Customizing DITA output

…isn't as hard as you may think °

Roger Fienhold Sheen

@infotexture

° within certain parameters…
under controlled conditions

— but that may be all you need.

Caveat № 2:
This is not a tutorial

— but it links to several

Agenda

- What you can change easily *(via parameters)*
- What if that's not enough *(simple plug-ins)*
- Where to go for help *(customization resources)*

What you can change easily *(via parameters)*

Parameter-based customizations

- Some apply to all transformations
- Others are common to HTML-based formats
- Certain parameters apply only to specific transformations
- Pass as options to the `dita` command using the `--parameter=value` syntax

See https://www.dita-ot.org/dev/parameters.

Customizing HTML with parameters

- Adding navigation
- Adding custom CSS
- Adding custom headers and footers

Adding navigation

In HTML5 output, you can set a parameter to include table-of-contents navigation in the `<nav>` element of each page.

Set the `nav-toc` parameter

- `partial` — current topic, parents, siblings & children
- `full` — embed navigation for entire map in each topic

Styling navigation

To render the navigation in a sidebar or menu, add CSS rules to define the placement and presentation:

```
nav[role="toc"] {
  float: left;
  width: 300px;
}

nav[role="toc"] li.active > a {
  font-weight: bold;
}
```

but how do I do that?

— I’m glad you asked…

Adding custom CSS

To modify the appearance of the default HTML output, you can reference a custom *Cascading Style Sheet* (CSS) file.

In your custom stylesheet, add rules with the typography, colors, and other presentation aspects that define your corporate identity.

(Like the position of the navigation in the previous example.)

Setting CSS parameters

1. Create a custom CSS file and store it with your source files.
2. Set the `args.css` parameter to the name of your CSS file.
3. Set `args.cssroot` to the path that contains your CSS.
4. Set the `args.copycss` parameter to `yes`.
5. Set `args.csspath` to the desired CSS output location.

Adding a custom header

Set `args.hdr` to include a running header above the page content with a publication title, company logo, or other common branding.

```
<div class="header">
  <a href="/" title="Go to the homepage">
    <svg role="img" class="c-brand-icon" title="MegaCorp logo">…</svg>
  </a>
</div>
```

Content is wrapped in an HTML5 `<header>` with the `@role` set to `banner`.

Adding a custom footer

Set `args.ftr` to include a running footer below the page content, with copyright information, legal boilerplate, or other fine print.

```
<div class="footer">
  <p>© 2018 · MegaCorp. All rights reserved.</p>
</div>
```

Content is wrapped in a `<footer>` with the `@role` set to `contentinfo`.

Styling headers & footers

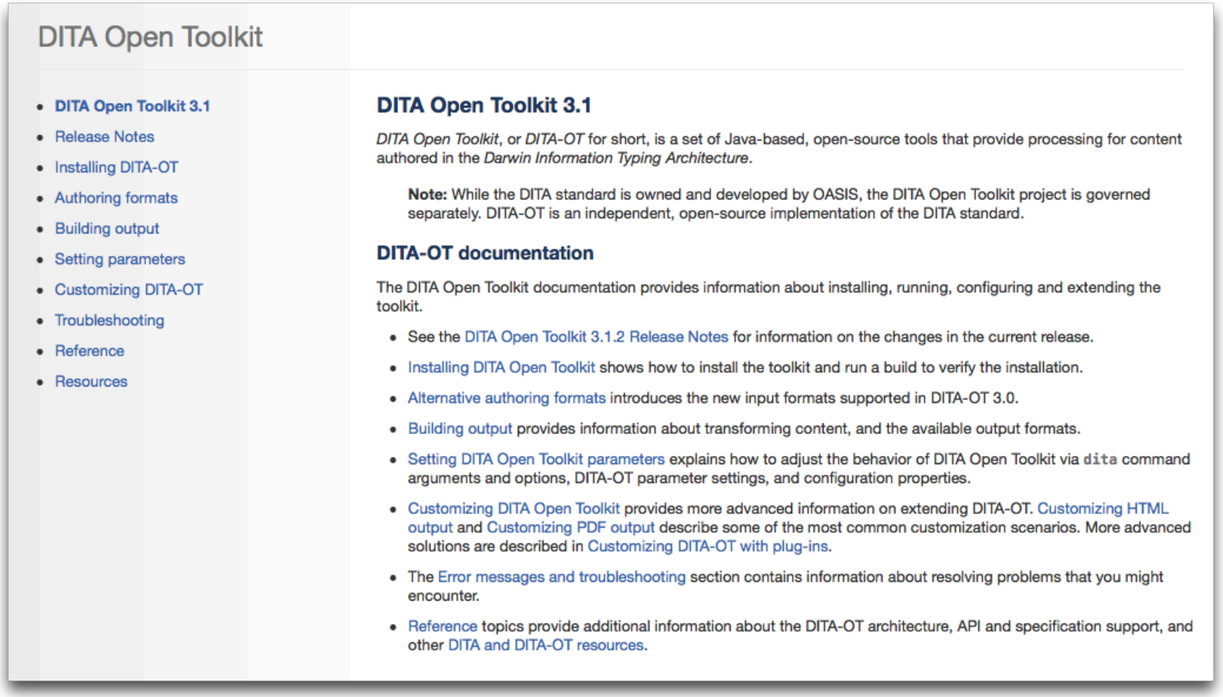
Add custom CSS rules to style headers and/or footers.

```
div.header p {
  color: #777;
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  line-height: 1.3;
  margin: 0;
}
```

For more sample header rules, see the dita-ot-doc.css file.

Sample settings & results

```
# Generate partial navigation in topics
nav-toc = partial

# Custom CSS file used to style output
args.css = dita-ot-doc.css

# Copy the custom CSS file to the output
args.copycss = yes

# Directory that contains the custom CSS
args.cssroot = resources

# Location of the copied CSS in output
args.csspath = css

# File with running header content
args.hdr = ${basedir}/resources/header.xml
```



What if that's not enough *(simple plug-ins)*

- Bundling CSS in a custom HTML plug-in
- Embedding web fonts in HTML output
- Inserting JavaScript in generated HTML
- New "DITA Bootstrap" plug-in

Bundling custom CSS

Create a plug-in that bundles custom CSS and related parameter settings to apply your corporate identity to any project:

```
<plugin id="com.example.html5-custom-css">
  <require plugin="org.dita.html5"/>
  <transtype name="html5-custom-css" extends="html5" desc="Custom CSS"/>
  <feature extension="ant.import" file="build_html5-custom-css.xml"/>
</plugin>
```

Sample Ant file

```
<project>
  <target name="dita2html5-custom-css"
        depends="dita2html5-custom-css.init,
                 dita2html5"/>
  <target name="dita2html5-custom-css.init">
    <property name="args.cssroot"
          location="${dita.plugin.com.example.html5-custom-css.dir}/css"/>
    <property name="args.css" value="custom.css"/>
    <property name="args.copycss" value="yes"/>
    <property name="args.csspath" value="css"/>
  </target>
</project>
```

No Tofu: □

— Noto

Adding web fonts to HTML5

Create a plug-in that bundles the Noto Sans web font and custom CSS to display characters in virtually any language:

```
<plugin id="com.example.html5-webfont">
  <require plugin="org.dita.html5"/>
  <transtype name="html5-webfont" extends="html5" desc="Noto Sans"/>
  <feature extension="ant.import" file="build_html5-webfont.xml"/>
</plugin>
```

Adding links to the HTML <head>

In your custom build target, set the `args.hdf` parameter:

```
<target name="dita2html5-webfont.init">
  <property name="args.hdf"
    location="${dita.plugin.com.example.html5-webfont.dir}/include/webfont.hdf.xml"/>
  <!-- More CSS parameters here -->
</target>
```

From the custom `<head>` content, link to the web font:

```
<div>
  <link href="https://fonts.googleapis.com/css?family=Noto+Sans" rel="stylesheet"></link>
</div>
```

Using the font

In your custom CSS file, define rules to apply the font:

```
body {
  font-family: 'Noto Sans', sans-serif;
}
```

You'll probably want different fonts for headings and body text,

— but you get the idea…

Adding JavaScript to HTML5

Create a plug-in that inserts JavaScript code in HTML pages to enable web analytics or dynamic content delivery:

```
<plugin id="com.example.html5-javascript">
  <require plugin="org.dita.html5"/>
  <transtype name="html5-javascript"
             extends="html5" desc="Add JavaScript"/>
  <feature extension="ant.import"
           file="build_html5-javascript.xml"/>
</plugin>
```

Adding links to the HTML `<footer>`

In your custom build target, set the `args.ftr` parameter:

```
<target name="dita2html5-javascript.init">
  <property name="args.ftr"
    location="${dita.plugin.com.example.html5-javascript.dir}/
              include/javascript.ftr.xml"/>
</target>
```

Note: If your JavaScript supports pre-loading and your app targets modern browsers that recognize the `async` script attribute, you may prefer to insert the JavaScript in the `<head>` element.

Place your JavaScript in the footer file

```
<div>
<!-- Google Analytics -->
<script>
console.log('Adding Google Analytics tracker');

(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
})(window,document,'script','https://www.google-analytics.com/analytics.js','ga');

ga('create', 'UA-XXXXX-Y', 'auto');
ga('send', 'pageview');
</script>
<!-- End Google Analytics -->
</div>
```

DITA Open Toolkit 3.2

DITA Open Toolkit, or *DITA-OT* for short, is a set of Java-based, open-source tools that provide processing for content authored in the *Darwin Information Typing Architecture*.

Note: While the DITA standard is owned and developed by OASIS, the DITA Open Toolkit project is governed separately. DITA-OT is an independent, open-source implementation of the DITA standard.

DITA-OT documentation

The DITA Open Toolkit documentation provides information about installing, running, configuring and extending the toolkit.

DITA Bootstrap

- See the DITA Open Toolkit 3.2 Release Notes for information on the changes in the current release.
- Installing DITA Open Toolkit shows how to install the toolkit and run a build to verify the installation.
- Alternative authoring formats introduces the new input formats supported in DITA-OT 3.0.
- Building output provides information about transforming content, and the available output formats.
- Setting DITA Open Toolkit parameters explains how to adjust the behavior of DITA Open Toolkit via dita command arguments and options, DITA-OT parameter settings, and configuration properties.
- Customizing DITA Open Toolkit provides more advanced information on extending DITA-OT. Customizing HTML output and Customizing PDF output describe some of the most common customization scenarios. More advanced solutions are described in Working with plug-ins.
- The Error messages and troubleshooting section contains information about resolving problems that you might encounter.
- Reference topics provide additional information about the DITA-OT architecture, API and specification support, and other DITA and DITA-OT resources.

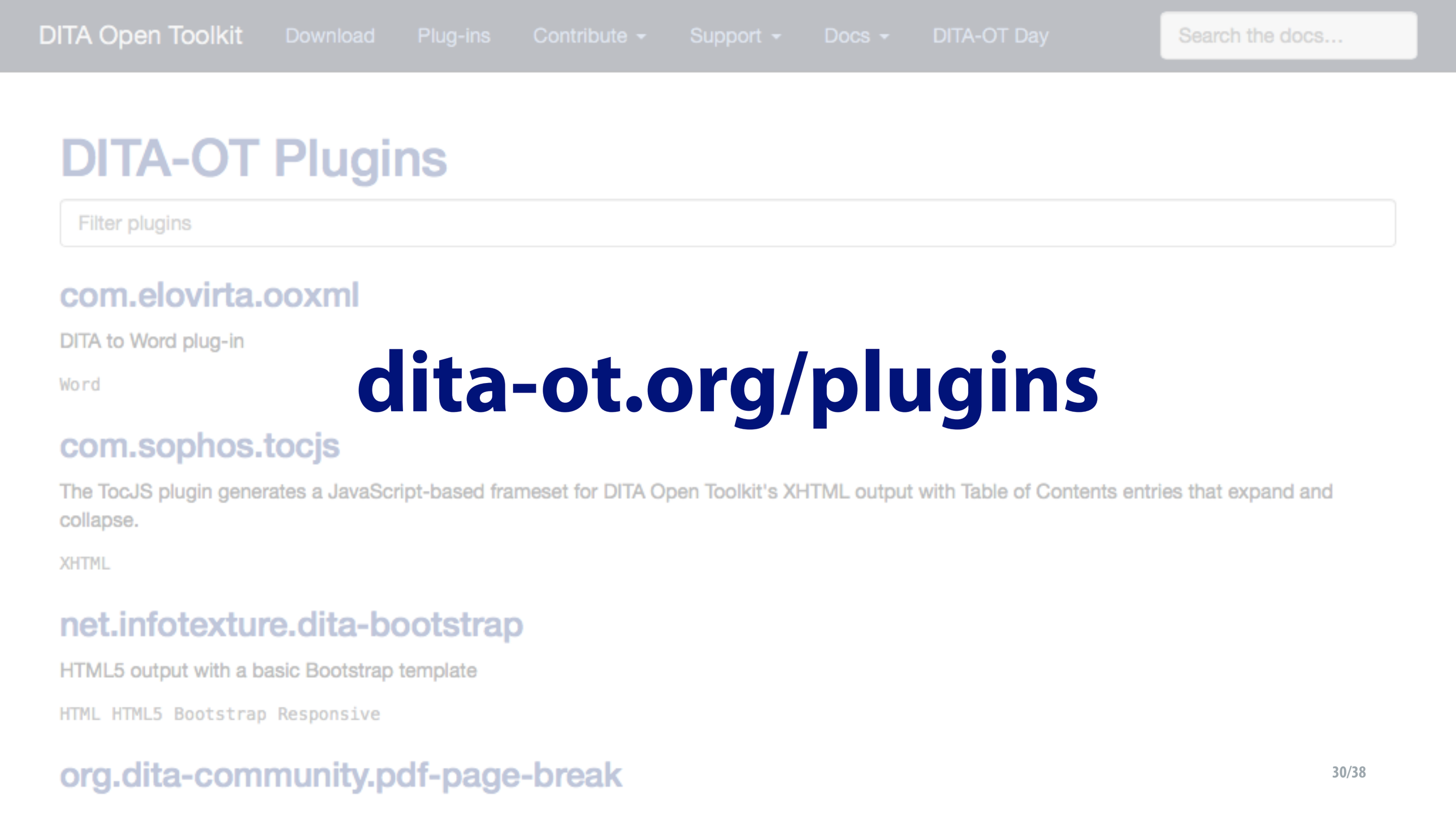
DITA Bootstrap — HTML5 with Bootstrap style

HTML5 with a basic Bootstrap template that includes responsive grid layout, global navigation menu and ToC:

🔗 https://github.com/infotexture/dita-bootstrap

The global navbar is implemented as a custom header file that can be inverted/overridden via `args.hdr` & CSS.

Where to go for help *(customization resources)*



dita-ot.org/plugins

✨ DITA-OT 3.2 — New Plug-in Registry

The plug-in registry at dita-ot.org/plugins makes it easier to discover & install plug-ins by name via the `dita` command:

```
dita --install net.infotexture.dita-bootstrap
```

The registry is a Git repository at github.com/dita-ot/registry.

New plug-ins or new versions can be added by sending a pull request that includes a new plug-in entry in JSON format.

Read The Docs

See the DITA-OT docs sections on customization:

- Bundling CSS in a custom HTML plug-in
- Embedding web fonts in HTML output
- Inserting JavaScript in generated HTML
- Customizing PDF output

Jarno Elovirta's PDF Plugin Generator

The online tool at dita-generator.elovirta.com allows you to adjust a variety of PDF settings and automatically creates a new custom PDF plug-in:

- Define the target environment *(current and 2 previous versions of DITA-OT)*
- Select the formatting engine *(FOP, Antenna House Formatter, or RenderX XEP)*
- Specify page size, columns, and margins
- Select options for headers and footers
- Specify layout options for chapters

Select fonts, colors and placement options

- Normal text
- Headings *(levels 1–4)*
- Titles for sections and examples
- Tables and figures
- Notes and examples
- Lists *(unordered, ordered, and definition)*
- Code blocks and pre-formatted text
- Inline elements such as links and trademarks

Leigh White's *"DITA for Print"*

A DITA Open Toolkit Workbook walks readers through developing a PDF customization from scratch.

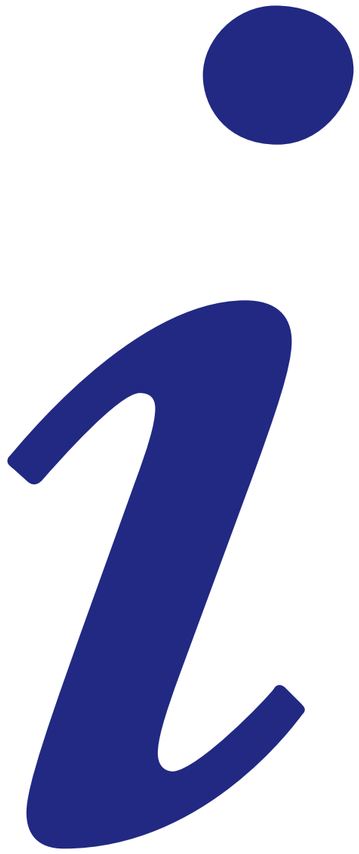
DITA for Print is for anyone who wants to learn how to create PDFs using the DITA Open Toolkit without learning everything there is to know about XSL-FO, XSLT, or XPath, or even about the DITA Open Toolkit itself.

Share your code

— Tag releases on GitHub for stable URLs

Publish your plug-ins

— Submit pull requests to the plug-in registry



infotexture