Développement Android

ESIEE / Key Consulting

Partie 2

v2.2 - 2012

INTENT

Intent

- Intent : intention
- Une Intent est une agrégation d'informations, décrivant de manière abstraite une action devant être effectué
- Informations principales
  - Une **action** : une String nommant l'action désirée
  - Une **donnée** : une String, sous la forme d'un URI, permettant d'identifier ou de localiser la ressource sujette de l'action
  - URI : scheme://host:port/path

Intent

- Informations secondaires :
  - Une ou plusieurs **catégories** : une catégorie donne une indication au système sur le genre de composant attendu
  - Un **type** : correspond à un type MIME, identifiant spécifiquement le type des données
  - Un **nom de composant** : défini le nom exact de la classe du composant qui sera utilisé
  - **Extras** : informations supplémentaires, sorte de HashTable

Intent - utilisation

- startActivity
- startActivityForResult
- startService
- BroadcastReceiver

Intent

- Une Intent est un message adressé :
  - Au système
  - À un composant : Service, Activity ou Broadcast Receiver
- Au système, l'intent indique à quel composant elle s'adresse
- Au composant, l'intent indique les paramètres et données annexes à utiliser

Utiliser une intent explicite

- Une intent est explicite si elle indique explicitement le nom du composant a utiliser
  - En donnant le nom complet de la classe
    - Package + nom de classe
  - En donnant le type de la classe
    - Uniquement dans ma propre application

EXEMPLE

```
Intent i1 = new Intent();
Intent.setClassName(“fr.myapp.mypkg”,
        “MyActivity”);


Intent i2 = new Intent();
Intent.setClass(getApplicationContext(),
        MyActivity.class);
```

Utiliser une intent implicite

- Un intent est implicite si elle ne précise pas le nom du composant à utiliser
- Permet de demander au système un composant dont on ne connaît pas le type exact

Intent - Exemple

- Exemple 1
  - action: android.intent.action.INSERT
  - data: content://com.google.provider.NotePad/notes
  - Demander l'activité permettant la création d'une nouvelle note dans la liste content://com.google.provider.NotePad/notes et permettant à l'utilisateur de l'éditer
- Exemple 2
  - action: com.android.notepad.action.EDIT_TITLE
  - data: content://com.google.provider.NotePad/notes/*ID*
  - Demander l'activité permettant d'afficher le titre de la note ID, et autorisant l'utilisateur à editer ce titre

Intent

- Appeler un composant de **son** application ou d'une **autre** application
- Pas d'appel direct au constructeur
- Limite les dépendances fortes entre composant
- Dépendance faible
  - Le composant a besoin d'une **fonctionnalité**
  - Et non plus d'une **implémentation**

Sélection d'un composant (implicite)

- Le système recherche parmi les applications installées une activité ou un service
- Chaque application, pour chaque activité et service, peut déclarer une balise <intent-filter>
- Cette balise est utilisé par le système pour déterminer si l'activité ou le service est un candidat potentiel

<INTENT-FILTER>

- Associé à une balise <activity>, <service>
- Décrit les intents auquel le composant peut répondre
- <intent-filter> peut contenir 3 types de balises filles :
  - <action> : filtre en fonction du nom de l'action
  - <category>: filtre en fonction du nom de la catégorie
  - <data> : filtre en fonction de l'URI en comparant le schéma, la source, le chemin d'accès sur l'hôte (en entier, par préfixe ou suffixe), ainsi que le type MIME

<INTENT-FILTER> - EXEMPLES

```
<intent-filter android:label="@string/resolve_title">
    <action
android:name="com.android.notepad.action.EDIT_TITLE" />
    <category android:name="android.intent.category.DEFAULT"
/>
    <category
android:name="android.intent.category.ALTERNATIVE" />
    <category
android:name="android.intent.category.SELECTED_ALTERNATI
VE" />
    <data
android:mimeType="vnd.android.cursor.item/vnd.google.note" />
</intent-filter>
```

Intent – Action prédéfini

- Voir la classe android.content.Intent

- Exemples :
  - ACTION_VIEW
  - ACTION_EDIT
  - ACTION_PICK
  - ACTION_DIAL
  - ACTION_SEARCH
  - …

Intéret pour l'utilisateur

- Remplacement des composants par défaut
- Faire apparaître votre fonctionnalité dans le menu d'autres applications
- Proposer des menus d'actions pour un type de données
  - Voir PackageManager
- Uniformisation du fonctionnement
  - Le même écran d'édition de contact peut être utilisé partout

Utiliser les services Android

Utiliser les services Android

- Android propose un nombre de service par défaut
- Ces services donnent accès aux fonctionnalités du téléphone
  - Wifi
  - Bluetooth
  - GPS
  - Accéléromètres
  - Recherche
  - Informations système (batterie, vibreur, alarme)

WifiManager

- Gestion de la listes des réseaux disponibles et de leur état de connexion
- Information sur le réseau actuellement connecté
  - Connexion/déconnexion
  - Force du signal
  - Nom/type de cryptage…
- Ajout de nouveaux réseau
  - Manuel
  - Recherche des réseaux environnants
- Activation/déactivitation du Wifi

WIFI – CONNEXION AU NET

- Package java.net

- Fonctionnement classique
  - Création d'un socket
  - connect()
  - getInputStream() / getOutputStream()
  - close()

LocationManager

- Réception périodique de la position GPS du téléphone
- Déclenchement d'une Intent lorsque le téléphone approche un certain endroit
- Récupération d'informations :
  - Nombre de satellites disponibles
  - Précision de la position
  - Etat d'activité du module GPS
- Voir la classe LocationListener

Google Maps

- Non disponible dans le SDK par défaut
- Nécessite l'installation d'API supplémentaires
  - Installation depuis Eclipse
- Nécessite une inscription
  - Procédure décrite : http://code.google.com/android/add-ons/google-apis/mapkey.html
- Fournit un vue UI : MapView

SearchManager

- Fourni une interface utilisateur de recherche unifiée
- Voir le tutorial
  - http://developer.android.com/guide/topics/search/index.html
- Recherche vocale
- Système de proposition de suggestion
- Recherche dans les données de votre application
- Recherche globale

AlarmManager

- Fonctionnalité « Reveil » du téléphone
- Programmer l'exécution de votre application à un certain moment dans le futur
  - Même lorsque votre application n'est pas active
- Le lancement de votre application se fait grâce à une Intent
  - Implémentation du BroadcastReceiver associé (utiliser les intents-filters!)

NotificationManager

- Indiquer à l'utilisateur qu'un événement intéressant est survenu
- Un icône persistante dans la barre de status
  - Lancement d'une application lorsque l'utilisateur appuie sur l'icône grâce à une Intent
- Allumer/Flasher l'une des LEDs de l'appareil
- Flash du rétro-éclairage, son, vibration

SensorManager

- Accès aux différents capteurs du téléphone
  - Accéléromètre, gyroscope, capteur de luminosité, boussole, baromètre
- Récupération des mesures d'un capteur en particulier
- Méthodes pour déterminer l'orientation du téléphone
  - Calcul de la matrice de rotation
  - Calcul du vecteur d'orientation

Utiliser les services

- Depuis une activité
  - getSystemService(String name)
  - Le paramètre « name » est une des constantes définies par la classe android.context.Context
  - Exemple

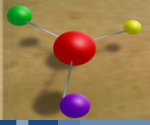
```
WifiManager wifi = getSystemService(
        Context.WIFI_MANAGER);
```

Utiliser les services

- La plupart des services propose des interfaces d'écoute
- Identifier l'interface/classe abstraite à implémenter ou étendre (appelé « listener »)
- Identifier la méthode du service permettant l'enregistrement auprès du service (en général addListener)

Utiliser les services

- Bien lire la documentation et les recommandations fournies par Google
- Plusieurs services consomment de l'énergie
- La documentation fournie des informations sur le fonctionnement attendue par l'utilisateur
  - Quand/comment le service est activé ?
  - Qui le désactive ?



Dalvik VM

VIRTUAL MACHINE

- VM : Virtual Machine
- Environnement d'exécution virtuel
- Cache les détails matériels de l'architecture
  - Jeu d'instruction indépendant du processeur -> bytecode
  - Indépendance vis-à-vis du type de RAM utilisé -> 32/64bits, endianness
- Fournit des services supplémentaires
  - Ramasse-miettes / Garbage collector (GC)
  - Sécurité : vérification à l'exécution

VM - Bytecode

- Code intermédiare, entre
  - L'assembleur (code natif/machine)
  - Et le langage de programmation
- Code interprété
  - Le processeur ne peut l'exécuter directement
  - Il doit être traduit dans le jeu d'instruction du processeur
- Une VM est donc un interpréteur de bytecode
- Problème : beaucoup plus lent que du code natif

JIT COMPILER

- Just – in – Time, compilateur juste à temps
- Compile le bytecode en code natif, a la volée, lors de l'exécution du programme
- Enormement d'implémentation différentes
  - Quand compiler ?
    - Installation, lancement, appel de la méthode, chargement d'une instruction
  - Quel objet compiler ?
    - Programme, dépendance, methode, trace, instruction

Facteurs de choix

- Contraintes d'une plateforme mobile
- Temps de compilation
  - Les applications ont un temps d'exécution court
- Espace mémoire utilisé
  - Le JIT ne doit pas altérer l'expérience utilisateur négativement
- Possibilités d'optimisation
  - Optimiser un code nécessite de comprendre son contexte d'exécution

Par méthode

- JVM, CLR

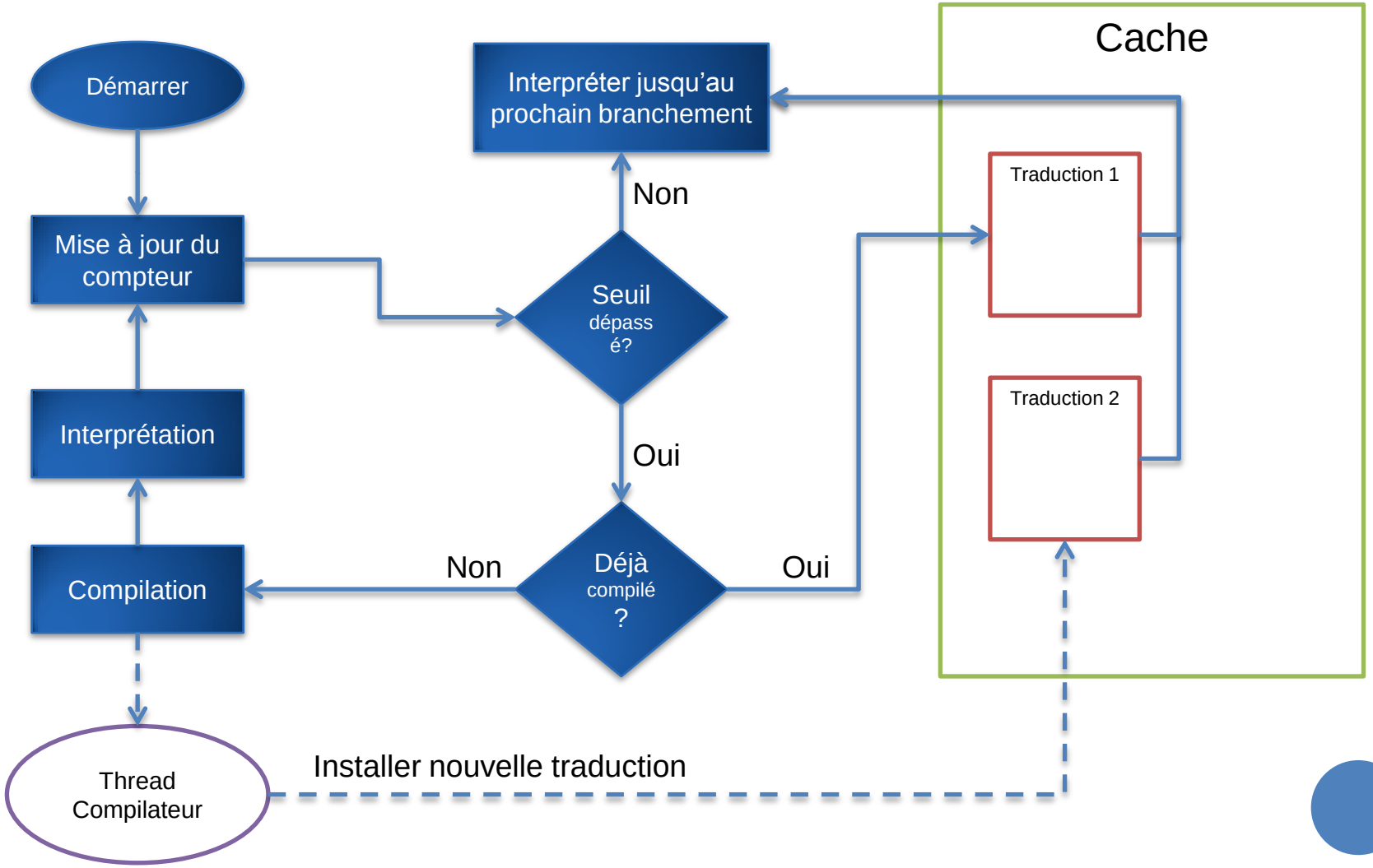
- Détection des méthodes les plus utilisés
- Compile et optimise par méthodes

- Points forts :
  - Bonne capacité d'optimisation
  - Méthode la plus simple
- Points faibles :
  - Le code mort dans les méthodes est optimisé
  - Délai important avant d'obtenir le boost de performance

Par trace

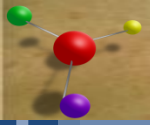
- Dalvik
- Identification de chemins d’exécution souvent utilisés
- Points forts :
  - Seul le code exécuté est optimisé
  - Le boost de performance arrive très vite
- Points faibles :
  - Optimisation plus difficile
  - Implémentation plus complexe

DALVIK TRACE JIT



Dalvik JIT - Performances

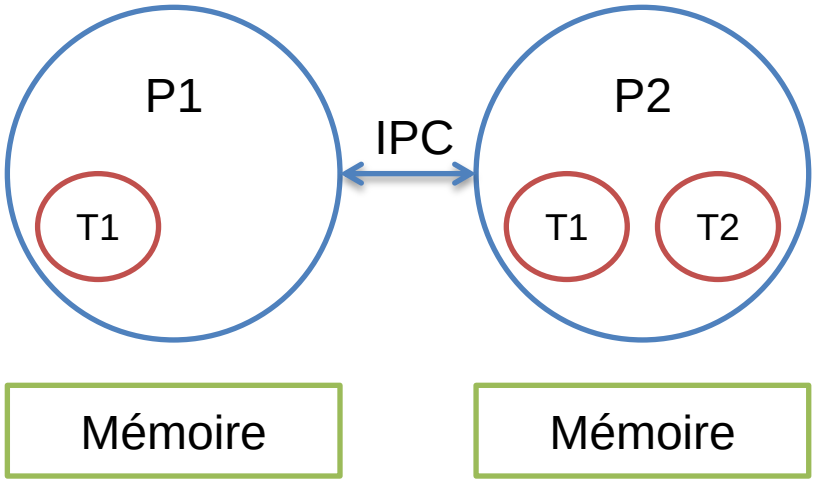
- Utilisation mémoire faible
- Gain de performance significatif
  - 2x - 5x sur des applications type calcul intensif
- Disponible à partir de Android 2.2



Threading

INTRODUCTION

- Parallélisme
- Thread
  - Fil d'exécution
  - Tâche
- Caractéristique
  - Dans un même processus
  - Partage le même espace mémoire



Anatomie d'une application Android

- Une application = un processus
- Par défaut, Android crée et lance un unique thread
- Ce thread est appelé le thread principal, ou aussi le thread UI
- Vous pouvez créer, lancer et gérer vos propres threads
  - Même API que J2ME, J2EE

Anatomie d'une application Android

- Le thread UI exécute un Looper
- Un Looper
  - Une file de messages à traiter
  - Une boucle traitant les message un par un
- Types de messages
  - Evénement utilisateur
  - Animations
  - Evénement déclenché par le développeur
- Types développeur
  - Message
  - Runnable

Threading

- Objectif : éviter de rendre l'interface utilisateur inactive
- Tout le code de l'UI s'exécute sur le même thread
  - Le code appelé depuis l'interface aussi
- Si le traitement est long, l'application ne répond plus aux interactions de l'utilisateur
- => Erreur ANR (Application Not Responding)

Threading

- Type d'opérations longue pouvant survenir :
- Calcul intensif
- Lecture/Ecriture dans la mémoire flash
  - Jusqu’à 200/400ms
- Lecture/Ecriture sur le réseau
  - Jusqu’à plusieurs secondes

EXEMPLE

```
saveButton.setOnClickListener(new OnClickListener() {
        void onClick(View v) {
                Thread t = new Thread() {
                        void run() {
                                Bitmap b = loadImageNet();
                                mImage.setImageBitmap(b);
                        }
                }
        }
        t.start();
});
```

Threading

- Malheuresement, le code précédant ne fonctionne pas
- Le framework UI n'est pas thread-safe
  - Modèle « single-thread »
- Manipuler l'UI depuis un autre thread est dangereux
  - Bugs apparaissant aléatoirement
  - Difficile et long à debugger

Threading - UI

- La manipulation de l'interface doit se faire sur le thread UI
- Si on a accès une instance de l'UI :
  - Activity.runOnUiThread(Runnable r)
  - View.post(Runnable r)
- Sinon
  - Solution générique : Handler
  - Solution spécifique : AsyncTask
- Détection de mauvaise utilisation
  - Strict Mode

Threading - AsyncTask

```
public void onClick(View v) {
  new DownloadImageTask().execute("http://example.com/image.png");
}

private class DownloadImageTask extends AsyncTask<string, void,
Bitmap> {
     protected Bitmap doInBackground(String... urls) {
         return loadImageFromNetwork(urls[0]);
     }

     protected void onPostExecute(Bitmap result) {
         mImageView.setImageBitmap(result);
     }
 }
```

Threading - Handler

- Permet l'ajout de Message ou Runnable dans la file d'un Looper
- Le Looper peut être passer au constructeur
- Par défaut, utilise le Looper du thread sur lequel il a été instancié

Threading - Exemple

```
Handler h = new Handler();
saveButton.setOnClickListener(new OnClickListener() {
  void onClick(View v) {
     Thread t = new Thread() {
        void run() {
             Bitmap b = loadImageNet();
             h.post(new Runnable() {
               mImage.setImageBitmap(b);
             }
         }
     }
  }
  t.start();
});
```

Applications web

Spécifiques pour Android

Généralités

- 2 méthodes pour apporter du contenu web :
  - WebApplication : un site web
  - Mixte : un application qui contient un WebView
- Possibilité d'adapter le contenu web spécifiquement pour Android
- Possibilité d'utiliser l'API Android via JavaScript

Web Apps et taille d'écran

- 2 éléments à prendre en compte :
  - La taille de la zone d'affichage et le niveau de zoom
  - La densité de l'écran (dpi)
- Via le navigateur :
  - Par défaut la page est affichée en entier (Overview mode)
  - Possibilité de surcharger ce comportement et de spécifier les bornes min et max du zoom.
- Via un WebView :
  - Par défaut, le zoom est réglé à 100%

Viewport Metadata

- Le ViewPort est la zone d'affichage de la page web.
- Il peut avoir une résolution indépendante de l'écran.

```
<meta name="viewport"
      content="
          height = [pixel_value | device-height] ,
          width = [pixel_value | device-width ] ,
          initial-scale = float_value ,
          minimum-scale = float_value ,
          maximum-scale = float_value ,
          user-scalable = [yes | no] ,
          target-densitydpi = [dpi_value | device-dpi |
                               high-dpi | medium-dpi | low-dpi]
          " />
```

Viewport Metadata : taille

- Height / Width
  - Le mode overview à une largeur par défaut de mini 800px
  - Spécifier la largeur permet de « coller » à la page web
- Width
  - Les valeurs > 10000 sont ignorées
  - Les valeurs <=320 sont interprétées comme « device-width »
- Heigth
  - Les valeurs <200 et >10000 sont ignorées
- device-height / device-width
  - Adpate le ViewPort à l'écran physique

Viewport Metadata : zoom

- initial-scale
  - Facteur de zoom de la page [0.01-10]
  - Le zoom par défaut est calculé pour faire correspondre la page à la taille du ViewPort.
- minimum-scale
  - Niveau de zoom minimum
- maximum-scale
  - Niveau de zoom maximum
- user-scalable
  - yes|no
  - Autorise ou non l’utilisateur à effectuer des zoom.

Viewport Metadata : Densité

- La densité par défaut est « medium »
  - Soit 0.75x sur un écran « low »
  - Soit 1.5x sur un écran « high »

- target-densitydpi
  - device-dpi : Densité de l'écran physique, sans zoom.
  - high-dpi / medium-dpi / low-dpi : Force une densité.
  - <value> : Force une densité exprimée en dpi.

- Afficher sans déformation :

```
<meta name="viewport" content="target-densitydpi=device-dpi,
width=device-width" />
```

Viewport Metadata : Densité


```
width=device-width
```
ou
```
initial-scale=1.0
```

```
width=device-width,
target-densitydpi=device-dpi
```

Régler la densité via CSS

- Possibilité de définir des CSS différentes pour chaque densité

```
<link rel="stylesheet" media="screen and (-webkit-
device-pixel-ratio: 1.5)" href="hdpi.css" />

<link rel="stylesheet" media="screen and (-webkit-
device-pixel-ratio: 1.0)" href="mdpi.css" />

<link rel="stylesheet" media="screen and (-webkit-
device-pixel-ratio: 0.75)" href="ldpi.css" />
```

Régler la densité via JavaScript

- Propriété du DOM « devicePixelRatio », gérée par WebView et le navigateur Android

```
if (window.devicePixelRatio == 1.5) {
  alert("This is a high-density screen");
} else if (window.devicePixelRation == 0.75) {
  alert("This is a low-density screen");
}
```

WebView : Exemple

- Activity Layout

```
<?xml version="1.0" encoding="utf-8"?>
<WebView  xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
/>
```

- Chargement d'une page

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.loadUrl("http://www.example.com");
```

- Autoriser l'application à utiliser Internet

```
<manifest ... >
    <uses-permission android:name="android.permission.INTERNET" />
    ...
</manifest>
```

WebView : JavaScript

- Désactivé par défaut

```
WebView myWebView = (WebView) findViewById(R.id.webview);
WebSettings webSettings = myWebView.getSettings();
webSettings.setJavaScriptEnabled(true)
```

- Possibilité de lier du JavaScript à du code Android (binding)
  - On donne l'accès au JavaScript à une instance de classe Java
  - WebView se charge d'initialiser l'interface, les fonctions sont directement accessible du JavaScript
  - Exécution dans un thread séparé

- Sécurité : Ne pas autoriser la navigation vers d'autres pages

WebView : JavaScript - Exemple

```
public class JavaScriptInterface {
    Context mContext;

    /** Constructeur : initialise le contexte */
    JavaScriptInterface(Context c) {
        mContext = c;
    }

    /** Affiche un toast sur la page */
    public void showToast(String toast) {
        Toast.makeText(mContext, toast, Toast.LENGTH_SHORT).show();
    }
}
```

```
webView.addJavascriptInterface(new JavaScriptInterface(this), "Android");
```

```
<input type="button" value="Say hello" onClick="showAndroidToast('Hello
Android!')" />

<script type="text/javascript">
    function showAndroidToast(toast) {
        Android.showToast(toast);
    }
</script>
```

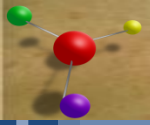
Web Apps : Debug

- Navigateur Android : utilisation des fonctions console de l'API JavaScript.
  - `console.log("Un message");`
  - Les messages apparaissent dans le logcat. C'est un log des messages systèmes.
    - Avec Eclipse :
    - Window > Show View > Other > Android > Logcat

- WebView
  - Implémenter WebChromeClient et surcharger la méthode onConsoleMessage()
  - Différent si API Level 7 ou 8 (Android 2.1 ou 2.2)

Web Apps : Bonnes pratiques

- Implémenter une version spécifique du site pour les smartphones
  - Redirection automatique
  - Frameworks multi-plateformes : jQuery
- Utiliser les meta-données pour configurer le viewport
- Limiter les accès à d'autres fichiers :
  - Inclure éventuellement la CSS dans la page
  - Compresser / optimiser : Minify
- Scrolling vertical uniquement



Frameworks

Open source et multiplateformes

Frameworks

- On chercher à développer une fois et à déployer sur plusieurs plateformes.

- 3 exemples :
  - PhoneGap
  - Titanium (Appcelerator)
  - Rhodes (Rhomobile)

PhoneGap

- Open source (MIT Licence)
- Gratuit
- Cibles : iOS, Android, BlackBerry, Symbian, Palm
- Langages : HTML, JavaScript, CSS

- Pas d'apparence « Native »

- APIs identiques à HTML5, le projet a une durée de vie limitée.

PhoneGap

rouge : pas possible pour cet appareil
jaune : en cours de développement
vert : fonctionnel

	iPhone	Android	Blackberry (OS 4.5)	Symbian	Windows Mobile	Palm
Geolocation						
Accelerometer			OS 4.7		HTC only	
Camera						
Vibration						
Contacts API						
SQLite Functionality						
XMPP API						
File system IO						Read only
Gesture / Multitouch		Android 2.0				
SMS API						
Telephone API						
Copy / Paste						
Sounds (Play)						
Sounds (Record)						
Bluetooth						
Wifi Adhoc connection						
Maps						
Orientation change						
Network availability						
Magnetometer	3GS only					
Storage						

Titanium (Appcelerator)

- Open source (Apache Public v2.0)
- Gratuit / $199 / $499
- Cibles : iOS, Android
- Langages : HTML, JavaScript, CSS

- Apparence Native

- Recompilation nécessaire pour chaque cible

Titanium (Appcelerator)

- API Graphique avancée

- Supporte :
  - Géolocalisation
  - Accéléromètre
  - Vibreur
  - Son (enregistrement et lecture)
  - Caméra
  - SQLite
  - Multitouch
  - Copier/Coller
  - Téléphone
  - Accès aux fichiers locaux (+contact et photos)

RHODES(RHOMOBILE)

- Open source (MIT Licence)
- Gratuit / $1000
- Cibles : iOS, Android, BlackBerry, Windows Mobile, Symbian
- Langages : HTML, Ruby

- Apparence Native

- Déploiement possible via RhoSync

- Meilleurs performances que les 2 précédents.

RHODES(RHOMOBILE)

- Open source (MIT Licence)
- Gratuit / $1000
- Cibles : iOS, Android, BlackBerry, Windows Mobile, Symbian
- Langages : HTML, Ruby

- Apparence Native

- Déploiement possible via RhoSync

- Meilleurs performances que les 2 précédents.

RHODES(RHOMOBILE)

Capability	iPhone	Windows Mobile	BlackBerry	Symbian	Android
GeoLocation	0.3	0.3	0.3	1.1	1.0
PIM Contacts	0.3	0.3	0.3	1.0	1.0
Camera	1.0	1.0	1.0	1.1	1.0
Date/Time picker	1.2.2	2.0	1.2	2.1	1.2
Native Menu/Tab Bar	1.2.2	2.0	1.2	2.1	1.5
Audio / Video capture	2.0	2.0	2.0	2.1	2.0
Bluetooth	2.0	2.0	2.0	2.1	2.0
Push / SMS	1.2	2.0	1.2	2.1	2.0
Landscape	2.0	2.0	2.0	2.1	2.0
Native Maps	1.4	2.0	1.4	2.0	1.5
Alerts / Audio File Playback	1.2	1.5	1.2	2.0	1.2

Frameworks

Framework		
PhoneGap	Nombre de plateformes supportées	Pas d'apparence native
Titanium	Apparence native	iOS et Android seulement
Rhodes	Nombre de plateformes supportées Performances	Nombre de fonctionalités