novice-variables-last" audience="novice">
  <p id="common-format-info">
    <varname>format</varname> is the output format (transformation type). This argument corresponds to the
    common parameter <xref keyref="parameters-base/transtype"/>. Use the same values as for the
    <parmname>transtype</parmname> build parameter, for example <option>html5</option> or
    <option>pdf</option>.</p>
</li>
```

An “audience” subject scheme controls the values that are available for the `@audience` attribute:

```
<subjectdef keys="audience">
  <subjectdef keys="novice"/>
  <subjectdef keys="expert"/>
  <subjectdef keys="xslt-customizer"/>
</subjectdef>
```

# DITA features — branch filtering

---

## Re-using and (branch) filtering profiled content

The *Installing DITA-OT* section pulls a subset of the build description, filtered to display only content deemed suitable for novice users under **Building output**:

```
<topicref href="using-dita-command.dita"
          keys="first-build-using-dita-command" locktitle="yes">
  <topicmeta>
    <navtitle>Building output</navtitle>
  </topicmeta>
  <ditavalref href="../resources/novice.ditaval">
    <ditavalmeta>
      <dvrResourcePrefix>first-build-</dvrResourcePrefix>
    </ditavalmeta>
  </ditavalref>
</topicref>
```

The same content appears later in **Building output using the dita command** with additional information on arguments, options and examples.



# DITA features — conref push

The docs build uses the conref push mechanism (specifically `@conaction="pushafter"`) to extend the parameter descriptions embedded in the default plug-ins:

```
<plentry id="args.csspath">
  <pt>
    <parname>args.csspath</parname>
  </pt>
  <pd conaction="mark" conref="parameters-base-html.dita#base-html/args.csspath.desc"/>
  <pd conaction="pushafter" audience="xslt-customizer">Corresponds to the XSLT parameter
    <parname>CSSPATH</parname>. DITA-OT will copy the file to this location.</pd>
</plentry>
```

The pushed content appears in the output after the default description:

```
args.csspath
Specifies the location of a copied .css file relative to the output directory.
Corresponds to the XSLT parameter CSSPATH. DITA-OT will copy the file to this location.
```

**TIP:** You could also use the same mechanism to extend the documentation with custom information that applies only to your company's toolkit distribution.

# DITA features — keys and key references

---

The `key-definitions.ditamap` defines keys for version references, re-usable links, etc.

This key definition defines the latest maintenance release:

```
<keydef keys="maintenance-version">
  <topicmeta>
    <keywords>
      <keyword>4.1.2</keyword>
    </keywords>
  </topicmeta>
</keydef>
```

In topics, the keyword is used in place of hard-coded version references:

```
<title>DITA Open Toolkit <keyword keyref="maintenance-version"/> Release Notes</title>
```

# DITA features — XML mention domain

---

The docs use the **XML mention domain** to mark up XML elements and attributes:

```
<li id="1777">  
  DITA 1.3: Initial support has been added for the <xmlatt>orient</xmlatt>  
  attribute on <xmlelement>table</xmlelement> elements. These changes allow  
  Antenna House Formatter to render tables in landscape mode when the  
  <xmlatt>orient</xmlatt> attribute is set to <option>land</option>. [...]  
</li>
```

When the toolkit generates output for the sample above:

- the XML element name is wrapped in angle brackets as `<table>`
- the attribute name is prefixed with an “at” sign as `@orient`

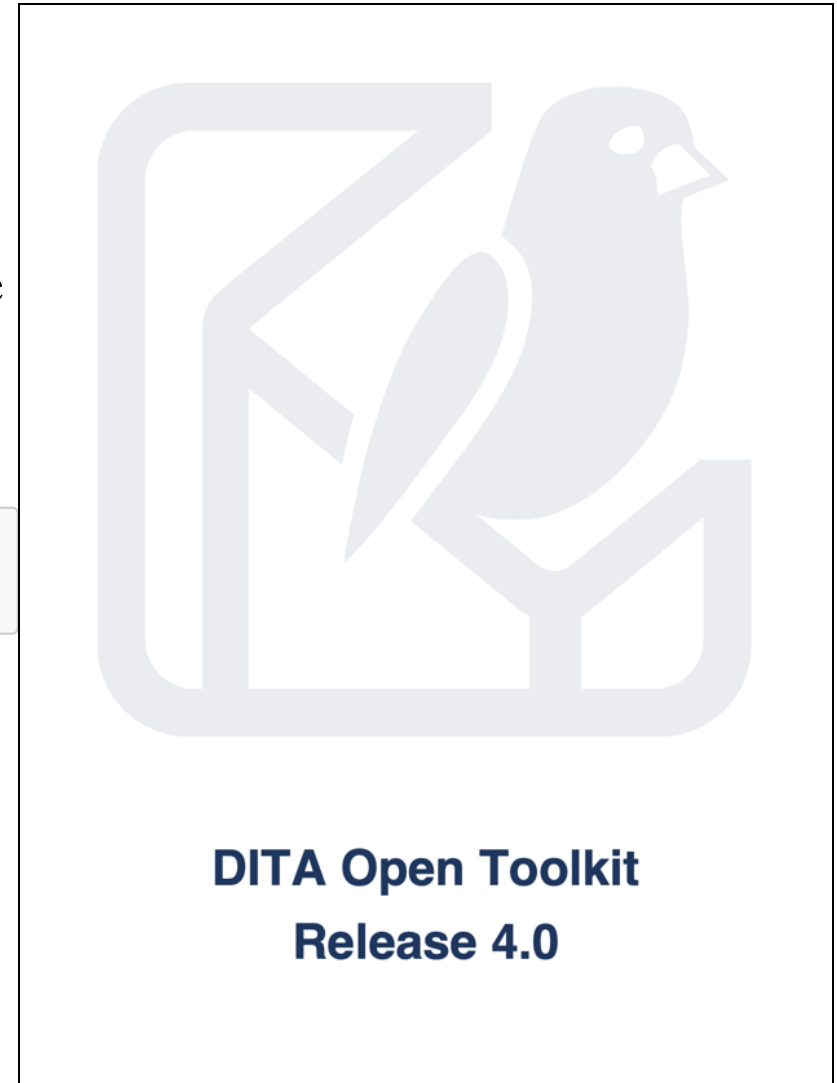
# PDF theme

---

As of DITA-OT 4.0, the PDF version of the distribution documentation is built with a custom **PDF theme**.

The `--theme` option takes a path to a theme file and changes the styling of the PDF output without changing XSLT stylesheets.

```
dita --project=samples/project-files/pdf.xml \  
     --theme=path/to/custom-theme-file.yaml
```



# Automatically generating topics

---

The docs build generates new topics from plug-in code via Ant & XSLT:

- **the error message overview** — *with Additional details column via conref push*
- **parameter listings**
- **extension points**

```
<antcall>
  <target name="generate-msg-topic"/>
  <target name="generate-params-topic"/>
  <target name="generate-extension-points-topic"/>
  <target name="generate-properties-file"/>
</antcall>
```

Plus an annotated properties file template you can use for your own builds:

```
<target name="generate-properties-file" depends="init" description="Regenerate annotated .properties file">
  <property name="propfile.xml" location="${basedir}/resources/properties-file.xml"/>
  <property name="propfile.input" location="${resource.dir}/plugins.xml"/>
  <property name="propfile.output" location="${doc.samples.dir}/properties/template.properties"/>
  <xslt in="${propfile.input}" out="${propfile.output}" style="${propfile.xml}" force="yes"/>
</target>
```

# Continuous integration — Gradle build tool

---

Gradle is a next-generation build tool that understands Ant files and offers significant performance advantages.



- Build caching makes builds faster
- Incremental builds — build only what has changed
- The `--continuous` option re-runs the build whenever source files change

## *DITA Open Toolkit Gradle Plugin*

The `dita-ot-gradle` plug-in by DITA-OT contributor Eero Helenius runs DITA-OT from Gradle, and significantly faster than running the toolkit directly.

You can publish all `.ditamap` files in a directory at once, or publish the same document multiple times with different parameters.

We use the DITA Open Toolkit Gradle Plugin to publish the docs on the website, and for local testing to ensure output is generated correctly before committing.

# Netlify CI builds for each push

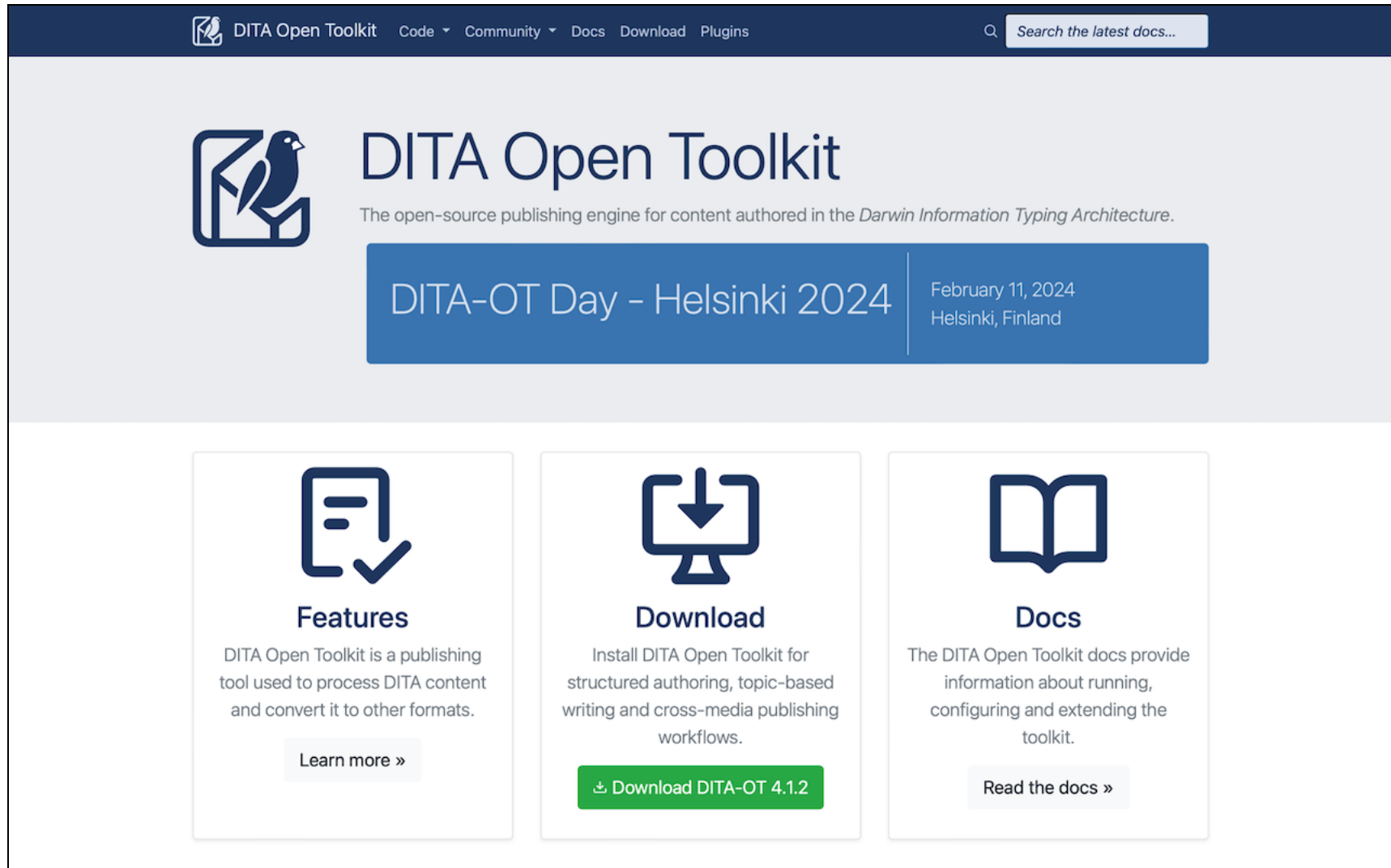
The Netlify platform builds and deploys sites to a global content delivery network from Git. Netlify supports DITA-OT with an unlimited plan for open source projects.



Netlify runs the Gradle site build whenever new commits are pushed to the `develop` branch and deploys the output to [dita-ot.org](https://dita-ot.org) when the build passes.

Production deploys >	Deploy Previews >
<b>Production: master@aecfe62</b> <span style="background-color: #28a745; color: white; padding: 2px;">Published</span> Oct 5: Bump postcss from 8.4.29 to 8.4.31 (#775) >	<b>Deploy Preview #775:</b> dependabot...@f7ff300 Oct 3: Bump postcss from 8.4.29 to 8.4.31 >
<b>Production: master@f39c168</b> Oct 3: Update JavaScript dependencies (#773) >	<b>Deploy Preview #774:</b> release/4.1.2@c20b346 Oct 2: Update site for 4.1.2 >
<b>Production: master@16ca136</b> Oct 2:  Deploy dita-ot/docs@c08ddba to 'dev' docs >	<b>Deploy Preview #774:</b> release/4.1.2@da2d9b8 Oct 2: Update site for 4.1.2 >
<b>Production: master@5aa6f10</b> Oct 2: Merge pull request #774 from dita-ot/release/4.1.2 >	<b>Deploy Preview #773:</b> test-depend...@b64602b Oct 1: Update JavaScript dependencies >

# DITA-OT website — dita-ot.org



The screenshot shows the homepage of the DITA Open Toolkit website. At the top is a dark blue navigation bar with the DITA logo, the text "DITA Open Toolkit", and menu items for "Code", "Community", "Docs", "Download", and "Plugins". A search bar on the right contains the text "Search the latest docs...". Below the navigation bar is a large light gray hero section. On the left is the DITA logo, a stylized bird. To its right is the main heading "DITA Open Toolkit" and a subheading "The open-source publishing engine for content authored in the Darwin Information Typing Architecture." Below this is a blue banner for "DITA-OT Day - Helsinki 2024" with the date "February 11, 2024" and location "Helsinki, Finland". The main content area below the hero section consists of three white boxes with rounded corners. The first box, titled "Features", has an icon of a document with a checkmark and a downward arrow, and a "Learn more »" button. The second box, titled "Download", has an icon of a monitor with a downward arrow and a green "Download DITA-OT 4.1.2" button. The third box, titled "Docs", has an icon of an open book and a "Read the docs »" button.

DITA Open Toolkit Code Community Docs Download Plugins Search the latest docs...

## DITA Open Toolkit

The open-source publishing engine for content authored in the *Darwin Information Typing Architecture*.

**DITA-OT Day - Helsinki 2024** February 11, 2024  
Helsinki, Finland

### Features

DITA Open Toolkit is a publishing tool used to process DITA content and convert it to other formats.

[Learn more »](#)

### Download

Install DITA Open Toolkit for structured authoring, topic-based writing and cross-media publishing workflows.

[Download DITA-OT 4.1.2](#)

### Docs

The DITA Open Toolkit docs provide information about running, configuring and extending the toolkit.

[Read the docs »](#)



# Building the project website

---

The DITA-OT project website is published via [Netlify](#) to [dita-ot.org](#).

The website is maintained in DITA, [Markdown](#) and HTML, versioned in Git and updated by pushing commits to the repository at [github.com/dita-ot/website](#).

The HTML version of the site is built with [Jekyll](#), an open source tool like DITA-OT that transforms files in one format with variables and templates, and generates output.



Like DITA keys in a project map, variables like `version: '4.1.2'` are defined in Jekyll's `_config.yml` file and referenced in source files using [Liquid](#) syntax: `{{site.version}}`.

```
<a href="https://github.com/dita-ot/dita-ot/releases/download/{{site.version}}/dita-ot-{{site.version}}.zip"></
```



Jekyll supports [Sass](#): “*Syntactically Awesome Style Sheets*”, which extends CSS with variables, nesting, partials, imports and inheritance.

# Building the documentation

---

The **Documentation** section is maintained in DITA using the source files from the DITA Open Toolkit documentation repository at [github.com/dita-ot/docs](https://github.com/dita-ot/docs).

The OT docs are transformed to HTML5 using the [org.dita-ot.html](#) plugin, which extends the default `html5` transformation with additional processing.

The site plug-in adds **Bootstrap** classes and YAML metadata to the generated HTML fragments, and Jekyll templates in the site repository provide the base layout:

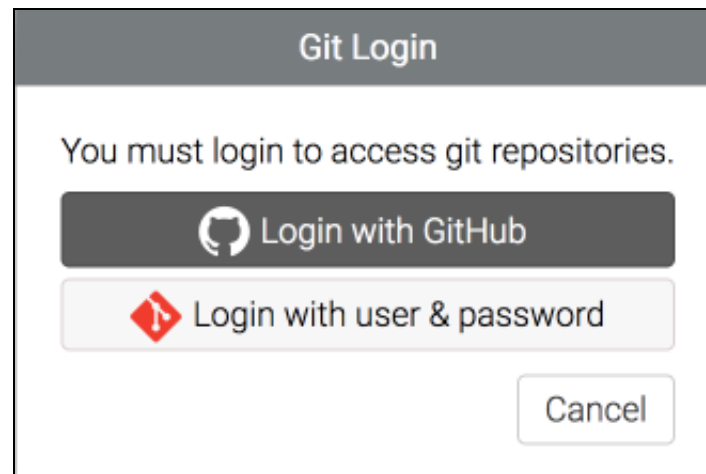
```
<body>
{% include header.html %}
<!-- : (conditional menu logic) : -->
<div id="content" class="container">
  <div class="row">
    <!-- ↓ Here there be DITA... -->
    {{ content }}
  </div>
</div>
{% include footer.html %}
{% include search.html %}
{% include help.html %}
</body>
```

# Edit This Page!

---

The page headers in the development documentation include **Edit this page** links that open the DITA source file for the topic in [oXygen XML Web Author](#).

The web-based authoring workflow prompts users to log in to GitHub and fork the [dita-ot/docs](#) repository if necessary.



Changes saved in the authoring environment are committed to a new branch, and a pull request is created to submit changes for review by the DITA-OT docs team.

# Feedback

---

Visit [www.dita-ot.org/dev/](http://www.dita-ot.org/dev/) for the latest docs.

## Create an Issue

If you find a bug — *and you don't know how to fix it*, **create an issue**.

## Create a Pull Request

*Or — if that all sounds too complicated — just click the [Edit this page link](#).*

## Slides

<https://infotexture.net/boston-dita-2023>

## Contact

✉ [roger@infotexture.net](mailto:roger@infotexture.net)  [GitHub](#)  [Twitter](#) **in** [LinkedIn](#) — [@infotexture](#)

