**TP**

Réalisation d’un lecteur de flux RSS du site legorafi (http://www.legorafi.fr/feed).

**Exercice 1 – création de classes**

Créer un nouveau projet sous Xcode 5 pour iPhone, **avec utilisation de Core Data**.

Créer une classe nommée RSSItem qui possède les propriétés suivantes :

* un titre
* une date
* un contenu (texte)
* une url de lien
* une méthode d'initialisation prenant 4 paramètres (titre, date, contenu, url)

Création d'une classe **RSSFeed** avec :

* un titre
* un tableau de **RSSItem**
* une méthode pour ajouter un item
* une méthode pour insérer un item à un indice
* une méthode pour supprimer un item à un indice
* une méthode pour supprimer tous les items

**Exercice 2 – délégation**

Créer une classe **RSSReader** qui va permettre de parser un flux XML (RSS) avec :

* une méthode d'initialisation avec une url
* une méthode pour démarrer le parsing
* une méthode pour stopper le parsing
* une propriété en lecture seule permettant de récupérer le flux (objet de type **RSSFeed**)

Cette classe doit implémenter le protocole **NSXMLParserDelegate**.

Cette classe doit parser le flux (http://www.legorafi.fr/feed), créer une instance de **RSSFeed**, et lui ajouter des instances de **RSSItem**. Les balises à utiliser sont : channel, title, item, title (au sein de la balise item, à gérer avec un booléen pour savoir si l’on parse un item ou le channel), pubDate, description (attention c'est du CDATA), et link.

Créer un protocole « RSSReaderDelegate » avec :

* une méthode pour indiquer que le parsing est fini
* une méthode indiquant qu'il y a eu une erreur de parsing

Tester le bon fonctionnement.

**Exercice 3 – blocks**

Ajouter dans la classe **RSSReader** une méthode pour démarrer le parsing et fournir le résultat via un block. La signature de la méthode est du type :

-(void) startParsingWithCompletion:(void (^)(RSSFeed \*feed, NSError \*error))completionBlock;

Tester le bon fonctionnement.

**Exercice 4 – notifications**

Ajouter ou modifier dans la classe **RSSReader** une méthode pour démarrer le parsing et indiquer la fin du résultat par une notification.

Tester le bon fonctionnement.

**Exercice 5 – contrôleurs**

Créer un storyboard et l’indiquer au projet comme fichier d’interface principal en sélectionnant le projet, puis onglet Général et « Main Interface » et sélectionner le fichier créé.

Ajouter un contrôleur de type navigation.

Remplacer le contrôleur de type **UITableViewController** par un contrôleur de type **UICollectionViewController**, il va permettre d'afficher la liste des éléments du flux RSS. Créer une classe **RSSFeedViewController** pour ce contrôleur.

Chaque cellule **RSSItemViewCell** de cette collection view doit afficher :

* titre (UILabel)
* date (UILabel et NSDateFormatter)
* contenu d'un item limité à 1 ligne ou 2 max (UILabel).

Créer un contrôleur **RSSItemViewController** qui va permettre d'afficher le détail d'un item. Celui-ci doit afficher :

* le titre (UILabel)
* la date (UILabel + NSDateFormatter)
* le contenu complet (UITextView)
* un bouton permettant d'ouvrir le lien de l'item (UIButton). Voir les  méthodes sharedApplication, canOpenURL : et openURL : de la classe UIApplication pour ouvrir un lien dans Safari.

**Exercice 6 – Core Data**

Créer un modèle de données Core Data avec :

* une entité **RSSItem avec :**
* un titre
* une date
* un contenu (texte)
* une url de lien
* une relation unaire vers son feed
* une entité pour la classe **RSSFeed** avec :
* un titre
* l’url du flux (channelTitle)
* une relation n-aire vers ses items (la suppression de l’élément entraîne la suppression de ses items)

Modifier la classe **RSSReader** afin qu’elle utilise maintenant les objets Core Data à la place des classes créées dans le premier exercice.

**Attention :** il faut dorénavant commencer par vérifier qu’il n’y a pas déjà dans le modèle de données de flux correspondant à celui qui va être téléchargé. Si c’est le cas, il faut supprimer les items avant d’ajouter ceux récupérés du flux.

Ajouter un nouveau contrôleur de type **UICollectionViewController** nommé **RSSFeedListViewController** à placer avant **RSSFeedViewController**. Ce contrôleur doit contenir la liste des **RSSFeed** contenu dans la base de données Core Data. La sélection d’un élément dans la liste doit afficher l’écran **RSSFeedController** avec le contenu du flux de l’élément sélectionné.

Il doit également permettre l’ajout d’un nouveau flux dans la liste. Ajouter un **UIBarButtonItem** de type **ajout** dans la barre de navigation du contrôleur. Associer une action à ce bouton pour afficher une **UIAlertView** dans laquelle il faut saisir l’url du flux RSS. L’url saisie sert à créer un nouveau feed dans Core Data. Une fois le feed créé, afficher la liste de ses articles.

**Exercice 7 – chaînes localisées**

Ajouter un nouveau fichier de chaînes dans le projet et le nommer **Localizable.strings**.

Localiser les éléments suivants :

* titre du premier écran (le texte affiché dans la barre de navigation)
* les textes affichés dans l’alerte pour l’ajout d’un nouveau flux (textes + boutons)
* le format de la date affichée dans la liste des éléments d’un flux ou dans l’écran de détail d’un élément de flux
* le titre du bouton pour voir l’article complet

Tester l’application pour vérifier que les chaînes sont bien affichées

Ajouter une nouvelle langue à l’application en allant dans les réglages du **projet** puis onglet **Info** et partie **Localizations**.

Sélectionner le fichier **Localizable.strings** et cliquer sur le bouton **Localize** dans le volet à droite.

Modifier la langue du simulateur (Settings -> General -> International -> Language).

Tester le bon changement de langue dans l’application.

***Si la langue ne change pas dans l’application, il faut essayer de la désinstaller du simulateur, de faire un clean sous Xcode et de réinstaller.***

**Exercice 8 – réglages**

Ajouter un nouveau bundle de réglages au projet (Add File -> iOS -> Resource -> Settings Bundle).

Y ajouter un réglage de type slider qui va permettre de régler la hauteur des items d’un flux.

Ne pas oublier de localiser les réglages dans les langues définies précédemment.

Modifier l’application pour que ce réglage soit pris en compte au lancement et en live (notification système **NSUserDefaultsDidChangeNotification**).