Ciao a tutti,

durante le mie frequenti divagazioni nel web, avevo letto di un modo per ospitare siti web statici su github .
Durante questo mio articolo daro' una personalissima risposta alle domande:
Perche' creare un sito statico quando tutto il mondo si evolve dinamicamente?
Perche' usare un generatore di siti statici?
Perche' usare Pelican e non altro?
e poi vi mostrero' un modo semplice e indolore per provare e sperimentare su Pelican in locale.

Perche' creare un sito statico quando tutto il mondo si evolve dinamicamente?

Non c'e' una risposta chiara, esaustiva e valevole per tutti, e' solo una questione di preferenze, ecco le mie:

- per scrivere e postare un articolo preferisco usare un editor di testo e poi pushare online le modifiche
- un sito statico e' molto piu' sicuro di uno dinamico e dunque non richiede particolare manutenzione
- non ho bisogno di un pannello di amministrazione perche' sono figo :P
- posso utilizzare un comodi strumenti per il backup e history del sito (git)

Perche' usare un generatore di siti statici?

perche' va bene avere il controllo di tutto, ma anche la pigrizia vuole la sua parte. E pigrizia vuole che venga generato auto-magicamente un intero sito a partire da pagine scritte in modo umano (Markdown).

Perche' usare Pelican e non un altro?

Come si legge, scorrendo un po' questa pagina , github consiglia Jekyll , un generatore scritto in ruby; io invece ho scelto, per le mie prove,
Pelican
. Perche'? perche' e' interamente scritto in python, e python e' simpatico, amichevole e installato su quasi tutte le distro linux, ruby invece no.

Posso testare Pelican senza scombussolare l'intero sistema?

Certamente, tutto il mondo python puo' essere testato in locale semplicemente creando una cartella e installandovi dentro un virtualenv - un ambiente virtuale in cui installare python e tutte le librerie che si desiderano. In questo modo le librerie di sistema non vengono toccate e tutto il mondo e' racchiuso in una cartella. Comodo?
Vi serve solo che sia installato il pacchetto
python-virtualenv

Iniziamo, aprite un emulatore di terminale, posizionatevi dove volete e scrivete

```
virtualenv --no-site-packages pelican  cd pelican  source ./bin/activate
```

* l'ultimo comando fa si che le variabili di ambiente del virtualenv vengano esportate nella vostra corrente shell, molto comodo per evitare di ripetere path relativi prima di ogni comando che lanceremo.

Adesso che il virtual environment e' stato generato ed attivato nella corrente shell, e' possibile installare il pacchetto pelican.

```
pip install pelican
```

* Nota bene: se non avessimo attivato la virtualenv e fossimo sotto l'utente root, queste librerie verrebbero installate nel sistema, andando a rompere l'aurea purezza della combo debian & dpkg

Tutte le dipendenze verranno scaricate e installate:

- feedgenerator, to generate the Atom feeds
- jinja2, for templating support
- pygments, for syntax highlighting
- docutils, for supporting reStructuredText as an input format
- pytz, for timezone definitions
- blinker, an object-to-object and broadcast signaling system
- unidecode, for ASCII transliterations of Unicode text
- six, for Python 2 and 3 compatibility utilities
- MarkupSafe, for a markup safe string implementation

opzionalmente, se si vuole utilizzare la sintassi di Markdown - io voglio - installate anche questo pacchetto

```
pip install Markdown
```

Tutto pronto, lasciamo a Pelican il compito di creare lo scheletro del nostro progetto. Per far cio' non dovremmo far altro che lanciare il comando e rispondere alle domande che ci verranno poste

```
pelican-quickstart
```

Adesso dovresto poter vedere la struttura del progetto

```
yourproject/
├── content
│   └── (pages)
├── output
├── develop_server.sh
├── fabfile.py
├── Makefile
├── pelicanconf.py       # Main settings file
└── publishconf.py       # Settings to use when ready to publish
```

supponendo che make sia disponibile nella vostra linux-box, lanciate il comando
make devserver

dovreste ottenere questo risultato

```
  (pelican)allanon@serin [14:38:26] [~VIRTUAL_ENV/provola]   -> % -> writing
/home/allanon/progetti/virtualenvs/pelican/provola/output/index.html  -> writing
/home/allanon/progetti/virtualenvs/pelican/provola/output/tags.html  -> writing
/home/allanon/progetti/virtualenvs/pelican/provola/output/categories.html  -> writing
/home/allanon/progetti/virtualenvs/pelican/provola/output/authors.html  -> writing
/home/allanon/progetti/virtualenvs/pelican/provola/output/archives.html  Done: Processed 0
articles and 0 pages in 0.37 seconds.
```

Iniziamo a popolare il sito! le pagine vanno collocate all'interno della sottocartella content.
Eventuali sottocartelle verranno interpretare come Categorie.
Scrivere i vostri file di prova, oppure copiateli da quanto scritto sotto, e poi connettetevi all'indirizzo http://localhost:8000

-> % cat content/pages/about.md
Title: About me

Io sono io e questa e' una prova

-> % cat content/provaCategoria/primopost.md
Title: Primo Articolo di prova
Date: 2014-04-28
Tags: prova
Slug: primo-post
Author: allanon
Summary: solo una prova

Nulla di importante

-> % cat content/provaCategoria/secondopost.md

Title: Secondo Articolo di Prova
Date: 2014-04-28
Tags: prova
Slug: secondo-post
Author: allanon
Summary: esempio di link tra pagin

Qui c'e' qualcosa

[Un esempio di link tra pagine]({filename}/provaCategoria/primopost.md)

Il risultato dovrebbe essere simile a questo http://www.alaimo.org/pelican/

Qui finisce la nostra breve introduzione a Pelican.
Feed, template, plugin, e altre sorprese sono contenute nella documentazione ufficiale, ed aspettano solo di essere scoperte,
Enjoy!