Glyph Handbook

by *Fabio Cevasco*

February 2010

Chapter I – Table of Contents

Chapter II – Introduction

Glyph is a *Rapid Document Authoring Framework*.

Think of it like a sort of Ruby on Rails but for creating text documents instead of web sites. With Glyph, you can manage your documents tidily in *projects* that can be used to generate files in a variety of output formats, such as HTML or PDF.

Glyph was created to provide an easy way to produce printable documents in HTML and PDF format, using technologies familiar to web developers like HTML and CSS, without using them explicitly. In this very specific context, you could consider Glyph as a possible alternative to LaTeX, bearing in mind that it offers far less features and it is not a typesetting system.

2.1 Rationale

At the time of writing, the most common ways to create a printable document like, say, a PDF file, are the following:

- Use a word processor like Microsoft Word convert it to PDF, for example using Adobe Acrobat — Perhaps the most user-friendly method, but relying on proprietary technologies and GUIs.
- Use LaTeX — It can be used to produce amazing documents, but it is not exactly the most user-friendly option: you have to learn LaTeX first, and you may not want to.
- Use HTML and a 3rd party renderer like PrinceXML.

The third option is arguably the best, because:

- It allows web developers to use what they know best (HTML and CSS).
- It does not rely necessarily on GUIs, thereby giving the author more control on the document formatting and its structure.

The main problem with this method is that writing HTML code can be tedious, but luckily there are plenty of lightweight markup languages like Textile or Markdown that can make writing HTML a fun task. Unfortunately though, a markup language alone is not enough, because:

- It is not a complete alternative to HTML: you have to fallback to HTML to do certain things like the document *head* and *body* tags, and often block-level elements like *div* or *table* are not supported at all.
- They cannot be easily extended to provide additional functionalities, like creating a TOC automatically, for example.

That's where Glyph comes in: Glyph uses a simple but effective macro language to handle those tasks that are not commonly managed by lightweight markup languages, such as:

- Creating TOC, index and bibliography automatically
- Defining the document structure
- Managing text snippets, notes, ...
- Validating links
- ...

Chapter III – Quick Start

Want to start writing immediately? This short chapter shows you how to get started with glyph in no time. To install Glyph, simply run `gem install glyph`, like with any other Ruby gem. Then, create a new directory and initialize a new Glyph project, like so:

```
mkdir my_glyph_project
cd my_glyph_project
glyph init
```

That's it. The `my_glyph_project` directory contains your brand new Glyph project. If you open the `document.glyph` file with your favorite text editor, you can start writing your book or article right away.

Glyph's dependencies

Glyph requires the following gems:

- extlib
- gli
- treetop
- rake

Additionally, some Glyph macros may require additional gems, such as:

- RedCloth (*textile* macro)
- Maruku *or* Kramdown *or* BlueCloth (*markdown* macro)
- Haml (if you want to load .sass files with the *style* macro)

3.1 Editing text files

...

3.2 Compiling your project

...

Chapter IV – Glyph Language

Glyph uses a simple macro language to perform advanced tasks like creating and validating links, generating Table of Contents and Indexes, and defining the document structure.

Every Glyph project contains a `document.glyph` file that is typically used to define the document structure, like the following:

```
document[
  head[
    style[default.css]
  ]
  body[
    titlepage[
      title[]
      author[]
      pubdate[]
    ]
    toc[]
    preface[
      header[Preface]
      @[preface.textile]
    ]
    chapter[
      header[Chapter #1]
      @[chapter1.textile]
    ]
    chapter[
      header[Chapter #2]
      @[chapter2.textile]
    ]
  ]
]
```

Even without knowing anything about Glyph Language, you can easily figure out nearly everything: there's a `document` with a `head` and a `body`, the `body` contains a title page, a Table of Contents, a Preface and some chapters.

4.1 Introducing Glyph Macros

By now you probably figured out what a macro looks like: it's an identifier of some kind that wraps a value within square brackets. More specifically:

- The macro identifier can contain *any* character except for: `[`, `]`, `\`, `|` or spaces.

Chapter V – Command Reference

Chapter VI – Macro Reference