2013

Plug-in Kalifast pour Sélénium IDE

DOCUMENTATION TECHNIQUE

GEOFFROY KLIMERAK

EISEGE | 15 Boulevard Anatole France 45200 MONTARGIS

Introduction

Le plug-in Kalifast est une extension Firefox, autrement dit un module complémentaire, permettant d'ajouter des fonctionnalités à firefox. Dans notre cas, l'objectif du plug-in était d'apporter des améliorations au plug-in Sélénium IDE afin d'intégrer notre solution en son sein.

Objectifs du plug-in

Le plug-in devait permettre de remplir fonctions :

- Paramétrer des connexions vers les comptes utilisateurs via les préférences.
- Fournir une authentification vers la solution Kalifast via les web services
- Récupérer la liste des profils/projets/scénarios associés à un compte.
- Charger un scénario sélectionné au préalable via 2 méthodes possibles.

Les prochaines étapes seront de fournir des logs d'exécution des scripts.

Environnement Technique

L'environnement technique est celui de tout module Mozilla puisqu'il comprend les fichiers XUL pour tout ce qui est template, le javascript pour l'intéraction et le css pour la mise en forme.

XUL

Les XUL est le langage d'interface utilisateur pour Mozilla basé sur XML. Il permet de construire des applications multiplateformes riches en fonctionnalités, pouvant être ou non connectées à Internet. Ces applications peuvent facilement être personnalisées au niveau du texte, des graphiques et de leur mise en page et peuvent ainsi être facilement rhabillées selon leur destination et être fournies en plusieurs langues. Les développeurs Web qui sont déjà familiarisés avec le HTML dynamique peuvent apprendre le XUL rapidement et être opérationnels immédiatement.

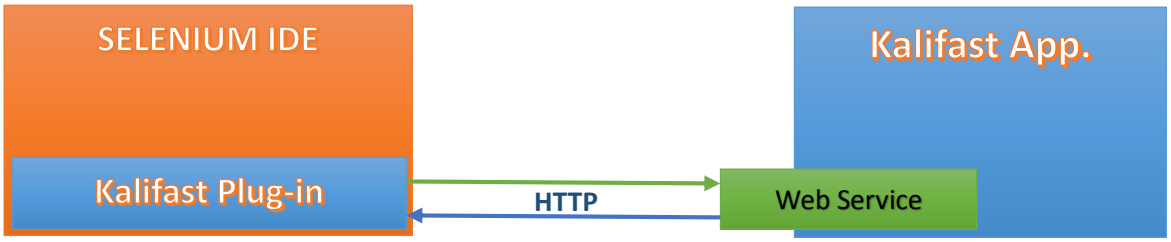
JAVASCRIPT

JavaScript (parfois abrégé en "JS") est un langage de script léger, orienté objet, principalement connu comme le langage de Scripting des pages web. Mais il est aussi utilisé dans de nombreux environnements extérieurs aux navigateurs web tels que node.js ou Couchbase.

Le standard pour JavaScript est ECMAScript. En 2012, tous les navigateurs modernes supportent complètement ECMAScript 5.1. Les anciens navigateurs supportent au minimum ECMAScript 3. Une 6e version majeure est en cours de préparation.

Architecture

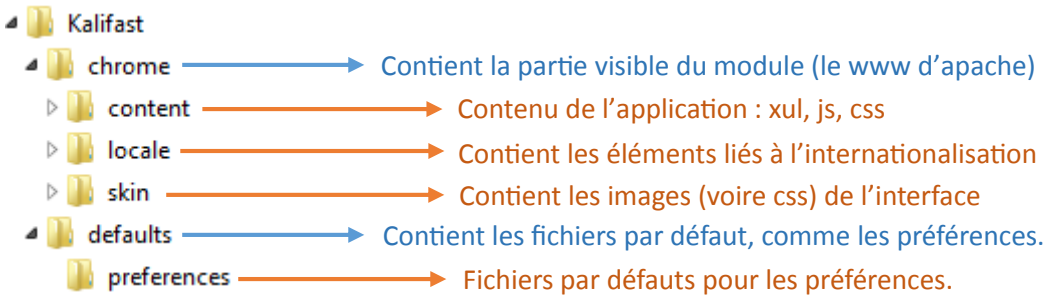
Architecture Globale



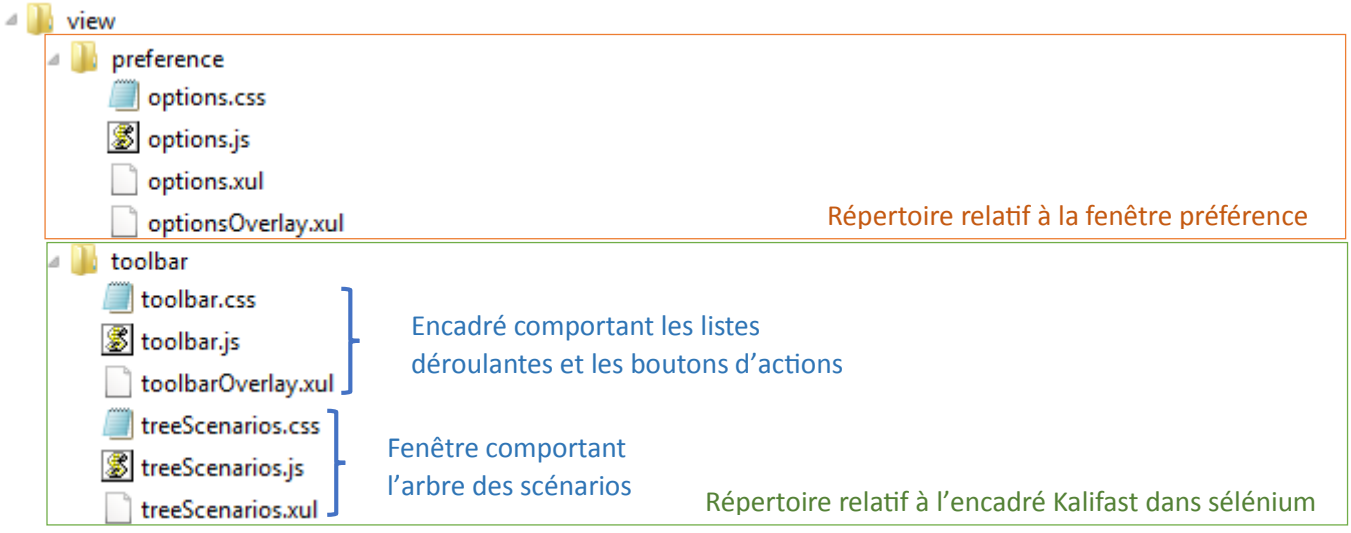
Le schéma ci-dessous traduit l'architecture du plug-in mais également son existence vis-à-vis de Sélénium IDE. Effectivement, il apparaît comme une surcharge du plugin et interagit avec la solution Kalifast par le biais de l'Ajax et de requêtes http.

Architecture de Fichiers

Le plug-in se présente sous la forme d'une archive XPI (comme toute extension Mozilla). Le schéma ci-dessous reprend l'architecture générale de fichiers de l'application :



Si nous parcourons un peu plus en détail le dossier content, ce dernier est constitué comme suit :



Dans le même dossier sont compris 4 fichiers javascripts gérant l'application :

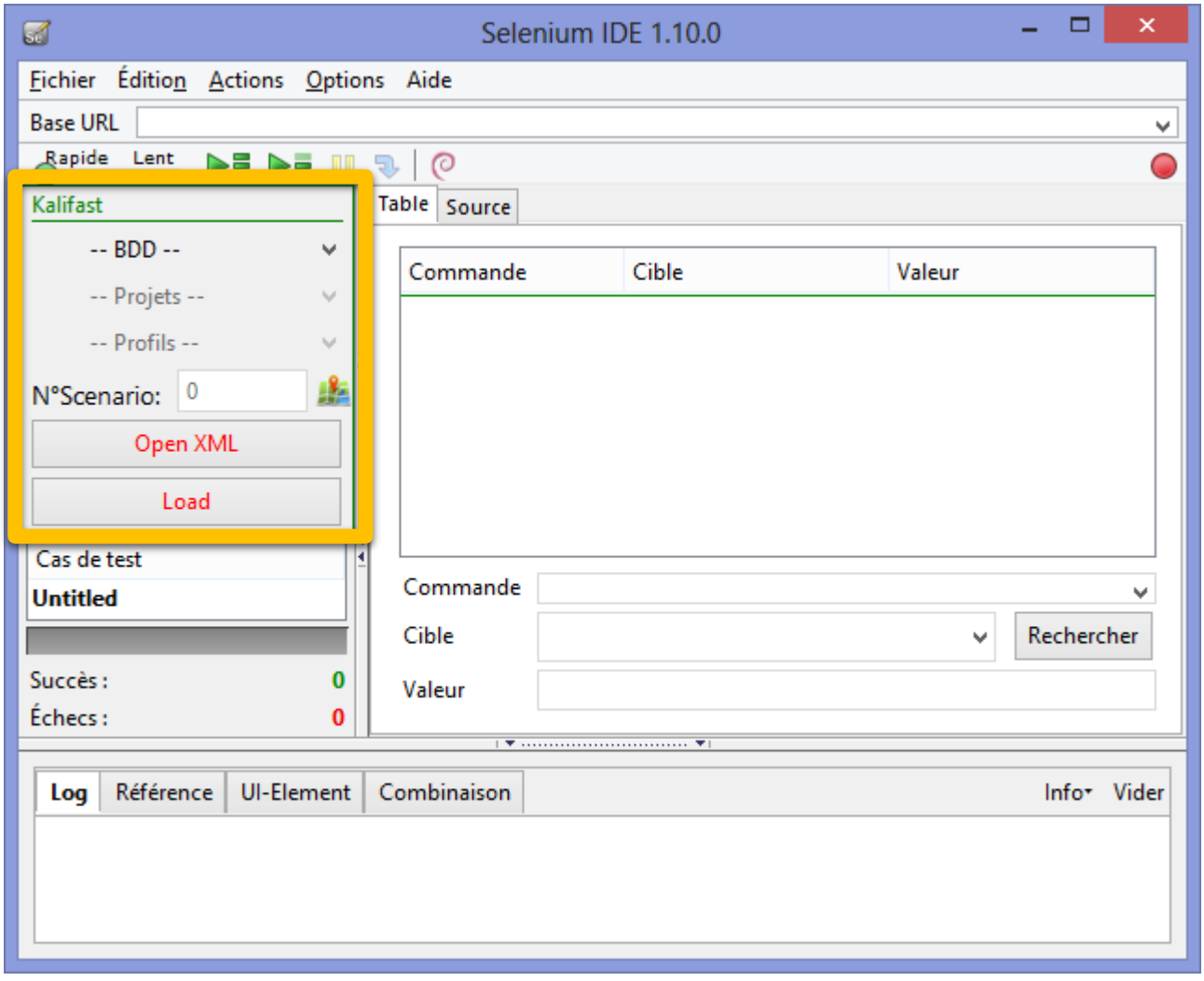
kalifast-application.js → Gère le comportement général de l'application.
kalifast-file-utils.js → Gestion des fichiers locaux/distants : lecture, écriture…
kalifast-format.js → Gestion du format Kalifast et transformation XML/XSL
preferences.js → Traitement sur validation des préférences

Deux fichiers très importants se trouvant à la racine du plug-in sont « chrome.manifest » et « install.rdf ». Ils sont indispensables.

Interface

Il est important de comprendre que le plug-in est encapsulé dans Sélénium IDE, de ce fait, on hérite de ses propriétés. Ainsi, pour ajouter des éléments IHM, il faut surcharger ses blocs.

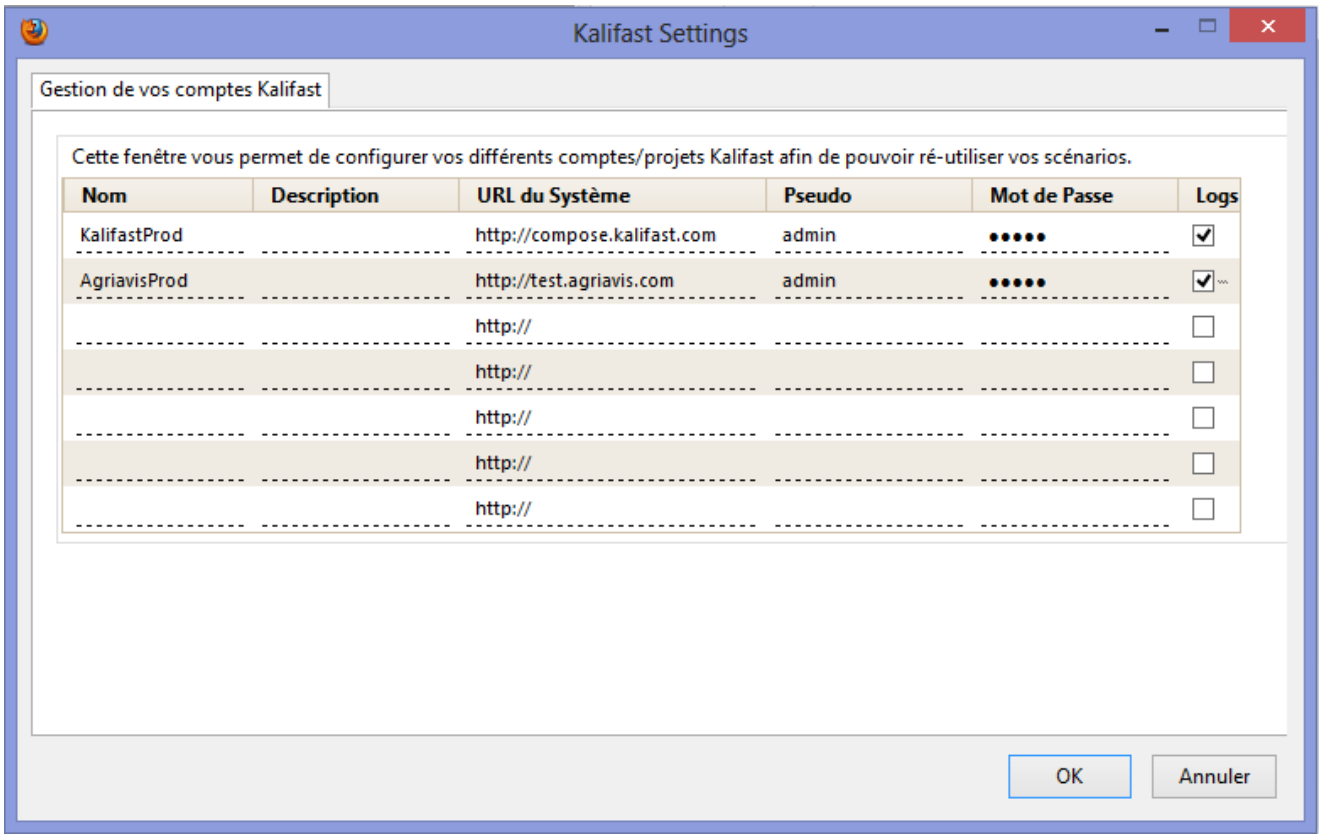
Accueil Sélénium IDE



L'interface ci-dessus est celle de Sélénium IDE. L'encadré jaune cible la partie où le plug-in Kalifast est intégré. Il contient 3 listes déroulantes : choix de la base de données (cf. préférences), choix du projet puis du profil. Une dernière étape étant le choix du scénario à charger. Deux méthodes sont possibles : saisir le

numéro de scénario directement ou cliquer sur la map et sélectionner dans le pop-up qui s'ouvre le scénario à ouvrir. Il suffit ensuite de cliquer sur load pour charger l'intégralité du scénario dans Sélénium IDE ;
Une autre alternative consiste à charger un document XML respectant la norme Kalifast avec les passages de paramètres à mentionner. (Pour plus d'informations, voir l'application Compose).

Préférences



La fenêtre des préférences est accessible via le menu « Options > Kalifast Options » ou directement dans la fenêtre des modules complémentaires en cliquant sur le bouton « Préférences » associé au plug-in.

Conception

Comme nous l'avons évoqué plus haut, pour paramétrer un plug-in Mozilla, il faut configurer deux fichiers à la racine du projet, à savoir : install.rdf et chrome.manifest. Le premier sert à paramétrer les informations sur le plug-in tandis que le second a davantage vocation à paramétrer l'application en elle-même et l'usage des URLs implicant l'application (chrome://).

```
content kalifast chrome/content/
locale kalifast en-US chrome/locale/en-US/
locale kalifast fr-FR chrome/locale/fr-FR/

skin kalifast classic/1.0 chrome/skin/classic/
style chrome://global/content/customizeToolbar.xul chrome://kalifast/content/view/toolbar/toolbar.css
overlay chrome://selenium-ide/content/selenium-ide-common.xul chrome://kalifast/content/view/preference/optionsOverlay.xul
overlay chrome://selenium-ide/content/selenium-ide.xul chrome://kalifast/content/view/toolbar/toolbarOverlay.xul
```

Indique que chrome://kalifast pointera sur le dossier chrome/content

Indique l'emplacement des locales

Surcharge de l'interface classic vers le dossier ciblé.

Surcharge CSS

Surcharge des layouts sélénium gérant le menu et l'interface

Nous allons maintenant détailler install.rdf :

```
<?xml version='1.0' encoding='UTF-8'?>
<RDF xmlns:em="http://www.mozilla.org/2004/em-rdf#" xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <Description about="urn:mozilla:install-manifest">
    <em:id>kalifast_selenium-ide@EISGE</em:id>
    <em:name>Kalifast (Selenium IDE)</em:name>
    <em:version>1.0</em:version>
    <em:creator>EISGE</em:creator>
    <em:description>A plugin for Selenium-IDE to play scenarios from our platform</em:description>
    <em:type>2</em:type>
    <!--Option Kalifast -->
    <em:optionsURL>chrome://kalifast/content/view/preference/options.xul</em:optionsURL>
    <em:homepageURL>http://kalifast.com/</em:homepageURL>

    <em:targetApplication>
      <Description>
        <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
        <em:minVersion>3.5</em:minVersion>
        <em:maxVersion>13.*</em:maxVersion>
      </Description>
    </em:targetApplication>

    <em:requires>
     <Description>
         <em:id>{a6fd85ed-e919-4a43-a5af-8da18bda539f}</em:id>
         <em:minVersion>1.0b2</em:minVersion>
         <em:maxVersion>1.*</em:maxVersion>
     </Description>
    </em:requires>
  </Description>
</RDF>
```

Nom officiel de l'application (Unique)

Nom du plug-in affiché dans le navigateur

Version du plugin

Le type d'application, ici, 2 pour extension

URL vers l'interface gérant les préférences

Application dans laquelle est incluse notre application. Ici, Firefox mais cela pourrait être thunderbird

Dépendances avec l'application. Ici, Sélénium IDE

Interface Préférences

Nous avons vu dans le fichier « chrome.manifest » que le menu de Sélénium avait été surchargé grâce à l'instruction « overlay chrome://selenium-ide/content/selenium-ide-common.xul chrome://kalifast/content/view/preference/optionsOverlay.xul ». L'objectif ici étant de pouvoir ajouter un menu dans l'interface de Sélénium IDE pour accéder aux préférences de Kalifast. Ainsi, le fichier optionsOverlay.xul va contenir notre menu supplémentaire :

```
<overlay id="selenium_overlay" xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
 <menupopup id="options-popup">
  <menuitem label="Kalifast Options..." accesskey="P"
        oncommand="window.open('chrome://kalifast/content/view/preference/options.xul','Kalifast Options','chrome=yes,,centerscreen=yes');" />
 </menupopup>
</overlay>
```

On lui indique ici de surcharger le bloc « selenium_overlay » et d'y ajouter un menu popup contenant l'item vers notre page des préférences.

Ainsi, dans options.xul, nous retrouvons l'interface permettant à l'utilisateur de saisir ses préférences :

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
<?xml-stylesheet href="chrome://kalifast/content/view/preference/options.css" type="text/css"?>

<!DOCTYPE prefwindow SYSTEM "chrome://kalifast/locale/options.dtd">

<prefwindow id="kalifast-prefs"
      title="Kalifast Settings"
      width="&windowWidth;"
      height="&windowHeight;"
      ondialogcancel="applyAndSaveChanges()"
      onload="loadPreferencesElements()"
      xmlns:html="http://www.w3.org/1999/xhtml"
      xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

  <script type="application/x-javascript" src="chrome://kalifast/content/preferences.js"/>
  <script type="application/x-javascript" src="chrome://kalifast/content/view/preference/options.js"/>

  <!-- Pref Pane -->
  <prefpane id="kalifast-panel" label="Kalifast Settings">
```

Déclaration des feuilles de style (le premier étant les feuilles de style du navigateur

Déclaration d'une fenêtre de préférence qui prend comme domaine d'internationalisation options.dtd

Création de la fenêtre de préférences avec un id & un titre (visible sur la fenêtre). On définit la taille de la fenêtre grâce aux variables définies dans options.dtd.

ondialogcancel & onload définissent les évènements qui surgissent lors du clic sur « annuler » et au chargement.

Le namespace global (xmlns) est basé sur un document XUL.

Déclaration des fichiers javascripts relatifs au chargement des options & sauvegarde des préférences.

Le code ci-dessous est la suite non-exhaustive qui permet d'enregistrer les préférences de l'utilisateur permettant d'associer et d'enregistrer des champs de formulaire dans ces variables. Ces dernières seront ensuite récupérable et enregistrées dans firefox (cf. about:config).

```
<prefpane id="kalifast-panel" label="Kalifast Settings">

  <preferences>
    <!-- Noms des connexions -->
    <preference id="pref_name0" name="extensions.kalifast.name.0" type="string" />
  </preferences>
```

Et ci-dessous un exemple d'association d'un champ de formulaire à cette préférence :

```
<html:tr>
    <html:td class="treeConnectionsKalifastName"><textbox preference="pref_name0" id="name.0" /></html:td>
</html:tr>
```

A retenir :

Les interfaces (fichiers XUL) fonctionnent comme des surcharges si elles ont été déclarées comme telles (overlay) dans le fichier chrome.manifest. Dans notre cas, on a choisi de surcharger selenium-ide-common.xul et l'interface générale selenium-ide.xul.

On peut ainsi hériter de toutes les variables javascript des applications parentes. Dans nos fichiers javascript, on aura accès plus ou moins à l'ensemble des variables déclarées par Sélénium. Il faudra bien gérer l'ordre d'importation des fichiers.

Notre fenêtre de préférences est constituée de deux fichiers javascripts : preferences.js & options.js. Nous allons tout d'abord observer le second qui gère l'action d'enregistrement des préférences tandis que le premier est plutôt un utilitaire. Nous observerons tout de même l'objet KalifastPlugin.PreferencesObserver qui y est déclaré.

```
if (KalifastPlugin.Preferences) {
   this.Preferences = KalifastPlugin.Preferences;
} else {
   this.Preferences = {
      getString: function(name, defaultValue) {
         return defaultValue;
      }
   };
}
```

Ci-dessus, la déclaration en haut du fichier options.js de notre objet KalifastPlugin.Preferences qui va stocker nos préférences. Cet objet sera accessible partout. Ensuite, on a créé la méthode applyAndSaveChanges() qui se charge de sauvegarder les valeurs saisies dans la fenêtre des préférences.

```
function applyAndSaveChanges(){
   var tableau_preferences = Array();
   var types = Array("name", "desc", "System.URL", "System.login", "System.password", "System.logWs");
   var elts, type, i;

   for(type in types){
      tableau_preferences[type] = new Array();
      for(i = 0; i <= 5; i++){
         if( document.getElementById(types[type]+"."+i).tagName == "textbox" ){
            tableau_preferences[type][i] = document.getElementById(types[type]+"."+i).value;
         }
         else{ tableau_preferences[type][i] = document.getElementById(types[type]+"."+i).checked; }
      }
   }

   this.Preferences.setArray(types[0], tableau_preferences[0]);
   …
   this.Preferences.setArray(types[5], tableau_preferences[5], "bool");
}
```

Ici, on possède un tableau contenant la liste des éléments présents dans notre fenêtre de préférences et on va parcourir chacun d'entre eux, créer un tableau de préférences et aller chercher dynamiquement en fonction de l'index sa valeur. Ainsi, on a vu plus haut un exemple de textbox dont l'identifiant était name.0. Sachant que l'on a 5 lignes, on va parcourir ce nombre de lignes et vérifier si un objet appelé name.{i} existe. Si c'est le cas, on l'ajoute à notre tableau.

Une fois le parcours de chacun de ces éléments réalisé et nos tableaux remplis, on les ajoutes à nos préférences via l'instruction : this.Preferences (sous-entendu KalifastPlugin.Preferences).setArray(le type (name), le tableau).

Ensuite, KalifastPlugin.PreferencesObserver contenu dans preferences.js sera notre objet chargé de nous prévenir dès qu'une modification est réalisée dans les préférences et de nous avertir. Ainsi, il est composé de trois méthodes :

- register : pour indiquer à Firefox d'utiliser notre observateur
- unregister : pour lui indiquer de ne plus tenir compte de notre observateur
- observe : action réalisée lors du changement de préférence chargeant ainsi notre tableau de listes de base de données dans l'interface UI Kalifast.

Volontairement, l'appel à « observe » n'a pas été mis en place pour le moment.

Interface Générale

```
<overlay id="toolbar_overlay" xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
    <script type="application/x-javascript" src="chrome://kalifast/content/preferences.js"/>
    <script type="application/x-javascript" src="chrome://kalifast/content/kalifast-format.js"/>
    <script type="application/x-javascript" src="chrome://kalifast/content/kalifast-file-utils.js"/>
    <script type="application/x-javascript" src="chrome://kalifast/content/kalifast-application.js"/>
    <script type="application/x-javascript" src="chrome://kalifast/content/view/toolbar/toolbar.js"/>

    <commandset id="xulKalifats-commandset">
      <command id="cmd_loadKalifatsFile" oncommand="chargerScenarioFromFile();"/>
    </commandset>

    <vbox id="suitePane">
      <vbox insertbefore="suiteTree" id="kalifastField">
            [ … ]
      </vbox>
    </vbox>
</overlay>
```

Tout comme notre fenêtre de preference, ici on lui indique que l'on surcharge un élément "toolbar_overlay" present dans l'interface Sélénium IDE. On lui indique l'ensemble de nos ressources javascript contenant : nos préférences, notre format, notre utilitaire de fichiers, notre application et l'interaction avec la toolbar (notre interface UI).
En orange, sont contenues les commandes qui seront utilisées dans les éléments xul et appelés via l'attribut « commande ». La seule commande définie permet de lancer une fenêtre de sélection d'un fichier (XML souhaité).

Ensuite, les vbox sont des surcharges des éléments présents dans l'interface Sélénium IDE et on indique qu'il faut ajouter la nôtre avant l'élément « suiteTree ». Ainsi, les éléments suivants seront intégrés dans l'interface de Sélénium IDE :

```
<label class="appname">Kalifast</label>
  <br />
  <!-- Choix Bdd -->
  <menulist id="kalifast-listeBdds" selectedItem="-1" class="kalifast-liste" insertbefore="kalifast-separator"
label="&kalifastIntituleListeBdds.tooltip;" tooltiptext="&kalifastIntituleListeBdds.tooltip;">
    <menupopup id="kalifast-listeBdds-menu">
      <menuitem label="&kalifastListeBdds.tooltip;" value="-1" />
    </menupopup>
  </menulist>

  <!-- Choix Projet -->
  <menulist id="kalifast-listeProjets" class="kalifast-liste" insertafter="kalifast-listeBdds"
label="&kalifastIntituleListeProjets.tooltip;" tooltiptext="&kalifastIntituleListeProjets.tooltip;">
    <menupopup id="kalifast-listeProjets-menu">
      <menuitem label="&kalifastListeProjets.tooltip;" value="-1" />
    </menupopup>
  </menulist>

  <!-- Choix Profil -->
  <menulist id="kalifast-listeProfils" class="kalifast-liste" insertafter="kalifast-listeProjets"
label="&kalifastIntituleListeProfils.tooltip;" tooltiptext="&kalifastIntituleListeProfils.tooltip;">
    <menupopup id="kalifast-listeProfils-menu">
      <menuitem label="&kalifastListeProfils.tooltip;" value="-1" />
    </menupopup>
  </menulist>
```

On a donc défini ici 3 listes déroulantes avec un ID nous permettant de le travailler en javascript. Dernière étape, le choix du scénario et les boutons d'actions :

```
<box class="kalifast-ligne-scenario">
  <label control="kalifast-inputScenarios">&kalifastIntituleNumScenarios.tooltip;:</label>
  <!—Textbox permettant de saisir le numéro de scénario kalifast -->
  <textbox id=”kalifast-inputScenarios” value=”” flex=”1” rows=”1” size=”10” class=”numberTxtBox” />
  <toolbarbutton id=”kalifast-button-openMap” flex=”1” class=”kalifast-button littleButton” disabled=”true”
image=”chrome://kalifast/skin/map.png” tooltiptext=”Open Map”/>
</box>
<!—Image Loader -->
<image id="loaderKalifastAjax" width="16" height="16" style="display:none; width: 16px; height: 16px;" />
<!—Bouton permettant de charger un fichier XML  -->
<button id="kalifast-button-openXML" class="kalifast-button" insertafter="kalifast-listeScenarios"
label="Open XML" tooltiptext="Open XML"/>
<!—Bouton permettant de charger en interne par web service -->
<button id="kalifast-button" class="kalifast-button" insertafter="kalifast-listeScenarios" label="Load"
tooltiptext="Load Default" command="cmd_selenium_testrunner"/>
```

Nous allons brièvement étudier les 3 différents processus pour charger un scénario :

[A venir…]