



Rendre utilisable FireFox sous KDE

Le 26 septembre 2007 à 00:10.

Oui, FireFox est *déjà* utilisable sous KDE mais dieu que c'est mal intégré. Alors pour ne pas noyer le bébé avec l'eau du bain, j'ai tenté de customiser un peu le monstre pour qu'il soit plus intégré au reste de KDE et plus en phase avec les performances de konqueror.

Améliorer le Look de FireFox

Synchroniser GTK avec KDE

Le look par défaut de FireFox est franchement horrible sous Linux/KDE. Et à mon humble avis, l'utilisation de thèmes aggrave encore les choses pour ceux qui aiment que l'ensemble des applications se ressemble.

Comme dit plus haut, par défaut, FireFox utilise le look de GTK. Il existe certains styles comme ClearLook qui ont été portés à la fois sur GTK et sur Qt. Mais la "vraie" solution réside plutôt d'ans l'utilisation de `gtk-qt`. `Qt-Gtk` est en réalité un thème pour GTK qui a la particularité de copier celui qui est en court d'utilisation sous Qt. En utilisant ce style, les applications GTK (FireFox, mais aussi Gimp, Evolution, etc...) deviennent alors d'un coup de baguette magique compatible avec le look général de KDE.

Pour l'installer sous Mandriva

```
|| urpmi gtk-qt-engine
```

Ceci fait, il faut paramétrer GTK pour utiliser notre nouveau style. Et là il y a quelques mystères... J'ai remarqué que pour un look quasi parfait de toutes les applications, j'ai besoin de trois choses.

Tout d'abord, d'aller dans le panneau de configuration de KDE, dans la section Apparence/Style et Police GTK, de choisir `Use my KDE style in GTK applications`, `Use my KDE fonts in GTK applications`, de cliquer sur `Install scrollbar fix...`, puis sur `Appliquer`.

Ensuite, je vais régler, en plus, le style GTK par l'application `gtk-chtheme`. Elle aussi se trouve dans votre distribution sans aucun doute. Une fois lancée, vous cherchez le style `Qt` et vous le sélectionnez.

Dernier point, installer le `gnome-settings-daemon`. Cet utilitaire semble important pour que le style passe bien sur certaines applications comme... `firefox`, mais aussi `Eclipse`. Le plus simple est d'installer le package `gnome-control-center`. Ceci fait, il faut simplement créer un lien symbolique dans votre dossier `AutoStart` :

```
|| ln [1] -s /usr/lib/gnome-settings-daemon ~/.kde/Autostart
```

Ceci fait, pour le lancer la première fois et pour re-re-re-configurer le style GTK, lancez `gnome-control-center`, cliquez sur `Thème` puis `Personnaliser...`, sélectionnez dans la liste de l'onglet `Contrôles` le style `Qt`, puis cliquez sur `Fermer/Fermer`

Voilà, ça devrait être bon, vérifiez à l'aide de la commande `ps -edaf` que le démon est bien lancé, quittez `firefox` et relancez le, normalement il devrait avoir le style `Qt`, même pour les onglets...

Modifier l'interface utilisateur

L'interface de FireFox est très ouverte car écrite en XUL (nous verrons cela plus en détail avec les extensions), un format de description hérité du XML qui emprunte beaucoup de ses comportements au couple DOM/Javascript. Cette particularité rend possible la modification de l'interface même de `firefox` à travers l'utilisation de simples feuilles de style. Et c'est très simple comme vous allez le voir.

Prenons un exemple. Sous KDE, j'utilise une fonte très particulière pour les menus, la `Square721 BT`, en 12pt. Et malheureusement, `Firefox` n'en a cure. Malgré l'utilisation de `GTK/QT`, il décide d'afficher le tout à sa sauce.

La solution se trouve dans votre dossier `./mozilla/firefox/XXXXX.defaults/chrome`. Vous trouverez là, deux fichiers `userChrome.css`. S'il n'existe pas il faut le créer. Ce fichier est ni plus ni moins qu'une feuille de style classique à la différence près qu'il ne modifie pas l'apparence d'une page web, mais de l'interface de `FireFox`. La syntaxe est strictement la même, seul les tags changent, car XUL est plus riche que HTML et contient par exemple des balises `bookmarks-tree`, `tabs` ou `menubar`. Nous allons donc ajouter un bloc de CSS pour modifier la fonte et la taille de celle-ci pour l'ensemble (ou presque) de `firefox`.

```
|| #main-window, #toolbar-menubar, menubar > menu, #bookmarks-menu, tabs,
|| #bookmarksPanel, bookmarks-tree, menubar > menu > menu
|| {
||   font-family:"Square721 BT" !important ;
||   font-size:12px;
|| }
```

Le mot clef `!important` est très... important 😊 En effet, il va indiquer à `firefox` que ce n'est pas la peine d'utiliser une autre police, c'est celle là que l'on veut. Bien évidemment, il faut redémarrer `firefox` pour voir le résultat (vous pouvez remplacer "ma" fonte par n'importe quelle autre police de votre goût).

Alors oui ce système est très puissant mais aussi assez obscure. Je n'ai en effet pas réussi à mettre la main sur une documentation complète et/ou à jour sur les balises et les IDs utilisables. Le bon moyen pour trouver votre voie, est d'utiliser

le men DOM Inspector et de saisir l'URL `chrome://browser/content/browser.xul` dans l'inspecteur. Là il vous faut naviguer pour trouver votre bonheur et le reporter dans la feuille de style.

Petit gag amusant, tapez la même URL (`chrome://browser/content/browser.xul`) non pas dans le DOM inspector, mais dans firefox directement 😊

Modifié le rendu des pages

Maintenant que l'interface nous convient mieux, il est temps de s'attaquer aux pages elle-mêmes. En effet il y a plusieurs choses qui ne vont pas. Les bordures des zones d'édition et des boutons sont épaisses et laides, les boutons cubiques, les radios et les combos semblent venir d'un autre âge.

Alors nous pourrions appliquer le même principe que pour l'interface en modifiant cette fois le fichier `userContent.css` qui se trouve au même endroit que `userChrome.css`. Par exemple si je veux que toutes les zones d'édition aient une bordure grise, il me suffit d'ajouter :

```
textarea
{
border: 1px solid gray !important;
}
```

Après redémarrage, toutes les zones de textes sont ainsi modifiées, quel que soit le site visité. Cette solution permet avec un peu d'effort de régler tous les problèmes. Mais l'avantage est que quelqu'un a déjà fait cela pour nous en fournissant un kit complet transformant tous les contrôles firefox avec un look kde. Cela s'appelle [FireFox Form Widgets](#) [2]. Une fois que vous avez téléchargé cette archive, il suffit de la décompresser, d'aller dans le dossier `KDEWidgets`, et de copier tout son contenu (5 images et une feuille css) dans le dossier `/usr/lib/firefox/res/`. Après redémarrage de firefox, tous les formulaires sont maintenant visuellement compatibles avec KDE.

Améliorer l'intégration avec KDE

L'intégration à KDE est à prendre au sens large du terme. Il s'agit autant de remplacer les dialogues GTK par ceux de KDE (ex. Impression) que de pouvoir appeler certaines applications KDE en lieu et place des fonctions internes de Firefox (ex. Téléchargement).

Utiliser KPrint pour l'impression

GTK et l'impression c'est un roman. Linus Torvald lui-même avait pété le plomb sur ce sujet tant l'équipe Gnome avait simplifié à outrance les dialogues au point de les rendre absolument inutilisables dans certains cas (avec deux imprimantes par exemple). KDE, en revanche, a un magnifique moteur d'impression nommé `kprint`. Nous allons donc tenter de l'utiliser. Pour cela il faut aller dans la boîte de dialogue d'impression de firefox (Fichier/Imprimer). Cliquer sur "propriétés". Dans la zone "commande", videz ce qui existe et saisissez juste `kprinter --stdin`. Validez et tentez une impression, vous devriez enfin déboucher sur la fenêtre d'impression de kde. simple et efficace (idée originale trouvée [ici](#) [3])

Ajouter les flux RSS dans akregator

Pour réaliser cela, il faut utiliser [ce script](#) [4]. Vous pouvez sauver ce script en `/usr/bin/xdg-addrss` et le rendre exécutable (`chmod +x /usr/bin/xdg-addrss`). Ensuite, dans les préférences de Firefox 3, allez dans l'onglet Applications, cherchez Flux WEB/Autres.. et sélectionnez votre nouveau script.

Comme souvent, l'effet est immédiat, il suffit d'aller sur un site disposant de flux, de cliquer sur l'icône orange dans la barre d'url et de sélectionner votre flux pour qu'il apparaisse sous akregator. Une bonne chose de faite, passons à la suite 😊

Appeler des programmes externes à partir de Firefox

Pour rendre Firefox encore un peu plus intégré à KDE, j'avais besoin de deux choses. Tout d'abord un menu contextuel sur un lien dans une page permettant de transférer l'URL vers Agregator. L'idée est d'exploiter pour se simplifier la vie, les fils de discussion au format RSS, de plus en plus nombreux. Ce menu devra exécuter le script présenté plus haut pour ajouter la conversation à akregator.

La deuxième chose qui me manque est l'archivage des pages web. L'idée est d'avoir un menu dans la barre principale appelant un script qui va télécharger la page avec ses images, ses styles, etc... et bien sûr compresser l'ensemble en un fichier `.war`. Ses fichiers `.war` sont directement lisibles par `konqueror` et me servent à garder trace de tous les articles qui vont un jour ou l'autre disparaître de la toile. Je peux garder `konqueror` pour les relire, mais je dois pouvoir les créer avec Firefox.

Alors aussi étrange cela puisse paraître, ajouter un menu pour Firefox, c'est la croix et la bannière. En réalité il n'existe aucun mécanisme standard permettant de faire cela simple (comme un simple fichier `.desktop` sous `konqueror`). Pour y arriver, la seule possibilité est de faire une extension... Sacré programme pour un bête menu 😊

Création d'un plugin basique

Heureusement, après avoir passé quelque temps à tenter de créer un plugin, j'ai découvert un [magnifique assistant qui fabrique cela tout seul](#) [5]. Il suffit de la

paramétrer pour lui demander de nous générer un fichier xpi contenant un squelette fonctionnel.

Dans notre cas nous allons donner les valeurs suivantes :

```
Your Name: Votre nom 😊
Extension Name: le nom du plugin, par exemple, handleit
Short Name: handleit
Extension: handleit A.T. _bad_karmalab.net
Cocher Create options dialog and preferences
Cocher Create context menu item
Cocher Create custom about box
```

Vous pouvez en mettre plus si cela vous tente mais c'est tout ce qui est nécessaire. Il ne reste plus qu'à cliquer sur Create Extension pour télécharger le fichier squelette handleit.zip

Ce fichier contient tout ce qu'il faut pour fabriquer un XPI mais pour l'instant, nous allons l'utiliser tel quel sous FireFox. Première étape, localiser votre dossier de profile. Il se trouve normalement dans le dossier ~/.mozilla/firefox et porte un nom ésothérique avec l'extension .default. J'imagine que c'est pour des raisons de sécurité. S'il vous en avez plusieurs, trouvez le bon en vous aidant du contenu du fichier bookmark.html qui se trouve à l'intérieur.

Dans ce dossier, que nous appellerons à partir de maintenant nrz806.default, se trouve un dossier extensions. Et dans ce sous-dossier sont stockés tous les plugins que vous avez installés. Pour utiliser notre plugin en cours de fabrication, nous allons donc procéder de la manière suivante :

```
cd
mkdir [6] workspace
cd workspace
unzip [7] ~/Desktop/handleit.zip [8]
cd handleit
echo $PWD/ > ~/.mozilla/firefox/nrz806.default/extensions/handleit\@karmalab.net
```

Ceci va créer un \"faux plugin\" dans le dossier des extensions qui pointe sur le plugin que l'on est en train de développer. Pour continuer, je vous conseille d'installer l'extension Restart FireFox. En effet, redémarrer firefox est le seul moyen de tester les modifications dans le plugin, youpi 😊

De toute façon, l'installation de cette extension va logiquement impliquer... de redémarrer FireFox. Lorsque ce dernier sera revenu à lui, vous aurez un nouveau menu Restart FireFox dans Fichier, un menu En rouge dans Outils et un nouveau menu dans votre click-droit. Les deux derniers ont été ajoutés par votre plugin.

Pour modifier les libelés, allez simplement changer leur valeur dans ~/workspace/handleit/locale/en-US/handleit.dtd. Mettez Archiver la page courante comme valeur à handleit.label, et Ajouter la conversation à Akregator dans handleitContext.label. Ceci fait, redémarrez FireFox pour admirez votre premier résultat.

Les plugins sont en javascript, pour modifier le comportement de nos deux menus (qui pour l'instant disent \"hello world\"), il faut modifier le fichier /home/yoran/workspace/handleit/content/overlay.js. Il nous faut tout d'abord y ajouter une fonction qui exécute une commande externe à FireFox (et oui, c'est possible...). Pour cela, ajoutez après la fonction onLoad:fuchions(){...}, le code suivante :

```
execute: function(command, args)
{
    var targetFile =
Components.classes['@mozilla.org/file/local;1'].createInstance(Components.interfaces.nsILocalFile)
;
    targetFile initWithPath(command);
    var process =
Components.classes['@mozilla.org/process/util;1'].getService(Components.interfaces.nsIProcess)
;
    process.init(targetFile);
    process.run(false, args, args.length);
},
```

La virgule finale est importante !! Ceci fait, nous allons vider le corps de la fonction onMenuItemCommand pour la faire coller à nos besoins. En l'occurrence il s'agit d'exécuter deux commandes différentes en fonction du type d'appel (par le menu principal ou par le menu contextuel), cela nous donne :

```
if (gContextMenu)
{
```

```

    var args = [gContextMenu.getLinkURL(), 'conversation'];
    handleit.execute('/usr/bin/xdg-addrss', args);
}
else
{
    var pos=document.title.indexOf("-");
    var title=document.title.substring(0, pos);
    var args = [window.content.document.location.href, title, document.characterSet];
    handleit.execute('/usr/bin/xdg-war-archive', args);
}
}

```

Notez que comparé à l'utilisation précédente, nous passons un paramètre en plus à xfg-addrss pour spécifier de mettre ce que l'on ajoute dans le dossier conversations d'Akregator.

Il ne nous reste maintenant plus qu'à créer le script /usr/bin/xdg-war-archive disponible [ici](#) [9]

Ce script va donc poser sur le bureau de l'utilisateur qui le lance un fichier portant comme nom, le titre de la page. Il consistera en une archive tar.gz complète au format .war lisible donc par konqueror. N'oubliez pas de faire un chmod +x /usr/bin/xdg-war-archive.

Il ne reste maintenant plus qu'à redémarrer FireFox pour tester tout cela. Si cela coince, il doit y avoir une faute de frappe, vous pouvez mettre des boîtes de dialogue dans le code pour tester pas à pas :

```

var promptService = Components.classes["@mozilla.org/embedcomp/prompt-service;1"]
.getService(Components.interfaces.nsIPromptService);
promptService.alert(window, "Debugger", "Message d'alerte");

```

Maintenant si vous voulez supprimer le faux plugin et installer "pour de vrai" votre création (ou pourquoi pas, la redistribuer, il y a de quoi être fier 😊), la marche à suivre est la suivante :

```

cd ~/workspace/handlit
chmod [10] +X build.sh [11]
./build.sh [11]

```

Vous pouvez maintenant supprimer le faux plugin de votre profile, et installer le fichier XPI fraîchement généré.

<http://artisan.karma-lab.net/node/1177>
(C) artisan numerique - CC BY-SA

Liens:

- [1] <http://pwet.fr/man/linux/commandes/ln>
- [2] <http://www.kde-apps.org/content/show.php/show.php?content=64392&vote=good&tan=88705725&PHPSESSID=def9661e2a761bc1a37bb007784b20bd>
- [3] <http://www.granneman.com/webdev/browsers/mozillafirefoxnetscape/linuxspecific/kdeprintinginmozilla/>
- [4] <http://artisan.karma-lab.net/node/1215>
- [5] <http://ted.mielczarek.org/code/mozilla/extensionwiz/>
- [6] <http://pwet.fr/man/linux/commandes/mkdir>
- [7] <http://pwet.fr/man/linux/commandes/unzip>
- [8] <http://pwet.fr/man/linux/commandes/zip>
- [9] <http://artisan.karma-lab.net/node/1216>
- [10] <http://pwet.fr/man/linux/commandes/chmod>
- [11] <http://pwet.fr/man/linux/commandes/sh>