executive knowledge

ICT Formation professionnelle Suisse occidentale

Module 335

Réaliser une application pour mobile

CIE Informaticiens CFC – Journée 2

Flavio Pacifico

CIE 335 : Sommaire

- Théorie sur les ressources
- Exercice en relation
- Théorie sur la persistance des données
- Exercice en relation

CIE 335 : Les ressources

Introduction

Une application pour téléphone comprend, en sus des différents écrans et du code, un certain nombre de données appelées ressources. Ces données peuvent être de natures diverses, les plus courantes étant le texte et les images.

Ces ressources sont incluses dans le package de l'application, distribuées avec elle et accessibles facilement depuis celle-ci. On peut les voir comme des fichiers d'accompagnement, invisibles à l'utilisateur et empaquetés dans l'application elle-même.

Ces ressources ont un intérêt supplémentaire : il est possible de spécifier dans quel contexte une ressource est utilisable. Suivant les systèmes, ces contextes sont :

- ✓ La langue : notamment les ressources « texte » peuvent être différentes suivant la langue du système
- ✓ La résolution et/ou la taille de l'écran : les images, par exemple, peuvent être différentes, adaptées à l'écran
- ✓ L'orientation (portrait ou paysage) de l'écran : ici encore, il est intéressant d'adapter les images !
- ✓ La présence ou non d'un capteur, d'un périphérique
- ✓ La version du système d'exploitation

CIE 335 : Les ressources Capacitor

Ressources texte

Comme Capacitor ne crée pas réellement d'exécutable mais un site web, il n'y a pas à proprement parler de ressources texte incorporées dans le code ! Il est néanmoins possible d'obtenir une application accessible en plusieurs langues en fonction de la langue du système.

Plusieurs possibilités coexistent :

- ✓ L'utilisation du plugin : *cordova-plugin-globalization*
- ✓ L'utilisation de la bibliothèque : *JQuery.localize*
- ✓ Les API : *ECMAScript*
- ✓ *Les plugins des différents framework*

CIE 335 : Les ressources Capacitor

Ressources texte

Les API *ECMAScript* se nomme I18N. C'est un ensemble des bonnes pratiques pour permettre à des produits ou des services d'être lisiblement adaptés à toute culture visée.

Pour l'installation :

npm

npm install i18next –save

yarn

yarn add i18next

Et bien-sûr on oublie pas de l'intégrer au projet ensuite

CIE 335 : Les ressources Capacitor

Ressources texte

Les fichiers de ressources sont des fichiers JSON simples qui contiennent autant de paires *nom-valeur* que de chaînes à stocker. Les fichiers doivent avoir le même nom (par exemple **common**) mais dans un dossier translations et des sous dossier avec le code de langue (fr pour le français, en pour l'anglais, etc...).

Si nous avons, par exemple, deux chaînes : une appelée « title » et une appelée « text », nous aurons les fichiers JSON suivants pour le français et l'anglais :

translations/fr/common.json	{JSON}	translations/en/common.json

```json
{
    "title": "Bonjour !",
    "text ": "Ceci est un texte en français"
}
```

```json
{
    "title": "Hello !",
    "text ": "This is an english text"
}
```

CIE 335 : Les ressources Capacitor

Ressources texte

Dans le fichier javascript, on déclare une variable resources pour indiquer les langues :

```
const resources = {
  en: {
    translation: common_en,
  },
  fr: {
    translation: common_fr,
  },
};
```

CIE 335 : Les ressources Capacitor

Ressources texte

Il suffit ensuite d'initialiser le i18n :

```
i18n.use(detector).init({
  resources,
  fallbackLng: "en",
});
```

CIE 335 : Les ressources Capacitor

Ressources texte

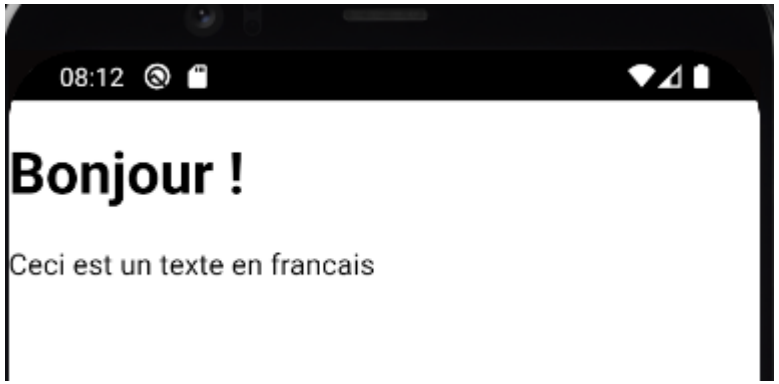
Une fois cela fait, il suffit de remplacer par un innerHTML les textes qui doivent être traduit grâce au i18n.t(‘clé’) :

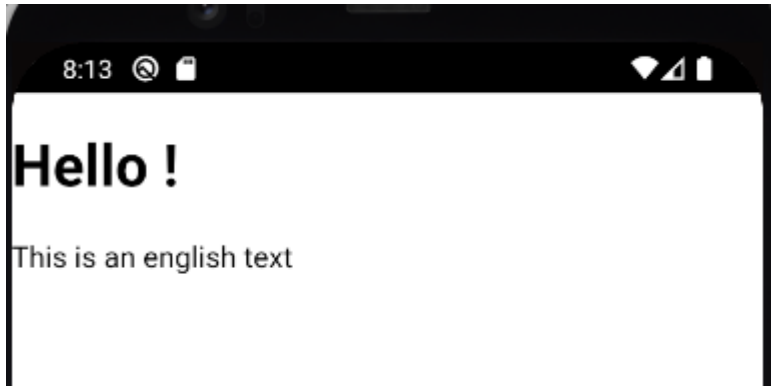
```
document.getElementById("sTitle").innerHTML = i18n.t("title");
document.getElementById("sText").innerHTML = i18n.t("text");
```

CIE 335 : Les ressources Capacitor

Ressources texte

Exemple d'une application utilisant les ressources texte :





CIE 335 : Les ressources Capacitor

Ressources texte

Il peut arriver que les ressources texte ne soient pas utilisées pour la traduction mais comme source de données. Dans ce cas, la ressource est un fichier stocké sur le serveur (Javascript est une API pour un navigateur). Pour y accéder, il faut donc utiliser une requête Ajax (un « *get* ») et récupérer la réponse (du texte brut, du json…)

Exemple de code Javascript (ECMAScript 6, utilisant les promesses pour simplifier la gestion de l'asynchronisme) qui lit un fichier texte (*word.txt*) et l'affiche dans un élément HTML de la page (un paragraphe dont l'identifiant est texte) :

```
fetch("/res/words.txt").then(function (reponse) {
    response.text().then(function (txt) {
        document.getElementById("texte").innerHTML = txt;
    });
}).catch(function (x) {
    alert(x);
});
```

CIE 335 : Les ressources Capacitor

Ressources image

Pour les ressources images, c'est encore plus simple puisqu'en HTML, une image utilise systématiquement un fichier extérieur ! La balise *img* contient donc directement le chemin vers la ressource, à savoir le fichier image utilisé pour l'affichage.

CIE 335 : Capacitor – Exercice 5

Exercice 5 – That’s ma traduction

Ict-moodle.ch

Jour 2 – Exercice 5

CIE 335 : La persistance des données

Introduction

Toutes les données manipulées par une application (mobile ou autre) sont présentes dans la mémoire de l'appareil. Lorsque l'application se ferme (fin normale, erreur, extinction de l'appareil...), celles-ci disparaissent. Il est important que certaines données persistent entre deux sessions : les paramètres de l'application (choix des couleurs par exemple), certaines données manipulées, etc.

CIE 335 : La persistance des données

Introduction

Il existe plusieurs méthodes pour avoir des données persistantes dans nos applications mobiles. Il est important d'assurer une sécurité minimale pour les données persistantes. Les systèmes mobiles sont (relativement) sécurisés sur ce principe : il ne faut pas qu'une application puisse manipuler les données d'une autre application, par exemple.

Néanmoins, il est parfois nécessaire à une application d'utiliser des données partagées. Prenons l'exemple des photos de votre téléphone : l'application gérant l'appareil photo doit pouvoir les écrire, une application permettant de les afficher doit pouvoir les lire, une application permettant de modifier une photo (supprimer les yeux rouges, par exemple) de les lire et de les écrire.

CIE 335 : La persistance des données

Introduction

Les systèmes mobiles découpent l'espace de stockage en plusieurs parties :

- ✓ L'espace réservé à l'application : inaccessible aux autres applications et accessible directement en lecture/écriture à l'application elle-même.
- ✓ L'espace partagé réservé aux images (photos, etc...) : accessible à toute application (lecture et écriture) qui indique dans son manifeste le souhait d'utiliser ce dossier.
- ✓ L'espace partagé réservé aux musiques : même principe que pour les images.
- ✓ L'espace partagé réservé aux vidéos (parfois confondu avec celui des images) : même principe.
- ✓ L'espace partagé réservé aux documents : même principe.
- ✓ L'espace réservé au système : accessible au système uniquement.
- ✓ Le reste de l'espace : accessible uniquement sur interaction visuelle avec l'utilisateur.

Il est intéressant de constater que les trois systèmes actuels (Android, Windows, iOS) sont similaires sur ces principes.

CIE 335 : La persistance des données

Paramètres de l’application

Tous les systèmes mobiles intègrent la notion de paramètres d'une application : chaque application peut gérer un petit nombre de données directement dans son espace privé avec un accès simple. Ce mode est à privilégier pour la persistance des données paramétrant l'application ou les données de celle-ci si elles sont peu nombreuses.

CIE 335 : La persistance des données

Paramètres Capacitor

HTML5 définit la possibilité pour un navigateur d'utiliser un stockage local simplement accessible par des paires clé/valeur. Ce stockage est limité à des chaînes (donc tout objet sérialisable) et à des petites quantités de données (<5M0).

L'accès à l'objet Javascript gérant le stockage local se fait par l'objet *window* :

```
var storage = window.localStorage;
```

CIE 335 : La persistance des données

Paramètres Capacitor

Cet exemple stocke un nom et un âge :

```
storage.setItem("nom","toto");
storage.setItem("age","15");
```

Et nous pouvons ensuite lire cela dans le stockage local :

```
var nom = storage.getItem("nom");
var age = storage.getItem("age");
```

CIE 335 : La persistance des données

Paramètres Capacitor

Dans le cas ou nous voulons sauvegarder plusieurs informations sur une même «key» nous devons mettre les données dans un tableau :

```
var namesArr = [];
namesArr.push('toto, Philippe, Gilbert');
storage.setItem('nom', JSON.stringify(namesArr));
```

Et pour lire les données :

```
var storedNames = JSON.parse(storage.getItem("nom"));
```

CIE 335 : La persistance des données

Fichiers

Sur l'ensemble des systèmes mobiles, une application ne peut accéder à l'intégralité des fichiers, pour des raisons de sécurité. La gestion de ceux-ci est donc bien moins souple qu'avec une programmation « classique » où il suffit de donner le chemin du fichier pour pouvoir y accéder.

CIE 335 : La persistance des données

Fichiers

Chaque application possède un dossier privé qui lui est associé dans lequel elle peut manipuler, en lecture/écriture, des fichiers de type quelconque. Ce dossier n'étant pas accessible aux autres applications, il est protégé contre toute intrusion, suppression, virus, etc.

CIE 335 : La persistance des données

Fichiers

Il existe également sur le téléphone des espaces « partagés » où toutes les applications peuvent lire et écrire des fichiers, sous réserve d'en avoir demandé l'autorisation :

- ✓ Les documents
- ✓ Les images
- ✓ Les musiques/sons
- ✓ Les vidéos
- ✓ Le support externe (carte SD par exemple)

CIE 335 : La persistance des données

Fichiers Capacitor

L'accès aux fichiers est possible en utilisant le plugin standard *@capacitor/filesystem*.

npm install @capacitor/filesystem

CIE 335 : La persistance des données

Fichiers Capacitor

L'objet *writeFile* du *FileSystem* permet d'accéde de sauver un texte dans le FileStorage

```
const writeAFile = async () => {
  await Filesystem.writeFile({
    path: 'secrets/text.txt',
    data: "This is a test",
    directory: Directory.Documents,
    encoding: Encoding.UTF8,
  });
};
```

CIE 335 : La persistance des données

Fichiers Capacitor

Pour lire un fichier texte et le transférer dans un paragraphe du document :

```
const readSecretFile = async () => {
  const contents = await Filesystem.readFile({
    path: 'secrets/text.txt',
    directory: Directory.Documents,
    encoding: Encoding.UTF8,
  });

  self.shadowRoot.getElementById('idDesc').innerHTML = contents.data;
};
```

CIE 335 : Capacitor – Exercice 6

Exercice 6 – Oh! You touch my data-ta

Ict-moodle.ch

Jour 2 – Exercice 6