**Sommaire :**

**L'univers Android …………………………………………………………………3**

* La création d'Android ……………………………………………………………………………..3

#### La philosophie et les avantages d'Android …………………………………………………5

#### Architecture ……………………………………………………………………………………….8

* Fonctionnement ……………………………………………………………………………….10

**Instalation et configuration ………………………………………………….11**

* Conditions initiales ………………………………………………………………………..…11

**Les différents types d’applications mobiles :……………………………12**

* Applications natives …………………………………………………………………………12
* Web Apps………………………………………………………………………………………….12
* Applications hybrides ………………………………………………………………………13

**Web Technologies ……………………………………………………………….14**

* jquery mobile…………………………………………………………………………………….14
* PhoneGap………………………………………………………………………………………….15
* Phonegap ou Cordova ?..........................................................15

***Ateliers ………………………………………………………………………………18***

* Media ………………………………………………………………………………………………..18
* *Paramètres………………………………………………………………………………….18*
* *Méthodes……………………………………………………………………………………..18*
* *Media.getCurrentPosition……………………………………………………………19*
* media.pause…………………………………………………………………………………20
* media.play ………………………………………………………………………………….21
* *media.release………………………………………………………………………………22*
* *media.stop……………………………………………………………………………………22*

## ***MediaError………………………………………………………………23***

* Géolocalisation ………………………………………………………………………………..24
* Méthodes……………………………………………………………………………………..25
* Autorisations……………………………………………………………………………….25
* geolocation.watchPosition ………………………………………………………….28
* geolocation.clearWatch……………………………………………………………….30
* Position ……………………………………………………………………………………….35
* PositionError ……………………………………………………………………………….37
* geolocationSuccess……………………………………………………………………..38
* Contacts ……………………………………………………………………………………………40
* 1 ère Etape …………………………………………………………………………………40
* 2ème Etape …………………………………………………………………………………41
* 3ème Etape …………………………………………………………………………………41
* Storage………………………………………………………………………………………………45

## Méthodes……………………………………………………………………..………………45

## Arguments…………………………………………………………………………………..45

## Objets………………………………………………………………………………………….46

## Permission…………………………………………………………………………………..46

## SQLTransaction…………………………………………………………..............56

## SQLResultSet……………………………………………………………………………..52

## SQLResultSetRowList…………………………………………………..............54

* SQLError……………………………………………………………………………………..56

Android est un système d'exploitation open source3 utilisant le noyau Linux, pour smartphones, tablettes tactiles, PDA et terminaux mobiles conçu par Android, une startup rachetée par Google, et annoncé officiellement le 5 novembre 2007.

**L'univers Android :**

* **La création d'Android :**

Quand on pense à Android, on pense immédiatement à Google, et pourtant il faut savoir que cette multinationale n'est pas à l'initiative du projet. D'ailleurs, elle n'est même pas la seule à contribuer à plein temps à son évolution. À l'origine, « Android » était le nom d'une PME américaine, créée en 2003 puis rachetée par Google en 2005, qui avait la ferme intention de s'introduire sur le marché des produits mobiles. La gageure, derrière Android, était de développer un système d'exploitation mobile plus intelligent, qui ne se contenterait pas uniquement de permettre d’envoyer des SMS et transmettre des appels, mais qui devait permettre à l'utilisateur d'interagir avec son environnement (notamment avec son emplacement géographique). C'est pourquoi, contrairement à une croyance populaire, il n'est pas possible de dire qu'Android est une réponse de Google à l'iPhone d'Apple, puisque l'existence de ce dernier n'a été révélée que deux années plus tard.

C'est en 2007 que la situation prit une autre tournure. À cette époque, chaque constructeur équipait son téléphone d'un système d'exploitation propriétaire. Chaque téléphone avait ainsi un système plus ou moins différent. Ce système entravait la possibilité de développer facilement des applications qui s'adapteraient à tous les téléphones, puisque la base était complètement différente. Un développeur était plutôt spécialisé dans un système particulier et il devait se contenter de langages de bas niveaux comme le C ou le C++. De plus, les constructeurs faisaient en sorte de livrer des bibliothèques de développement très réduites de manière à dissimuler leurs secrets de fabrication. En janvier 2007, Apple dévoilait l'iPhone, un téléphone tout simplement révolutionnaire pour l'époque. L'annonce est un désastre pour les autres constructeurs, qui doivent s'aligner sur cette nouvelle concurrence. Le problème étant que pour atteindre le niveau d'iOS (iPhone OS), il aurait fallu des années de recherche et développement à chaque constructeur...

C'est pourquoi est créée en novembre de l'année 2007 l'Open Handset Alliance (que j'appellerai désormais par son sigle OHA), et qui comptait à sa création 35 entreprises évoluant dans l'univers du mobile, dont Google. Cette alliance a pour but de développer un système *open source* (c'est-à-dire dont les sources sont disponibles librement sur internet) pour l'exploitation sur mobile et ainsi concurrencer les systèmes propriétaires, par exemple Windows Mobile et iOS.

Cette alliance a pour logiciel vedette Android, mais il ne s'agit pas de sa seule activité.

L'OHA compte à l'heure actuelle 80 membres.

[](http://uploads.siteduzero.com/files/318001_319000/318544.jpg)

Le logo de l'OHA, une organisation qui cherche à développer des standards open source pour les appareils mobiles

Android est à l'heure actuelle le système d'exploitation pour smartphones et tablettes le plus utilisé.

Les prévisions en ce qui concerne la distribution d'Android sur le marché sont très bonnes avec de plus en plus de machines qui s'équipent de ce système. Bientôt, il se trouvera dans certains téléviseurs (vous avez entendu parler de Google TV, peut-être ?) et les voitures. Android sera partout.

#### La philosophie et les avantages d'Android :

###### **Open source :**

Le contrat de licence pour Android respecte les principes de l'open source, c'est-à-dire que vous pouvez à tout moment télécharger les sources et les modifier selon vos goûts ! Bon, je ne vous le recommande vraiment pas, à moins que vous sachiez ce que vous faites... Notez au passage qu'Android utilise des bibliothèques open source puissantes, comme par exemple SQLite pour les bases de données et OpenGL pour la gestion d'images 2D et 3D.

###### **Gratuit (ou presque) :**

Android est gratuit, autant pour vous que pour les constructeurs. S'il vous prenait l'envie de produire votre propre téléphone sous Android, alors vous n'auriez même pas à ouvrir votre porte-monnaie (mais bon courage pour tout le travail à fournir !). En revanche, pour poster vos applications sur le Play Store, il vous en coûtera la modique somme de 25$. Ces 25$ permettent de publier autant d'applications que vous le souhaitez, à vie

###### **Facile à développer :**

Toutes les API mises à disposition facilitent et accélèrent grandement le travail. Ces APIs sont très complètes et très faciles d'accès. De manière un peu caricaturale, on peut dire que vous pouvez envoyer un SMS en seulement deux lignes de code (concrètement, il y a un peu d'enrobage autour de ce code, mais pas tellement).

Une API, ou « interface de programmation » en français, est un ensemble de règles à suivre pour pouvoir dialoguer avec d'autres applications. Dans le cas de Google API, il permet en particulier de communiquer avec Google Maps.

###### **Facile à vendre :**

Le Play Store(anciennement Android Market) est une plateforme immense et très visitée ; c'est donc une mine d'opportunités pour quiconque possède une idée originale ou utile.

###### **Flexible :**

Le système est extrêmement portable, il s'adapte à beaucoup de structures différentes. Les smartphones, les tablettes, la présence ou l'absence de clavier ou de trackball, différents processeurs... On trouve même des fours à micro-ondes qui fonctionnent à l'aide d'Android !    
Non seulement c'est une immense chance d'avoir autant d'opportunités, mais en plus Android est construit de manière à faciliter le développement et la distribution en fonction des composants en présence dans le terminal (si votre application nécessite d'utiliser le Bluetooth, seuls les terminaux équipés de Bluetooth pourront la voir sur le Play Store).

###### **Ingénieux :**

L'architecture d'Android est inspirée par les applications composites, et encourage par ailleurs leur développement. Ces applications se trouvent essentiellement sur internet et leur principe est que vous pouvez combiner plusieurs composants totalement différents pour obtenir un résultat surpuissant. Par exemple, si on combine l'appareil photo avec le GPS, on peut poster les coordonnées GPS des photos prises.

#### Les difficultés du développement pour des systèmes embarqués :

Il existe certaines contraintes pour le développement Android, qui ne s'appliquent pas au développement habituel !

Prenons un cas concret : la mémoire RAM est un composant matériel indispensable. Quand vous lancez un logiciel, votre système d'exploitation lui réserve de la mémoire pour qu'il puisse créer des variables, telles que des tableaux, des listes, etc. Ainsi, sur mon ordinateur, j'ai 4 Go de RAM, alors que je n'ai que 512 Mo sur mon téléphone, ce qui signifie que j'en ai huit fois moins. Je peux donc lancer moins de logiciels à la fois et ces logiciels doivent faire en sorte de réserver moins de mémoire. C'est pourquoi votre téléphone est dit limité, il doit supporter des contraintes qui font doucement sourire votre ordinateur.

Voici les principales contraintes à prendre en compte quand on développe pour un environnement mobile :

* Il faut pouvoir interagir avec un système complet sans l'interrompre. Android fait des choses pendant que votre application est utilisée, il reçoit des SMS et des appels, entre autres. Il faut respecter une certaine priorité dans l'exécution des tâches. Sincèrement, vous allez bloquer les appels de l'utilisateur pour qu'il puisse terminer sa partie de votre jeu de sudoku ?
* Comme je l'ai déjà dit, le système n'est pas aussi puissant qu'un ordinateur classique, il faudra exploiter tous les outils fournis afin de débusquer les portions de code qui nécessitent des optimisations.
* La taille de l'écran est réduite, et il existe par ailleurs plusieurs tailles et résolutions différentes. Votre interface graphique doit s'adapter à toutes les tailles et toutes les résolutions, ou vous risquez de laisser de côté un bon nombre d'utilisateurs.
* Autre chose qui est directement lié, les interfaces tactiles sont peu pratiques en cas d'utilisation avec un stylet et/ou peu précises en cas d'utilisation avec les doigts, d'où des contraintes liées à la programmation événementielle plus rigides. En effet, il est possible que l'utilisateur se trompe souvent de bouton. Très souvent s'il a de gros doigts.
* Enfin, en plus d'avoir une variété au niveau de la taille de l'écran, on a aussi une variété au niveau de la langue, des composants matériels présents et des versions d'Android. Il y a une variabilité entre chaque téléphone et même parfois entre certains téléphones identiques. C'est un travail en plus à prendre en compte.

Les conséquences de telles négligences peuvent être terribles pour l'utilisateur. Saturez le processeur et il ne pourra plus rien faire excepté redémarrer ! Faire crasher une application ne fera en général pas complètement crasher le système, cependant il pourrait bien s'interrompre quelques temps et irriter profondément l'utilisateur.

Il faut bien comprendre que dans le paradigme de la programmation classique vous êtes dans votre propre monde et vous n'avez vraiment pas grand-chose à faire du reste de l'univers dans lequel vous évoluez, alors que là vous faites partie d'un système fragile qui évolue sans anicroche tant que vous n'intervenez pas. Votre but est de fournir des fonctionnalités de plus à ce système et faire en sorte de ne pas le perturber.

Bon, cela paraît très alarmiste dit comme ça, Android a déjà anticipé la plupart des âneries que vous commettrez et a pris des dispositions pour éviter des catastrophes qui conduiront au blocage total du téléphone.  Si vous êtes un tantinet curieux, je vous invite à lire l'annexe sur l'architecture d'Android pour comprendre un peu pourquoi il faut être un barbare pour vraiment réussir à saturer le système.

**Architecture :**



Android est basé sur un kernel linux 2.6 mais ce n'est pas linux. Il ne possède pas de système de fenêtrage natif (X window system), la glibc n'est pas supporté, Android utilise une libc customisé appelé Bionic libc.  
Enfin Android utilise un kernel avec différents patches pour la gestion de l'alimentation, le partage mémoire, etc. permettant une meilleurs gestion de ces caractéristiques pour les appareils mobiles.  
  
Android n'est pas linux mais il est basé sur un kernel linux. Pourquoi sur un kernel linux ?

* Le kernel linux a un système de gestion mémoire et de processus reconnu pour sa stabilité et ses performances.
* Le model de sécurité utilisé par linux, basé sur un système de permission, connu pour être robuste et performant. Il n’a pas changé depuis les années 70
* Le kernel linux fournit un système de driver permettant un abstraction avec le matériel. Il permet également le partage de librairies entre différent processus, le chargement et le déchargement de modules à chaud.
* le kernel linux est entièrement open source et il y a une communauté de développeurs qui l'améliorèrent et rajoute des drivers.

C'est pour les points cités ci-dessus que l'équipe en charge du noyau a décidé d'utiliser un kernel linux.

**Patches :**

Le kernel Android a été patchés avec différents patchs :

**Alarme**

Ce patch fournit un certain nombre de timeurs permettant par exemple de "réveiller l'appareil quand il est en veille"

**Ashmem :**

Ce patch permet aux applications de partager de la mémoire. Cette gestion est faite au niveau kernel du fait que le partage mémoire est très utilisé dans la plateforme Android car la mémoire dans les appareils mobiles est limitée par rapport à des PC. Le partage mémoire est essentiellement utilisé par le Binder (lien).

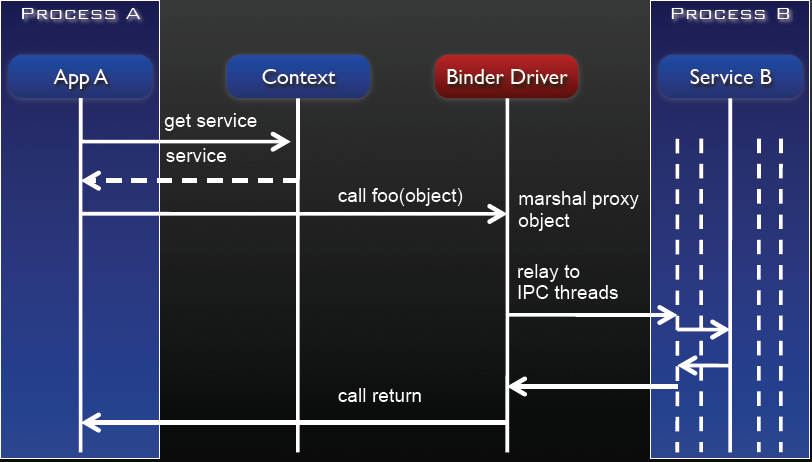
**Binder - Android IPC :**

La communication interprocessus (IPC) peut entrainer des trous de sécurité, c'est pour cela qu'Android à son propre IPC, le Binder et que la communication interprocessus n'est pas laissé aux développeurs d'application. De plus, avoir un système IPC centralisé permet une maintenance plus facile et une correction des problèmes de sécurités générales.

Dans Android chaque application est lancée dans un processus différent. Ces différents processus ont besoin de communiquer ensemble, de partager des données. Cet IPC est possible avec le Binder.

Il permet à plusieurs processus de partager des données, de communiquer entre eux en utilisant le partage mémoire (ashmem driver). Cette technique permet des performances accrue par rapports à de la recopie en mémoire des données, ou de la sérialisation.  
  
Les problèmes de concurrence, lorsque plusieurs processus essaye d'accéder en même temps à une "même zone mémoire"  (au même objet java) sont gérés par le Binder. Tous les appels sont synchronisés entre les processus.

**Fonctionnement :**



L'application A récupère une référence vers le Service B à l'aide du Context Manager. Le Context Manager peut être comparé à un DNS. Il permet de récupérer à l'aide d'un nom, une référence vers un objet java. Pour ceux qui connaissent RMI (Remote Method Invocation), c'est le registry. Si on veut partagé des objets, il faut au préalable les enregistrer dans le Context Manager.  
Une fois la référence vers le service B récupérée, la méthode foo() du service B est appelée par l'application A. Le binder intercepte cet appel, et à l'aide d'un des threads libres présent dans sa thread pool (piscine de thread ou réservoir de threads), il va exécuter la méthode sur le service B.

**Instalation et configuration :**

**Conditions initiales :**

De manière générale, n'importe quel matériel permet de développer sur Android du moment que vous utilisez Windows, Mac OS X ou une distribution Linux. Il y a bien sûr certaines limites à ne pas franchir.

Voyons si votre système d'exploitation est suffisant pour vous mettre au travail.  
Pour un environnement Windows, sont tolérés XP (en version 32 bits), Vista (en version 32 et 64 bits) et 7 (aussi en 32 et 64 bits). Officieusement (en effet, Google n'a rien communiqué à ce sujet), Windows 8 est aussi supporté en 32 et 64 bits.

*Et comment savoir quelle version de Windows j'utilise ?*

C'est simple, si vous utilisez Windows 7 ou Windows Vista, appuyez en même temps sur la touche Windowset sur la touche R. Si vous êtes sous Windows XP, il va falloir cliquer sur Démarrer puis sur Exécuter. Dans la nouvelle fenêtre qui s'ouvre, tapez winver. Si la fenêtre qui s'ouvre indique Windows 7 ou Windows Vista, c'est bon, mais s'il est écrit Windows XP, alors vous devez vérifier qu'il n'est écrit à aucun moment 64 bits. Si c'est le cas, alors vous ne pourrez pas développer pour Android.

Sous Mac, il vous faudra Mac OS 10.5.8 ou plus récent **et** un processeur x86.

Sous GNU/Linux, Google conseille d'utiliser une distribution Ubuntu plus récente que la 10.04. Enfin de manière générale, n'importe quelle distribution convient à partir du moment où votre bibliothèque GNU C (glibc) est au moins à la version 2.7. Si vous avez une distribution 64 bits, elle devra être capable de lancer des applications 32 bits.

**Les différents types d’applications mobiles :**

**Applications natives :**

Les applications natives sont des logiciels conçus spécifiquement pour une plate-forme mobile, en utilisant le SDK propre à celle-ci. Les applications ainsi crées sont ensuite téléchargeables depuis une plateforme dédiée au système, généralement un magasin d’application type App Store d’Apple ou Android Market.

Ces applications permettent de tirer parti de toute la puissance et toutes les possibilités du device mobile : accélération matérielle, capteurs, caméra, accès aux contacts, au fonctions de téléphonie, etc.

**Web Apps**

Il s’agit de sites web optimisés pour mobile, souvent conçus pour ressembler à de « vraies » applications.

On accède à ces applications web via le navigateur internet du device mobile.

Aujourd’hui grâce notamment au support d‘HTML5 et de CSS3 il est possible d’aller assez loin dans l’utilisation des capacités des devices. HTML5 permet notamment de gérer le multitouch, la géolocalisation, l’utilisation de l’accéléromètre, la mise en cache de ressources statiques, et même la synchronisation offline, après que l’appareil ait perdu puis retrouvé sa connexion. HTML5 gère également le stockage de données en local.

Google et Facebook notamment utilisent avec succès cette technologie et proposent des version mobiles de leurs applications web.

Ces applications web peuvent être génériques (toute plate-forme mobile) ou bien dédiées à un type de support particulier, par exemple beaucoup d’applications et de frameworks pour mobiles ne fonctionnent que sur un navigateur basé sur Webkit (iOS, Android, BlackBerry>=6.0, WebOS), exemple : touch.www.linkedin.comLe système iOS permet d’utiliser une web app à la manière d’une application, en plaçant un raccourci vers le site web mobile directement sur le bureau, avec une icône. Le navigateur lance alors la web app en plein écran, comme une vraie application.

Android gère en partie ce mécanisme, il permet la mise en raccourci sur le bureau avec icône, mais le navigateur ne se lance pas en plein écran.

**Applications hybrides :**

Une application hybride est un mélange de code natif et d’affichage de vues HTML/javascript. Concrètement toutes les plateformes mobiles proposent un composant de type WebView, permettant d’afficher du contenu web soit sur une partie de l’écran, soit en plein écran, et en utilisant le moteur HTML du navigateur intégré au système.

Ces applications hybrides peuvent être distribuées sur les stores des systèmes mobiles.

**Plusieurs stratégies sont alors possibles, selon que l’on place le curseur plus du côté natif ou plus du côté HTML :**

ne réaliser que certains écrans voir même que certains composants d’IHM en HTML.

réaliser touts les écrans en HTML mais garder la logique applicative en code natif, notamment les effets de transitions entre écrans et la gestion du scrolling.

réaliser les écrans en HTML, et les transitions / scrolling en javascript. Le code natif peut alors se cantonner à quelques composants techniques très ciblés. De la même manière selon les applications la logique métier peut être codée en javascript ou bien en code natif.

Quelques exemples d’applications hybrides :

LinkedIn

Microsoft Bing pour mobiles

Applications Flash

Flash Builder / Flex en version 4.5 permettent de packager des applications Android, iOS, et BlackBerry Tablet OS.

Concrètement ces applications Flash embarquent le runtime AIR dans leur code.

Comme pour les applications HTML5/javascript il s’agit d’une solution de développement multiplateformes, qui permet d’accéder à la plupart des ressources des smartphones et tablettes : multitouch, accéléromètre, GPS, bases de données locales SQLite.

Attention cependant Adobe a officiellement stoppé le développement du SDK Flex pour mobile et l’a offert à la fondation Apache en novembre 2011, on peut donc légitimement s’interroger sur la pérennité de cette technologie, surtout face à HTML5.

**Web Technologies :**

**jquery mobile:**

jQuery Mobile est un tactile optimisée framework web (plus connu sous le nom de bibliothèque JavaScript ou un cadre mobile) en cours d'élaboration par le jQuery équipe de projet. Le développement se concentre sur la création d'un cadre compatible avec une grande variété de smartphones et tablettes , rendue nécessaire par la tablette en croissance, mais hétérogène et marché des smartphones. Le framework jQuery Mobile est compatible avec d'autres cadres d'applications mobiles et de plateformes telles comme PhoneGap , Phare de travail et plus encore.

**Caractéristiques :**

* Compatible avec toutes les principales plates-formes mobiles ainsi que tous les navigateurs de bureau majeurs, y compris Ios, Android, BlackBerry, Web OS, Symbian, Windows Phone 7, et plus encore.
* Construit au-dessus de jQuery noyau, de sorte qu'il a une courbe d'apprentissage minimale pour les personnes déjà familiarisées avec la syntaxe jQuery.
* Thématisation cadre qui permet de créer des thèmes personnalisés.
* Dépendances limitées et légers pour optimiser la vitesse.
* La même base de code sous-jacent est automatiquement ajustée à n'importe quel écran
* HTML5 axée sur la configuration de la mise en page avec des scripts minimale
* Ajax-powered navigation avec les transitions de page animés qui fournit la capacité de nettoyer les URL par pushState.
* Widgets d'interface utilisateur qui sont tactile optimisée et la plate-forme agnostique.

**PhoneGap:**

PhoneGap est un cadre de développement mobile produit par Nitobi acheté par Adobe Systems .Il permet logiciels programmeurs de construire des applications pour les appareils mobiles utilisant JavaScript , HTML5 et CSS3 , au lieu de dispositif langages spécifiques tels que l'Objective-C . Les applications qui en découlent sont hybrides, ce qui signifie qu'ils ne sont ni vraiment natif (parce que tout le rendu disposition se fait via vues web au lieu d'origine de la plate-forme du cadre de l'assurance-chômage), ni purement basée sur le Web (parce qu'ils ne sont pas des applications simplement du Web, mais sont emballés que les applications de distribution et avoir accès à de périphérique natif API ). De 1,9 ultérieur version, il est même possible de mixer librement des extraits de code indigènes et hybrides.

Le logiciel sous-jacent PhoneGap est Apache Cordova . Le logiciel a été précédemment appelé simplement «PhoneGap", puis "Rappel Apache". Apache Cordova est un logiciel open source .

**Phonegap ou Cordova ?**

**A l’occasion de MAX 2011, Adobe a annoncé le rachat de PhoneGap, un framework de développement d’applications mobiles multi plates-formes reposant sur les standards du Web. Une façon d’effacer d’un revers de la main la dualité qui oppose Flash à HTML5, et de donner un signal clair au marché de son implication dans les standards. Adobe et PhoneGap comptent faire don du code du framework à la Fondation Apache.**

La bataille entre Flash et HTML 5 n’existe pas, scande haut et fort Adobe, à l’occasion du second jour de Max 2011, la conférence développeurs de l’éditeur qui se tient à Los Angeles du 3 au 5 octobre. Et comme pour justifier son propos, l’éditeur de la plate-forme Flash annonce le rachat de la société Nitobi Software, éditeur du framework Open Source PhoneGap, localisé à Vancouver. Un outil bien connu des développeurs – on parle de 600 000 téléchargements depuis son lancement -  de par ses capacités à exploiter les standards ouverts HTML 5, CSS et Javascript notamment, pour le développement d’applications mobiles natives. Et ce sur les principaux OS du marché (iOS,  Android, BlackBerry, Windows Phone 7,  webOS,  Symbian et Bada).

Pour Adobe, il est clair que cette initiative vient sceller dans le marbre son implication au niveau des standards. Standards que le groupe de San Jose entend faire cohabiter avec son produit phare, le Flash – et surtout Adobe AIR (Adobe Integrated Runtime). Le groupe souhaite ainsi être une porte d’entrée dans le développement multi-plateforme, et ce, quelles soient les technologies, nous révélait le Pdg du groupe, Shantanu Narayen, à l’occasion d’une session de questions / réponses avec la presse. Surtout confirme un autre responsable du groupe, outre le fait qu’HTML 5 propose désormais un vaste pan de fonctions innovantes et surprenantes, il est également trop couteux de développer pour deux technologies différents. L’argument économique a en effet de quoi séduire.

« Cette acquisition étend les possibilités de création d’ applications multi plates-formes. Adobe a historiquement fourni des outils au dessus de ses plates-formes, comme Flash et PDF. La tendance aujourd’hui est à la mutation du hardware, avec notamment les tablettes et les mobiles. Et en même temps, la nouvelle plate-forme HTML 5 est devenue une référence dans l’industrie. Il est donc nécessaire de disposer des outils et de la plate-forme pour ces nouveaux terminaux et de la faire évoluer au rythme du marché », explique Dave McAllister, directeur et gourou des standards chez Adobe, dans un entretien avec la rédaction.

Selon lui, ce qui distingue également PhoneGap de la concurrence, c’est également sa simplicité d’utilisation, « qui ne nécessite pas d’investir dans des compétences pointues », pour la réalisation d’applications. A terme, les outils de PhoneGap seront sous forme de Creative Services, présents dans le Creative Cloud annoncé hier lors de cette même conférence développeurs.  
 **Un code donné à la fondation Apache :**

L’un des points clé du rachat est le choix d’Adobe et de Nitobi de confier le code de PhoneGap à la fondation Apache, au sein de son ‘incubateur à projet. Pour l’heure, le projet est référencé sous le nom Callback, mais ne semble pas définitif, comme l’indique le Wiki de la fondation. Si Mc Allister reconnait les travaux de la fondation Open Source ainsi que son identité très communautaire, il souligne également d’Adobe est fortement impliqué dans quelque 23 projets de l’institution Open Source, en tant que « contributeur principal dans certains, supporter dans d’autres ». Un choix logique donc.

Pour autant, les motivations de placer le code de PhoneGap dans une institution clé de l’Open Source a également un autre intérêt : celui de rassurer la communauté existante de Phonegap : « nous souhaitons conserver la communauté intacte, de façon à ce que les utilisateurs n’aient pas se poser la question quant aux intentions d’Adobe. […] Il s’agit de rassurer la communauté en leur garantissant l’ouverture, tout en nous permettant d’utiliser le code comme nous l’entendons, explique-t-il, faisant ainsi référence à la licence Apache, l’une des licences Open Source les moins permissives. Apache apparait ici comme le contrat de confiance de PhoneGap.

***Ateliers :***

**Appareil photo**

**Contacts**

**Événements**

**Dossier**

**Stockage**

**Géolocalisation**

* **Media :**

L'objet Media offre la possibilité de lire et d'enregistrer des fichiers audio sur un mobile.

var media = new Media(src, mediaSuccess, [mediaError], [mediaStatus]);

Remarque : L'implémentation actuelle de n'est pas conforme à la spécification du W3C, et n'est fournie que pour des raisons pratiques. Une prochaine implémentation sera conforme à la dernière spécification du W3C et pourrait rendre obsolète l'API courante.

# ***Paramètres***

* **src**: L'URI d'un fichier audio. *(DOMString)*
* **mediaSuccess**: (Facultatif) La fonction de callback qui est appelée à l'issue d'une action de lecture/enregistrement ou arrêt réalisée sur l'objet Media. *(Function)*
* **mediaError**: (Facultatif) La fonction de callback qui est appelée en cas d'erreur. *(Function)*
* **mediaStatus**: (Facultatif) La fonction de callback qui est appelée lorsque qu'un changement d'état a lieu. *(Function)*

# ***Méthodes***

* **media.getCurrentPosition:** Retourner la position courante dans le fichier audio.
* **media.getDuration**: Récupérer la durée du fichier audio.
* **media.play:** Démarrer ou reprendre la lecture du fichier audio.
* **media.pause**: Mettre en pause la lecture du fichier audio.
* **media.release:** Libérer la ressource audio dans le système d'exploitation.
* **media.seekTo:** Déplacer la position courante dans le fichier audio.
* **media.startRecord**: Démarrer l'enregistrement d'un fichier audio.
* **media.stopRecord:** Arrêter l'enregistrement du fichier audio.
* **media.stop:** Arrêter la lecture du fichier audio.

# ***Media.getCurrentPosition***

Récupérer la position courante dans un fichier audio.

media.getCurrentPosition(mediaSuccess, [mediaError]);

# *Paramètres*

* **mediaSuccess**: La fonction de callback qui est appelée avec en argument la position en secondes.
* **mediaError**: (Facultatif) La fonction de callback qui est appelée en cas d'erreur.

# ***Description***

La fonction media.getCurrentPosition est une fonction asynchrone qui retourne la position courante dans le fichier audio qu'un objet Media représente. Elle met également à jour le paramètre \_position de l'objet Media.

## **Exemple :**

***// Lecteur audio  
    //*** *var* ***my\_media* =***new*Media**(*src*, *onSuccess*, *onError*); *// Récupérer la positon courante une fois par seconde*** *var* ***mediaTimer* = *setInterval*(***function***(){ *// Récupérer la position courante  
        my\_media*.*getCurrentPosition*( *// Fonction de callback de succès*** *function***(*position*){***if***(*position* >-*1*){ *console*.*log*((*position*)+ *" sec"*);}}, *// Fonction de callback d'erreur*** *function***(*e*){ *console*.*log*(*"Erreur lors de la récupération de la position = "* + *e*);});}, *1000*);**

## **media.pause**

Mettre en pause la lecture d'un fichier audio.

media.pause();

# ***Description***

La fonction media.pause est une fonction synchrone qui met en pause la lecture d'un fichier audio.

## **Exemple :**

***// Lire le clip audio***  
***//  
function playAudio(url) {  
    // Lire le clip audio désigné par l'URL  
    var my\_media = new Media(url,  
        // Callback en cas de succès  
        function() {  
            console.log("playAudio() : Réussite");  
        },  
        // Callback en cas d'erreur  
        function(err) {  
            console.log("playAudio() : Erreur : "+err);  
    });  
  
    // Lire le clip audio  
    my\_media.play();  
  
    // Mettre en pause au bout de 10 secondes  
    setTimeout(function() {  
        media.pause();  
    }, 10000);          
}***

# **media.play :**

Démarrer ou reprendre la lecture du fichier audio.

media.play();

# **Description**

La fonction media.play est une fonction synchrone qui démarre ou reprend la lecture d'un fichier audio

# **Exemple :**

***// Lire le clip audio  
//  
function playAudio(url) {  
    // Lire le clip audio désigné par l'URL  
    var my\_media = new Media(url,  
        // Callback en cas de succès  
        function() {  
            console.log("playAudio() : Réussite");  
        },  
        // Callback en cas d'erreur  
        function(err) {  
            console.log("playAudio() : Erreur : "+err);  
    });  
  
    // Lire le clip audio  
    my\_media.play();  
}***

# ***media.release***

Libérer la ressource audio dans le système d'exploitation.

media.release();

# ***Description***

La fonction media.release est une fonction synchrone qui libère la ressource audio dans le système d'exploitation du mobile. Cette fonction est particulièrement importante sous Android, puisqu'il y a un nombre fini d'instances de OpenCore pour la lecture audio. Les développeur doivent donc appeler la fonction 'release' lorsqu'ils n'ont plus besoin de la ressource Media.

# ***Exemple :***

***// Lire le clip audio désigné par src  
    //  
    var my\_media = new Media(src, onSuccess, onError);  
  
    my\_media.play();  
    my\_media.stop();  
    my\_media.release();***

# ***media.stop***

Arrêter la lecture du fichier audio.

media.stop();

# ***Description***

La fonction media.stop est une fonction synchrone qui arrête la lecture d'un fichier audio.

# ***Exemple :***

***// Lire le clip audio  
//  
function playAudio(url) {  
    // Lire le clip audio désigné par l'URL  
    var my\_media = new Media(url,  
        // Callback en cas de succès  
        function() {  
            console.log("playAudio() : Réussite");  
        },  
        // Callback en cas d'erreur  
        function(err) {  
            console.log("playAudio() : Erreur : "+err);  
    });  
  
    // Lire le clip audio  
    my\_media.play();  
  
    // Arrêter la lecture après 10 secondes  
    setTimeout(function() {  
        my\_media.stop();  
    }, 10000);          
}***

## ***MediaError***

Un objet MediaError est passé en argument de la focntion de callback mediaError lorsqu'une erreur survient.

## ***Propriétés***

* **code:** Un des codes d'erreur prédéfinis ci-dessous.
* **message:** Un message explicitant l'erreur survenue.

## ***Constantes***

* MediaError.MEDIA\_ERR\_ABORTED
* MediaError.MEDIA\_ERR\_NETWORK
* MediaError.MEDIA\_ERR\_DECODE
* MediaError.MEDIA\_ERR\_NONE\_SUPPORTED

## ***Description***

* L'objet MediaError est retourné à l'utilisateur via la fonction de callback mediaError lorsqu'une erreur survient.
* **Géolocalisation :**

La géolocalisation ou géoréférencement est un procédé permettant de positionner un objet (une personne, etc) sur un plan ou une carte à l'aide de ses coordonnées géographiques.

Cette opération est réalisée à l'aide d'un terminal capable d'être localisé (grâce à un système de positionnement par satellites (et un récepteur GPS par exemple) ou à d'autres techniques) et de publier (en temps réel ou de façon différée) ses coordonnées géographiques (latitude/longitude). Les positions enregistrées peuvent être stockées au sein du terminal et être extraites postérieurement, ou être transmises en temps réel vers une plateforme logicielle de géolocalisation. La transmission temps réel nécessite un terminal équipé d'un moyen de télécommunication de type GSM, GPRS, UMTS, radio ou satellite lui permettant d'envoyer les positions à des intervalles réguliers. Ceci permet de visualiser la position du terminal au sein d'une carte à travers une plateforme de géolocalisation le plus souvent accessible depuis internet.

Géolocalisation fournit des informations de localisation de l'appareil, telles que la latitude et la longitude. Les sources communes d'informations de localisation comprennent Global Positioning System (GPS) et le lieu de déduire des signaux de réseau tels que l'adresse IP, RFID, WiFi et Bluetooth adresses MAC et ID de cellules GSM / CDMA. Aucune garantie n'est donnée quant à l'API renvoie emplacement réel de l'appareil.

Cette API est basée sur le W3S Géolocalisation spécification API. Certains appareils (Android, BlackBerry, Bada, Windows Phone 7, webOS et Tizen, pour être précis) fournissent déjà une implémentation de cette spécification. Pour ces appareils, la prise en charge intégrée est utilisée au lieu de la remplacer par la mise en œuvre de Cordoue. Pour les appareils qui ne supportent pas la géolocalisation, la mise en œuvre Cordova adhère à la spécification du W3C.

**Méthodes**

• geolocation.getCurrentPosition

• geolocation.watchPosition

• geolocation.clearWatch

Arguments

• geolocationSuccess

• geolocationError

• geolocationOptions

Objets (Read-Only)

• Position

• PositionError

• Coordonnées

**Autorisations**

Androïde :

app / res / xml / config.xml

<plugin name="Geolocation" value="org.apache.cordova.GeoBroker" />

app/AndroidManifest.xml:

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<uses-permission android:name="android.permission.ACCESS\_LOCATION\_EXTRA\_COMMANDS" />

geolocation.getCurrentPosition

Retourne la position actuelle de l'appareil en tant que Position objet.

navigator. geolocation.getCurrentPosition ( geolocationSuccess , [ geolocationError ], [ geolocationOptions ]);

Paramètres

• geolocationSuccess : Le rappel qui est appelée avec la position actuelle.

• geolocationError : (Facultatif) Le rappel qui sera appelée si une erreur s'est produite.

• geolocationOptions : (Facultatif) Les options de géolocalisation.

**Description**

geolocation.getCurrentPosition est une fonction asynchrone. Il retourne la position actuelle de l'appareil à l' geolocationSuccess rappel avec un Position d'objet comme paramètre. S'il y a une erreur, le geolocationError rappel est appelé avec un PositionError objet.

Plates-formes prises en charge

• Androïde

• BlackBerry WebWorks (OS 5.0 et plus)

• iOS

• Windows Phone 7 et 8

• Bada 1.2 et 2.x

• webOS

• Tizen

• Windows 8

**Exemple complet :**

<!DOCTYPE html>

<html>

<head>

<title>Device Properties Example</title>

<script type="text/javascript" charset="utf-8" src="cordova-2.5.0.js"></script>

<script type="text/javascript" charset="utf-8">

// Wait for Cordova to load

//

document.addEventListener("deviceready", onDeviceReady, false);

// Cordova is ready

//

function onDeviceReady() {

navigator.geolocation.getCurrentPosition(onSuccess, onError);

}

// onSuccess Geolocation

//

function onSuccess(position) {

var element = document.getElementById('geolocation');

element.innerHTML = 'Latitude: ' + position.coords.latitude + '<br />' +

'Longitude: ' + position.coords.longitude + '<br />' +

'Altitude: ' + position.coords.altitude + '<br />' +

'Accuracy: ' + position.coords.accuracy + '<br />' +

'Altitude Accuracy: ' + position.coords.altitudeAccuracy + '<br />' +

'Heading: ' + position.coords.heading + '<br />' +

'Speed: ' + position.coords.speed + '<br />' +

'Timestamp: ' + position.timestamp + '<br />';

}

// onError Callback receives a PositionError object

//

function onError(error) {

alert('code: ' + error.code + '\n' +

'message: ' + error.message + '\n');

}

</script>

</head>

<body>

<p id="geolocation">Finding geolocation...</p>

</body>

</html>

**geolocation.watchPosition :**

Montres de modification de la position actuelle de l'appareil.

var watchId = navigator. geolocation.watchPosition ( geolocationSuccess , [ geolocationError ], [ geolocationOptions ]);

**Paramètres**

• geolocationSuccess : Le rappel qui est appelée avec la position actuelle.

• geolocationError : (Facultatif) Le rappel qui sera appelée si une erreur s'est produite.

• geolocationOptions : (Facultatif) Les options de géolocalisation.

Résultats

• String: retourne un identifiant montre que les références de l'intervalle surveillance de position. L'id montre doit être utilisé avec geolocation.clearWatch d'arrêter de regarder pour des changements de position.

Description

geolocation.watchPosition est une fonction asynchrone. Il renvoie la position actuelle de l'appareil quand un changement de position n'a été détecté. Lorsque l'appareil a récupéré un nouvel emplacement, le geolocationSuccess rappel est appelé avec une Position d'objet comme paramètre. S'il y a une erreur, le geolocationError rappel est appelé avec un PositionErrorobjet.

**Exemple complet :**

<!DOCTYPE html>

<html>

<head>

<title>Device Properties Example</title>

<script type="text/javascript" charset="utf-8" src="cordova-2.5.0.js"></script>

<script type="text/javascript" charset="utf-8">

// Wait for Cordova to load

//

document.addEventListener("deviceready", onDeviceReady, false);

var watchID = null;

// Cordova is ready

//

function onDeviceReady() {

// Throw an error if no update is received every 30 seconds

var options = { timeout: 30000 };

watchID = navigator.geolocation.watchPosition(onSuccess, onError, options);

}

// onSuccess Geolocation

//

function onSuccess(position) {

var element = document.getElementById('geolocation');

element.innerHTML = 'Latitude: ' + position.coords.latitude + '<br />' +

'Longitude: ' + position.coords.longitude + '<br />' +

'<hr />' + element.innerHTML;

}

// onError Callback receives a PositionError object

//

function onError(error) {

alert('code: ' + error.code + '\n' +

'message: ' + error.message + '\n');

}

</script>

</head>

<body>

<p id="geolocation">Watching geolocation...</p>

</body>

</html>

**geolocation.clearWatch**

Arrêtez de regarder pour des changements à l'emplacement de l'appareil référencé par le watchID paramètre.

navigator. geolocation.clearWatch (watchID);

Paramètres

• watchID: L'id de l' watchPosition intervalle à effacer. (String)

Description

geolocation.clearWatch arrête les modifications à regarder à l'emplacement de l'appareil en ouvrant la geolocation.watchPosition référencé par watchI

**Exemple complet :**

<!DOCTYPE html>

<html>

<head>

<title>Device Properties Example</title>

<script type="text/javascript" charset="utf-8" src="cordova-2.5.0.js"></script>

<script type="text/javascript" charset="utf-8">

// Wait for Cordova to load

//

document.addEventListener("deviceready", onDeviceReady, false);

var watchID = null;

// Cordova is ready

//

function onDeviceReady() {

// Get the most accurate position updates available on the

// device.

var options = { enableHighAccuracy: true };

watchID = navigator.geolocation.watchPosition(onSuccess, onError, options);

}

// onSuccess Geolocation

//

function onSuccess(position) {

var element = document.getElementById('geolocation');

element.innerHTML = 'Latitude: ' + position.coords.latitude + '<br />' +

'Longitude: ' + position.coords.longitude + '<br />' +

'<hr />' + element.innerHTML;

}

// clear the watch that was started earlier

//

function clearWatch() {

if (watchID != null) {

navigator.geolocation.clearWatch(watchID);

watchID = null;

}

}

// onError Callback receives a PositionError object

//

function onError(error) {

alert('code: ' + error.code + '\n' +

'message: ' + error.message + '\n');

}

</script>

</head>

<body>

<p id="geolocation">Watching geolocation...</p>

<button onclick="clearWatch();">Clear Watch</button>

</body>

</html>

Coordonnées

Un ensemble de propriétés qui décrivent les coordonnées géographiques d'une position.

Propriétés

• latitude: latitude en degrés décimaux (Number).

• longitude: Longitude en degrés décimaux (Number).

• altitude: Hauteur de la position en mètres au-dessus de l'ellipsoïde (Nombre).

• Précision: Niveau de précision des coordonnées de latitude et de longitude en mètres (Nombre).

• altitudeAccuracy: Niveau de précision des coordonnées altitude en mètres (Nombre).

• rubrique:. Sens de la marche, indiquée en degrés de comptage horaire par rapport au nord vrai (Nombre)

• Vitesse: la vitesse au sol actuelle de l'appareil, indiquée en mètres par seconde (Nombre).

Description

Le Coordinates objet est créé et peuplé de Cordoue, et attaché à la Position objet. La Position d'objet est ensuite renvoyé à l'utilisateur par le biais d'une fonction de rappel.

**Exemple complet :**

<!DOCTYPE html>

<html>

<head>

<title>Geolocation Position Example</title>

<script type="text/javascript" charset="utf-8" src="cordova-2.5.0.js"></script>

<script type="text/javascript" charset="utf-8">

// Set an event to wait for Cordova to load

//

document.addEventListener("deviceready", onDeviceReady, false);

// Cordova is loaded and Ready

//

function onDeviceReady() {

navigator.geolocation.getCurrentPosition(onSuccess, onError);

}

// Display `Position` properties from the geolocation

//

function onSuccess(position) {

var div = document.getElementById('myDiv');

div.innerHTML = 'Latitude: ' + position.coords.latitude + '<br/>' +

'Longitude: ' + position.coords.longitude + '<br/>' +

'Altitude: ' + position.coords.altitude + '<br/>' +

'Accuracy: ' + position.coords.accuracy + '<br/>' +

'Altitude Accuracy: ' + position.coords.altitudeAccuracy + '<br/>' +

'Heading: ' + position.coords.heading + '<br/>' +

'Speed: ' + position.coords.speed + '<br/>';

}

// Show an alert if there is a problem getting the geolocation

//

function onError() {

alert('onError!');

}

</script>

</head>

<body>

<div id="myDiv"></div>

</body></html>

**Position :**

Contient Position coordonnées et l'horodatage, créé par l'API de géolocalisation.

Propriétés

• coords: Un ensemble de coordonnées géographiques (. Coordonnées )

• timestamp: horodatage de création pour les coords (date).

Description

La Position objet est créé et peuplé de Cordoue, et renvoyée à l'utilisateur par le biais d'une fonction de rappel.

<!DOCTYPE html>

<html>

<head>

<title>Device Properties Example</title>

<script type="text/javascript" charset="utf-8" src="cordova-2.5.0.js"></script>

<script type="text/javascript" charset="utf-8">

// Wait for Cordova to load

//

document.addEventListener("deviceready", onDeviceReady, false);

// Cordova is ready

//

function onDeviceReady() {

navigator.geolocation.getCurrentPosition(onSuccess, onError);

}

// onSuccess Geolocation

//

function onSuccess(position) {

var element = document.getElementById('geolocation');

element.innerHTML = 'Latitude: ' + position.coords.latitude + '<br />' +

'Longitude: ' + position.coords.longitude + '<br />' +

'Altitude: ' + position.coords.altitude + '<br />' +

'Accuracy: ' + position.coords.accuracy + '<br />' +

'Altitude Accuracy: ' + position.coords.altitudeAccuracy + '<br />' +

'Heading: ' + position.coords.heading + '<br />' +

'Speed: ' + position.coords.speed + '<br />' +

'Timestamp: ' +

position.timestamp + '<br />';

}

// onError Callback receives a PositionError object

//

function onError(error) {

alert('code: ' + error.code + '\n' +

'message: ' + error.message + '\n');

}

</script>

</head>

<body>

<p id="geolocation">Finding geolocation...</p>

</body>

</html>

**PositionError :**

Un PositionError objet est retourné à la geolocationError rappel lorsqu'une erreur se produit.

**Propriétés :**

code: Un des codes d'erreur prédéfinis listés ci-dessous.

message: message d'erreur décrivant les détails de l'erreur rencontrée.

Constantes

PositionError .PERMISSION\_DENIED

PositionError .POSITION\_UNAVAILABLE

PositionError .TIMEOUT

Description

Le PositionError objet est retourné à l'utilisateur par l'intermédiaire du geolocationError fonction de rappel lorsqu'une erreur se produit avec la géolocalisation.

PositionError .PERMISSION\_DENIED

Renvoyé lorsque l'utilisateur ne laissez pas votre application de récupérer des informations de position. Cela dépend de la plate-forme.

PositionError .POSITION\_UNAVAILABLE

Renvoyé lorsque l'appareil n'a pas pu récupérer une position. En général, cela signifie que l'appareil n'a pas de connectivité réseau et / ou ne peuvent pas obtenir un point satellite.

PositionError .TIMEOUT

Renvoyé lorsque l'appareil n'a pas pu récupérer une position dans le délai précisé dans les geolocationOptions ' timeout propriété. Lors de l'utilisation en conjonction avec geolocation.watchPosition , cette erreur pourrait être appelé dans le geolocationError rappel tous timeout millisecondes.

**geolocationSuccess**

Fonction de rappel de l'utilisateur qui est appelée quand une position de géolocalisation devient disponible (lors de l'utilisation avec geolocation.getCurrentPosition ), ou lors d'un changement de position (lors de l'utilisation avec geolocation.watchPosition ).

function(position) { // Do something }

Paramètres

position: La position de géolocalisation renvoyée par le dispositif. ( Position )

Exemple

function geolocationSuccess (position) { alert('Latitude: ' + position.coords.latitude + '\n' + 'Longitude: ' + position.coords.longitude + '\n' + 'Altitude: ' + position.coords.altitude + '\n' + 'Accuracy: ' + position.coords.accuracy + '\n' + 'Altitude Accuracy: ' + position.coords.altitudeAccuracy + '\n' + 'Heading: ' + position.coords.heading + '\n' + 'Speed: ' + position.coords.speed + '\n' + 'Timestamp: ' + position.timestamp + '\n'); }

geolocationError

Fonction de rappel de l'utilisateur qui est appelée quand il ya une erreur pour les fonctions de géolocalisation.

function(error) { // Handle the error }

Paramètres

erreur: L'erreur renvoyée par le dispositif. ( PositionError )

geolocationOptions

Les paramètres optionnels pour personnaliser la récupération de la géolocalisation Position .

{ maximumAge: 3000, timeout: 5000, enableHighAccuracy: true };

Options de

enableHighAccuracy: Fournit une indication que la demande souhaitez recevoir les meilleurs résultats possibles. Par défaut, l'appareil tentera de récupérer une Position en utilisant des méthodes basées sur le réseau. Définir cette propriété à true indique le cadre à utiliser des méthodes plus précises, telles que le positionnement par satellite. (Boolean)

timeout: La longueur maximale du temps (millisecondes) qui est autorisé à passer de l'appel à geolocation.getCurrentPosition ou geolocation.watchPosition jusqu'à ce que le correspondant geolocationSuccess rappel est invoquée. Si le geolocationSuccess callback n'est pas appelé dans ce délai, la geolocationError callback sera appelée avec un PositionError .TIMEOUT code d'erreur. REMARQUE: lorsqu'il est utilisé en conjonction avec geolocation.watchPosition , le geolocationError rappel pourrait être appelé sur un intervalle tous timeout millisecondes (Nombre)!

maximumAge: Accepter un poste en cache dont l'âge ne dépasse pas le temps spécifié en millisecondes (Nombre).

Quirks Android

Les simulateurs Android 2.x ne retourne pas de résultat sauf si l'option géolocalisation enableHighAccuracy est définie sur true.

{ enableHighAccuracy: true }

**Contacts :**

Création d’une petite application mobile de gestion des contacts du télèphone : l’utilisateur aura la possibilité de saisir les coordonnés d’un contact, et le récupérer par la suite via l’application Contact installés par défault dans tout les Smartphone Android, iOS, ..etc

le test est fait sur Android mais devra marché de sur d’autres terminaux mobiles avec des configurations spécèfique à chaque OS Mobile.  
  
Outils : HTML, CSS, Javascripte, PhoneGap  
  
  
**1 ère Etape :**

Création d’un formulaire HTML avec une décoration CSS :

<form >

<table>

<tr>

<td> <label for=*"nom"*> Nom: </label> </td>

<td> <input type=*"text"* id=*"lenom"* n/></td>

</tr>

<tr>

<td> <label for=*"prenom"*> Prénom: </label></td>

<td> <input type=*"text"* id=*"leprenom"* /></td>

</tr>

<tr>

<td><label for=*"note"*> Note</label> </td>

<td><input type=*"text"* id=*"lanote"* /></td>

</tr>

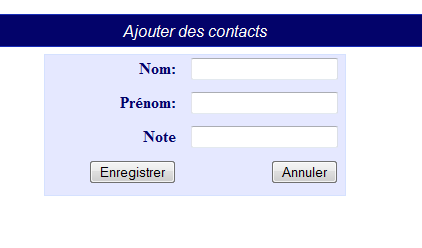
<tr>

<td><input type=*"button"* id=*"BtnSave"* value=*"Enregistrer"* /></td>

<td><input type=*"button"* id=*"BtnAnnuler"* value=*"Annuler"* /></td>

</tr>

</form>



**2ème Etape :**

Configuration du fichier config.xml et AndroidManifest.xml pour obtenir les permessions requises pour accéder aux contacts du télèphone

**app/res/xml/config.xml**

<plugin name="Contacts" value="org.apache.cordova.ContactManager" />

**app/AndroidManifest.xml**

<uses-permission android:name="android.permission.GET\_ACCOUNTS" />  
<uses-permission android:name="android.permission.READ\_CONTACTS" />  
<uses-permission android:name="android.permission.WRITE\_CONTACTS" />

**3ème Etape :**

Création de la méthode SaveContact qui nous permettra de stocker les informations saisies dans le formulaire dans les contacts de notre télèphone .

* Récupérer les données saisies dans notre formulaire :

**var** fullname = document.getElementById('lenom').value + " "+document.getElementById('leprenom').value;

**var** note = document.getElementById('lanote').value;

* Création d’une variable contact qui accéde aux contacts de notre télèphonen on lui donne comme valeurs nos variables fullname et note :

**var** contact = navigator.contacts.create({"displayName" : fullname});

contact.note = note;

* On enregistre les changements effectués à l’aide de la méthode prédèfinie Save, on lui donne en arguments deux fonctions qui nous permettront de davoir si le stockage a été bien effectué ou non

contact.save(onSaveSuccess,onSaveError);

**function** onSaveSuccess(contact) {

alert("le contact a été enregistré avec succès");

}

**function** onSaveError(contactError) {

alert("Error = " + contactError.code);

}

Notre fonction SaveContact est dèsormais comme suit :

**function** saveContact()

{

**var** fullname = document.getElementById('lenom').value + " "+document.getElementById('leprenom').value;

**var** note = document.getElementById('lanote').value;

**var** contact = navigator.contacts.create({"displayName" : fullname});

contact.note = note;

contact.save(onSaveSuccess,onSaveError);

}

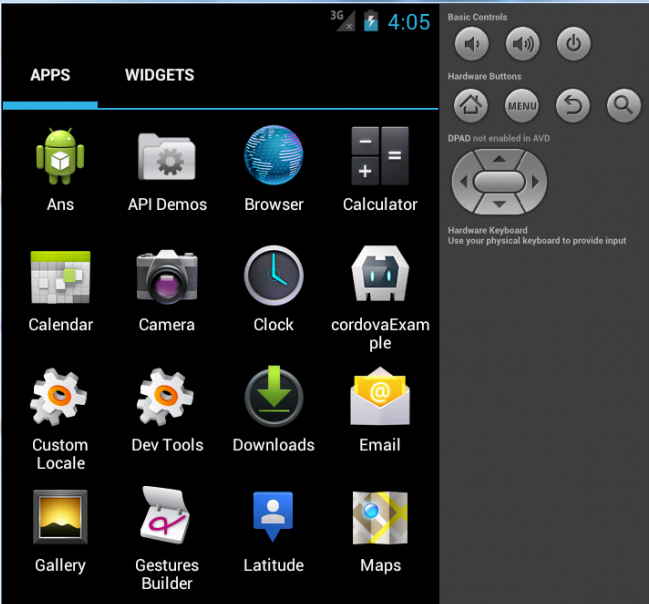
On ajoute un listner sur notre bouton enregistrer, si l’utilisateur clique sur le bouton enregistrer, la méthode saveContact est alors appellé :

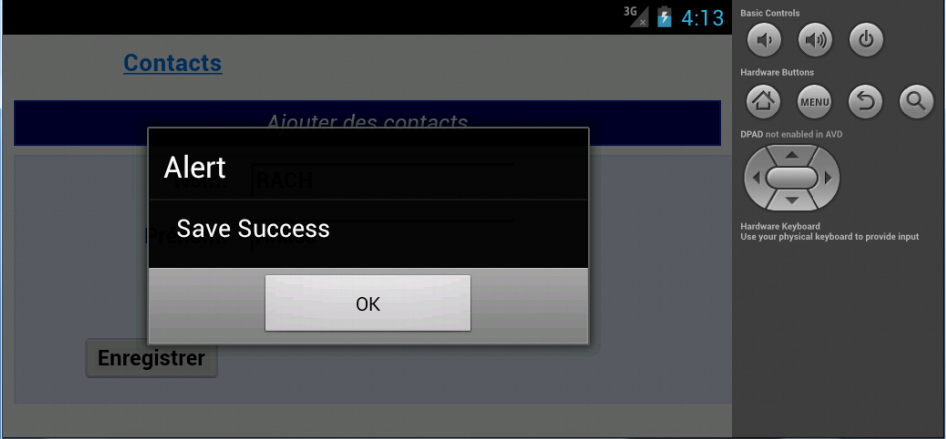
**function** onDeviceReady() {

document.getElementById('BtnSave').addEventListener('click',saveContact,**false**);

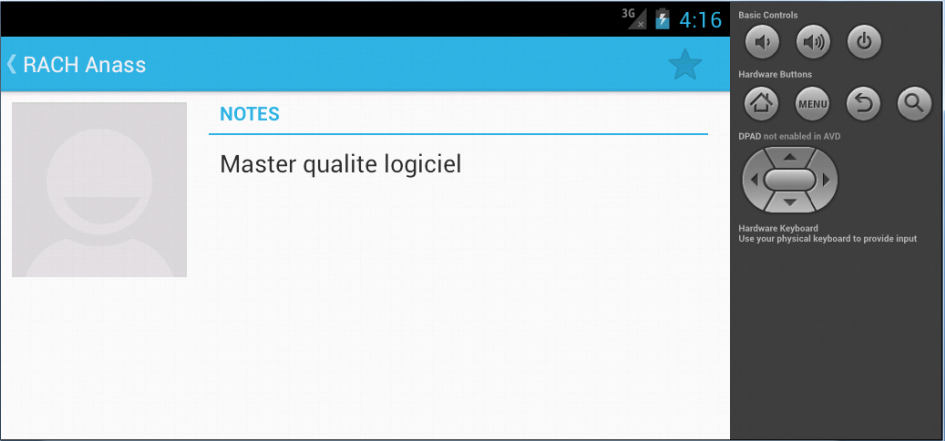
}

Execution de notre application sur un télèphone virtuel AVD Android :

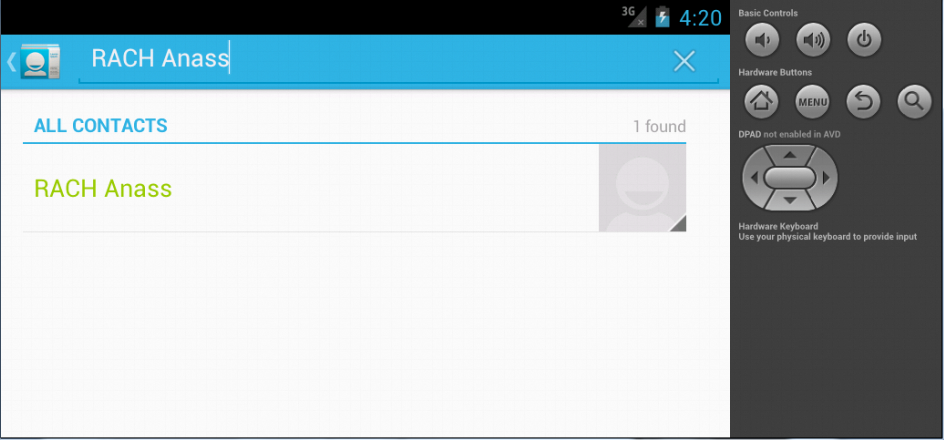


Saisie des données :

Confirmation du stockage :

Vérification, dans les contacts de notré télèphone virtuel on remarque l’ajout d’un nouveau contact :

On peut également retrouver le contact en effectuant une recherche sur la barre des contacts :



**Stockage :**

permet d'accéder aux options de stockage des appareils

Cette API est basée sur le W3C Web SQLDatabase Spécification et W3C Web Storage API Specification. Certains appareils offrent déjà une implémentation de cette spécification. Pour ces appareils, la prise en charge intégrée est utilisée au lieu de la remplacer par la mise en œuvre de Cordova. Pour les appareils qui n'ont pas de support de stockage, la mise en œuvre de Cordova devrait être compatible avec la spécification du W3C.

## Méthodes

* openDatabase

## Arguments

* database\_name
* database\_version
* database\_displayname
* database\_size

## Objets

* Base de données
* SQLTransaction
* SQLResultSet
* SQLResultSetRowList
* SQLError
* localStorage

## **Permissions**

### Android

#### app / res / xml / config.xml

<plugin name=" Storage " value="org.apache.cordova. Storage " />

### Bada

Aucune autorisation n'est requise.

### BlackBerry WebWorks

#### www / config.xml

<feature id="blackberry.widgetcache" required="true" version="1.0.0.0" />

### iOS

Aucune autorisation n'est requise.

### webOS

Aucune autorisation n'est requise.

### Windows Phone

Aucune autorisation n'est requise.

### Tizen

Aucune autorisation n'est requise.

# openDatabase

Retourne un nouveau objet base de données .

var dbShell = window. openDatabase ( database\_name , database\_version , database\_displayname , database\_size );

## Description

**window.** openDatabase retourne un nouveau objet base de données.

Cette méthode va créer un nouveau SQL Lite Base de données et renvoyer un objet base de données.

l’objet base de données retourné est utiliser pour manipuler les données.

## Plates-formes supportées

* Android
* BlackBerry WebWorks (OS 6.0 et supérieur)
* iPhone
* webOS
* Tizen

## Exemple rapide

var db = window. openDatabase ("test", "1.0", "Test DB", 1000000);

## **Exemple complet:**

<!DOCTYPE html>  
<html>  
  <head>  
    <title>Storage Example</title>  
  
    <script type="text/javascript" charset="utf-8" src="cordova-2.5.0.js"></script>  
    <script type="text/javascript" charset="utf-8">  
  
    // Wait for Cordova to load  
    //  
    document.addEventListener("deviceready", onDeviceReady, false);  
  
    // Cordova is ready  
    //  
    function onDeviceReady() {  
        var db = window.openDatabase("test", "1.0", "Test DB", 1000000);  
    }  
  
    </script>  
  </head>  
  <body>  
    <h1>Example</h1>  
    <p>Open Database</p>  
  </body>  
</html>

# database\_name

# Le nom de la base de données.

# database\_version

La version de la base de données.

# database\_displayname

# Le nom d'affichage de la base de données.

# database\_size

# La taille de la base de données en octets.

# Database

Contient des méthodes qui permettent à l'utilisateur de manipuler la base de données

## Méthodes

* **transaction:** Exécute une transaction de la base de données.
* **changeVersion:** méthode permet de vérifier automatiquement le numéro de la version et de le changer.

## Plates-formes supportées

* Android
* BlackBerry WebWorks (OS 6.0 et supérieur)
* iPhone
* webOS
* Tizen

## Transaction Exemple rapide

function populateDB(tx) {  
     tx.executeSql('DROP TABLE IF EXISTS DEMO');  
     tx.executeSql('CREATE TABLE IF NOT EXISTS DEMO (id unique, data)');  
     tx.executeSql('INSERT INTO DEMO (id, data) VALUES (1, "First row")');  
     tx.executeSql('INSERT INTO DEMO (id, data) VALUES (2, "Second row")');  
}  
  
function errorCB(err) {  
    alert("Error processing SQL: "+err.code);  
}  
  
function successCB() {  
    alert("success!");  
}  
  
var db = window.openDatabase("Database", "1.0", "Cordova Demo", 200000);  
db.transaction(populateDB, errorCB, successCB);

## Changer Version Exemple rapide

var db = window. openDatabase (" Database ", "1.0", "Cordova Demo", 200000); db.changeVersion("1.0", "1.1");

## 

## Exemple complet

<!DOCTYPE html>  
<html>  
  <head>  
    <title>Storage Example</title>  
  
    <script type="text/javascript" charset="utf-8" src="cordova-2.5.0.js"></script>  
    <script type="text/javascript" charset="utf-8">  
  
    // Wait for Cordova to load  
    //  
    document.addEventListener("deviceready", onDeviceReady, false);  
  
    // Cordova is ready  
    //  
    function onDeviceReady() {  
        var db = window.openDatabase("Database", "1.0", "Cordova Demo", 200000);  
        db.transaction(populateDB, errorCB, successCB);  
    }  
  
    // Populate the database   
    //  
    function populateDB(tx) {  
         tx.executeSql('DROP TABLE IF EXISTS DEMO');  
         tx.executeSql('CREATE TABLE IF NOT EXISTS DEMO (id unique, data)');  
         tx.executeSql('INSERT INTO DEMO (id, data) VALUES (1, "First row")');  
         tx.executeSql('INSERT INTO DEMO (id, data) VALUES (2, "Second row")');  
    }  
  
    // Transaction error callback  
    //  
    function errorCB(tx, err) {  
        alert("Error processing SQL: "+err);  
    }  
  
    // Transaction success callback  
    //  
    function successCB() {  
        alert("success!");  
    }  
  
    </script>  
  </head>  
  <body>  
    <h1>Example</h1>  
    <p>Database</p>  
  </body>  
</html>

# **SQLTransaction**

Contient des méthodes qui permettent à l'utilisateur d'exécuter des instructions SQL sur la base de données .

## Méthodes

* **executeSql:** exécute une instruction SQL .

## Plates-formes supportées

* Android
* BlackBerry WebWorks (OS 6.0 et supérieur)
* iPhone
* webOS
* Tizen

## **Exemple complet :**

<!DOCTYPE html>  
<html>  
  <head>  
    <title>Storage Example</title>  
  
    <script type="text/javascript" charset="utf-8" src="cordova-2.5.0.js"></script>  
    <script type="text/javascript" charset="utf-8">  
  
    // Wait for Cordova to load  
    //  
    document.addEventListener("deviceready", onDeviceReady, false);  
  
    // Cordova is ready  
    //  
    function onDeviceReady() {  
        var db = window.openDatabase("Database", "1.0", "Cordova Demo", 200000);  
        db.transaction(populateDB, errorCB, successCB);  
    }  
  
    // Populate the database   
    //  
    function populateDB(tx) {  
         tx.executeSql('DROP TABLE IF EXISTS DEMO');  
         tx.executeSql('CREATE TABLE IF NOT EXISTS DEMO (id unique, data)');  
         tx.executeSql('INSERT INTO DEMO (id, data) VALUES (1, "First row")');  
         tx.executeSql('INSERT INTO DEMO (id, data) VALUES (2, "Second row")');  
    }  
  
    // Transaction error callback  
    //  
    function errorCB(err) {  
        alert("Error processing SQL: "+err);  
    }  
  
    // Transaction success callback  
    //  
    function successCB() {  
        alert("success!");  
    }  
  
    </script>  
  </head>  
  <body>  
    <h1>Example</h1>  
    <p>SQLTransaction</p>  
  </body>  
</html>

# SQLResultSet

Lorsque la méthode ExecuteSQL d'un SQLTransaction est appelée, elle invoquer son rappel avec un SQLResultSet .

## **Propriétés**

* **insertId:** l'id de la ligne inserer par l'objet SQLResultSet dans la base de donnees
* **rowsAffected:** le nombre des lignes changé par une instruction sql. si aucune ligne n'est changée il est mis à 0.
* **rows:** une SQLResultSetRowList représentant les lignes retournées. Si aucune ligne n'est renvoyée l'objet sera vide.

## Plates-formes prises en charge

* Androïde
* BlackBerry WebWorks (OS 6.0 et supérieur)
* iPhone
* webOS
* Tizen

## **Exemple complet :**

<!DOCTYPE html>  
<html>  
  <head>  
    <title>Storage Example</title>  
  
    <script type="text/javascript" charset="utf-8" src="cordova-2.5.0.js"></script>  
    <script type="text/javascript" charset="utf-8">  
  
    // Wait for Cordova to load  
    //  
    document.addEventListener("deviceready", onDeviceReady, false);  
  
    // Populate the database   
    //  
    function populateDB(tx) {  
        tx.executeSql('DROP TABLE IF EXISTS DEMO');  
        tx.executeSql('CREATE TABLE IF NOT EXISTS DEMO (id unique, data)');  
        tx.executeSql('INSERT INTO DEMO (id, data) VALUES (1, "First row")');  
        tx.executeSql('INSERT INTO DEMO (id, data) VALUES (2, "Second row")');  
    }  
  
    // Query the database  
    //  
    function queryDB(tx) {  
        tx.executeSql('SELECT \* FROM DEMO', [], querySuccess, errorCB);  
    }  
  
    // Query the success callback  
    //  
    function querySuccess(tx, results) {  
        console.log("Returned rows = " + results.rows.length);  
        // this will be true since it was a select statement and so rowsAffected was 0  
        if (!results.rowsAffected) {  
            console.log('No rows affected!');  
            return false;  
        }  
        // for an insert statement, this property will return the ID of the last inserted row  
        console.log("Last inserted row ID = " + results.insertId);  
    }  
  
    // Transaction error callback  
    //  
    function errorCB(err) {  
        console.log("Error processing SQL: "+err.code);  
    }  
  
    // Transaction success callback  
    //  
    function successCB() {  
        var db = window.openDatabase("Database", "1.0", "Cordova Demo", 200000);  
        db.transaction(queryDB, errorCB);  
    }  
  
    // Cordova is ready  
    //  
    function onDeviceReady() {  
        var db = window.openDatabase("Database", "1.0", "Cordova Demo", 200000);  
        db.transaction(populateDB, errorCB, successCB);  
    }  
  
    </script>  
  </head>  
  <body>  
    <h1>Example</h1>  
    <p>Database</p>  
  </body>  
</html>

# **SQLResultSetRowList**

L'une des propriétés de la SQLResultSet contenant les lignes renvoyées par une requête SQL.

## Propriétés

* **length:** le nombre de lignes retournées par la requête SQL

## Méthodes

* **item:** retourne la ligne à l'index spécifié représenté par un objet JavaScript.

## Détails

Le SQLResultSetRowList contient les données renvoyées par une instruction SQL select. L'objet contient une propriété length qui vous permet de savoir combien de lignes de l'instruction select a été retourné. Pour obtenir une ligne de données que vous appelleriez l' item méthode spécifiant un index. La méthode renvoie un objet JavaScript Object qui est des propriétés sont les colonnes de la base de données de l'instruction select a été exécuté contre.

## Plates-formes prises en charge

* Androïde
* BlackBerry WebWorks (OS 6.0 et supérieur)
* iPhone
* webOS
* Tizen

## **Exemple complet :**

<!DOCTYPE html>  
<html>  
  <head>  
    <title>Storage Example</title>  
  
    <script type="text/javascript" charset="utf-8" src="cordova-2.5.0.js"></script>  
    <script type="text/javascript" charset="utf-8">  
  
    // Wait for Cordova to load  
    //  
    document.addEventListener("deviceready", onDeviceReady, false);  
  
    // Populate the database   
    //  
    function populateDB(tx) {  
        tx.executeSql('DROP TABLE IF EXISTS DEMO');  
        tx.executeSql('CREATE TABLE IF NOT EXISTS DEMO (id unique, data)');  
        tx.executeSql('INSERT INTO DEMO (id, data) VALUES (1, "First row")');  
        tx.executeSql('INSERT INTO DEMO (id, data) VALUES (2, "Second row")');  
    }  
  
    // Query the database  
    //  
    function queryDB(tx) {  
        tx.executeSql('SELECT \* FROM DEMO', [], querySuccess, errorCB);  
    }  
  
    // Query the success callback  
    //  
    function querySuccess(tx, results) {  
        var len = results.rows.length;  
        console.log("DEMO table: " + len + " rows found.");  
        for (var i=0; i<len; i++){  
            console.log("Row = " + i + " ID = " + results.rows.item(i).id + " Data =  " + results.rows.item(i).data);  
        }  
    }  
  
    // Transaction error callback  
    //  
    function errorCB(err) {  
        console.log("Error processing SQL: "+err.code);  
    }  
  
    // Transaction success callback  
    //  
    function successCB() {  
        var db = window.openDatabase("Database", "1.0", "Cordova Demo", 200000);  
        db.transaction(queryDB, errorCB);  
    }  
  
    // Cordova is ready  
    //  
    function onDeviceReady() {  
        var db = window.openDatabase("Database", "1.0", "Cordova Demo", 200000);  
        db.transaction(populateDB, errorCB, successCB);  
    }  
  
    </script>  
  </head>  
  <body>  
    <h1>Example</h1>  
    <p>Database</p>  
  </body>  
</html>

# **SQLError :**

Un SQLError objet est levée lorsqu'une erreur se produit.

## Propriétés

* **code:** Un des codes d'erreur prédéfinis listés ci-dessous.
* **message:** Une description de l'erreur.

## Constantes

* SQLError .UNKNOWN\_ERR
* SQLError .DATABASE\_ERR
* SQLError .VERSION\_ERR
* SQLError .TOO\_LARGE\_ERR
* SQLError .QUOTA\_ERR
* SQLError .SYNTAX\_ERR
* SQLError .CONSTRAINT\_ERR
* SQLError .TIMEOUT\_ERR

## Description

Le SQLError objet est levée lorsqu'une erreur se produit lors de la manipulation d'une base de données.

# localStorage

Donne accès à un interface de stockage W3C (http://dev.w3.org/html5/webstorage/ # l'option-localStorage-attribut)

var storage = window. localStorage ;

## Méthodes

* **key:** Retourne le nom de la clé à la position spécifiée.
* **getItem:** Retourne l'élément identifié par sa clé.
* **setItem:** Sauvegarde et point à la clé fournie.
* **removeItem:** Supprime l'élément identifié par sa clé.
* **clear:** efface toutes les paires clé-valeur.

## Détails

localStorage fournit une interface pour un W3C stockage d'interface. Il permet de sauvegarder les données clé-valeur.

## Plates-formes prises en charge

* Androïde
* BlackBerry WebWorks (OS 6.0 et supérieur)
* iPhone
* Windows Phone 7 et 8
* webOS
* Tizen

Exemple key

var keyName = window. localStorage .key(0);

Exemple Set Item

window. localStorage .setItem("key", "value");

Exemple Get Item

var value = window. localStorage .getItem("key");

// value is now equal to "value"

Exemple Remove Item

window. localStorage .removeItem("key");

Exemple Clear

window. localStorage .clear();

**Exemple complet :**

<!DOCTYPE html>

<html>

<head>

<title>Storage Example</title>

<script type="text/javascript" charset="utf-8" src="cordova-2.5.0.js"></script>

<script type="text/javascript" charset="utf-8">

// Wait for Cordova to load

//

document.addEventListener("deviceready", onDeviceReady, false);

// Cordova is ready

//

function onDeviceReady() {

window.localStorage.setItem("key", "value");

var keyname = window.localStorage.key(i);

// keyname is now equal to "key"

var value = window.localStorage.getItem("key");

// value is now equal to "value"

window.localStorage.removeItem("key");

window.localStorage.setItem("key2", "value2");

window.localStorage.clear();

// localStorage is now empty

}

</script>

</head>

<body>

<h1>Example</h1>

<p>localStorage</p>

</body>

</html>

## Windows Phone 7 Quirks

* notation par points n'est pas disponible sur Windows Phone 7. Veillez à utiliser: Fenêtre localStorage . .setItem / getItem, et non la spécification définie w3 appelle à la fenêtre localStorage . someKey = 'someValue »;

**Dossier :**

Cet objet contient des attributs d'un fichier unique.

**Propriétés**

**name**: Le nom du fichier (DOMString).

**fullPath**: Le chemin complet du fichier, y compris le nom du fichier (DOMString).

**Type**: Le type MIME du fichier (DOMString).

**lastmodifieddate**: La dernière fois que le fichier a été modifié (date).

**Taille**: La taille du fichier en octets (long).

**Méthodes**

**tranche**: sélectionner seulement une partie du fichier à lire.

**Détails**

**Tranche :**Retour d'un nouveau fichier objet, pour lequel FileReader ne renverra que la partie spécifiée du fichier. Les valeurs négatives pour début ou de fin sont mesurés à partir de la fin du fichier. Les indices sont toujours relatives à la tranche actuelle (voir l'exemple complet).

* Quelques concepts de base d’une l'API du fichier :
  + vous demandez un système de fichiers. Vous pouvez demander un système de fichiers persistante ou temporaire .
  + L'API prend en charge de base "CRUD" opérations pour les fichiers et les dossiers.
  + L'API prend en charge la lecture et l'écriture de fichiers, à la fois binaire et texte brut.
  + Probablement l'aspect le plus difficile, est que chaque opération est asynchrone. Donc, pour obtenir et lire un fichier implique environ 3 ou 4 niveaux de rappels

API du fichier PhoneGap est vraiment une mise en œuvre de l' API W3 fichier . Les docs PhoneGap parler de quelque chose de similaire dans le domaine de base de données il est donc logique pour le fichier docs être mis à jour, en plus c’est une API pour lire, écrire et naviguer hiérarchies du système de fichiers.

Le fichier objet contient des attributs d'un fichier unique. Vous pouvez obtenir une instance d'un fichier objet en appelant la méthode de fichier d'un FileEntry objet.

**Autorisations**

**Androïde**

**app / res / xml / config.xml**

<pluginname="File"value="org.apache.cordova.FileUtils" />

<pluginname=”FileTransfer”

value=”org.apache.cordova.FileTransfer”/>

**app / AndroidManifest.xml**

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

Plates-formes prises en charge

Androïde

BlackBerry WebWorks (OS 5.0 et plus)

iOS

Windows Phone 7 et 8

Windows 8

**Atelier :**

j'ai construit une application simple qui montre certains de ces principes.J'ai commencé avec quelques boutons simples qui me permettent de tester les opérations de base sur les fichiers:



Afin de faire quelque chose, j'ai besoin d'accéder au système de fichiers, et ce qui doit être fait après PhoneGap déclenche l'événement ***deviceready****:*

**Exemple*:***

function onDeviceReady() {

//request the persistent file system

window.requestFileSystem(LocalFileSystem.PERSISTENT, 0, onFSSuccess, onError);

}

function init() {

document.addEventListener("deviceready", onDeviceReady, true);

}

**LocalFileSystem( )**

* Cet objet fournit un moyen d'obtenir des systèmes de fichiers racine.
* **Méthodes :**
  + **requestFileSystem:** *Demande à un système de fichiers. (Fonction)*
  + **resolveLocalFileSystemURI:** Récupérer un DirectoryEntry ou FileEntry utilisant URI local. (Fonction)

Si le système de fichier est chargé, ***onFSSuccess*** gérera stocker un pointeur vers elle tout en la mise en place mes gestionnaires d'événements:

**Exemple :**

*function onFSSuccess(fs) {*

*fileSystem = fs;*

*getById("#dirListingButton").addEventListener("touchstart",doDirectoryListing);*

*getById("#addFileButton").addEventListener("touchstart",doAppendFile);*

*getById("#readFileButton").addEventListener("touchstart",doReadFile);*

*getById("#metadataFileButton").addEventListener("touchstart",doMetadataFile);*

*getById("#deleteFileButton").addEventListener("touchstart",doDeleteFile);*

*logit( "Got the file system: "+fileSystem.name +"*

*" +*

*"root entry name is "+fileSystem.root.name + "<p/>")*

*doDirectoryListing();*

*}*

Comme une petite parenthèse, getById est simplement un wrapper pour document.getElementById.

Notre objet système de fichiers a quelques propriétés que nous pouvons afficher, comme son nom par exemple. Il a également une propriété racine qui est un pointeur vers le répertoire racine.

**Root entry name :**

Ce gestionnaire d'événement se déclenche puis s'éteint ***doDirectoryListing***. Ceci est normalement exécuté par le bouton «Répertoire de matières Show" mais je automatiquement de fonctionner après le système de fichiers est ouvert.

**Exemple :**

function gotFiles(entries) {

var s = "";

for(var i=0,len=entries.length; i<len; i++) {

//entry objects include: isFile, isDirectory, name, fullPath

s+= entries[i].fullPath;

if (entries[i].isFile) {

s += " [F]";

}

else {

s += " [D]";

}

s += "

";

}

s+="<p/>";

logit(s);

}

function doDirectoryListing(e) {

//get a directory reader from our FS

var dirReader = fileSystem.root.createReader();

dirReader.readEntries(gotFiles,onError);

}

**Description :**

La fonction ***logit*** est simplement l'ajout dans un DIV sur la page HTML comme une technique mise au point rapide.

**Exemple :**

function appendFile(f) {

f.createWriter(function(writerOb) {

writerOb.onwrite=function() {

logit("Done writing to file.<p/>");

}

//go to the end of the file...

writerOb.seek(writerOb.length);

writerOb.write("Test at "+new Date().toString() + "\n");

})

}

function doAppendFile(e) {

fileSystem.root.getFile("test.txt", {create:true}, appendFile, onError);

}

**Description :**

 La lecture de bas en haut, le gestionnaire d'événement commence par créer un objet lecteur hors de la propriété root de l'objet système de fichiers. Pour obtenir les fichiers, vous ***readEntries*** appels simples, et d'utiliser un callback pour gérer le résultat. Les entrées (qui peuvent être des fichiers ou des répertoires) sont un simple tableau d'objets.

Que dire alors de la lecture et de l'écriture du fichier ? Ouverture d'un fichier est simple. Vous pouvez simplement lancer getFile (nom) et l'API peut (si vous voulez) également créer le fichier s'il n'existe pas. Cela simplifie un peu les choses. Voici le gestionnaire d'événements et de rappeler aux cliquant sur "Créer / Annexer au test de fichier".

**FileReader()**

FileReader : est un objet qui permet de lire un fichier.

* **Propriétés :**
  + ***Onload:*** Appelé lorsque la lecture a réussi *(Fonction).*
  + ***Onloadend:*** Appelé lorsque la demande est terminée (soit dans le succès ou l'échec) *(Fonction)*
  + ***Erreur****:* Une des erreurs d'objet contenant *( FileError )*
  + ***Onerror:*** Appelé lorsque la lecture a échoué *(Fonction).*
* **Méthodes**
  + ***annuler:*** Annule la lecture du fichier.
  + ***readAsDataURL*:** le fichier et les données de retour comme une url de données.
  + ***readAsText*:** Lit le fichier texte

Encore une fois - s'il vous plaît lire de bas en haut. Vous pouvez voir l'utilisation de getFile ici avec les options après pour assurer une erreur ne sera pas levée si elle n'existe pas. Ajout d'un fichier se fait en créant un objet écrivain. Je tenais à vous - et je vis moi-même ce - si vous ne cherchez pas à la fin du fichier que vous aurez réellement écraser les données, par opposition à l'ajout. Maintenant, penchons-nous sur la lecture:

**Exemple :**

function readFile(f) {

reader = new FileReader();

reader.onloadend = function(e) {

console.log("go to end");

logit("<pre>" + e.target.result + "</pre><p/>");

}

reader.readAsText(f);

}

function doReadFile(e) {

fileSystem.root.getFile("test.txt", {create:true}, readFile, onError);

}

Description :

Comme précédemment, on commence par ouvrir le fichier, et dans le rappel de réussite, créer un objet FileReader. Vous pouvez lire des données texte ou binaire en fonction de vos besoins. Dans cet exemple, notre contenu est tout le texte afin que nous readAsText et en ce rappel de l'ajouter à notre div.

***FileError( )***

'FileError' objet est défini en cas d'erreur dans l'une des méthodes de l'API de fichiers.

* ***Propriétés :***
  + **code**: L'un des codes d'erreur prédéfinis listés ci-dessous.
* **Constantes**
  + FileError .NOT\_FOUND\_ERR
  + FileError .SECURITY\_ERR
  + FileError .NOT\_READABLE\_ERR
  + FileError .ENCODING\_ERR
  + FileError .SYNTAX\_ERR
  + FileError .INVALID\_MODIFICATION\_ERR
  + FileError .PATH\_EXISTS\_ERR

**FileEntry( )**

Cet objet représente un fichier sur un système de fichiers. Il est défini dans des systèmes de spécification.

* **Méthodes**
  + ***getMetadata*:** Consulter les métadonnées d'un fichier.
  + ***setMetadata*:** Définir les métadonnées d'un fichier.