Asciidoctor
manuale d'uso

INDICE

COS'è ASCIIDOCTOR

Asciidoctor è un linguaggio di markup che consente di creare documenti in HTML5, pdf, Docbook 5 o 4.5, ed Epub3 in modo sempolice. Per lavorare con asciidoctor serve ruby, e la relativa gem; serve inoltre un editor di testo semplice. Per produrre file in ofrmato PDF inoltre, è necessario utilizzare **asciidoctor-pdf**, che estende le funzionalità di asciidoctor, permettendo la conversione. Quando effettua la conversione di un file dal formato di asciidoctor, `.adoc` in uno dei vari formati che supporta, asciidoctor applica un proprio stile di default, personalizzabile creando file in CSS o YAML.

INSTALLARE ASCIIDOCTOR

Asciidoctor può essere installato tramite il comando `gem`, con il package manager delle varie distribuzioni di linux o via Bundler. L'installazione richiede la presenza di ruby nel sistema; asciidoctor funziona con diverse versioni di ruby, incluse le API Jruby, Rubinius e Opal.

LINUX E MAC

Ci sono diversi modi di installare asciidoctor, ma il più semplice è utilizzando il comando `gem` da riga di comando:

```
gem install asciidoctor
```

Un altro modo di installare asciidoctor è utilizzando i package manager di linux, che variano a seconda della distribuzione:

Debian o Ubuntu

```
apt-get asciidoctor
```

Fedora 21 o precedente

```
sudo yum install asciidoctor
```

Fedora 22 o successivo

```
sudo dnf install asciidoctor
```

una volta installato asciidoctor, per effettuare il render di documenti in formato PDF, è necessario scaricare asciidoctor-pdf, che è disponibile come pre-release su rubygems.org. L'installazione è possibile solo da riga di comando:

```
gem install asciidoctor-pdf --pre
```

WINDOWS

Per isntallare asciidoctor su un sistema Windows, bisogna prima installare ruby; esiste un progetto che ne rende l'installazione molto semplice, chiamato Ruby Installer. Una volta scaricato ed eseguita l'installazione, sarà disponibile una command prompt apposita, dalla quale sarà possibile scaricare ed installare la gem di asciidoctor:

```
gem install asciidoctor
```

Una volta scaricato ed installato asciidoctor, se si vuole produrre documenti in formato PDF, è necessario installare anche la gem asciidoctor-pdf, la quale è disponibile solo come pre-release:

```
gem install asciidoctor-pdf --pre
```

SOURCE HIGHLIGHTING

Qualora volessimo uno strumento per l'highlighting del codice nel documento, asciidoctor supporta CodeRay, Rouge e Pygments, i quali sono scaricabili come gem da riga di comando:

```
gem install coderay

gem install rouge

gem install pygments.rb
```

Una volta scaricati uno o più di questi strumenti, possiamo dichiarare nel nostro documento cosa utilizziamo per l'highlighting utilizzando l'attributo `:source-highlighter:` nel documento.

L'EDITOR

Quando decidiamo di lavorare con asciidoctor, una decisione importante è la scelta dell'editor di testo da utilizzare: editor di testo complessi come Word o Open office sono sconsigliati, ma allo stesso tempo su piattaforme windows è scosigliato l'uso di trumenti come wordpad o notepad. Questi editor infatti possono complicare la conversione o causare errori nell'interpretazione del file. Gli editor consigliati sono TextMate per Mac

OS X, o GEdit per i sistemi Linux, mentre per Windows è consigliato Notepad++. Alternativamente è possibile utilizzare Vim, Emacs o Sublime Text. Questi editor di testo sono consigliati perché supportano l'highlighting della sintassi di asciidoctor.

TIPOLOGIA DEL DOCUMENTO

Asciidoctor permette di scrivere documenti con diverse tipologie, che si differenziano per la struttura. Le tipologie, sono impostate dichiarando un attributo nel documento, o alla conveersione:

Articolo
: la tipologia di default, che può contenere le sezioni abstract, appendice, bibliografia, glossario ed indice, oltre al corpo dell'articolo. L'attributo che definisce l'articolo è `:article:` e può essere omesso, in quanto è la tipologia di default.

Libro
: il libro può contenere più sezioni di livello 0, ed oltre alle sezioni dell'articolo, il libro contene il colofone (note sulla produzione del libro), dedica e prefazione. L'attributo per definire un documento come libro è `:book:`.

Manuale
: il manuale, o MAN page, è utilizzato per produrre documenti simili alla documentazione dei manuali dei sistemi Unix, ha una formattazione particolare, e viene dichiarato con `:manpage:`

STRUTTURA DEL DOCUMENTO

Un documento asciidoctor è composto da diversi elementi, suddivisi in elementi block o inline. Ognuno di questi elementi ha una serie di stili, opzioni e funzioni applicabili al loro contenuto; i block element sono elementi che possono occupare più righe, e possono contenere al loro interno altri blocchi. Gli elementi inline invece effettuano delle operazioni su una parte del contenuto di un blocco. Alcuni elementi di tipo block comprendono sezioni, titpoli, header, tabelle e liste, mentre gli elementi inline comprendono quotes, macro o sostituzioni di caratteri.

QUOTES

Con quotes in asciidoctor si intende una variazione nella formattazione del testo, ad esempio per rendere una parte del testo in grassetto, o in monospace. Ci sono due tipologie di quotes: vincolati (constrained) e liberi (unconstrained).

Per quotes vincolati si intende i quotes che comprendono una o più parole nella loro

interezza, e non compaiono altri caratteri subito prima o subito dopo dei simboli che delimitano i quotes.

Venogno utilizzati con parole singole,

```
Questa macchina è *veloce*
```

con più parole,

```
Questa macchina è *davvero veloce*
```

o quando una parola è seguita da un segno di punteggiatura

```
Non ho mai guidato una macchina *così veloce*!
```

i quotes mostrati nell'esempio rendono il testo che racchiudono in grassetto. Il risultato delle frasi degli esempi è il seguente:

Questa macchina è **veloce** Questa macchina è **davvero veloce** Non ho mai guidato una macchina **così veloce**!

I quotes liberi invece servono ad evidenziare parti di una parola o più parole, e vengono usate nei seguenti casi:

- se una lettera, un numero o un underscore precedono o seguono la parte da comprendere nel quote
- se il simbolo di apertura del quote è preceduto da un punto e virgola (;)
- se ci sono degli spazi subito dopo il simbolo di apertura e subito prima il simbolo di chiusura del quote

```
La parola sc**i**enza si scrive con la *i*
```

```
Oggi è il _23_&#8722;__05__&#8722;__2016__
```

```
Ho bisogno di più `` spazio ``
```

Come mostrano gli esempi, i quotes liberi sono delimitati con due simboli invece che uno.

Un caso particolare si presenta se vogliamo alterare una o più parole che sono comprese tra i doppi apici:

```
"`@` "
"``@`` "
"```@``` "
```

Dato che i doppi apici non sono lettere, numeri o underscore, verrebbe da utilizzare un quote vincolato, ma in questo caso va utilizzato un quote libero. La terza coppia di accenti viene interpretata dal parser di asciidoctor come parte dei doppi apici. Se effettuassimo un render dell'esempio otterremmo il testo seguente:

"@" "@" "@"

ATTRIBUTI

Gli attributi sono dichiarazioni effettuate generalmente subito dopo una sezione di livello 0, e che influenzano l'intero documento dalla dichiarazione dell'attributo in poi, tramite comportamenti o stili particolari, come ad esempio la creazione di un indice, o la numerazione delle sezioni del documento. Gli attributi si dividono in 6 categorie, in base alla loro funzione:

- Attributi ambientali (?)
    - Sono attributi che asciidoctor definisce automaticamente, come la data di creazione del documento, o il percorso del file da convertire. Generalmente sono da considerare attributi di sola lettura, anche se possono essere modificati.
- Attributi integrati
    - Si tratta di attributi definibili ovunque nel documento, ad eccezione di una parte, chiamata attributi dell'header, che vanno definiti all'inizio del documento. Un attributo integrato è visibile e viene applicato solo dopo la sua definizione, e non può essere definito in più punti del documento, se non con il prefisso `@`, ad eccezione dell'attributo `sectnums` che può essere definito più volte nello stesso documento.
- Attributi predefiniti
    - Gli attributi predefiniti vengono utilizzati per sostituire alcuni caratteri se necessario.
- Attributi definiti dall'utente
    - Tutti gli attributi dichiarati e definiti dall'autore; utili per inserire rapidamente contenuto che va utilizzato più volte nel documento.
- API e attributi da riga di comando

- Attributi appartenenti alle altre categorie ma che possono essere definiti alla conversione, come ad esempio l'attributo ambientale `:backend:` che può essere definito con l'opzione `-b` da riga di comando, o un attributo che definisce la tipologia del documento, definibile con l'opzione `-d` deella riga di comando.

- Attributi degli elementi
  - Attributi definiti in un elemento come una lista o una tabella, i quali hanno validità solo per quell'elemento ed hanno la precedenza sugli attributi definiti nel documento.

ASSEGNAZIONE DEGLI ATTRIBUTI

Gli attributi hanno un ordine di interpretazione preciso:

1. Attributi impostati dall'API o dalla riga di comando
2. Attributi impostati nel documento
3. Valore di default degli attributi

È possibile gestire questo ordine in un certo senso: se ad un attributo nell'interfaccia a riga di comando viene aggiunta "`@`" alla fine, la precedenza viene assegnata all'attributo assegnato nel documento, e, qualora non sia presente o assegnato, passa di nuovo alla CLI (command line interface, interfaccia a riga di comando).

Gli attributi vanno definiti con la seguente sintassi:

```
:attributo: valore
```

Come detto in precedenza, gli attributi in asciidoctor possono richiedere che venga assegnato loro un valore, che può essere numerico, o una stringa, un percorso, un URL o riferimenti ad altri attributi. Inoltre è possibile "disattivare" un attributo impostato in precedenza, inserendo un `!` nell'attributo stesso.

```
:sectnums:
:leveloffset: 3
il valore di leveloffset è {leveloffset}
:!sectnums: :sectnums!:
:imagesdir: ./Immagini
```

Nell'esempio qui sopra vediamo un attributo che non richiede l'inserimento di valori, `:sectnums:` ed un attributo che invece richiede un valore numerico. L'attributo compreso tra parentesi graffe, `{leveloffset}` rappresenta un riferimento al valore

dell'attributo `leveloffset`. Nella penultima riga invece, sono riportati i due modi di "disattivare" l'attributo `:sectnums:`; il punto esclamativo per negare l'attributo precedentemente impostato, può essere inserito subito prima o subito dopo il nome dell'attributo stesso, il risultato non cambia. Infine, nell'ultima riga è mostrato un esempio di sintassi che descrive un percorso.

SOSTITUIZIONE DEGLI ATTRIBUTI

Una delle feature di asciidoctor è quella di poter utilizzare sostituzioni di caratteri come i caratteri speciali; queste sostituzioni sono disponibili anche negli attributi, e possono essere utilizzatio per creare del contenuto da richiamare più volte nel documento utilizzando solo il riferimento all'attributo, così da non digitarne il contenuto; le sostituzioni verranno viste più nel dettaglio in seguito, ma per ora vediamo un esempio:

```
:app-name: pass:quotes[MyApp^(C)^]
```

Nell'esempio riportato qui sopra, la macro pass applica la sostituzione, e se dovessimo fare riferimento all'attributo app-name, otterremmo questo risultato: MyApp©

ATTRIBUTI SU PIù RIGHE

In certi casi, come ad esempio la creazione di un attributo definito dall'utente per inserire automaticamente nel documento elementi lunghi come paragrafi interi o righe di codice, può essere utile dividere il contenuto dell'attributo in più righe in modod da renderlo facilmente leggibile da chi andrà a vedere il documento in formato .adoc. Un attributo del genere è definito come ogni altro attributo, ed ogni riga termina con una backslash (`\`).

```
:attributo-lungo: questo è un attributo lungo, è talmente lungo che \
per facilitare la lettura del contenuto di questo attributo molto lungo \
a chi dovesse vedere il documento non renderizzato, \
quindi il documento in formato originale, è stato diviso in più righe, \
altrimenti la sua lettura potrebbe risultare difficile.
```

LIMITI DEGLI ATTRIBUTI

Gli attributi di asciidoctor, seppur molto utili e versatili, hanno delle limitazioni riguardo al loro contenuto; e certi elementi non sono supportati all'interno dell'attributo stesso.

Cos'è supportato:

- contenuto semplice

        - un numero, una stringa, un percorso o un URL
    - riferimenti ad altri attributi
    - formattazione testuale
        - testo in **grassetto**, *corsivo* o `monospace` e sostituzione testuale
    - macro

Cosa non è supportato:

- liste
- paragrafi multipli
- tipologie di markup che necessitano di whitespace

ATTRIBUTI DEGLI ELEMENTI

È possibile assegnare ad un elemento inline o block, oppure una macro, uno o più attributi, e questo si ottiene attraverso l'uso di liste di attributi, le quali hanno la precedenza sugli attributi impostati nel documento per l'elemento specifico a cui fanno riferimento. Una lista di attributi è un insieme di attributi specifici, separati tra loro da una virgola, e compresi tra delle parentesi quadre:

```
[positional-attribute, positional-attribute, named-attribute="valore"]
```

Positional attribute: il positional attribute in un elemento inline, viene chiamato *role*, mentre in una macro e un elemento di tipo block come una tabella o un paragrafo è chiamato *style*.

Named attribute i named attribute sono attributi a cui viene assegnato, tramite l'uso di un `=` un valore compreso tra doppi apici. Un esempio di named attribute è l'attributo `cols` che indica il numero di colonne di una tabella. Per rendere un named attribute indefinito, se in precedenza era stato definito, basta assegnargli il valore `none`.

ROLE

Il role è utilizzato principalmente per l'output HTML. L'attributo role infatti, una volta effettuato il render in HTML, diventa la classe di un elemento. Per dichiarare un role ci sono 3 modi: il primo è quello di precedere il nome del role da assegnare con un `.`, il secondo è quello di utilizzare il named attribute `role`, ed il terzo, che è valido solo per gli elementi inline è quello di inserirlo per primo nella lista degli attributi di quell'elemento. Come la classe in HTML, anche il role può contenere più valori:


```
[.role1.role2.role3]<elemento generico>
[role="role1, role2, role3"]<elemento generico>
[role]<elemento inline>
[.role1.role2.role3]<elemento inline>
```

STYLE

Lo style viene utilizzato per cambiare l'aspetto o il comportamento di un intero elemento di tipo block o macro. In una lista di attributi, è il primo elemento se la lista fa riferimento ad un block o ad una macro. Ad un paragrafo ad esempio può essere assegnato l'attributo `source` per fare in modo che l'intero paragrafo venga renderizzato come un blocco di codice (come è stato fatto per tutti gli esempi di questo manuale).

ID

L'id di un elemento ha come scopo principale quello di fornire un'"ancora" per la creazione di cross reference, e nel caso l'output sia HTML, viene inserito come id dell'elemento. Oltre a questa funzione però l'id permette l'applicazione di uno stile particolare ad un elemento. L'id di un elemento è definito con un `#`, compreso come il role tra parentesi quadre. possiamo inoltre definire assieme l'id di un elemento ed il suo role:

```
[#id.role]<elemento>
```

ATTRIBUTI MANCANTI

Se viene fatto un riferimento ad un attributo che non è stato definito, asciidoctor generalmente non mostra la riga che contiene quell'attributo; tuttavia, per evidenziare questi problemi, nelle ultime release, sono stati inseriti due attributi nuovi: *attribute-missing* e *attribute-undefined*, che permettono all'utente di specificare il comportamento che deve seguire asccidoctor quando incontra attributi mancanti o non definiti.

ATTRIBUTE-MISSING

Questo attributo viene utilizzato per definire il comportamento di asciidoctor quando viene fatto un riferimento ad un attributo non esistente. L'attributo accetta 4 possibili valori: `skip`, `drop`, `drop-line` e `warn`.

- `skip`
    - l'impostazione di default, il riferimento viene mostrato così come è stato scritto;

- `drop`
    - il riferimento viene rimosso;
- `drop-line`
    - l'intera riga contenete il riferimento viene rimossa;
- `warn`
    - viene mostrato un messaggio di avviso che il riferimento manca;

Valore	Risultato
skip	Ciao, {nome}!
drop	Ciao, !
drop-line	
warn	WARNING: skipping reference to missing attribute: XYZ