

TP 6

RSS

À la fin de ce TP :

- Faire une archive contenant les projets Xcode des exercices
- Envoyer l'archive à ahumiliere@captaintrain.com avec l'objet : [DANT] TP 6 – Prénom Nom
- Si le TP est fait à plusieurs, préciser les noms et adresses mail de chacun

Le TP se base sur le projet de base « TP6 - Initial Project » disponible sur adhumi.fr/teaching.

Exercice 1

- Observer le contenu du projet. Il n'y a ni Storyboard ni View Controller, mais un **main.swift**. Il s'agit d'un projet Xcode « vide ».
- Lancer l'application. Un écran noir apparaît.
- Toutes les programmes Swift ont un point d'entrée (« application entry point »). Il s'agit de **main.swift**. Il ne fait qu'appeler la fonction `UIApplicationMain()`. `UIApplicationMain()` prend en paramètres les arguments de la ligne de commande et le nom de la class `AppDelegate` du projet. Voir la documentation de cette fonction. `UIApplicationMain` instancie l'objet `UIApplication`, lui assigne un delegate, lance la boucle principale et charge le Storyboard si il a été configuré.
- Supprimer le fichier **main.swift**. Lancer l'application et noter l'erreur de compilation.
- Il existe en Swift `@UIApplicationMain` qui synthétise le point d'entrée et permet d'éviter l'utilisation de **main.swift**.
- Ajouter `@UIApplicationMain` au dessus de la définition de la classe `AppDelegate`.
- Lancer l'application. L'écran noir apparaît (L'application n'a pas de Storyboard, l'écran noir est l'absence d'interface par défaut).

Exercice 2

- Ajouter un nouveau Storyboard appelé **Main.storyboard**, l'enregistrer dans le dossier **Base.lproj**.
- Sélectionner le projet **RSSReader** dans la barre de gauche et définir **Main** comme l'interface principale.
- Lancer l'application et observer le résultat. Voir le message d'erreur dans la console. Un Storyboard principal est défini, mais il n'a pas de ViewController par défaut.
- Dans **Main.storyboard**, ajouter un nouveau Tab Bar Controller. Interface Builder ajoute automatiquement deux scènes supplémentaires.
- Sélectionner le tab bar controller, ouvrir l'Attributes Inspector (⌘4), s'assurer que **Is Initial View Controller** est bien coché. Interface Builder affiche une flèche sur le controller « par défaut ».
- Lancer l'application et observer le résultat.

Exercice 3

- Ouvrir **Images.xcassets** et ajouter les images fournies avec les ressources. Elles sont automatiquement triées. Chaque image existe en 3 résolutions, correspondant aux différentes densités de pixels des écrans (1pt = 1px, @2x : 1pt = 2px, @3x : 1pt = 3px).
- Dans Interface Builder, sélectionner le tab bar button « Item 1 » en bas du view controller « Item 1 ». Dans l'Attributes Inspector, changer le titre en **Top Song** et l'image en **TopSongIcon**. Observer les changements.
- Pour « Item 2 » choisir **Top Album** et **TopAlbumIcon**.
- Ajouter un nouveau view controller au Storyboard et le nommer **Top App**.



- Ajouter une connexion entre le tab bar controller et ce nouveau view controller. Sélectionner une relation **view controllers**. Observer le résultat dans Interface Builder.
- Le tab bar controller gère un tableau de view controllers qui sont affichés comme des onglets.
- Changer les attributs de titre et d'image en **Top App** et **TopAppIcon**.
- Répéter la même procédure pour ajouter un controller **Top Movie** dans un quatrième onglet.
- Lancer l'application et interagir avec les onglets.

Exercice 4

- Chaque view controller devra afficher le titre, l'artiste et l'image du ou de la meilleur-e chanson/album/app/film. Une possibilité serait de réaliser quatre implémentations séparées, mais chaque view controller fera exactement la même chose, simplement avec des données différentes.
- Dans Interface Builder, ajouter à Top Song un label (Title), un label (Artist) et une image view. Placer les contraintes Autolayout pour un rendu harmonieux. L'image view devra avoir un format carré.
- Lancer l'application. Les labels apparaissent, mais pas l'image view, puisqu'aucune image ne lui est attribuée.
- Sélectionner les trois éléments et les copier/coller dans la vue Top Album. Noter comment les contraintes entre ces éléments sont conservées, mais celles affectant le conteneur doivent être recréées.
- Recréer les contraintes manquantes.

- Effectuer la même opération pour les deux autres controllers.
- Ajouter une nouvelle classe au projet appelée `TopMediaController` qui hérite de `UIViewController`.
- Avec Interface Builder, sélectionner chaque view controller et utiliser l'Identity Inspector (⌘⌘3) pour définir la classe comme étant `TopMediaController`.
- Dans l'implémentation de `TopMediaController`, déclarer trois outlet pour les éléments d'interface.

```
@IBOutlet weak var titleLabel: UILabel!  
@IBOutlet weak var artistLabel: UILabel!  
@IBOutlet weak var imageView: UIImageView!
```

- Effectuer les connections entre les éléments des différents view controllers et ces outlets.
- Dans l'implémentation de `TopMediaController`, modifier `viewDidLoad` pour éditer `titleLabel.text`.

```
override func viewDidLoad() {  
    super.viewDidLoad()  
    titleLabel.text = "Media Title"  
}
```

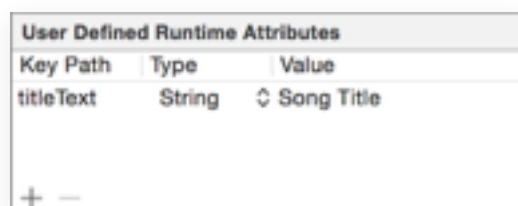
- Lancer l'application et interagir avec les différents onglets. Chaque Title label doit afficher **Media Title**.

Exercice 5

- Nous allons avoir besoin que `TopMediaController` affiche de données différentes pour chaque view controller. Pour le moment, nous utilisons un string « en dur » comme texte pour `titleLabel`.
- Ajouter une propriété `titleText` à la classe `TopMediaController`.

```
var titleText: String?
```

- La propriété utilise `var` et un type optionnel parce qu'elle ne sera pas assignée pendant l'initialisation de la classe `TopMediaController`.
- Avec Interface Builder, sélectionner le view controller `Top Song`. Dans l'Identity Inspector, ajouter une nouvelle ligne dans « User Defined Runtime Attributes » avec le Key Path `titleText`, le Type `String` et la Value `Song Title`.



User Defined Runtime Attributes		
Key Path	Type	Value
titleText	String	↻ Song Title

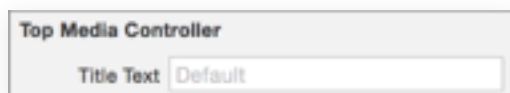
- Répéter cette opération pour les autres controllers, avec des valeurs adaptées.
- Mettre à jour l'implémentation de `viewDidLoad` pour utiliser la propriété `titleText`.

```
titleLabel.text = titleText!
```

- Lancer l'application et vérifier que les titres sont différents dans tous les onglets.
- Les « user defined runtime attributes » d'Interface Builder sont automatiquement assignées aux propriétés du view controller à l'initialisation.
- Interface Builder permet également d'assigner des valeurs aux propriétés du controller, directement depuis l'interface de Xcode et de l'Attributes Inspector.
- Ajouter l'attribut `@IBInspectable` à la propriété `titleText`.

```
@IBInspectable var titleText: String?
```

- Supprimer les « user defined runtime attributes » pour `titleText` et utiliser **Title Text** dans l'Attributes Inspector.



- Lancer l'application et observer le résultat.

Exercice 6

- Dans un navigateur, visiter <http://www.apple.com/rss/>.
- Choisir **Top 10 Songs**, et observer l'output.
- Dans le navigateur, modifier l'URL et remplacer `limit=10` par `limit=1` et `/xml` par `/json`. Le JSON est un format simple de données structurées.
- Ces données peuvent simplement être utilisées par une application iOS.
- Mettre à jour l'implémentation de `TopMediaController viewDidLoad` pour extraire les données JSON.

```
override func viewDidLoad() {
    super.viewDidLoad()
    let feedURL = "http://ax.itunes.apple.com/WebObjects/
MZStoreServices.woa/ws/RSS/topsongs/limit=1/json"
    let request = URLRequest(URL: NSURL(string: feedURL)!)
    NSURLConnection.sendAsynchronousRequest(request,
        queue: NSOperationQueue.mainQueue()) { response, data, error in
        if let jsonData = data,
            feed = (try? NSJSONSerialization.JSONObjectWithData(jsonData,
                options: .MutableContainers)) as? NSDictionary,
            title = feed.valueForKeyPath("feed.entry.im:name.label") as?
String,
            artist = feed.valueForKeyPath("feed.entry.im:artist.label")
as? String {
                self.titleLabel.text = title
                self.artistLabel.text = artist
            }
        }
}
```

- Lancer l'application et observer le résultat. Le titre et le nom de l'artiste apparaissent dans les champs.
- Discuter ce code avec l'enseignant.

- Comment va s'afficher le titre si il est vraiment très long ? Modifier l'implémentation de `viewDidLoad` pour simuler un titre de chanson arbitrairement long. Lancer l'app et observer le résultat.

```
...
self.titleLabel.text = "A Very Long Song Title (Long Title Remix)"
...
```

- Sélectionner le title label et, dans l'Attributes Inspector assigner l'attribut Autoshrink à une **Minimum Font Size** de **10**. Pour s'adapter, le label doit avoir une largeur définie ou maximale. Si ce n'est pas déjà le cas, modifier les contraintes en conséquences.
- Répéter ces modifications sur les différents controllers.
- Lancer l'application et observer le résultat.
- Pour une interface plus agréable, les labels devraient être masqués avant d'afficher le contenu de la requête. Dans Interface Builder, sélectionner les labels et cocher la case Hidden.
- Modifier l'implémentation de `TopMediaController viewDidLoad` pour afficher les labels quand la requête se termine.

```
...
self.titleLabel.text = title
self.titleLabel.hidden = false
self.artistLabel.text = artist
self.artistLabel.hidden = false
...
```

- Lancer l'application et observer le résultat.

Exercice 7

- Les flux RSS songs, albums, apps et movies ont des structures de données similaires, mais chaque view controller doit utiliser une URL de flux RSS différente.
- Dans `TopMediaController`, changer la propriété `titleText` en `feedURL`.
- Dans Interface Builder, indiquer les URL des feeds (voir **feedurls.txt**) pour chaque view controller.
- Mettre à jour `viewDidLoad` pour utiliser `feedURL`.

```
let request = URLRequest(URL: NSURL(string: feedURL!))
```

- Lancer l'application et observer le résultat.

Exercice 8

- Les flux RSS permettent d'accéder à des images pour chaque ressource. Ouvrir une des images dans un navigateur. Cette URL pourrait être utilisée pour créer une autre `NSURLConnection` qui récupérerait les données de l'image.
- Mettre à jour `viewDidLoad` pour récupérer l'URL de l'image dans le RSS, créer un objet `NSURL` et passer cette `NSURL` à une autre méthode du controller qui se chargera de télécharger l'image.

```
...
artist = feed.valueForKeyPath("feed.entry.im:artist.label") as?
String,
imageURLs = feed.valueForKeyPath("feed.entry.im:image") as?
[NSDictionary] {
    if let imageURL = imageURLs.last,
        imageURLString = imageURL.valueForKeyPath("label") as? String
    {
        self.loadImageFromURL(NSURL(string:imageURLString)!)
    }
    self.titleLabel.text = title
}
...
```

- Implémenter loadImageFromURL:

```
func loadImageFromURL(URL: NSURL) {
    let request = NSURLRequest(URL: URL)
    NSURLConnection.sendAsynchronousRequest(request,
        queue: NSOperationQueue.mainQueue()) { response, data, error
    in
        if let imageData = data {
            self.imageView.image = UIImage(data: imageData)
        }
    }
}
```

- Discuter ce code avec l'enseignant.
- Lancer l'application et *enjoy*.

