Apache OFBiz Documentation Guidelines

The Apache OFBiz Project

Version Trunk

Table of Contents


License Apache 2.0

1. Intro

These guidelines serve as a general style guide and collection of examples of how we are documenting the project. This is intentionally not a complete user manual, but lists the subset of functionality and formatting options we would like to use.

For further reference and more examples see

- Official Asciidoc User Guide [1]
- Asciidoc Writers Guide [2].
- Asciidoc Quick Reference [3]
- Asciidoc Recommended Practices [4]

If you would like to help out with the documentation of the project, please see the following wiki page [5] for further information and how we are organised.

[1] http://asciidoc.org/userguide.html

[2] https://asciidoctor.org/docs/asciidoc-writers-guide/

[3] https://asciidoctor.org/docs/asciidoc-syntax-quick-reference/

[4] https://asciidoctor.org/docs/asciidoc-recommended-practices/

[5] https://cwiki.apache.org/confluence/display/OFBIZ/OFBiz+Documentation+Team

2. General rules

2.1. Document configuration

Documents that will be published as standalone guides (e.g. developer manual, user manual) should contain common configuration so that the output will be generated in exactly the same way for all documents.

This is not necessary for documents that will only be included as part of parent documents. In this case the documents will inherit the configuration from the parent.

Please see below for details of the proposed configuration:

```
The Apache OFBiz Project ①
:imagesdir: ./images ②
ifdef::backend-pdf[] ③
:title-logo-image: image::OFBiz-Logo.svg[Apache OFBiz Logo, pdfwidth=4.25in,
align=center] ④
:source-highlighter: rouge ⑤
endif::[] ⑥
```

① author

② global definition of the image directory

③ begin block of configurations only for pdf rendering

④ define the title logo image

⑤ use the Rouge source code highlighter

⑥ end block of configurations only for PDF rendering

The following configuration options are set globally in the Gradle build file. They are not configured in the document itself and are listed for reference only:

build.gradle

```
'doctype': 'book', ①
'revnumber': 'Trunk', ②
'experimental': '', ③
'icons': 'font', ④
'sectnums': '', ⑤
'chapter-label': '', ⑥
'toc': '', ⑦
'toclevels': '5' ⑧
```

① doctype book

② target release, indicates the release for which the documentation is valid

③ allow experimental features like keyboard shortcuts

④ make font awesome icons available

⑤ number chapters and sections automatically

⑥ do not prefix the chapters

⑦ insert a table of contents

⑧ max levels to show in the table of contents

2.2. Apache License Header

Each .adoc file must contain the Apache license header (put between "//// license... ////"). You can just copy the following block:

```
////
Licensed to the Apache Software Foundation (ASF) under one
or more contributor license agreements.  See the NOTICE file
distributed with this work for additional information
regarding copyright ownership.  The ASF licenses this file
to you under the Apache License, Version 2.0 (the
"License"); you may not use this file except in compliance
with the License.  You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an
"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
KIND, either express or implied.  See the License for the
specific language governing permissions and limitations
under the License.
////
```

2.3. file organization

There are one main file per application (located in the 'src/docs/asciidoc' directory), which include some other files located in 'src/docs/asciidoc/_include'.

Example 1. For each component, there are these directories:

```
src
  |- docs
      |- asciidoc
          |- _include
          |- images
              |- ${componentName}
```

So, for include file use

```
include::_include/xxxxxxx
```

and for image, it's necessary to add on top of main file

```
:imagesdir: ./images/${compnentName}
```

and so include them with

```
image::/xxxxxxx
```

During generating process images are copied on the root asciidoc/images directory

2.4. File name conventions

We will be implementing a consistent naming convention for the documentation content files. Filenames should be in lower case with the extension `.adoc`.

Each guide will be named after the component / module name (e.g. humanres.adoc, accounting.adoc, manufacturing.adoc, party.adoc etc). Lower level files that are in the include directory will include a prefix/shortname indicating the component name, separated by dashes (e.g `hr-intro.adoc`, `hr-glossary.adoc`, etc.) Similar pages will have consistent naming. We will have several intro, glossary, FAQ, settings, security pages, so the naming format will be ([shortname]-intro, [shortname]-glossary, [shortname]-faq, [shortname]-settings, [shortname]-security etc.)

Example 2. For Human Resources this will be as follows:

```
humanres.adoc
    |- hr-intro.adoc
    |- hr-employee-evaluations.adoc
    |- hr-glossary.adoc
    |- hr-employee-positions.adoc
    |- hr-employees.adoc
    |- hr-employments.adoc
    |- hr-performance-review.adoc
    |- hr-positions.adoc
    |- hr-qualifications.adoc
    |- hr-recruitment.adoc
    |- hr-skills.adoc
    |- hr-resumes.adoc
    |- hr-training.adoc
    |- hr-leave.adoc
    |- hr-security.adoc
    |- hr-global-settings.adoc
```

2.5. General formatting

- It is recommended to write one sentence per line and/or manually break the line after approximately 80 to 120 characters.
- Section titles will use asymmetric atx style
  (e.g. `== This is an example of an Asymmetric Section Title`)
- When including another file using the `include` directive, please ensure that there is a blank line between each include line.

3. Text formatting

3.1. Quoted Text

Words and phrases can be formatted by enclosing inline text with quote characters:

Emphasized text

Word phrases enclosed in 'single quote characters' (acute accents) or *underline characters* are emphasized.

Strong text

Word phrases **enclosed in asterisk characters** are rendered in a strong font (usually bold).

Monospaced text

Word phrases enclosed in plus characters are rendered in a monospaced font. Word phrases `enclosed in backtick characters` (grave accents) are also rendered in a monospaced font but in this case the enclosed text is rendered literally and is not subject to further expansion (see inline literal passthrough).

‘Single quoted text’

Phrases enclosed with a `single grave accent to the left and a single acute accent to the right' are rendered in single quotation marks.

“Double quoted text”

Phrases enclosed with ``two grave accents to the left and two acute accents to the right'' are rendered in quotation marks.

Unquoted text

Placing hashes around text does nothing, it is a mechanism to allow inline attributes to be applied to otherwise unformatted text.

3.2. Paragraphs

You can indicate special information with an eye catching symbol. Please don’t overuse this (less is more).

```
[TIP]
This is a tip or idea.
```

This is a tip or idea.

You can also have multiple lines and empty lines inside the paragraph using a whitespace and a plus sign:

```
[TIP]
This is a tip or idea. +
This is another idea. +
 +
More ideas...
```

This is a tip or idea.
This is another idea.

More ideas...

```
[NOTE]
This is an information note.
```

This is an information note.

```
[IMPORTANT]
This indicates important information.
```

This indicates important information.

```
[WARNING]
This is a warning or something to pay attention to.
```

This is a warning or something to pay attention to.

```
[CAUTION]
This is something you should treat with caution.
```

This is something you should treat with caution.

```
[normal]
This is a Normal paragraph.
```

This is a Normal paragraph.

```
[literal]
This is a Literal paragraph.
```

```
This is a Literal paragraph.
```

```
[quote]
This is a Quote paragraph.
```

> This is a Quote paragraph.

```
[listing]
This is a Listing paragraph.
```

```
This is a Listing paragraph.
```

```
[abstract]
This is an Abstract paragraph.
```

This is an Abstract paragraph.

```
[comment]
This is a Comment paragraph. It does not show up in the rendered text ;-)
```

```
[example]
This is a Example paragraph.
```

This is a Example paragraph.

```
[sidebar]
This is a Sidebar paragraph.
```

This is a Sidebar paragraph.

```
[source]
This is a Source paragraph. See Code formatting for further examples.
```

```
This is a Source paragraph. See Code formatting for further examples.
```

This indicates a simple description headline

This is the text for the simple description headline

Example 3. This indicates an example inside a block

> Just a simple block example.

3.3. Keyboard shortcuts

You can also define keyboard shortcuts with the `kbd` macro.

```
kbd:[Alt+F11] - Toggle Full Screen Mode in the Eclipse IDE
```

The result is the following:

`Alt` + `F11` - Toggle Full Screen Mode in the Eclipse IDE

The result will be different depending on the rendering (PDF, HTML).

3.4. Using inline icons

You can also add the `:icons: font` directive to the top of your document, which allows you to use the icons from http://fortawesome.github.io/Font-Awesome/icons/ directory via a macro using any of the following icon sets:

- fa - https://fontawesome.com/v4.7.0/icons (default)
- fas - Font Awesome - Solid
- fab - Font Awesome - Brands
- far - Font Awesome - Regular
- fi - Foundation Icons
- pf - Payment font

The fa icon set is deprecated. Please use one of the other three FontAwesome icon sets.
when icon is not available in fa icon set but in an other, it's possible to give the correct set.

The generate Document process generate a INFOS message, if icon is available in an other set and it is not given.

```
icon:comment[set=far] This is a comment icon
icon:file[set=far] And a file icon
icon:battery-full[set=fas] And a battery icon
```

Example 4. The output looks like the following

This is a comment icon
And a file icon
And a battery icon

4. Code formatting

Asciidoc and the used highlighter provide support for code syntax highlighting of several programming languages and formats. The following are examples for code which is widely used within OFBiz.

Java source code formatting

```
[source,java]
----
public class Foo {
  public String bar;
  public String bar() {
   return bar;
  }
}
----
```

Renders to

```java
public class Foo {
  public String bar;

  public String bar() {
    return bar;
  }
}
```

Java source code formatting (numbered)

```java
1 public class Foo {
2   public String bar;
3
4   public String bar() {
5     return bar;
6   }
7 }
```

Java source code formatting (with explanation callouts)

```java
public class Foo {
  public String bar; ①

  public String bar() {
    return bar; ②
  }
}
```

① Declares the `bar` field

② Returns the `bar` value

Groovy

```
selected = UtilHttp.parseMultiFormData(parameters)
selected.each { row ->
    payment = from("Payment").where("paymentId", row.paymentId).queryOne()
    if (payment) {
        payments.add(payment)
    }
}
context.payments = payments
```

XML document

```
<author id="1">
  <personname>
    <firstname>Lazarus</firstname>
    <surname>het Draeke</surname>
  </personname>
</author>
```

Cascading Stylesheet (CSS)

```
body {
    background: transparent url(/images/ofbiz_logo.png) no-repeat scroll left top;
    color: #000;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: .75em;
    line-height: 1.5em;
    padding: 50px 0 0;
    bgcolor: #ffffcc;
}
```

Javascript

```
function msg(){
    alert("Hello OFBiz");
}
```

JSON

```
{
    "id": 1,
    "name": "A green door",
    "price": 12.50,
    "tags": ["home", "green"]
}
```

Properties files

```
foo = bar
```

SQL

```
SELECT * FROM FOO;
```

HTML

```
<!DOCTYPE doctype PUBLIC "-//w3c//dtd html 4.0 transitional//en">
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   </head>
   <body style="background-color: rgb(255, 255, 255);">
      <h1>Headline</h1>
  </body>
</html>
```

5. Importing files or file includes

You can import files via the include macro. This also works for source code includes.
It's possible to use [listing] ---- before include to have the correct style.
The following example imports a file formatted as Java source code.

```
[source,java]
----
include::resource/source.java[]
----
```

```
/*******************************************************************************
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed under the License is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * KIND, either express or implied.  See the License for the
 * specific language governing permissions and limitations
 * under the License.
 *******************************************************************************/

System.out.println("Hello");
```

5.1. Using leveloffset

Via the `leveloffset` attribute you can change the section offset, for example a `=` section will become `==` if you add the following statement `:leveloffset: 1`. Use `:leveloffset: 0` to reset the offset.

5.2. Importing files partially

It is also possible to include partial files. For this, please mark the part of the file to be included with a tag similar to the following:

article.adoc

```
# tag::tagname[]
This should be included!
# end::tagname[]
This text will not be included!
```

and include the file with the tagname in the brackets like this:

```
include::resource/article.adoc[tags=tagname]
```

The result would be

```
This should be included!
```

5.3. Importing images

You can import images with

```
image::
```

For the HTML output you can add the alt text within the brackets []. If the image is located in the images folder then the import would look like this:

```
image::OFBiz-Logo.svg[Apache OFBiz Logo, pdfwidth=3.0in, align=left]
```

Result:

Please notice that the images folder is specified in the document header configuration and is therefore not provided here again.

The import of an image with `image::` will add the image in a new line. If you want to add an image to be inline then use the macro `image:`

```
This is just a text image:OFBiz-Logo.svg[Apache OFBiz Logo, width=40%] to show inline
images.
```

Result:

This is just a text  to show inline images.

6. How to write a...

6.1. FAQ or Glossary

FAQ's and glossaries should be written as a labeled list. For hyperlinking, they should also have an ID which can be linked within a list.

For example

```
<<faq_entry_1,FAQ entry 1>>
<<faq_entry_2,FAQ entry 2>>

[#faq_entry_1]
FAQ entry 1::
This is entry #1 in our example FAQ.

[#faq_entry_2]
FAQ entry 2::
This is entry #2 in our example FAQ.
```

Renders to

FAQ entry 1
FAQ entry 2

FAQ entry 1
: This is entry #1 in our example FAQ.

FAQ entry 2
: This is entry #2 in our example FAQ.

6.1.1. Glossary

There is one Glossary per Component and a General Glossary in user-manual.

For glossary entries ID should be in Uppercase with space replace by "_"

7. Asciidoc FAQ

7.1. How to reset Heading Counter in Asciidoc

You can deactivate the counter for a section:

```
:sectnums!:

== Preface

:sectnums:

== First Chapter
```