

POD, Un moyen simple pour tout faire simplement ?

Le POD, à la base fut créé pour écrire la documentation d'un programme en Perl au sein même du code.

Il existait déjà des passerelles mais il en manquait au moins une, celle vers le format ODF, c'est ce travail qui a mené à cet article

1. Introduction

Le POD pouvait déjà se traduire par des passerelles vers du Texte (pod2text), du Man (pod2man), du Latex (pod2latex) , du HTML (plusieurs modules), SGML (Pod::DocBook), PDF (POD::PDF). Rappelons aussi que le POD peut aussi s'afficher directement par l'utilitaire tkpod.

Fournir le moyen d'aller vers le format ODF ouvrirait également d'autres perspectives telles que traduire le POD pour faire des articles pour GMLF, générer la présentation associée sans passer par un générateur spécialisé et pouvoir l'afficher dans un navigateur.

Donc à partir d'une seule source nous pouvions obtenir :

Une documentation en POD

Une présentation

Une documentation en HTML, PDF

Un article pour GMLF

C'est pourquoi vous trouverez le premier élément de la présentation ci-après et par la suite tous les autres.



Fig.1 : Accueil Présentation PodODFtools

2. Informations utiles

Le dépôt concernant ce package se situe sur l'URL <https://github.com/fastrack/tools> et non comme d'habitude sur le CPAN. Passer à un degré plus formel de diffusion ne m'est pas paru utile car l'installation ne nécessite que la récupération du fichier et les packages qui sont utilisés.

Les figures (ou "diapositives") sont soit des captures d'écran au format .jpg soit réalisées avec le logiciel Inkscape et conservées au format .svg

3. Quelques Concepts de base pour l'écriture

3.1 Transformer des scripts en package objet

L'idée étant de rassembler l'ensemble des outils dans un même package, nous étions dans la situation de transformer des scripts en sous-programmes selon le schéma classique ci-dessous :

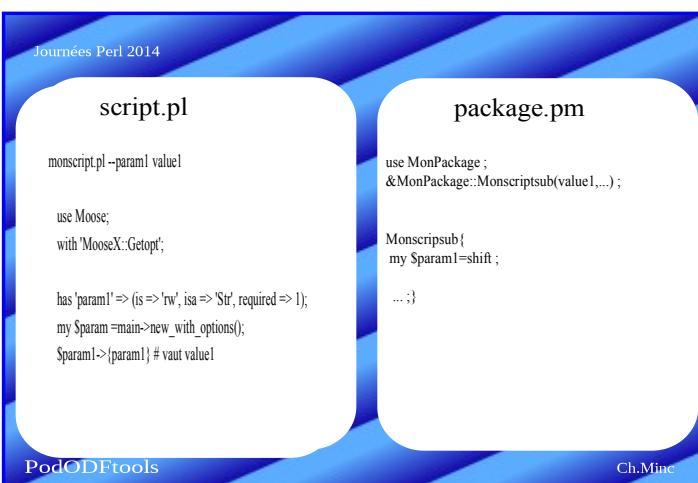


Fig.2 : Transformation script-package

Cette solution manquait cependant d'élégance, alors que Moose permettait de traiter simplement le problème par l'intermédiaire du trigger :

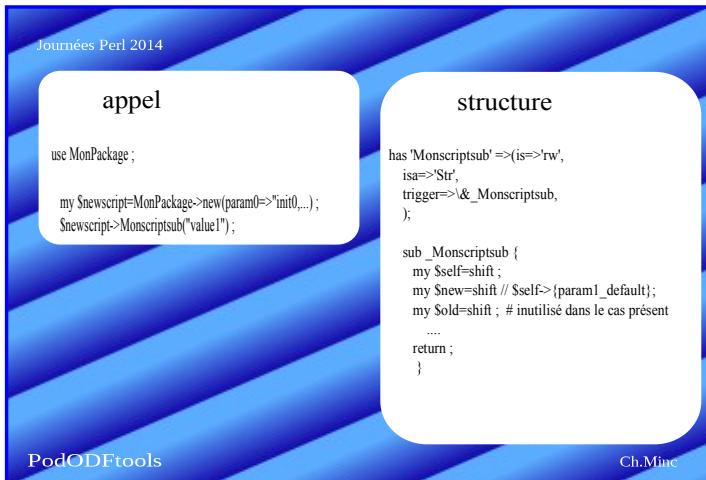


Fig.3 : Transformation script-package avec Moose

C'est donc ce schéma qui a été adopté pour la réalisation de PodODFtools.pm. Il offre, à notre avis, plus de souplesse avec une meilleure séparation entre les variables générales et celles spécifiques à l'appel d'une méthode.

3.2 Les principaux packages utilisés

Deux packages sont à la base de PodODFtools.pm :

Pod::HtmlEasy

ODF::IpOD

Rappelons que Pod::HtmlEasy transforme du POD en HTML, mais grâce à sa structure ouverte, nous pouvons l'utiliser juste comme un "Parser". Techniquement, il suffit de modifier ou d'ajouter les tags dans l'appel :

```

my $podhtml =
    Pod::HtmlEasy->new(
        on_H  => sub {
            votre code
            return ;
        },
        etc...
    );

```

La surcharge permet ainsi de modifier la fonctionnalité du package, dans l'exemple ci-dessus du tag "H" créé pour les besoins de GMLF avec le "callback "on_H".

La partie IpOD::ODF permet alors d'assurer la transition du texte ainsi sélectionné et analysé vers le format ODF. Ainsi dans notre cas :

```

my $podhtml =
    Pod::HtmlEasy->new(
        on_H => sub {
            my ( $this, $text )
= @_ ;
            my
$txt=decode_entities($text) ;
            "chapeau"
            # H as hat for style
$p=odf_paragraph->create(text=>$txt,style=>"chapeau") ;
            my $contexte =
$PodODFtools::doc->get_body;
            $contexte->append_element($p);
            return ;
        },
        etc...
    );

```

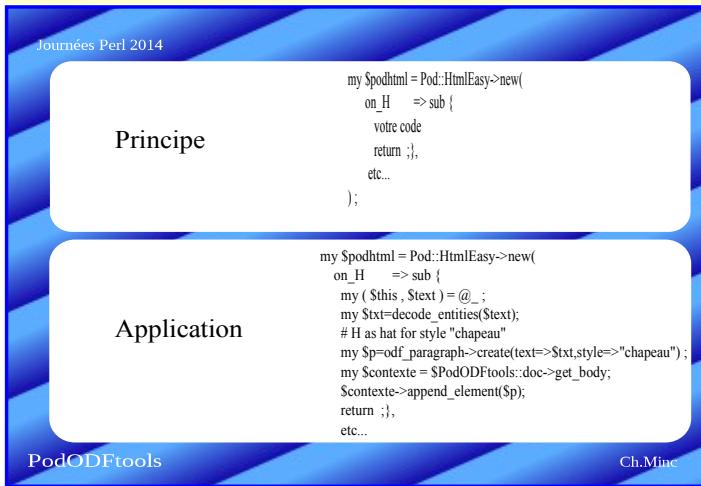


Fig.4 : Pod::HtmlEasy.on_H

La déclaration des variables dans le programme principal assure leur persistance au sein des sous-programmes ou des différents packages. De ce fait, il n'est pas la plupart du temps nécessaire de faire de déclaration de globalité. Cela simplifie amplement le code.

4. Les méthodes

De ce qui précède, les fonctions sont des *méthodes* pour exécuter les tâches décrites auparavant, ainsi nous avons :

Création d'un objet :

`my $tools=PodODFtools->new(%paramètres)`

Génération d'un fichier ODF :

`$tools->pod2odf("monlibreoffice.odt")`

Génération d'un fichier gmlf-ODF :

`$tools->pod2gmlf("mongmlf.odt")`

Génération de la présentation :

`$tools->show("ppt_page")`

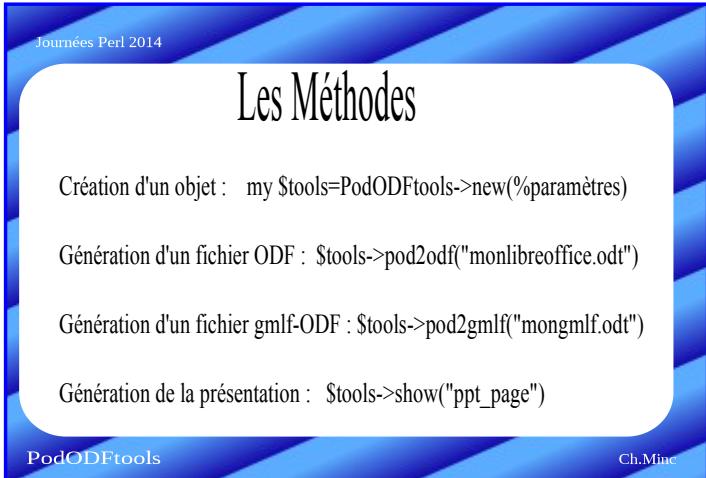


Fig.5 : Les Méthodes

L'utilisateur ayant plusieurs packages PERL à sa disposition pour obtenir un fichier HTML et en outre, comme il peut également le faire de façon triviale à partir d'un fichier ODF, nous n'avons pas cru nécessaire d'ajouter cette fonctionnalité.

4.1 Les paramètres globaux

Les paramètres globaux sont initialisés dans la création de l'objet :

```

my $tools=PodODFtools->new(
    #nom du fichier "model" utilisé et spécifié par GMLF -
    nécessaire.
    model=>"monmodel.ott",

    # nom du fichier pod source - nécessaire.

    #bien que les paramètres pod_car et odf_car soient censés
    permettre de choisir le type des caractères en entrée et en sortie
    #pour des raisons encore obscures, les caractères doivent
    être en ISO8859-1 pour compatibilité avec ODF::lpoD,

    pod=>'./animation-ISO8859-1.pod',

    # nom du fichier racine pour les "slides" en HTML
    #      showfile=> './showpage',
    #  screen_w définit la largeur de l'écran d'affichage
    #      screen_w=>1280,
    #  screen_h définit la hauteur de l'écran d'affichage

    #      screen_h=>800 ,

    # ce paramètre permet d'ajuster la taille des images dans le
    # fichier ODF par une réduction d'échelle
    #      scaledown=>2,

    # une référence tableau permet de définir le répertoire où se
    # trouve l'éditeur ODF - nécessaire

sofficedir=>["usr","bin","libreoffice"],
    # cette variable donne le nom de l'exécutable pour éditer les
    # fichiers ODF - nécessaire
    soffice_exe=>"swriter,
)
;
```

Les paramètres non nécessaires possèdent une valeur par défaut qui est celle de l'exemple ci-dessus, sauf

en ce qui concerne l'éditeur ODF pour des raisons évidentes de dépendance d'implantation ou de version.

4.2 La méthode show

Les inclusions graphiques dans le fichier POD se font par l'intermédiaire d'une directive "=for html". C'est une convention d'écriture de notre utilisation. Le "=for" autorise l'utilisation d'un autre langage, précisé ici par le complément "html", qui n'a pas d'interprétation dans le POD.

L'idée est que ce sont les figures incluses dans le texte de l'exposé qui permettent à un conférencier d'illustrer son discours au même titre que son écrit.

Partant de ce postulat, il suffit donc de filtrer les lignes contenant ces items afin de former une suite de fichiers HTML et constituer ainsi le squelette de la présentation. Ces derniers constitueront une suite numérotée qui sera parcourue en cliquant dans la partie gauche de la figure insérée, pour revenir en arrière ou à droite, pour avancer.

Une souris ou un écran tactile seront donc utilisables pour effectuer les transitions.

Les fichiers sources peuvent être du SVG, PNG, JPG etc... Le fait qu'ils soient affichables dans un navigateur assure une portabilité plus grande d'un système à un autre ; de plus les paramètres screen_w et screen_h permettent d'adapter les vues à la géométrie de l'écran, pour plus de spécificité, en relation avec la taille de l'image.

Ainsi la première image dont la ligne de code est la suivante :

```
=for html <p class="illustration"></p>
```

produira le code HTML :

```
<p class="illustration"></p>
<map name="navigmap">
<area shape="rect" coords="0,0,464.117647058824px,640px" href="JP2014_CLM_page_0.html" alt="predpage">
<area shape="rect" coords="464.117647058824px,0,928.235294117647px,640px" href="JP2014_CLM_page_1.html" alt="nextpage">
```

Mentionnons que les directives avec "=for comment" ne sont visibles que dans le pod source et constituent des commentaires uniquement destinés au(x) rédacteur(s). Par exemple :

```
=for comment <p class="illustration"></p>
```

5. pod2odf et pod2gmlf

De fait, les simples instructions suivantes suffisent à produire les fichiers au format correspondant, respectivement :

```
$tools->pod2odf("./pod2odf.odt");
$tools->pod2gmlf("./gmlf.odt");
```

On pourra faire la comparaison de ces deux fichiers entre eux puis avec le source en pod.

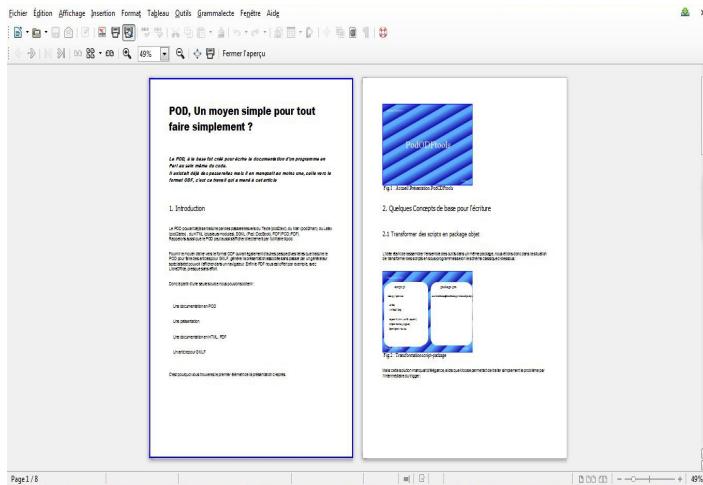


Fig.6 : Les deux premières pages en ODF

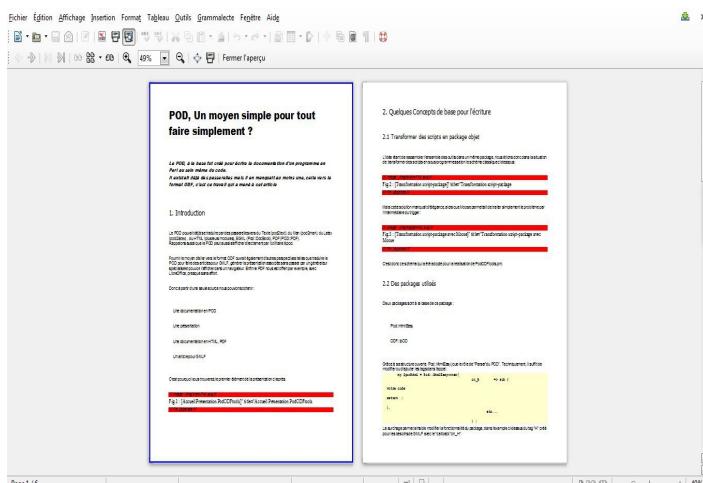


Fig.7 : Les deux premières pages en gmlf-ODF

```
=pod
Q<POD, Un moyen simple pour tout faire simplement >
=meta copyright 2014 - Charles MINC.

H<Le POD, à la base fut créé pour écrire la documentation d'un programme en Perl au sein même du code.
Il existait déjà des passerelles mais il manquait au moins une, celle vers le format ODF, c'est ce travail qui a mené à cet article>

=head1 Introduction

Le POD pouvait déjà se traduire par des passerelles vers le Texte (pod2text), du Man (pod2man), du Latex (pod2latex), du HTML (plusieurs modules), SGML (Pod::DocBook), PDF (POD::PDF).

Rappelons aussi que le POD peut aussi s'afficher directement par l'utilitaire ipod.

Fournir le moyen d'aller vers le format ODF ouvrait également d'autres perspectives telles que traduire le POD pour faire des articles pour GMLF, générer la présentation associée sans passer par un générateur spécialisé et pouvoir l'afficher dans un navigateur. Enfin le PDF nous est offert par exemple, avec LibreOffice, presque sans effort.

Donc à partir d'une seule source nous pouvions S<obtenir >

=over 4
=item Use documentation en POD
=item Use présentation
=item Use documentation en HTML, PDF
=item Un article pour GMLF
=back

C'est pourquoi vous trouverez le premier élément de la présentation ci-après.
```

```
=fix html <p class="illustration"></p>
```

Fig.8 : Les premières lignes en POD

Vous ne remarquez rien ? Il y a deux tags qui ne font pas partie du POD "Q" et "H". Ils ont été ajoutés pour les besoins de GMLF pour appliquer le formatage "Titre" et celui de "chapeau".

De manière commune, les paragraphes et les figures sont automatiquement numérotés puisque dans le POD source cette indication est absente.

6. Méthode pod2odf

Dans notre cas, le PDF nécessite que le fichier ODF soit présent. En effet, il devient le fichier source ; le fichier destination porte alors le même nom racine, seule l'extension est modifiée.

La commande suivante :

```
$tools->pod2pdf("chemin_absolu/pod2odf.odt") ;
```

nous permet de le générer.

Comme précédemment nous vous présentons les deux premières pages.

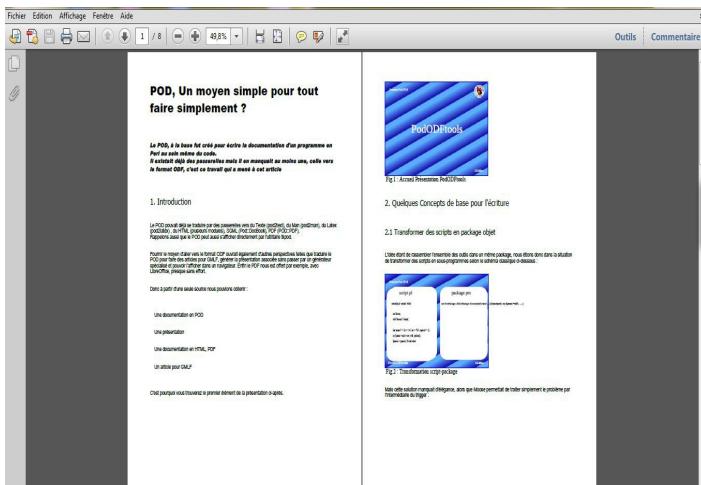


Fig.9 : Les deux premières pages en PDF

7. Exemple in extenso d'un programme pour réaliser l'ensemble des générations

```
#####
### File           : gmlfPodODFtools.pl
### Author         : Ch.Minc
### Purpose        : Article gmlf/Journées Perl 2014
### Version        : 1.0 2014/06/03
### copyright      : GNU license
#####

our $VERSION = '1..0';
use 5.12.3;
use strict ;
use warnings ;

use PodODFtools ;

# création de l'objet PodODFtools ,
# sélection du modèle (feuille de style de GMLF dans ce cas)
# et réduction d'échelle des figures
my $tools=
```

```

PodODFtools->new(
    model=>"diamond_editions.ott",
    pod=> './gmlfPodTools.pod',
    scaledown=>3) ;

# sortie des noms des fichiers temporaires

say $tools->{target} ;
say $tools->{pregmlf} ;
say $tools->{html} ;

# indication de la taille de l'écran
say $tools->{screen_w} ;
say $tools->{screen_h} ;

# génération des fichiers
$tools->show("JP2014_CLM_page") ;
$tools->pod2odf("./pod2odf.odt") ;
$tools->pod2gmlf("./gmlf.odt") ;

# passage à la taille écran Sony Xperia SP
$tools->{screen_w}=720 ;
$tools->{screen_h} =1280 ;

# création d'une nouvelle présentation pour le mobile
$tools->show("JP2014_CLM_SmartPhone") ;

# création du fichier pdf
$tools->pod2pdf("/pod2odf.odt") ;

```

8. Conclusion

La multiplicité des supports et des outils amène à trouver des moyens permettant de générer à partir d'une même source plusieurs formats. En outre, les particularités matérielles sont de plus en plus diversifiées (TV, PC, Smartphone ...) , il s'ensuit que la production des documents devient aussi tributaire des écrans. La présence de PERL sur la plupart des plateformes (seul Android semble encore un peu expérimental, à ma connaissance) permet de rendre la génération relativement indépendante du système.

Le moyen d'élaboration simple que nous avons présenté semble donc remplir ces objectifs. Certains trouveront les caractéristiques du POD insuffisantes comparées à celles des éditeurs classiques, mais le semblant "Make it simple, make it stupid" peut faire gagner pas mal de temps par rapport à des possibilités superfétatoires. C'est notre but.



Fig.10 : Comme dans un fauteuil



Fig.11 : Conclusion

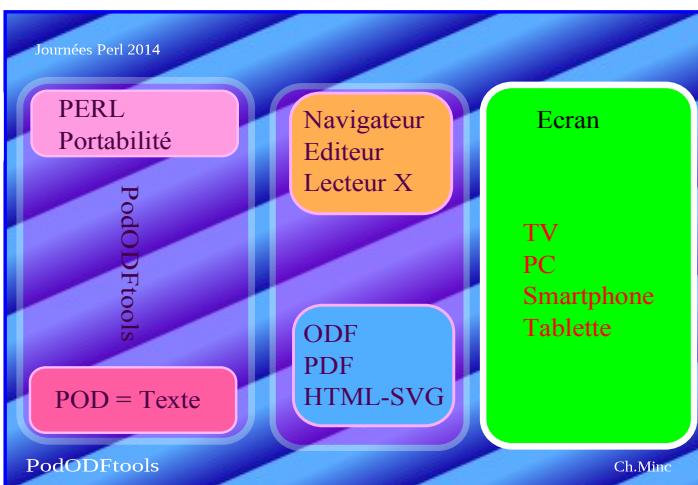


Fig.12 : Résumé

9. Bibliographie ou références

POD, <http://perldoc.perl.org/perlpod.html>

Editeur SVG, <https://code.google.com/p/svg-edit/>

Editeur SVG, <http://www.inkscape.org/fr/>

Tutoriel inkscape <http://www.inkscape.org/fr/apprendre/didacticiels/>

Dépôt PodODFtools, <https://github.com/fastrack/tools>

GNU Linux Magazine France, http://www.ed-diamond.com/articles/guide_auteur.pdf



Fig.13 : Bibliographie

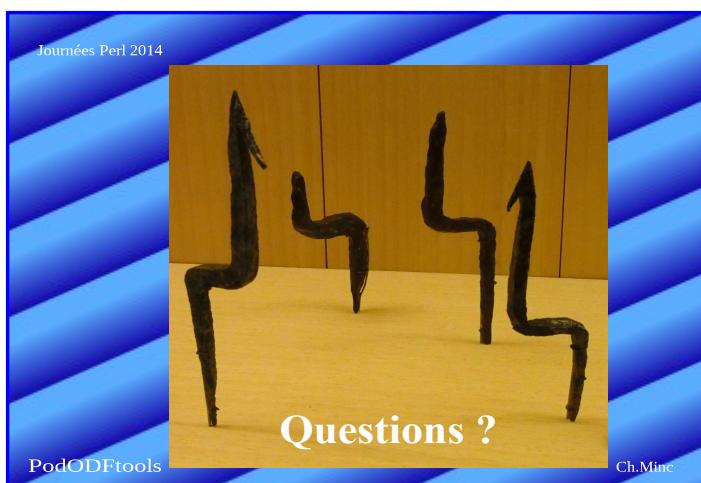


Fig.14 : Questions



Fig.15 : Pause