Introduzione all'uso di source-higlight 2.9

Autore: Giacomo Mengucci

Versione del 25 maggio 2008

Sommario dei contenuti

A cosa serve source highlight

I programmatori hanno spesso bisogno, per scopi documentativi, di pubblicare il proprio codice con la sintassi evidenziata, in modo che, attraverso i colori o specifici formati di carattere, al lettore risulti immediato identificare ciascuna parte del codice (istruzioni, operatori, operandi, funzioni, classi, istanze ad una classe, commenti, stringhe, numeri ecc).

Il documento con la sintassi evidenziata deve poter essere generato in vari formati: HTML, LATEX, DocBook TEXINFO.

Lo scopo di source-higlight è proprio quello di generare, partendo dal file che contiene il codice sorgente, uno di questi documenti.

Source-higlight è in grado di trattare file sorgenti relativi a molteplici linguaggi di programmazione ed è in grado di generare documenti con la sintassi evidenziata in molteplici formati di output.

Conoscere i linguaggi di programmazione supportati

Per conoscere quali sono i linguaggi di programmazione supportati è sufficiente eseguire il seguente comando:

```
source-highlight --lang-list
```

Il comando restituirà la lista dei linguaggi supportati (uno per riga) nel seguente formato:

```
python = python.lang
```

La parte a destra del carattere = indica il linguaggio supportato, nel senso che source highlight è in grado di trattare codici sorgenti scritti in quel linguaggio.

La parte a sinistra indica l'estensione dei file contenenti il codice, nel senso che i file aventi quell'estensione saranno considerati come file contenenti del codice nel linguaggio specificato nella parte destra.

Conoscere i formati di output supportati

Per conoscere quali sono i formati di documento che possono essere generati è sufficiente eseguire il seguente comando:

```
source-highlight --outlang-list
```

La logica del funzionamento di source-highlight

Il funzionamento di source-highlight, da un punto di vista logico, può essere sintetizzato in tre passi fondamentali:

1. L'analisi del file che contiene il codice sorgente per identificarne le varie parti semantiche;
2. La definizione dei colori o dei formati di carattere da utilizzare per evidenziare, nel documento di output, le parti semantiche in precedenza individuate;

3. La generazione del documento di output nel formato specificato (HTML, Latex ecc.); a tal proposito source highlight è in grado di generare le strutture sintattiche tipiche del formato di output prescelto (marcatori HTML, marcatori Latex ecc.), in modo che le parti semantiche del codice sorgente siano evidenziate, in fase di visualizzazione del documento generato, con i colori e i formati di carattere in precedenza definiti.

L'individuazione delle strutture semantiche del linguaggio

L'individuazione delle parti di codice che corrispondono ai vari elementi semantici del linguaggio di programmazione è effettuata per mezzo delle espressioni regolari.

Infatti, per ogni linguaggio di programmazione supportato, esiste un file con estensione `.lang` in cui sono classificati tutti gli elementi semantici del linguaggio che source highlight è in grado di individuare.

In corrispondenza di ciascuna classificazione è indicata un espressione regolare che ha lo scopo di catturare, nel codice sorgente dato in input, la parte di testo che corrisponde all'elemento semantico a cui la classificazione si riferisce; ciò è possibile perché ogni elemento semantico è caratterizzato, nel linguaggio considerato, da una specifica sintassi e l'espressione regolare, riferendosi a quella sintassi, è costruita in modo da “catturare” la parte di codice che corrisponde a quella determinato elemento.

La gran parte dei linguaggi di programmazione hanno i medesimi elementi semantici: stringhe, numeri, nomi di variabili, nomi di funzioni, commenti, nomi di classi, nomi di istanze alle classi ecc.; ciò che diversifica ciascun linguaggio è la sintassi con cui tali elementi sono rappresentati nel codice.

L'output del comando `source-highlight --lang-list` consente di conoscere quale sia, per i vari linguaggi supportati, il file che contiene la classificazione degli elementi semantici del linguaggio e le corrispondenti espressioni regolari che sono in grado di catturarli nel codice sorgente:

```
python = python.lang
```

Questo esempio si riferisce al linguaggio python; il file che contiene le classificazioni e le relative espressioni regolari è `python.lang` che si trova, insieme a tutti gli altri file, nella directory `/usr/local/share/source-high-light`[1]

Specificare quale file `.lang` utilizzare

Il linguaggio di programmazione da considerare, e di conseguenza il relativo file `.lang` da utilizzare, può essere specificato con l'opzione `--src-lang=<codice linguaggio>`.

Il `<codice linguaggio>` è la stringa che sta a sinistra del carattere = nell'output del comando `source-highlight --lang-list`.

Con il comando:

1 Ovviamente se l'installazione è avvenuta con prefisso /usr, la directory da considerare è `/usr/share/source-highlight/`.

```
source-highlight --input=”myprogram.py” --output=”myprogram.xhtml” --src-lang=python --out-format=xhtml
```

si specifica che Il programma è contenuto nel file “myprogram.py” (--input), che è scritto nel linguaggio python (--src-lang), che dovrà essere generato un documento xhtml (--out-format) ed in fine che il documento generato dovrà essere memorizzato nel file “myprogram.xhtml” (--output).

Inoltre source highlight ha un meccanismo di selezione automatica del corretto file .lang, in relazione all'estensione del file che contiene il codice sorgente da processare.

Nell'esempio, precedente, in realtà non sarebbe stato necessario specificare in modo espresso (con l'opzione --src-lang) quale file .lang utilizzare, in quanto source highlight l'avrebbe rilevato automaticamente grazie all'estensione .py del file sorgente indicato nell'opzione --input.

Esiste infatti il file di configurazione lang.map che contiene tutte le corrispondenze fra estensioni e file .lang da utilizzare; così, ad esempio, l'estensione .py e .python sono associate al file python.lang.

L'utente può anche creare la propria mappa personalizzata, specificando nuove ed ulteriori estensioni da accoppiare ai vari file .lang; in tal caso è necessario utilizzare l'opzione --lang-map per indicare quale mappa utilizzare.

```
source-highlight --input=”myprogram.py” --output=”myprogram.xhtml”--out-format=xhtml --lang-map="personal-lang.map"
```

Le principali classificazioni degli elementi semantici

Di seguito un elenco, non esaustivo, delle principali classificazioni di elementi semantici che di norma sono definite per la gran parte dei linguaggi di programmazione:

Classificazione	Significato
keyword	Parole chiavi del linguaggio
comment	Commenti del programmatore
string	Stringhe di testo fisse
variable	Nomi che identificano variabili od oggetti istanza di una classe
preproc	Direttive al preprocessore o direttive di importazione (esistono solo in alcuni linguaggi di programmazione)
function	Nomi che identificano le funzioni (sia nella sintassi di dichiarazione o definizione, che nella sintassi di chiamata)
number	Numeri
normal	Testo normale, cioè che non corrisponde a nessun elemento semantico del linguaggio considerato.

Per vedere quali elementi semantici sono “catturati” da source-highlight per un determinato linguaggio di programmazione, si può utilizzare l'opzione `--show-lang-elements`.

```
Source-highlight --show-lang-elements=python.lang
```

Come si può vedere, l'opzione richiede che sia specificato il file `.lang` del linguaggio considerato.

Definizione dei vari stili di visualizzazione

Il secondo passo fondamentale è la definizione dei formati con cui devono essere evidenziate le parti di codice che corrispondono ai vari elementi semantici del linguaggio utilizzato.

Questi formati di visualizzazione saranno poi utilizzati durante la fase di generazione del documento HTML, Latex ecc. secondo la scelta dell'utente.

La definizione dei formati di visualizzazione consiste, in sintesi, nell'accoppiare alle varie classificazioni di ciascun elemento semantico gestito, un codice di colore e/o un codice di stile.

Questi accoppiamenti vengono realizzati attraverso apposite direttive da inserire in uno o più file di stile che hanno come estensione `.style`.

Ad esempio se si vuole che tutte le parole chiavi siano visualizzate con il colore rosso ed in grassetto è sufficiente specificare la seguente direttiva:

```
keyword red b;
```

Il primo campo della direttiva è l'elemento semantico da evidenziare, il secondo campo è il colore da assegnare all'elemento semantico ed in fine il terzo campo è lo stile del carattere.

Per ogni elemento semantico che si vuole evidenziare è quindi necessario definire una direttiva simile alla precedente.

La specificazione dello stile di caratteri (terzo campo) è facoltativa, mentre se si volesse specificare solamente lo stile di carattere senza colore, è sufficiente indicare il codice di stile come secondo campo senza specificare alcun colore; è anche possibile specificare più di un codice di stile ed in tal caso l'uno andrà separato dall'altro con una virgola (senza spezi intermedi).

```
comment i,b;
```

In questa direttiva all'elemento `comment` non è accoppiato alcun codice di colore, ma sono accoppiati due codice di stile, per cui tutti i commenti saranno visualizzati in corsivo grassetto.

Può anche capitare che uno stile di visualizzazione è comune a più elementi semantici; in tal caso è sufficiente indicare nel primo campo tutti gli elementi semantici da considerare, l'uno separato dall'altro da una virgola (senza spazi intermedi).

```
preproc,keyword yellow b;
```

In questo esempio le direttive al preprocessore e le parole chiavi del linguaggio sono evidenziate in giallo ed in stile grassetto.

Le direttive di formattazione possono essere utilizzate anche per impostare il colore dello sfondo dell'intero documento; a tal scopo è sufficiente utilizzare nel primo campo lo pseudo elemento `bgcolor`.

```
bgcolor black;
```

I codice di formato supportati

I colori supportati sono i seguenti:

```
black
red
darkred
brown
yellow
cyan
blue
pink
purple
orange
brightorange
green
brightgreen
darkgreen
teal
gray
darkblue
```

Quando in output si genera un documento HTML o XHTML, oltre ai colori indicati, si hanno anche a disposizione tutti i colori che è possibile generare con i codici tipicamente utilizzati in questi linguaggi di marcatura, come ad esempio "#769796".

Gli stili di carattere supportati sono i seguenti:

`b` = stile grassetto

`i` = stile corsivo

`u` = stile sottolineato

`f` = caratteri a dimensione fissa

Impostazioni di stile predefinite e personalizzate

I vari linguaggi di programmazione hanno molti elementi semantici in comune e quindi di default viene fornito un file che contiene delle direttive già predefinite (il file `default.style`); nulla vieta, però, che per ogni linguaggio possa essere creato uno specifico file di stile, come ad esempio `python.style`.

L'opzione `--style-file` consente di specificare quale file di stile utilizzare:

```
source-highlight --input=”myprogram.py” --output=”myprogram.xhtml” --src-
lang=python --out-format=xhtml --style-file="python.style"
```

In questo esempio, anziché utilizzare il file di stile di default, viene specificato, grazie all'opzione citata, un file di stile appositamente creato dall'utente, in cui sono state inserite le proprie direttive di formato.

Utilizzare CSS per specificare i formati

Gli stili di visualizzazione, oltre che per mezzo delle direttive di formato contenute nei file `.style`, possono essere definiti anche attraverso i fogli di stile CSS (Cascading Style Sheet).

In realtà source highlight è in grado di comprendere solo un sottoinsieme minimo delle innumerevoli proprietà esistente in CSS.

Il foglio di stile deve essere memorizzato in un file e richiamato attraverso l'opzione `--style-css-file`.

```
source-highlight --input=”myprogram.py” --output=”myprogram.xhtml” --src-
lang=python --out-format=xhtml --style-css-file="python.css"
```

In questo esempio i formati da utilizzare per generare il documento XHTML sono definiti per mezzo dei fogli di stile nel file che `python.css`.

Di seguito è riportato un esempio di foglio di stile correttamente supportato da source highlight.

```
/* Stili per l'intero corpo del documento */
body {
      background-color: black; /* Sfondo nero */
      color: white; /* colre di default del testo */
      font-family: monospace; /*font di default del testo */
      font-style: normal; }
/* Stili per i commenti (nessun colore solo corsivo grassetto) */
.comment {
      font-weight: bold;
      font-style: italic; }
/* Stili per le paroli chiavi e le direttive di importazione */
.keyword, .preproc {
      color: yellow;
      font-weight: bold; }
```

Le cose principali da notare sono le seguenti:

- I commenti sono supportati
- Gli elementi semantici da evidenziare sono tradotti in altrettanti selettori CSS di tipo classe (il punto davanti ad ogni elemento è quindi essenziale)
- Nel selettore `body` (di tipo tag) è impostato il colore dello sfondo e gli altri formati da applicare se non diversamente disposto dai selettori di tipo classe relativi ai diversi elementi semantici trattati

Le proprietà CSS riconosciute da source highlight sono le seguenti:

`background-color`, `font-family`, `font-weight`, `font-style`, `text-decoration`, `color`

Il formato del documento da generare in output

In questa fase source highlight genera il documento di output in relazione al formato selezionato, tenendo conto degli elementi semantici individuati nel file sorgente e tenendo conto dei formati di evidenziazione specificati per tali elementi.

Ogni formato supportato implica la generazione di un codice che rispetti la sintassi del linguaggio di marcatura che sta alla base del formato selezionato.

Ad esempio, il linguaggio di marcatura HTML o XHTML è quello utilizzato per creare le pagine WEB; tale linguaggio ha le proprie regole sintattiche che, in qualche modo, devono essere acquisite e conosciute da source highlight se si vuole che esso generi il relativo codice per evidenziare correttamente gli elementi semantici del file sorgente.

A questo scopo vengono in aiuto i file `.outlang`, i quali specificano come deve essere generato il documento in relazione al formato di output selezionato ed in relazione alle evidenziazioni in precedenza scelte.

Il formato del documento da generare in output è scelto con l'opzione `--out-format=<codice formato>`.

`<codice formato>` è il codice che si può vedere nella parte sinistra del carattere = dell'output del comando `source-highlight --outlang-list`.

Ad esempio, relativamente ai linguaggi di marcatura HTML, XHTML e Latex fra l'altro, si otterrà sicuramente anche il seguente output:

```
html = html.outlang
xhtml = xhtml.outlang
latex = latex.outlang
```

Ciascuno dei codici a sinistra del carattere = può essere utilizzato come argomento dell'opzione `--out-format`.

```
source-highlight --input=”myprogram.py” --output=”myprogram.xhtml” --src-
lang=python --out-format=xhtml –style-file="python.style"
```

In realtà source highlight è in grado di riconoscere automaticamente il tipo di formato da utilizzare per la generazione del documento di output; ciò è possibile grazie all'estensione del nome del file che viene passato come argomento dell'opzione `--output`.

Nell'esempio precedente, il nome del file di output, `myprogram.xhtml`, ha estensione `.xhtml` e quindi source highlight avrebbe automaticamente selezionato il formato XHTML, anche se fosse stata omessa l'opzione `--out-format`.

Generare in output solo la parte relativa al codice sorgente oppure un documento finale validamente formato.

Molti dei formati di output che source highlight è in grado di trattare hanno una propria struttura logica, nel senso che i relativi documenti, per poter essere considerati come validamente formati, devono contenere tutte le sezioni logiche che la sintassi del formato prevede.

Ad esempio, il formato HTML o XHTML è caratterizzato da una sezione di intestazione (marcata dai tag `<head>` e `</head>`) e da un corpo del testo (marcato dai tag `<body>` e `</body>`); la prima sezione contiene una serie di informazioni relative al documento (titolo, set di caratteri utilizzato, collegamenti a fogli di stile esterni ed altro ancora), mentre la seconda sezione fa riferimento al contenuto specifico del documento (nel caso di source highlight il contenuto è il testo del codice sorgente).

Source highlight può operare in due modalità; con la prima modalità, quella predefinita, sono generati solo i marcatori relativi al codice sorgente, ma non sono invece generati i marcatori che si riferiscono alla struttura logica del formato considerato; ad esempio, per il formato HTML o XHTML, non sono generati i tag di struttura come `<head> </head>` e `<body> </body>`, come non sono generati i marcatori contenuti all'interno dell'intestazione (`<title> </title>`, `<meta /> <link />` ecc.)

La ragione per cui la modalità predefinita non genera un documento validamente formato è che in tal caso l'output di source highlight può essere facilmente utilizzato come input di altri comandi o programmi che, magari, generano le parti mancanti del documento in maniera più flessibile o adatta alle esigenze particolari della situazione specifica.

La seconda modalità con cui source highlight può operare è la generazione di un documento validamente formato; in tal caso sono generati anche i marcatori che caratterizzano la struttura logica del tipo di formato: per HTML o XHTML è generata l'intestazione e i suoi elementi costitutivi (`<head> <title> <meta> <link>` ecc.).

La modalità utilizzata da source highlight dipende dal tipo di file `.outlang` selezionato per mezzo dell'opzione `--out-format`.

Esaminando l'output del comando `source-highlight --outlang-list`, si potrà notare che, per la gran parte dei formati supportati, sono riportati più codici identificativi; in particolare, insieme al codice che identifica il formato di output, ve ne sono altri in cui lo stesso codice è seguito dal suffisso `-doc`.

Ad esempio per il formato HTML, XHTML e Latex si potrà notare la presenza dei seguenti codici (la parte a sinistra del carattere =):

```
html = html.outlang
html-doc = htmldoc.outlang
xhtml = xhtml.outlang
xhtml-doc = xhtmldoc.outlang
latex = latex.outlang
latex-doc = latexdoc.outlang
```

I codici con suffisso `--doc` generano documenti formati validamente, cioè generano, oltre ai marcatori per evidenziare il sorgente di input, anche i marcatori relativi alle sezioni logiche che caratterizzano la struttura del formato scelto, consentendo così di utilizzare in modo autonomo il documento prodotto.

I codici senza suffisso `--doc` (modalità predefinita), si limitano invece a generare solo i marcatori relativi all'evidenziazione del codice sorgente; un documento così prodotto non può essere utilizzato autonomamente, in quanto manca di alcune componenti fondamentali che caratterizzano la struttura logica del formato scelto.

```
source-highlight --input=”myprogram.py” --output=”myprogram.xhtml” --src-
lang=python --out-format=xhtml-doc
```

In questo esempio viene generato un documento XHTML validamente formato per il codice sorgente contenuto in `myprogram.py`.

In alternativa si potrebbe usare anche il seguente comando, che produce esattamente gli stessi effetti:

```
source-highlight --input=”myprogram.py” --output=”myprogram.xhtml” --src-
lang=python --doc
```

Come si può notare manca l'opzione `--out-format` ed è invece presente l'opzione `--doc`; source highlight, grazie all'estensione del nome del file passato come argomento dell'opzione `--output`, è in grado di riconoscere in automatico il formato da utilizzare (in tal caso XHTML), mentre l'utilizzo dell'opzione `--doc`, consente di generare un documento validamente formato senza utilizzare l'opzione `--out-format`.

Per quanto riguarda i documenti HTML o XHTML validamente formati, source highlight genera la seguente intestazione:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="GENERATOR" content="GNU source-highlight 2.9
by Lorenzo Bettini
http://www.lorenzobettini.it
http://www.gnu.org/software/src-highlite">
<title>pyjoin.py</title>
</head>
```

La prima cosa importante da notare è la meta informazione `charset=iso-8859-1`, la quale serve per comunicare al browser il set di caratteri utilizzato per creare il documento HTML.

A tal proposito, source highlight genera il documento HTML utilizzando lo stesso set di caratteri con cui l'utente ha creato il file che contiene il codice sorgente.

Il problema nasce dal fatto che la meta informazione `charset` è inserita in modo statico ed è sempre `charset=iso-8859-1`, anche se, ad esempio, il documento HTML è in realtà generato con il set di caratteri `utf-8`.

Tenuto conto di ciò, se l'utente è abituato ad utilizzare il set di caratteri utf-8, conviene modificare manualmente i file `htmldoc.outlang`, `htmlcss.outlang`, `xhtmldoc.outlang`, `xhtmlcss.outlang` e sostituire `iso-8859-1` con `utf-8`[2].

Nell'intestazione di un file HTML sono presenti anche i tag `<title>` e `</title>` che servono per marcare il titolo assegnato al documento HTML; source highlight in automatico assegna come titolo del documento, il nome del file che contiene il codice sorgente (argomento dell'opzione `--input`).

Con l'opzione `--title` l'utente può assegnare un proprio titolo che sarà inserito all'interno dei tag `<title>` e `</title>` (l'uso dell'opzione `--title` implica la generazione di un documento validamente formato).

```
source-highlight --input=”myprogram.py” --output=”myprogram.xhtml” --src-
lang=python --doc --title="Il mio programma"
```

La generazione di documenti HTML o XHTML secondo la DTD strict

Source highlight è particolarmente utile per la generazione di documenti HTML e XHTML.

Da questo punto in poi, quando si farà riferimento all'output HTML, implicitamente si farà riferimento anche all'output XHTML, a meno che non sia diversamente specificato.

Si è già visto che un output HTML può essere generato, o solo con i codici di marcatura relativi alle parti del codice sorgente, oppure anche con i codici di marcatura che definiscono le parti strutturali di un documento HTML validamente formato.

In entrambi i casi, source highlight utilizza i seguenti marcatori HTML per evidenziare le varie parti del codice sorgente:

- per evidenziare con il colore: **** *parte di codice sorgente* ****
- per evidenziare con il grassetto: *****parte di codice sorgente*****
- per evidenziare con il corsivo: *****parte di codice sorgente*****
- per evidenziare con il sottolineato: **<u>***parte di codice sorgente***</u>**

La *parte di codice sorgente* è la parte che corrisponde ad una struttura semantica specifica del linguaggio di programmazione utilizzato, individuata da source highlight grazie ai file `.lang`.

2 Se la directory di installazione è `/usr/local`, i file si trovano in `/usr/local/share/source-highlight/`

I colori e gli stili di carattere con cui la parte semantica estratta da source highlight deve essere evidenziata sono specificati nei file `.style`.

Quando source highlight conosce quali parti semantiche evidenziare o come evidenziarle, genera i marcatori HTML come sopra indicati grazie alle descrizioni contenute nei file `.outlang`.

Ad esempio, se si è specificato, nel file `.style`, che le parole chiavi (`keyword`) del linguaggio python debbano essere evidenziate in blu ed in grassetto, ogni volta che source highlight, grazie ai file `.lang`, individua una parte di codice sorgente che può essere classificata come `keyword`, genera il seguente codice di marcatura:

```
<b><font color="#0000FF">print</font></b>
```

In tal caso, la parola `print`, è classificata come `keyword` nel file `python.lang`.

I tag di marcatura sopra indicati, sebbene siano riconosciuti da tutti i browser, non sono pero conformi allo standard W3C per la generazione di codice HTML secondo la DTD strict.

La DTD strict raccomanda, infatti, che tutti gli stili di rappresentazione, compresi i colori, non siano specificati direttamente con i tag HTML, ma piuttosto siano definiti attraverso specifici fogli di stile di tipo CSS (Cascading Style Sheet).

Per questa ragione, source highlight è in grado di generare codice HTML idoneo a riferirsi a stili di evidenziazione definiti in un separato foglio di stile CSS.

L'utente ha il compito di definire un foglio di stile CSS, con il quale siano specificate le evidenziazioni da applicare alle diverse tipologie di parti semantiche che source highlight è in grado di estrarre dal codice sorgente.

Ad esempio si può definire un foglio di stile per il linguaggio Python, in un file di nome `python.css`, il quale può avere il seguente contenuto:

```
/* Stili per l'intero corpo del documento */
body {
      background-color: black; /* Sfondo nero */
      color: white; /* colre del testo bianco */
      font-family: monospace;
      font-style: normal; }
/* Stili per i commenti (nessun colore solo corsivo grassetto) */
.comment {
      font-weight: bold;
      font-style: italic; }
/* Stili per le paroli chiavi e le direttive di importazione */
.keyword, .preproc {
      color: yellow;
      font-weight: bold; }
```

In tal caso vengo definiti specifici stili di evidenziazione per le parti semantiche di tipo `comment`, `keyword` e `preproc`, mentre per tutte le altre parti semantiche si applicheranno gli stili definiti per il tag <body>.[3]

Da notare come ciascuna classificazione semantica, per la quale si ha la necessità di definire uno specifico stile di evidenziazione, corrisponda ad un selettore di tipo classe nel foglio di stile CSS.

Il comando per utilizzare la modalità CSS è il seguente:

```
source-highlight --input=”myprogram.py” --output=”myprogram.html” --out-format=”html” --css="python.css"
```

L'opzione **`--out-format`**=”`html`” specifica di generare un documento HTML.

L'opzione **`--css`**=”`python.css`” specifica di generare codice HTML che utilizzi, per gli stili di evidenziazione, un foglio di stile esterno contenuto nel file `python.css`.

Se con l'opzione `--output` viene specificata anche l'estensione `.html`, si può fare a meno di utilizzare l'opzione `--out-format` ed usare solamente l'opzione `--css`.

Quando si utilizza la modalità CSS i codici di marcatura generati sono i seguenti per ciascuna parte semantica estratta da source highlight nel codice sorgente:

```
<span class=”tipologia semantica”>parte semantica estratta</span>
```

Il tag `span` con l'attributo `class` è l'elemento di collegamento fra parte semantica estratta e stili di rappresentazione definiti nel foglio di stile CSS.

Il valore dell'attributo `class`, infatti, corrisponde da un lato al tipo semantico estratto da source highlight e dall'altro lato ad un selettore di tipo classe definito nel foglio di stile.

3 I fogli CSS sono a cascata, per cui un determinato elemento eredità gli stili assegnati all'elemento padre, salvo che non vi sia una assegnazione specifica applicabile in modo diretto all'elemento considerato.

Se ad esempio source highlight estrae la parola chiave (`keyword`) `print` e nel foglio di stile CSS è stato specificato un selettore di tipo classe `.keyword`, il marcatore in concreto generato è il seguente:

```
<span class=”keyword”>print</span>
```

Quando si utilizza la modalità CSS, source highlight genera in automatico un documento validamente formato; l'intestazione generata in tal caso ha alcune differenze interessanti:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
    "http://www.w3.org/TR/REC-html40/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta name="GENERATOR" content="GNU source-highlight 2.9
by Lorenzo Bettini
http://www.lorenzobettini.it
http://www.gnu.org/software/src-highlite">
<title>PickTest.py</title>
<link rel="stylesheet" href="python.css" type="text/css">
</head>
```

La prima differenza importante è la presenza del tag `<link>` con cui viene indicato quale file CSS utilizzare.

```
<link rel=”stylesheet” href=”python.css” type=”text/css”>
```

Si può notare che l'argomento passato con l'opzione `--css` è riportato nell'attributo `href` del tag `<link>`.

Da notare anche la differente `Doctype` con cui viene appunto specificata la rispondenza alla DTD strict del documento HTML generato.

Licenza d'uso