

Créez vos widgets Dashboard

par Romain Guy ([Romain Guy Home](#))

Date de publication : 21/06/2007

Dernière mise à jour : 21/06/2007

Dashboard est un nouveau bureau virtuel intégré à Tiger, la dernière version en date de Mac OS X. Quelques semaines ont suffi pour que des centaines d'applications Dashboard apparaissent sur Internet. Si vous n'avez pas encore trouvé votre bonheur il est temps de créer le vôtre.

Introduction

II - Structure des widgets

III - Création du widget

IV - Gestion des préférences

V - Code

VI - Screenshots

Introduction

Dashboard figure parmi les nouveautés les plus importantes de Tiger pour les utilisateurs de Mac OS X, aux côtés de Spotlight et d'Automator. Il s'agit d'un environnement d'exécution pour de mini applications appelées widgets, disponibles à tout moment dans un espace virtuel auquel vous pouvez accéder d'une simple touche. Si vous possédez Tiger, vous pouvez afficher le Dashboard et ses widgets en appuyant sur la touche F12. Ce nouvel espace, qui s'affiche en transparence par-dessus votre bureau, permet de ranger des applications comme une calculatrice, un calendrier, un dictionnaire, des notes, ... en résumé tous les outils dont vous avez besoin régulièrement mais que vous ne laissez pas ouverts pour ne pas encombrer votre espace de travail. Une caractéristique très intéressante de ces widgets est qu'ils ne sont pas de véritables applications, mais des pages Web exécutées en dehors d'un navigateur Internet. Quel que soit le widget que vous utilisez, vous interagissez avec du code HTML et du JavaScript. Il s'agit là d'une utilisation formidable du nouveau paradigme de développement qui fait fureur depuis quelques années, les clients légers. La simplicité de ces technologies permet en outre de créer facilement et rapidement ses propres widgets, surtout si vous savez développer des sites Web. Puisqu'il serait difficile de créer des outils complexes avec ses seules technologies, Dashboard propose également des API pour sauvegarder des préférences, localiser les widgets et accéder au système.

II - Structure des widgets

Les widgets Dashboard possèdent une structure semblable aux applications régulières. Ils apparaissent sous la forme d'un fichier portant l'extension `.wdgt` dans le Finder mais sont en réalité des répertoires contenant au moins quatre fichiers : un fichier `Info.plist` décrivant le widget, un document HTML et deux images PNG. La plupart des widgets utilisent en outre une feuille de style CSS. Le [listing 1](#) présente le contenu de notre widget `#ProgX Library`. Parmi les 4 images présentes, seules `Default.png` et `Icon.png` sont indispensables. La première est affichée lorsque vous glissez votre widget depuis la barre de sélection sur l'environnement Dashboard. Cette image est affichée pendant le chargement du widget et permet d'en simuler la présence. La seconde image, `Icon.png`, correspond à l'icône décrivant le widget dans la barre de sélection.

Pour créer le fichier `Info.plist`, lancez l'application `Property List Editor`, créez un nouvel élément racine puis ajoutez les cinq éléments présentés dans le [listing 2](#). Ces propriétés donnent le nom du widget à afficher, son identificateur unique, son nom de bundle (qui doit être celui du dossier `.wdgt`), sa version et le nom du fichier HTML principal. Certains paramètres optionnels sont également disponibles. Vous pouvez ainsi modifier la taille du widget, qui est par défaut la même que `Default.png`, en créant les éléments `Width` et `Height`.

III - Création du widget

Le widget #ProgX Library que nous allons étudier affiche une liste de livres, classés par genre, auteur et titre, téléchargée sur un site Internet. Le [listing 3](#) contient son code HTML. Pour simplifier le développement, toutes les propriétés de présentation se trouvent dans la feuille de style Library.css, importée dans l'en-tête du document. Le script Library.js contient pour sa part toute la logique applicative. Outre une organisation plus claire, cette séparation permet d'obtenir un document HTML très simple et facile à maintenir. Il contient ici une image, qui fera office d'arrière plan, et quatre conteneurs `</div>` contenant respectivement la liste des genres, la liste des auteurs, la liste des titres et une description détaillée du livre sélectionné.

Comme nous travaillons dans un environnement connu, nous n'avons pas à nous soucier de la gestion de différentes résolutions ou des moteurs de rendu HTML des navigateurs. Chaque widget utilise le WebKit, le moteur de rendu de Safari. De fait, vous pouvez simplement essayer votre code directement dans Safari avant de l'utiliser en tant que véritable widget. #ProgX Library place ainsi tous ses composants de manière absolue dans le widget comme le montre l'extrait de la CSS présenté dans le [listing 4](#). En utilisant des positions absolues nous pouvons ainsi manipuler l'ordre d'affichage des éléments. Le `</div>` contenant le livre sélectionné se trouve ainsi placé toujours placé par-dessus l'image de fond.

Le code JavaScript de l'application est le même que celui présenté dans le numéro de juillet/août dans l'article intitulé "Redécouvrez le web avec AJAX". La technologie AJAX est largement répandue dans la réalisation de widgets puisqu'elle permet d'interagir efficacement avec des sites Web. Le [listing 5](#) contient un extrait du code utilisé pour contacter le site <http://www.progx.org> et récupérer les listes de tous les genres, auteurs et titres. L'objet XMLHttpRequest émet une requête GET ou POST à une URL donnée et fournit la réponse sous forme d'un arbre DOM, ici la variable doc. Le JavaScript n'a donc qu'à parcourir cet arbre, extraire les données recherchées et modifier le code HTML du widget, mais vous pouvez également manipuler l'arbre DOM :

```
document.getElementById("genres").innerHTML = code1;
```

Dashboard propose en outre des API JavaScript spécifiques aux widgets. Pour rafraîchir régulièrement le contenu du widget sans obliger l'utilisateur à cliquer sur un bouton particulier, nous utilisons les événements onshow et onhide de l'objet widget, deux fonctions invoquées quand le widget est affiché ou caché. Le [listing 6](#) contient le code JavaScript qui invoque la fonction de remplissage des listes dès que le widget apparaît. Le test if (widget.onshow) est nécessaire pour éviter la levée d'exceptions lorsque vous testez votre code dans Safari. Vous pouvez également gérer le déplacement du widget par l'utilisateur, avec ondragstart et ondragstop, et savoir quand il est retiré de l'environnement Dashboard, avec onremove.

Vous pouvez enfin tester le widget en copiant le dossier .wdgt dans le dossier ~/Library/Widgets ou /Library/Widgets. Invoquez Dashboard puis ouvrez la barre des widgets pour l'installer. Vous pouvez vérifier que l'icône correspond à Icon.png et que l'image affichée lorsque vous installez le widget est Default.png. En essayant votre widget en conditions d'utilisation réelles, vous rencontrerez sûrement des problèmes. Par exemple, #ProgX Library ne parviendra jamais à récupérer les données depuis le serveur distant. Dashboard propose en effet un environnement sécurisé dans lequel les widgets doivent demander des droits d'accès spéciaux. Il existe six permissions que vous devez ajouter à Info.plist comme des éléments booléens :

AllowFileAccessOutsideOfWidget, AllowFullAccess, AllowInternetPlugins, AllowJava, AllowNetworkAccess et AllowSystem. Notre widget n'a besoin que d'AllowNetworkAccess.

Invoquer constamment Dashboard peut se révéler particulièrement fastidieux lorsque vous devez résoudre des problèmes dans votre code. Mac OS X propose un paramètre spécial qui permet d'utiliser les widgets directement sur le bureau. Pour l'activer, ouvrez un terminal et saisissez la commande suivante :

```
defaults write com.apple.dashboard devmode YES
```

Après avoir relancé le Finder, en vous déconnectant par exemple, vous pouvez invoquer Dashboard, commencer à déplacer un widget, fermer Dashboard et relâcher le widget sur le bureau. Vous pouvez en outre recharger un widget sans le fermer en lui donnant le focus puis en appuyant sur les touches Pomme-R.

IV - Gestion des préférences

De nombreux widgets conservent leur état même si vous les fermez ou proposent des options au dos. Toute la gestion des préférences est réalisée en JavaScript grâce à deux méthodes l'objet widget : `setPreferenceForKey()` et `preferenceForKey()`. La première permet d'enregistrer une valeur identifiée par une clé tandis que la seconde récupère une valeur à partir d'une clé, comme dans l'exemple suivant :

```
if (window.widget) {  
    window.setPreferenceForKey("server", "http://www.progx.org");  
    var baseURL = window.preferenceForKey("server");  
}
```

Bien que rien ne vous y contraigne, Apple préconise d'afficher les options au dos du widget. Vous devez pour cela intégrer les deux faces du widget dans le même code HTML en alternant l'affichage de chacune. Le [listing 7](#) présente la fonction `showPrefs()` utilisée par le widget. Ce code prépare le widget pour la transition puis invoque `widget.performTransition` pour afficher l'animation elle-même. Vous trouverez de nombreux exemples semblables dans la documentation d'Apple pour Dashboard, comme la gestion du dimensionnement des widgets, l'animation des boutons ou la création de contrôles personnalisés pour remplacer les composants Aqua. Nous vous conseillons également de consulter le contenu des widgets les plus intéressants que vous trouverez sur le Web pour apprendre par exemple à invoquer du code Python ou Perl.

V - Code

*

Listing 1

```
ProgXLibrary.wdgt\  
  Library.css  
  Library.html  
  Library.js  
  Info.plist  
  Book.png  
  Default.png  
  Icon.png  
  Library.png
```

*

Listing 2

```
CFBundleDisplayName: #ProgX Library  
CFBundleIdentifier: org.progx.widget.Library  
CFBundleName: #ProgX Library  
CFBundleVersion: 1.0  
MainHTML: Library.html
```

*

Listing 3

```
<html>  
  <header>  
    <script type="text/javascript" src="Library.js"></script>  
    <style type="text/css">  
      @import "Library.css";  
    </style>  
  </header>  
  <body onload="setup()">  
      
  
    <div id="genres" class="list">  
    </div>  
  
    <div id="authors" class="list">  
    </div>  
  
    <div id="titles" class="list">  
    </div>  
  
    <div id="selectedBooks">  
        
    </div>  
  </body>  
</html>
```

*

Listing 4

```
#selectedBooks {  
  width: 190px;  
  height: 178px;  
  /* ... */  
  position: absolute;  
  top: 22px;  
  left: 400px;  
  /* ... */  
  z-index: 5;  
}
```

*

Listing 5

```
function fillLists() {
    var xmlRequest = new XMLHttpRequest();
    var url = baseUrl + "/bookshelfEngine.php?init=true";
    xmlRequest.open("GET", url , true);
    xmlRequest.onreadystatechange = function() {
        if (xmlRequest.readyState == 4) {
            var doc = xmlRequest.responseXML;
            // transformation et affichage
        }
    };
    xmlRequest.send(null);
}
```

*

Listing 6

```
function onshow() {
    fillLists();
}

if (window.widget) {
    widget.onshow = onshow;
}
```

*

Listing 7

```
function showPrefs() {
    var front = document.getElementById("front");
    var back = document.getElementById("back");

    if (window.widget) {
        widget.prepareForTransition("ToBack");
    }

    front.style.display = "none";
    back.style.display = "block";

    if (window.widget) {
        setTimeout ('widget.performTransition();', 0);
    }
}
```

VI - Screenshots



Dashboard est un bureau supplémentaire sur lequel vous pouvez déposer des outils appelés widgets.



Un widget est un simple répertoire contenant un document HTML.



Lorsque vous installez votre widget, Dashboard utilise l'image Default.png.



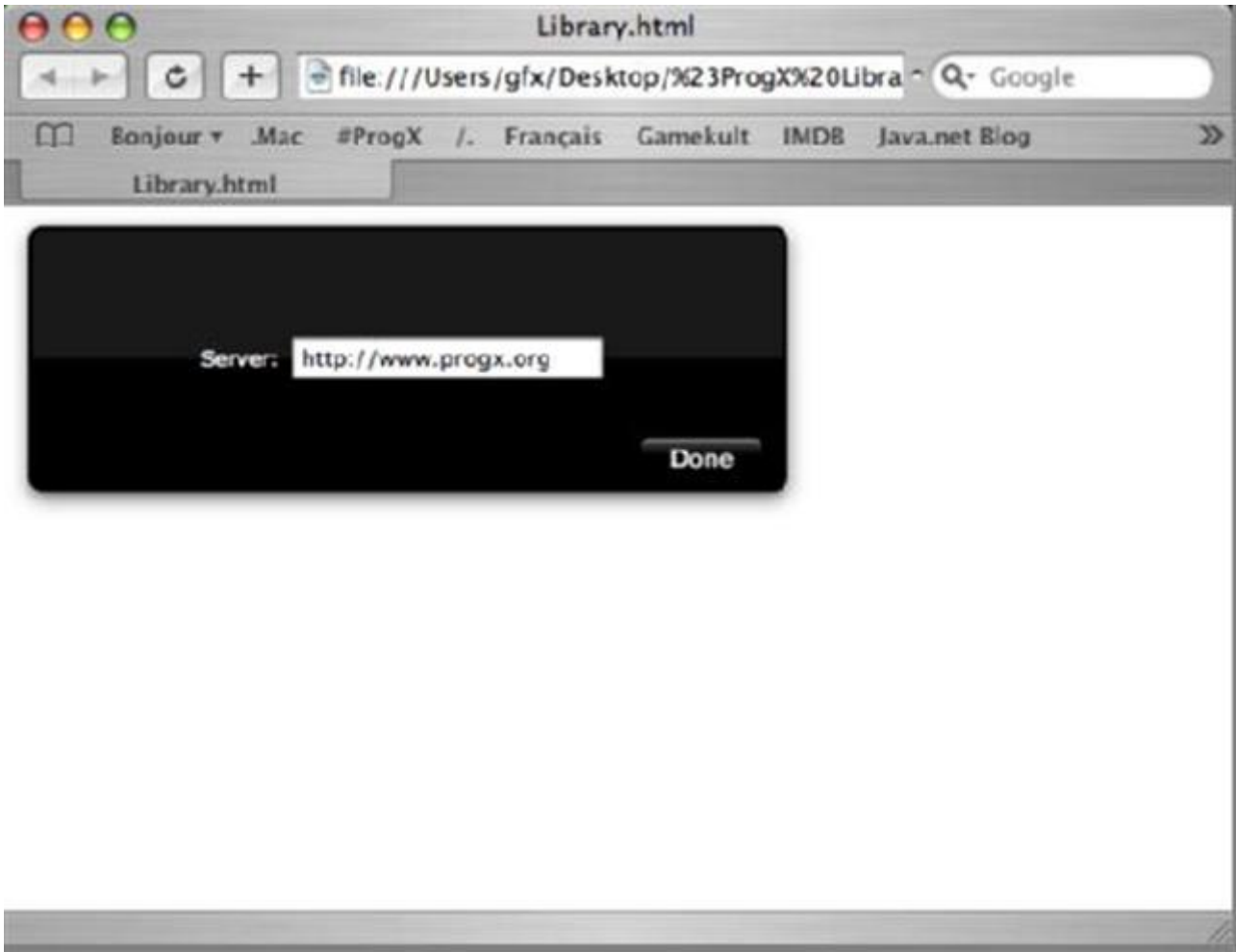
Le widget #ProgX Library affiche une liste de livres téléchargée depuis un serveur.



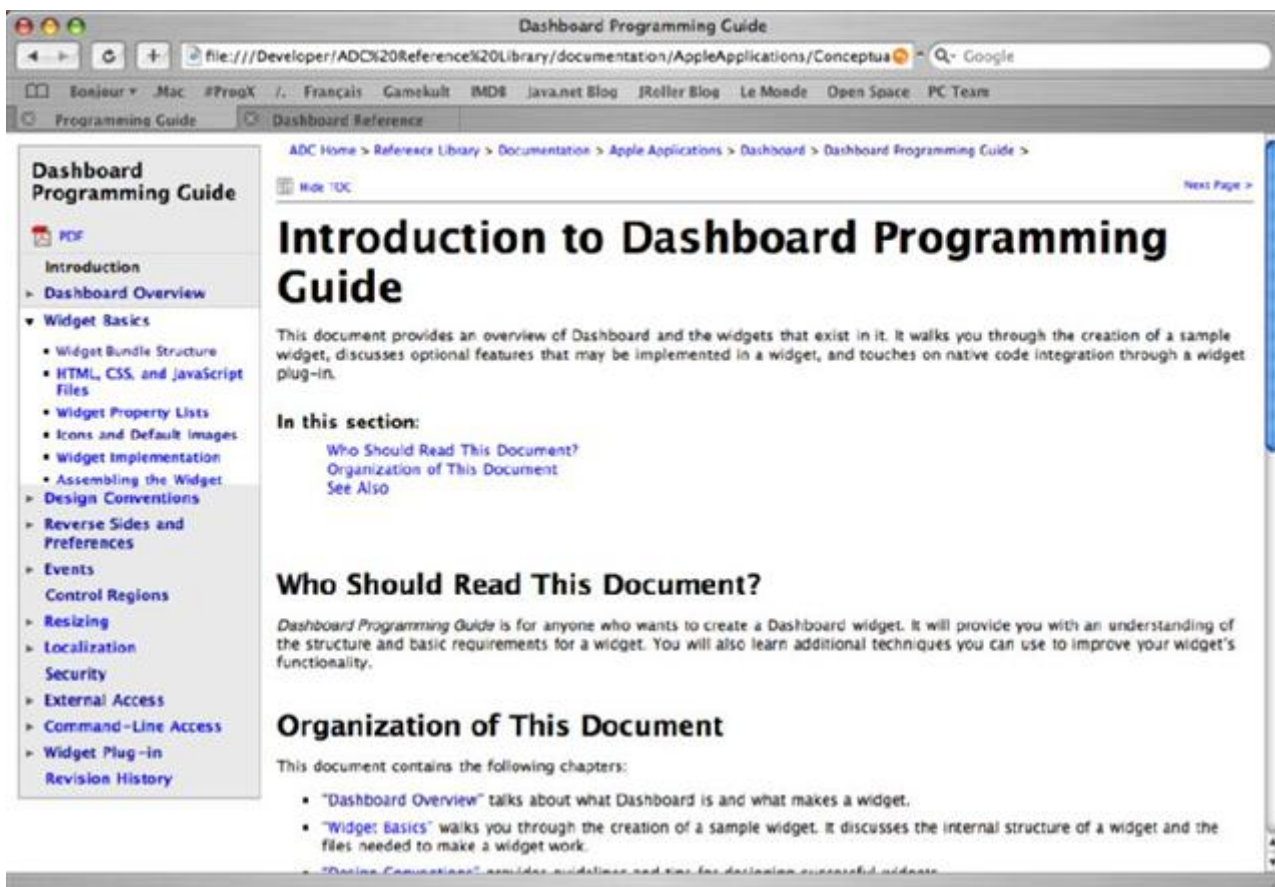
Le site d'Apple propose des centaines de widgets téléchargeables gratuitement.



Une option spéciale de Mac OS X permet de déposer les widgets sur le bureau.



Les options des widgets se trouvent sur leur face arrière.



L'Apple Developer Connection propose des documents de référence très complets pour Dashboard.

