

HTML5 offline browsing and storage

Mohamad Monzer

PLAN

<u>INTRODUCTION</u>	<u>3</u>
<u>HTML5 APPLICATION CACHE.....</u>	<u>4</u>
<u>AVANTAGES DE L'UTILISATION DU CACHE WEB</u>	<u>4</u>
<u>SUPPORT NAVIGATEUR</u>	<u>4</u>
<u>UTILISATION DE LA CACHE D'APPLICATION</u>	<u>5</u>
<u>CREATION DU FICHIER MANIFEST</u>	<u>6</u>
<u>EXEMPLE DU FICHIER MANIFEST</u>	<u>6</u>
<u>MISE A JOUR DU CACHE</u>	<u>7</u>
<u>ETATS DE LA CACHE D'APPLICATION.....</u>	<u>8</u>
<u>RESUME DE LA PARTIE CACHE DE L'APPLICATION</u>	<u>9</u>
<u>BESOIN DES OUTILS COMPLEMENTAIRE POUR LE TRAVAIL SANS CONNEXION.....</u>	<u>9</u>
<u>HTML5 LOCAL STORAGE.....</u>	<u>10</u>
<u>SUPPORT NAVIGATEUR</u>	<u>11</u>
<u>LOCAL STORAGE ET SESSION STORAGE.....</u>	<u>11</u>
<u>POINTS FORTS ET POINTS FAIBLES DU LOCAL STORAGE.....</u>	<u>12</u>
<u>INDEXED DATABASE (INDEXEDDB).....</u>	<u>12</u>
<u>AVANTAGES.....</u>	<u>12</u>
<u>INCONVENIENT.....</u>	<u>13</u>
<u>EXEMPLE D'ARCHITECTURE D'UNE APPLICATION WEB QUI SUPPORT LE TRAVAIL HORS CONNEXION</u>	<u>13</u>
<u>CONCLUSION</u>	<u>14</u>
<u>FIGURES.....</u>	<u>15</u>

Introduction

L'internet dans nos jours est devenu une nécessité pour la plupart des gens. On trouve ainsi des projets qui visent à partager l'internet même avec les pays tiers monde gratuitement.

Et grâce aux bénéfices de l'internet, un grand nombre des entreprises ont changé leurs utilisations des applications du bureau vers les applications web.

Une application web est une application accessible aux utilisateurs à partir d'un navigateur web (ou bien un agent spécifique), en utilisant des requêtes http avec des liens spécifiques qui seront mappés vers des serveurs qui à leur tour traduisent les demande des utilisateurs et les renvoient des pages html qui peuvent être afficher par le navigateur.

L'application web peut être composé de plusieurs couches (selon l'architecture utilisée pour la création de cette application). Un exemple typique est une application composées des trois couches : présentation, métier (business) et data, comme c'est montré dans la Figure 5: Architecture trois tiers.

La couche présentation contient l'interface utilisateur, la couche métier contient la partie logique de l'application, tandis que la couche data s'occupe de la gestion des données et la communication avec les bases des données utilisées.¹

Les utilisateurs de l'internet sont habitués à la notion qu'une application web nécessite avoir une connexion à internet pour qu'elle peu être utilisée.

Cependant, un grand nombre des applications web sont utilisés par les utilisateurs d'une façon régulière, et l'utilisateur de ces applications demande généralement les mêmes ressources chaque fois (images, voix, vidéo, et même des données stockées dans des bases existant chez le serveur).

Et ici plusieurs questions peuvent être demandées :

1. Est-ce-t' il vrai qu'une application web nécessite toujours de l'internet ?
2. Pourquoi ne pas sauvegarder les ressources qui doivent être utilisés fréquemment par le client?

¹ <https://msdn.microsoft.com/en-us/library/ee658099.aspx>

HTML5 Application Cache

Avec html5, on peut aujourd'hui créer une version déconnectée de l'application web en utilisant le cache de l'application « application cache ».

Avantages de l'utilisation du cache web

« Application cache » est le processus de la mise en cache des documents web tel que les médias, les pages web (html, scripts, librairies, ..) ce qui peut être utile pour :

Réduire la consommation de la bande passante, puisque les ressources de la page web seront téléchargées uniquement la première fois qu'on ouvre la page, ensuite le navigateur n'a besoin que de télécharger les changements et les mises à jour des ressources.

Réduire la charge du serveur web, puisque les clients vont travailler sans connexion, donc sans l'intervention du serveur pour chaque événement, donc ces clients ne nécessitent plus de ressources tel que le processeur et la mémoire centrale.

Naviguer sans connections, puisque le client a déjà téléchargé toutes les ressources nécessaires pour que l'application (ou bien une partie de cette application) puisse travailler, donc il pourrait utiliser cette application même s'il n'a pas une connexion à internet.

Diminuer le temps nécessaire pour la consultation des pages web, puisque le chargement des ressources déjà téléchargées nécessite un temps négligeable en le comparant au temps du téléchargement de ces ressources.

Support navigateur

Le tableau suivant montre les versions depuis lesquelles chaque navigateur du Desktop commence à supporter le cache web.






API					
Application Cache	4.0	10.0	3.5	4.0	11.5

Figure 1 : support navigateur²

Pour Google chrome: la version 4.0 est disponible depuis l'année 2010

Pour IE : la version 10.0 est disponible depuis l'année 2012

Pour Mozilla : la version 3.5 est disponible depuis l'année 2009

Pour Safari: la version 4 est disponible depuis l'année 2009

Pour Opéra : la version 11.5 est disponible depuis l'année 2011.

Encore pour les mobiles, le tableau suivant montre les versions des navigateurs qui supportent le cache web :

Feature	Android	Firefox Mobile (Gecko)	Firefox OS	IE Mobile	Opera Mobile	Safari Mobile
Basic support	2.1	(Yes)	1.0.1	11.0	11.0	3.2

Figure 2 : support navigateur pour sur mobile³

Donc on peut dire que cette propriété est maintenant supportée par les navigateurs les plus utilisés dans nos jours.

En plus,

Utilisation de la cache d'application

Le cache d'application repose principalement sur l'utilisation d'un fichier « manifest » et à l'ajouter aux pages HTML du site web.

Pour activer le cache d'application dans une page web, il suffit d'ajouter un attribut « manifest » avec le chemin vers le fichier manifest à la balise <html>, comme c'est montré dans l'exemple suivant :

² http://www.w3schools.com/html/html5_app_cache.asp

³ https://developer.mozilla.org/en-US/docs/Web/HTML/Using_the_application_cache

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
...
</html>
```

Chaque page contenant l'attribut manifest serait mise en cache quand l'utilisateur la visite.

Note : une recommandation par le standard est d'utiliser l'extension « .appcache » pour le fichier manifest.

Création du fichier manifest

Le fichier manifest est un fichier texte, dans le quel on déclare les ressources qui doivent être mises en cache et celles qui doivent être téléchargées à chaque fois du serveur.

Ce fichier comprend 3 sections dont la première est requise :

Cache Manifest : dans cette section on spécifie les fichiers qui doivent être mis en cache.

Network : dans cette section on spécifie les ressources qui doivent être prises à chaque fois du serveur.

Fallback : dans cette section on choisit une page html qui doit être utilisée par le client en cas où on ne peut pas établir une connexion à internet.

Note : on peut utiliser « # » pour mettre une ligne en commentaire.

Exemple du fichier manifest

CACHE MANIFEST

#2012-02-21 v1.0.0

#ici on place les ressources qui doivent être mises en cache

/theme.css

/logo.gif

/main.js

NETWORK :

#ici on place les ressources et les pages qui nécessitent une connexion à internet

login.asp

#on peut toujours mettre une étoile « * » pour dire « toutes les autres ressources doivent
#être mise en cache »

*

FALLBACK :

/html/ /offline.html

Note : il faut faire attention de ce qu'on va mettre en cache, car lorsqu'un fichier est mise en cache, le navigateur continue à utiliser la version mise en cache du fichier, même si ce fichier change chez le serveur.

Mise a jour du cache

Il existe 3 méthode pour changé se qui est mise en cache par le navigateur :

1. L'utilisateur efface la cache du navigateur
 - Par exemple en Google chrome (pour les utilisateur de Windows) : si l'utilisateur tape sur son clavier « ctrl » + « alt » + « del », puis il click sur le bouton « clear browsing data » après avoir choisir « cached images and file » de la liste des option.
2. Le fichier manifest change et **non pas les ressources qui sont misent en cache**
 - Autrement dit : si on a mis en cache une image nommé logo.gif, puis on l'a par une autre, et on n'a pas changé le fichier manifest, le client de cette application ne va pas avoir la nouvelle image, mais il va utilisé celle qui est caché dans son navigateur.
3. Le cache application est mis à jour par le code.
 - **Note :** les commentaires utilisés dans le fichier manifest peuvent avoir un autre objectif que de créer un code bien organisé et compréhensible.
 - **Par exemple :** Le changement de la date et de la version dans une ligne de commentaire est une autre méthode pour demander au navigateur de remettre a jour les fichier misent en cache.

La documentation de Firefox dit que le meilleur entraînement (best practice) est de changer la version mise en commentaire pour mettre à jour le fichier manifest, après un changement dans une des ressources.⁴

Faire attention :

Un changement dans le fichier manifest n'aboutit pas directement à une mise à jour de la cache, mais nécessite une demande de mise à jour de la part client, qui se fait en général par un rechargement de la page, ou bien par la commande « `applicationCache.update()` » ;

États de la cache d'application

La cache d'application possède un statu qui indique son état à un moment donné, qui est un des suivants :

- **UNCACHED** : une valeur qui indique qu'un objet de la cache de l'application n'est pas complètement chargée.
- **IDLE** : Le cache de l'application n'est pas dans le processus de mise à jour.
- **CHECKING** : le manifest est entrain d'être récupéré pour vérifier le besoin de mettre à jour la cache.
- **DOWNLOADING** : les ressources sont en cour de téléchargement pour être ajouté à la cache à cause d'un changement dans les ressources du manifest.
- **UPDATEREADY** : ce statu indique qu'on a une nouvelle version du manifest

Remarque importante : toutes les caches qui partagent le même manifest possèdent le même état.

Pour la vérification du statu courant de la cache, appeler la fonction : « `applicationCache.status` ».

Pour demander une mise à jour du statu, appeler la fonction :

⁴ http://www.w3schools.com/html/html5_app_cache.asp

« `applicationCache.update()` ».

En plus, on peut ajouter un auditeur de statu (listener) pour faire des évènements automatiquement lors du changement de l'état de la cache.

La figure ci-dessous montre l'exemple de l'auditeur du statu « `updateready` » qui lance un alerte qu'une nouvelle version existe.⁵

```
1 function onUpdateReady() {  
2     alert('found new version!');  
3 }  
4 window.applicationCache.addEventListener('updateready', onUpdateReady);  
5 if(window.applicationCache.status === window.applicationCache.UPDATEREAD  
6     onUpdateReady();  
7 }
```

Figure 3 : auditeur du changement de l'état de la cache

Résumé de la partie Cache de l'application

La cache de l'application est une belle option apporté pas HTML5, qui utilise un fichier manifest qui énumère les ressources qu'on veut les mettre en cache chez le client, pour qu'elles soient utilisées dans les prochains appels sans avoir à les ré-télécharger du serveur un deuxième fois (si elles ne sont pas changé), ce réduit la consommation de la bande et diminue le temps nécessaire pour la consultation de la page web, et qui permet même de consulter ces ressources sans connexion à internet.

Besoin des outils complémentaire pour le travail sans connexion

Donc jusqu'à maintenant, est-ce qu'on peut dire qu'on peut avoir une application web qui peut être utiliser complètement sans connexion ?

Pour répondre à cette application, prenant l'exemple des deux scénarios qui suivent :

⁵ https://developer.mozilla.org/en-US/docs/Web/HTML/Using_the_application_cache

Dans le premier scénario : on a une application qui est un montre des images, des rapports, et même des audio qui sont rarement changé, et on a décidé d'offrir à nos clients la possibilité de consulter notre site web même s'ils n'ont pas une bonne connexion à internet.

Alors dans ce cas, l'utilisation de la cache de l'application peut être parfait.

Mais d'un autre scénario, où on a une application qui contient une base de donnée, et qui nécessite une interaction avec le client, le cache d'application ne peut être suffisant pour offrir une application qui peut travailler complètement sans connexion.

Par exemple : si le client veut à besoin de faire des demande à la base de données, d'ajouter et d'effacer des informations même s'il est perd sa connexion, on veut qu'il puisse travailler et on synchronise sont travail lors de la retour de la connexion, la cache d'application n'est pas suffisant toute seule, puisque le téléchargement des ressources ne lui donne pas un accès aux données stocké dans la base de données, donc il va perdre les fonctionnalités principaux de l'application.

Donc, on conclut qu'on a besoin d'un stockage des informations chez le client, pour profiter en plus de la cache afin d'avoir plus d'utilisation pour cette propriété.

HTML5 Local Storage

Lorsque vous ouvrez le site w3school.com sur la partie du local storage, vous trouvez dans la première ligne la phrase suivante : « HTML local storage, better than cookies ».

Alors, qu'est-ce que l'HTML local storage, et pourquoi sa comparaison avec les cookies ?

Premièrement, on peut dire que cookies, étaient la seule option pour le stockage des informations chez le client, mais elle avait beaucoup des limites :

Les cookies avaient une taille trop limitée (maximum 4 kb), en plus qu'elle doit être incluse dans chaque requête vers le serveur, tandis que HTML5 local storage offre une taille plus grande (minimum 5MB), sans avoir à inclure ses informations dans les requêtes du serveur.

En plus, Local storage est par origine, donc toutes les pages d'une même application partagent l'accès à ses informations stockées localement chez le client, et qui n'ont pas de date d'expiration.

Support navigateur

Local storage est supporté par les navigateurs depuis les versions suivants :






API					
Web Storage	4.0	8.0	3.5	4.0	11.5

Figure 4 : local storage support des navigateurs⁶

En plus, on peut vérifier le support des navigateurs par code, en vérifiant la condition suivante :

```
type(Storage) !== 'undefined'
```

Local storage et Session storage

HTML local storage offre deux objets pour le stockage des informations chez le client :

1. `Window.localStorage` pour stocker des informations sans date d'expiration
2. `Window.sessionStorage` pour les informations qui doivent être stocké pour une session, donc qui seront effacés lors de la fermeture de la tab du navigateur.

Local storage stocke les informations sous-forme de clé/valeur.

Pour ajouter une information peut être faite par une des méthodes suivantes :

« `localStorage.clé = valeur` » ou bien « `localStorage.setItem('clé', 'valeur')` »

« `sessionStorage.clé = valeur` » ou bien « `sessionStorage.setItem('clé', 'valeur')` »

Et la méthode `removeItem('clé')` pour les effacer.

⁶ http://www.w3schools.com/html/html5_webstorage.asp

Points forts et points faibles du local storage

Points forts	Points faibles
Est supporté par tous les navigateurs modernes, en plus de l'androïde et l'iOS	Mauvaise performance pour les larges et complexe data. Pas d'indexation
Simple utilisation : <code>localStorage.key = value ;</code>	Pas de structuration des données, ce qui nécessite une vérification de la cohérence et l'intégrité des données

Donc, on peut dire que l'utilisation du local storage est efficace pour le stockage des informations précis tel que les préférences, les paramètres de l'application, ...

Indexed Database (IndexedDB)

Indexed Database est une collection des « object stores » où vous pouvez simplement déposer des objets.

Les stores sont quelque chose similaire aux tables de l'SQL, sans les contraintes de la structuration des objets. Donc l'IndexedDB est similaire au localStorage avec l'avantage de la possibilité d'avoir autant de bases de données dont on a besoin, et autant des stores dans chaque base de données.

La différence entre IndexedDB et le localStorage est que l'IndexedDB offre un API asynchrone, avec la possibilité de créer des Indexes pour augmenter la vitesse de recherche.

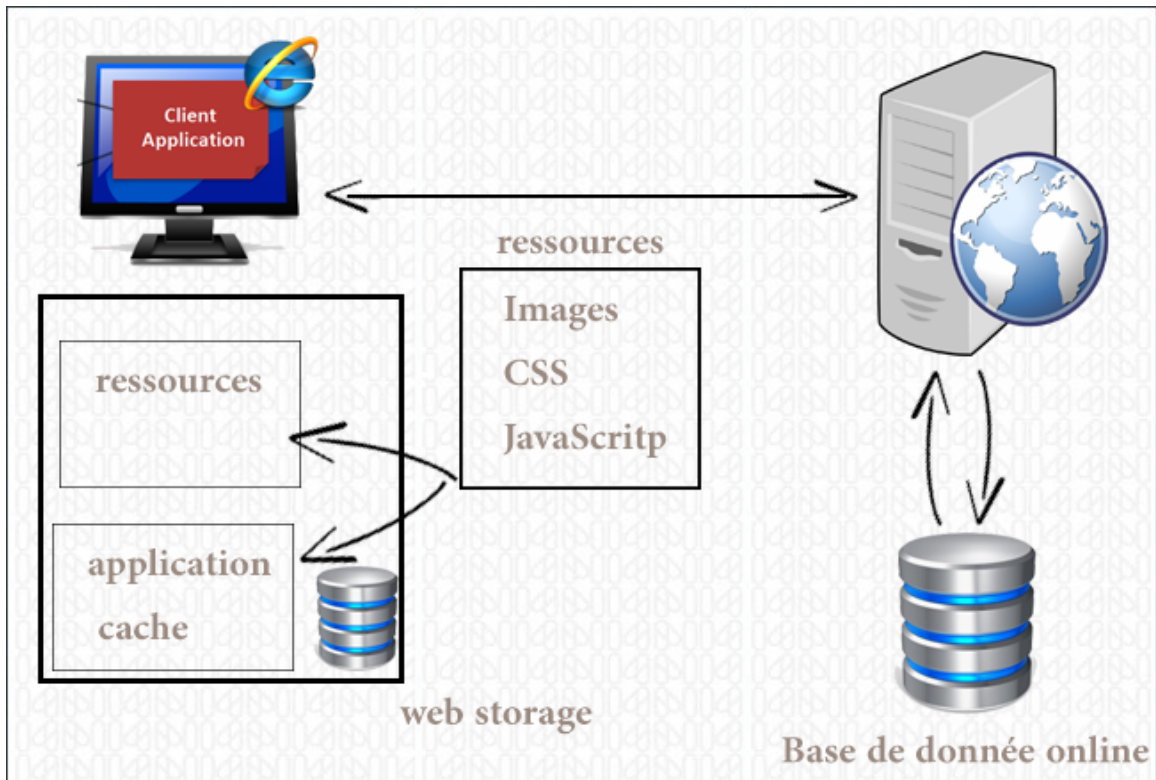
Avantages

1. Une bonne performance, due de l'utilisation des api asynchrone qui ne freeze pas l'utilisateur après chaque requête à la base de donnée.
2. Bonne vitesse de recherche, du par l'indexation des données.
3. Support du versioning.
4. Robuste puisqu'il supporte le model transactionnel des bases de données.
5. Supporté par les navigateurs modernes.

Inconvénient

L'**inconvénient** de l'utilisation de l'IndexedDB est que le code peut devenir très complexe à cause du grand nombre des fonctions callback imbriquées.⁷

Exemple d'architecture d'une application web qui support le travail hors connexion



⁷ <http://www.html5rocks.com/en/tutorials/offline/storage/>

Conclusion

La création d'une application web qui peut travailler sans avoir une connexion à l'internet est devenue possible, mais il nécessite la combinaison de plusieurs outils.

Par exemple, pour une application qui nécessite une interaction entre l'utilisateur et la base de données on a principalement besoin d'utiliser la cache d'application pour le téléchargement des ressources et ne plus référer au serveur pour le chargements de ces ressources tel que les fichiers JavaScript, CSS, les images, les audio, et même les actions (navigation entre les page).

Ensuite, on a besoins d'un stockage des informations chez le client, on peut utilisé le localStorage pour les préférences et les paramètres de l'application, et l'IndexedDB pour le stockage et la recherche des grands quantités des informations.

Et naturellement, on a à utiliser les scripts pour faire l'interaction avec les bases de données stockées chez le client.

Et pour ne pas perdre l'avantage d'avoir une application web, autrement dit, pour ne pas convertir notre application à une application de bureau, on a besoin de faire des synchronisations des données avec les bases de données globales lors du retour de la connexion.

Figures

Web Application

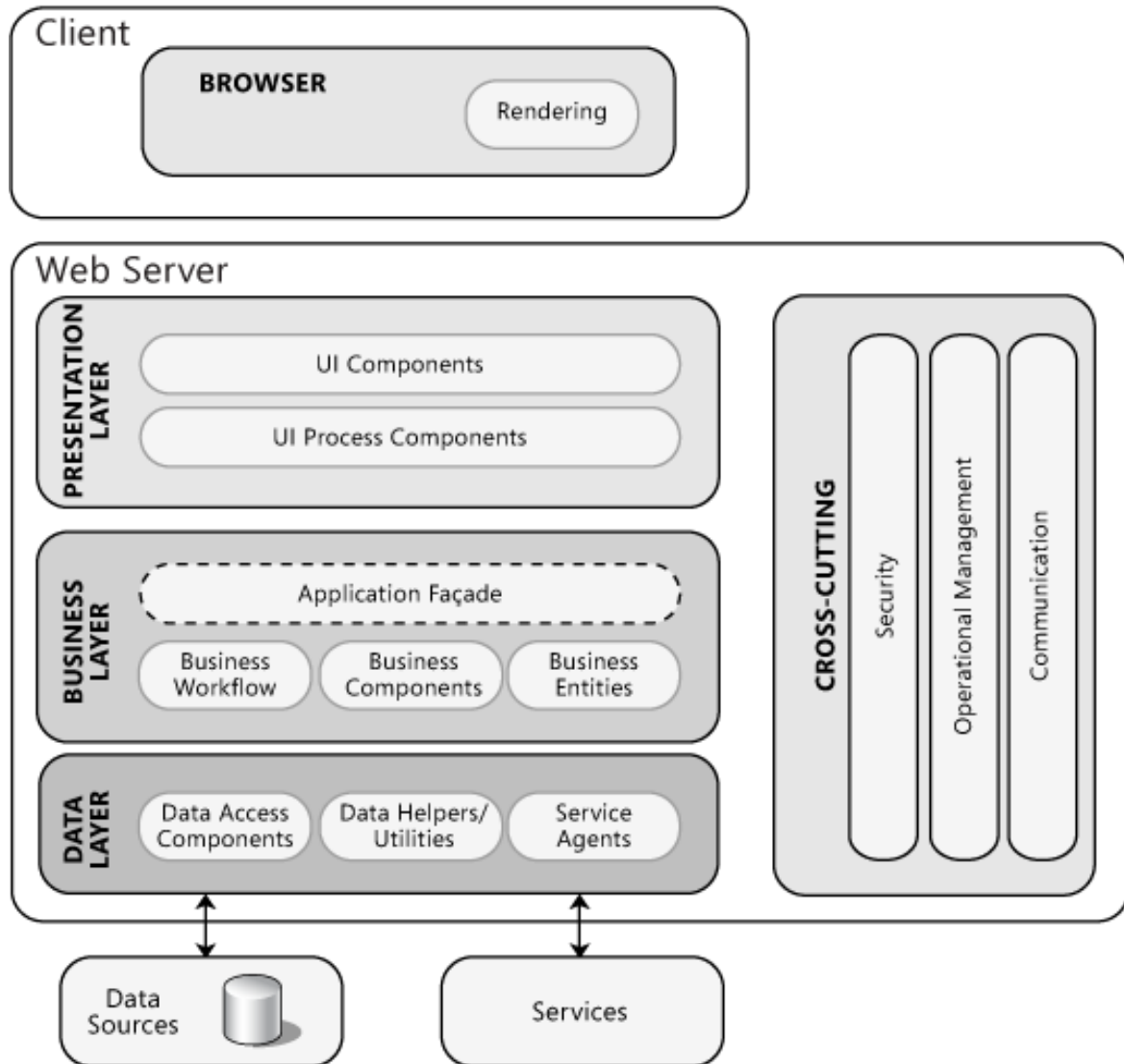


Figure 5: Architecture trois tiers⁸

⁸ <https://msdn.microsoft.com/en-us/library/ee658099.aspx>