Cours n°3 - Bases d'une application

Le modèle MVC

Les vues

Les vues représentent les éléments visuels de votre application. Les deux principales choses à retenir sont que les vues peuvent être parentes d'autres vues et que, côté SDK, toutes les vues héritent de la classe `UIView`.

Trois grandes façons de designer vos applications (c'est à dire, définir et placer vos vues) existent : via les storyboards, en les créant dynamiquement avec du code ou en utilisant Swift UI.

La conception utilisant les storyboards consiste à glisser-déposer des éléments dans Xcode pour constituer l'interface de votre application. Chaque écran est appelé « scène » et peut contenir une ou plusieurs vues. (c.f. diapositives 5 et 6)

La conception utilisant des vues de façon dynamique consiste à créer des éléments par le biais des classes Swift proposées par le SDK. Il est possible de créer des designs plus complexes ou plus adaptatifs par ce biais mais cela s'avère plus fastidieux que d'utiliser les storyboards ou Swift UI. (c.f. diapositives 7 et 8)

Swift UI est outil relativement récent qui offre un bon compromis entre les storyboards et la création d'éléments avec du code. Cependant, l'outil étant relativement récent, il n'est utilisable que pour des applications supportant les deux dernières versions d'iOS. (c.f. diapositives 9 et 10)

Les contrôleurs

Les contrôleurs permettent de gérer les vues de notre application. Ces derniers sont représentés par des classes héritant de la classe `UIViewController`. A l'instar des composants en développement web par exemple, les contrôleurs transitent par différents états ; des méthodes sont appelées avant et après la transition d'un état à l'autre. Il est possible de définir ces méthodes (c.f. diapositive 12 à 14) au niveau du contrôleur pour intervenir dans le cycle. Par convention, les méthodes déclenchée avant la transition vers le nouvel état contiennent le mot `Will` alors que celles étant déclenchées après contiennent le mot `Did`.

Les modèles

Les modèles représentent les données présentes dans l'application. Leur modélisation se fait avec des classes Swift simples même si le SDK propose des facilités pour stocker, traiter et filtrer les données.

Les données peuvent être stockées sur différents médias et dans différents formats ou peuvent être échangées via une API.

Pour rappel, une API désigne ici un site avec lequel on échange des données. La plupart du temps, ces données sont échangées au format JSON (qui est un simple système de clé-valeur). (c.f. diapositive 16)

L'assemblage de ces trois éléments (modèle, vue et contrôleur) constituent les briques du pattern MVC. On retrouve au centre le contrôleur qui permet de faire le lien entre la partie affichage et les données. C'est également le contrôleur qui va gérer les évènements liés aux vues. (c.f. diapositive 17)

Développement d'une application

Les storyboards

Par défaut, une application possède deux fichiers de storyboards. Le fichier `Launchscreen` permet de designer l'apparence de l'écran affiché pendant le chargement de l'application. Le fichier `Main` contient toutes les scènes de notre application.

Au delà de la scène, deux concepts sont à noter ici : la « safe area » et le « storyboard entry point ».
La safe area représente la zone dans laquelle un élément est assuré d'être affiché quelque soit la configuration de l'écran actuellement (barre d'état située tout en haut, présence d'une barre de navigation, etc.).
Le storyboard entry point représente simplement le point d'entrée de votre application ; il pointe vers la scène qui sera affichée une fois l'application lancée.

Les vues dynamiques

Du côté des vues dynamiques, le SDK propose une classe pour chaque vue disponible par défaut. La chose importante à garder en tête est que chaque contrôleur possède une vue racine (disponible via sont attribut `view`) à laquelle il faut ajouter tous les éléments que l'on crée dynamiquement pour qu'ils soient visibles.

Lors de la création d'un élément dynamiquement (exemple diapositive 33), il faut généralement préciser ses dimensions et sa position. Ceci peut se faire par le biais de deux propriétés : `frame` ou `bounds`. Ces deux propriétés sont identiques à un détail près : `frame` renvoie les coordonnées de la position par rapport à la vue parente. Sauf rares exceptions, `bounds` renvoie `(0, 0)`. (Schéma diapositive 36).

Création d'un application

Des diapositives 37 à 41, simple démarche pour créer un nouveau projet dans Xcode.

Structure d'une application

Au niveau de la diapositive 42, on retrouve la structure par défaut d'une application iOS qu'il sera possible d'enrichir avec d'autres contrôleurs, des modèles, etc.

Connexion entre vues et contrôleurs

Des diapositives 43 à 47, simple démarche pour connecter une vue contenue dans un storyboard à un contrôleur.

Gestion des évènements

Les évènements s'appliquent, en théorie, uniquement aux éléments héritant de la classe `UIControl` (étant elle-même une sous classe de `UIView`). Comme pour le reste, il est possible de passer soit par les storyboards, soit par du code pour gérer les évènements.

La solution par les storyboards suit la même démarche que celle pour connecter un élément à un contrôleur ; il faudra simplement sélectionner « Action » dans la liste déroulante. (c.f. diapositives 49 et 50)

Note: Xcode part du principe que les propriétés sont définies au début de la classe du contrôleur et les gestionnaires d'évènements / actions à la fin. Le choix par défaut dans la liste déroulante ne sera pas le même selon l'endroit pointé dans le contrôleur de ce fait.

Lorsque l'on passe par du code pour définir un gestionnaire d'évènement (c.f. diapositive 51), la syntaxe est un peu particulière car la méthode gérant l'évènement doit être exposée côté Objective-C. Il faut donc l'annoter avec l'attribut `@objc` (plus ou moins l'équivalent des annotations en Java).

Simulateur

Pour tester l'application, le simulateur s'avère extrêmement pratique même si certaines fonctionnalités ne sont pas testables comme le gyroscope par exemple.

Il est possible d'accéder cependant à des fonctionnalités un peu plus spécifiques comme Face Id via la menu « Hardware » sous Xcode 11 ou « Features » sous Xcode 12.