Sélectionner le bon format texte

Gazette Linux n°98 - Janvier 2004

Dean Wilson

Article paru dans le n°98 de la Gazette Linux de janvier 2004.
Traduction française par Simon Depiets <2df CHEZ tuxfamily POINT org>.
Relecture, réécriture et correction de la traduction française par Joëlle Cornavin <jcornavi CHEZ club TIRET internet POINT fr>.

Table des matières

1. Formats binaires 1

2. Formats orientés présentation 2

3. Formats à balisage fixe 2

4. Formats à balisage dynamique 3

5. Conclusion 4

J'ai travaillé sur un projet qui nécessite le stockage de données textuelles de façon à faciliter la recherche et la présentation des informations dans divers formats. Les informations que vous voyez ici ont été écrites pour m'aider à expliquer à d'autres personnes quels étaient les choix disponibles, en montrant leurs avantages et leurs inconvénients.

1. Formats binaires

Ce sont des formats dans lesquels il vous faut des informations supplémentaires pour donner du sens aux données. L'exemple le plus courant est le format `.doc` qu'utilise Microsoft Word. Si vous examinez un document Word avec un éditeur de texte, vous ne voyez pas votre texte. Ce problème à lui seul rend ce format inutile pour un système qui doit rechercher et présenter des données.

Comme il n'y a pas de raison de s'étendre sur ce format, j'aimerais aborder un autre problème applicable à beaucoup d'autres situations que celle dont je parle ici. Il est assez fréquent pour les fabricants de changer de format au fil du temps, en revendiquant qu'ils doivent le faire pour améliorer les capacités de leurs programmes. Il se peut que ce soit vrai ou non, mais il est facile d'imaginer qu'une future version

de leur programme n'utilisera pas le même format, ce qui rendra vos anciennes données inutiles ou rendra impossible d'en créer de nouvelles dans le format que vous aviez choisi à l'origine.

Comme ces types de formats ne conviennt pas pour le système sur lequel je travaille, je n'entrerai pas plus dans les détails. Les formats suivants reposent tous sur du texte.

2. Formats orientés présentation

Ces langages vous permettent de voir vos informations textuelles originelles à l'intérieur du document, qui contient des informations sur la manière dont le texte doit être présenté à l'intérieur du texte lui-même. Des exemples de ce type de format sont troff, TeX, RTF (*Rich Text Format*) et PostScript. Parmi ces quatre exemples, troff et TeX sont de loin les plus anciens et également les plus faciles pour y extraire du texte.

Troff a été écrit initialement pour produire une sortie imprimante sur une photocomposeuse particulière. Il existe un programme connexe, nroff, qui a été conçu pour prendre le même format de base de document et produire une sortie sur une imprimante classique. TeX (et LaTeX) ont été conçus pour composer des documents complexes contenant des symboles mathématiques spéciaux non disponibles en texte ASCII. Au fur et à mesure que le contenu du document devient plus compliqué, l'extraction le devient également.

Tôt dans la guerre des traitements de texte, Microsoft a créé un nouveau standard (en d'autres termes, ils l'ont appelé ainsi) pour l'échange de documents appelé RTF. Contrairement à troff ou TeX, l'utilisateur n'était pas supposé créer des documents en RTF, ce format étant juste à employer comme support d'échange entre différents traitements de texte.

Enfin, PostScript a été écrit pour servir de langage descriptif pour les fichiers, qui serait indépendant des périphériques de sortie. Par exemple, le même document PostScript pourrait être imprimé sur une imprimante laser relativement peu onéreuse avec une résolution de 300 points par pouce ou sur une imprimante haut gamme 3000 points par pouce ou plus.

Dans tous ces langages, l'accent est mis sur la description de ce à quoi vous voulez que ressemble le document. Vous décrivez les tailles des polices, les styles de caractères et le positionnement dans la page. Pour rechercher le texte originel du document, vous devez expurger toutes ces informations de formatage.

3. Formats à balisage fixe

Comme vous pouvez le voir dans les descriptions précédentes, aucun de ces formats ne rend les informations originelles accessibles à une recherche aisée. De plus, il vous faudra un programme de conversion pour traduire ces formats de documents dans les divers formats de présentation.

Bien avant que Microsoft ne crée le « standard RTF », SGML était la règle. SGML est un langage généralisé de balisage de documents sources conçu pour spécifier ce qu'il y a dans un document plutôt que la manière de l'afficher ou de l'imprimer. SGML, toutefois, est assez général au point d'être compliqué pour l'utilisateur et pour un ordinateur trop lent pour fonctionner avec.

HTML est pratiquement un dialecte de SGML. Pratiquement, car HTML n'obéit pas à toutes les conventions de base de SGML et un dialecte car HTML définit un balisage de documents spécifique que

l'on peut utiliser. Par exemple le balisage <p> sert à identifier le début d'un paragraphe. Il y a deux problèmes ici par rapport à ce que je dois faire :

- Bien que la balise de fermeture <p> soit maintenant autorisée en HTML, elle n'est pas obligatoire. Elle complique le processus.
- Bien que je sache où commence un paragraphe, je ne sais pas ce qu'il serait susceptible de représenter.

De plus, HTML a évolué dans des proportions telles qu'il y a un grand nombre de balises. Beaucoup d'entre elles se rapportent à la présentation mais n'offrent aucune information sur l'utilisation réelle du contenu du document. Des exemples ici sont les balises `strong` ou `bold` (gras), `italic` (italique).

4. Formats à balisage dynamique

Faisons la liste de ce que nous avons appris jusqu'ici :

- Les formats binaires ne fonctionnent pas pour la recherche.
- Il est difficile d'extraire les informations du document originel à partir de langages conçus pour décrire la présentation.
- Avoir un balisage qui décrit la fonction d'une donnée plutôt que ce à quoi elle doit ressembler est bien préférable.

Il serait relativement facile d'écrire un tel balisage. En fait, je l'ai fait de nombreuses fois pour des projets spécifiques. Je me souviens d'un système très basique où j'utilisais une seule lettre suivie de deux points pour identifier le type de données que contenait un enregistrement. Les enregistrements étaient séparés par un saut de ligne. Même si nous connaissions tous les types d'enregistrements possibles, il y a toujours une limitation très significative dans ce type d'implémentation. Vous ne pouvez pas décrire les relations à l'intérieur des données.

Un exemple plutôt évident est une adresse. Si vous lui imposez une structure qui est applicable à une adresse en France, vous pourriez obtenir le résultat suivant :

Nom|Adr1|Adr2|Ville|Code postal

Si l'on réfléchit en termes plus globaux, vous pourriez ajouter le nom du pays à la fin des données. Malheureusement, vous découvririez qu'en Espagne, par exemple, le code postal se trouve avant le nom de la ville.

Même si vous pouviez ensuite écrire le code permettant de remplir le septième champ (pays) pour l'Espagne et modifier la manière dont les informations sont affichées, vous constateriez rapidement qu'il existe de nombreuses autres exceptions. Avec le coût relativement bas du stockage de données aujourd'hui, une meilleure approche serait d'ajouter davantage d'informations à celles qui se trouvent dans l'enregistrement de données. Si ces informations portaient sur ce que sont les données plutôt que sur la manière de les traiter et si elles étaient présentées d'une manière bien structurée, il serait très facile de travailler avec.

Entre en scène XML, qui signifie *Extensible Markup Language*. XML est conçu pour faire précisément ce travail. Si l'on en revient au modèle d'adresse, vous pourriez la présenter en XML de la manière suivante :

Il n'y a rien de spécial sur l'espacement et l'indentation. Il s'agit juste de clarifier pour le lecteur ce que je suis en train de faire. Le seul point important est que les informations commencent par <address> et se terminent par </address>.

Ajouter de la place pour le nom du pays est aussi facile que définir la balise <country>. Les règles de présentation n'ont pas à être placées dans les données elles-mêmes. Il existe un autre langage, XSLT (*XML Stylesheet Language Transformations*), qui vous permet de définir des règles de traitement pour traduire du XML dans les formats de sortie souhaités.

5. Conclusion

La partie la plus importante de cet exercice a été pour moi d'être en mesure d'étudier les formats existants pour choisir une bonne solution. Parfois, une nouvelle approche ou un nouveau format nécessite d'être développé (PostScript est un bon exemple), mais vous aurez toujours moins de travail si vous commencez avec un format qui existe déjà.

Du fait que XML est extensible, vous ne choisissez pas un format qui répond totalement à vos besoins. Vous sélectionnez en fait une concordance exacte qui vous permet d'appréhender ce dont vous avez besoin. Avec tous les outils différents disponibles pour XML et XSLT et le nombre d'utilisations croissant chaque jour, développer vos applications autour de ce format sera plus aisé dans le futur.

Dean Wilson est administrateur système dans une entreprise où le patron (qui n'a aucune idée de ce que fait Dean) dit simplement « Faites en sorte que cela fonctionne ».