Utiliser le Text to Speech dans une application Android

par Sylvain Berfini (Mes tutoriels) (Blog)

Date de publication : 05/12/2011

Dernière mise à jour :

Le but de cet article est de vous apprendre à utiliser la fonction Text to Speech (littéralement Texte vers Parole) pour permettre à vos applications de lire du texte.

I - A quoi ça peut bien servir ?

Aujourd'hui, dans le monde du développement d'application, il y a un secteur trop peu connu : celui de l'interaction Hommes-machines. Le but de ceux qui s'y intéressent est de sensibiliser et valider les bonnes pratiques pour permettre aux gens comme Madame Michu de pouvoir utiliser un logiciel.

Mais en plus de l'ergonomie et de l'utilisabilité, il y a également l'accessibilité. Il est de notre responsabilité de développeurs de prendre en compte toutes les formes d'interactions avec l'utilisateur afin de permettre au plus large spectre d'utilisateurs de pouvoir se servir d'une application, quelle qu'elle soit.

En utilisant (intelligemment) une fonctionnalité comme le TTS (Text to Speech), on permet non seulement aux personnes malvoyantes de pouvoir interagir avec son téléphone ou son ordinateur, mais également à n'importe quelle personne de pouvoir utiliser une application sans forcément devoir avoir les yeux sur l'écran. Et c'est ce petit plus qui fera de votre application la prochaine "Killer-app".

Laissons maintenant de côté l'aspect éthique et théorique, pour nous concentrer sur la pratique.

Je ne détaillerai pas ici le développement d'une application Android, je suppose que vous avez les bases nécessaires. Si ce n'est pas le cas, je vous invite à consulter les autres tutoriels disponibles sur ce site.

II - Vérifier la présence d'un moteur de TTS sur le terminal

Avant de pouvoir se servir du TTS, il est nécessaire de vérifier la présence d'un "moteur" de TTS, puisque par défaut Android ne le propose pas. Certains constructeurs comme Samsung proposent le leur, d'autres incluent de base une version développée par d'autres. Quoi qu'il en soit, cette étape est nécessaire si vous ne désirez pas voir des dizaines de retour utilisateurs mécontents.

Pour vérifier la présence d'un moteur de TTS, rien de plus simple, puisqu'Android propose, à défaut d'un moteut intégré par défaut, une action pour vérifier la présence ou non de ce fameux moteur. Il suffit pour cela de surcharger la méthode onActivityResult de son Activity, et de regarder le résultat, comme ci-après.

Vérification de la présence du TTS sur le terminal

```
Intent checkIntent = new Intent();
checkIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
startActivityForResult(checkIntent, 0x01);

private TextToSpeech mTts;
protected void onActivityResult(
        int requestCode, int resultCode, Intent data) {
      if (requestCode == 0x01) {
          if (resultCode == TextToSpeech.Engine.CHECK_VOICE_DATA_PASS) {
              // Succès, au moins un moteur de TTS à été trouvé, on l'instancie
              mTts = new TextToSpeech(this, this);
          } else {

  // Echec, aucun moteur n'a été trouvé, on propose à l'utilisateur d'en installer un depuis le Market
              Intent installIntent = new Intent();
              installIntent.setAction(TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA);
              startActivity(installIntent);
          }
      }
     }
}
```

Vous remarquerez qu'en plus de faire une vérification, on propose le cas échéant à l'utilisateur d'installer in moteur de TTS depuis l'Android Market.

A noter ici qu'il faut que l'Activity dans laquelle ce code est placé doit implémenter TextToSpeech.OnInitListener afin que l'appel à mTts = new TextToSpeech(this, this) soit correct ! Dans le cas contraire, il vous faudra remplacer le deuxième paramètre par le bon listener.

III - Configuration

Maintenant que l'on a obtenu une instance valide de notre moteur, passons à la configuration.

Tout d'abord, il pourrait être intéressant de pouvoir choisir la langue dans laquelle le texte sera prononcé. En effet, bien que la langue dépendent du texte lu, le résultat ne sera pas le même suivant la configuration du moteur. Par exemple, Paris ne se prononce pas de la même manière en Français qu'en Anglais.

Attention, il faut également penser à vérifier que la langue pour laquelle on configure le moteur est disponible !

Vérification et configuration du moteur TTS

```
if (mTts.isLanguageAvailable(Locale.FRANCE) == TextToSpeech.LANG_COUNTRY_AVAILABLE) {
 mTts.setLanguage(Locale.FRANCE);
}
```

En plus de la langue, il est également possible de paramétrer d'autres paramètres, comme la vitesse de lecture ou bien le ton de la voix. Voyez plutôt.

Paramètres additionnels

```
mTts.setSpeechRate(1); // 1 est la valeur par défaut. Une valeur inférieure rendra l'énonciation plus lente, une
mTts.setPitch(1); // 1 est la valeur par défaut. Une valeur inférieure rendra l'énonciation plus grave, une valeu
```

Bon, maintenant il est temps de tester tout ça avec un test !

IV - C'est le moment de faire parler pour de vrai notre application !

Comme précisé plus haut, il est nécessaire d'implémenter quelque part TextToSpeech.OnInitListener, qui oblige d'avoir une méthode onInit(int status). C'est dans cette méthode que je mettrais ma lecture, libre à vous de la déplacer ailleurs.

Lecture de texte

```
public void onInit(int status) {
        if (status == TextToSpeech.SUCCESS) {
            mTts.speak("Ceci est un test grandeur nature du tutoriel sur l'énonciation de texte.",
 TextToSpeech.QUEUE_FLUSH, null);
            mTts.speak("Ceci est un deuxième test !", TextToSpeech.QUEUE_ADD, null);
        }
}
```

Comme vous vous en doutez, il est nécessaire que le texte soit rédigé proprement pour qu'il soit lu correctement.

Comme nous venons de le voir, il suffit d'appeler la méthode speak pour énoncer du texte. Mais à quoi servent les deux autres paramètres ?

Il faut savoir que le moteur TTS ajoute dans une pile les textes à lire dans le cas où on lui demanderait de lire un texte alors qu'il est déjà en train d'en lire un autre. Le deuxième paramètre permet d'ajouter dans cette pile un texte, ou au contraire de lui demander de vider la pile pour ne prononcer que les textes dont les appels suivront. Dans mon exemple ci-dessus, mon premier appel vide la pile (au cas où), et le deuxième se contente de se rajouter derrière en attendant son tour.

Le dernier paramètre permet de spécifier à Android, via une table de hachage, quel flux de sortie vous souhaitez utiliser. Je ne m'attarde pas dessus, ce n'est pas nécessaire au bon déroulement de ce tutoriel. Vous pourrez trouver plus d'information à ce sujet si ça vous intéresse sur Internet.

V - Pour terminer

Pour finir ce tutoriel correctement, il me reste deux points à aborder. Le premier est de bien penser à fermer le moteur TTS une fois son utilisation terminée.

Fermeture du moteur TTS

Le deuxième point est qu'il est possible d'appeler du code une fois l'énonciation terminée, via un listener spécial : TextToSpeech.OnUtteranceCompletedListener.

VI - Conclusion et remerciements

Voilà la fin de ce tutoriel, j'espère qu'il vous aura plût. Si vous avez des questions, j'essaierai d'y répondre sur le forum.

Je remercie ?? pour sa relecture.