Développement natif sous iOS

Cours n°7 - Médias, graphismes et localisation

iOS

Médias et graphismes

- Possibilité d’inclure des ressources statiques dans votre application

- Différents kits pour la gestion des graphismes (2D, jeux, réalité augmentée, etc.)

- Différents kits pour la gestion des médias (audio, vidéo, etc.)

Ressources

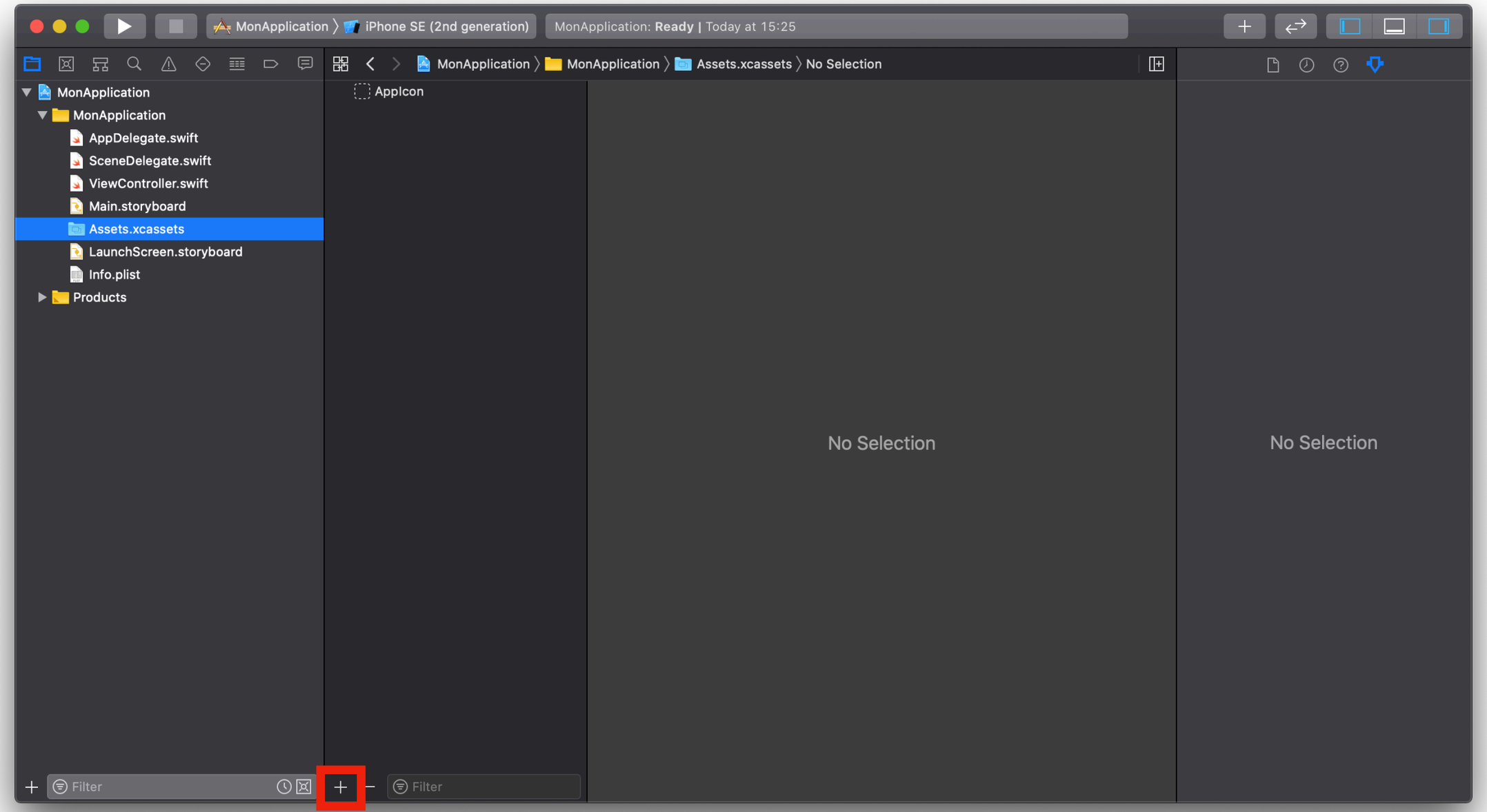
- Possibilités d’ajouter des fichiers divers (e.g. fichier de base de données SQLite3), accessibles depuis l’application

- Assets catalog : stockage de ressources graphiques
  - Images
  - Icônes
  - Textures
  - Set de couleurs

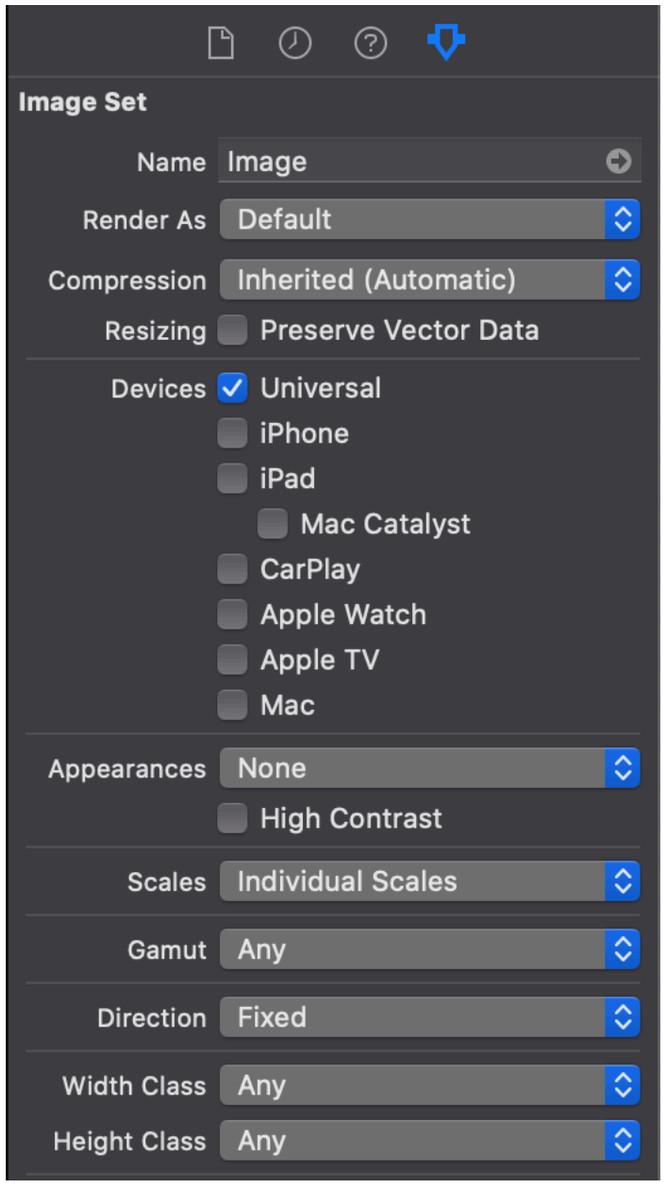
Ressources

- Les assets peuvent être fournies globalement ou :
    - Par appareil
        - Selon le thème (light ou dark)
        - Selon la langue
        - Par densité de pixels (pour les images)
- Elles sont stockées dans `Assets.xcassets`

Images


MonApplication 〉 iPhone SE (2nd generation)
MonApplication: Ready | Today at 15:25
MonApplication 〉 MonApplication 〉 Assets.xcassets 〉 No Selection
MonApplication
MonApplication
AppDelegate.swift
SceneDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products
AppIcon
No Selection
No Selection
Filter
Filter

Images



- Devices : Appareils sur lesquels l’image sera disponible
- Appearances : Thème clair ou sombre
- Direction : Pour les appareils affichant le text de gauche à droite ou de droite à gauche (e.g. en arabe)
- Width/Height class : classes de tailles (c.f. cours n°4)

Images

```
let image = UIImage(named: "monImage")
imageView.image = image
```

Icône de l’application

- Conseils :
    - Garder le fond simple et ne pas utiliser de transparence
    - Gardez les coins de votre icône carrés
    - Testez votre icône avec différents fonds d’écran
- https://developer.apple.com/design/human-interface-guidelines/ios/icons-and-images/app-icon/

Sets de couleurs

- Permettent de définir une couleur réutilisable facilement
- La création suit le même principe que pour les images
- La couleur peut être définie selon :
    - L’appareil
    - La langue
    - Le thème (clair ou sombre)

Sets de couleurs

- La couleur est ensuite utilisable depuis le storyboard
- Ou via le code :

```
let color = UIColor(named: "maCouleur")
view.backgroundColor = color
```

Graphismes

- Le SDK comporte de nombreux kits pour le traitement de ressources :
    - Core Animation : animations
    - Core Graphics : graphismes 2D
    - SceneKit, MetalKit : graphismes 3D
    - Core Image, PDFKit : traitement d’images/vidéos et PDF
    - ARKit, RealityKit : réalité augmentée

Core Graphics

- Gestion des graphismes 2D au sens large :
    - Création d’images
    - Affichage de texte
    - Création de fichiers PDF
    - Gestion des ombres et dégradés sur les objets
- La création d’images est très similaire aux canvas que l’on trouve en HTML/JavaScript

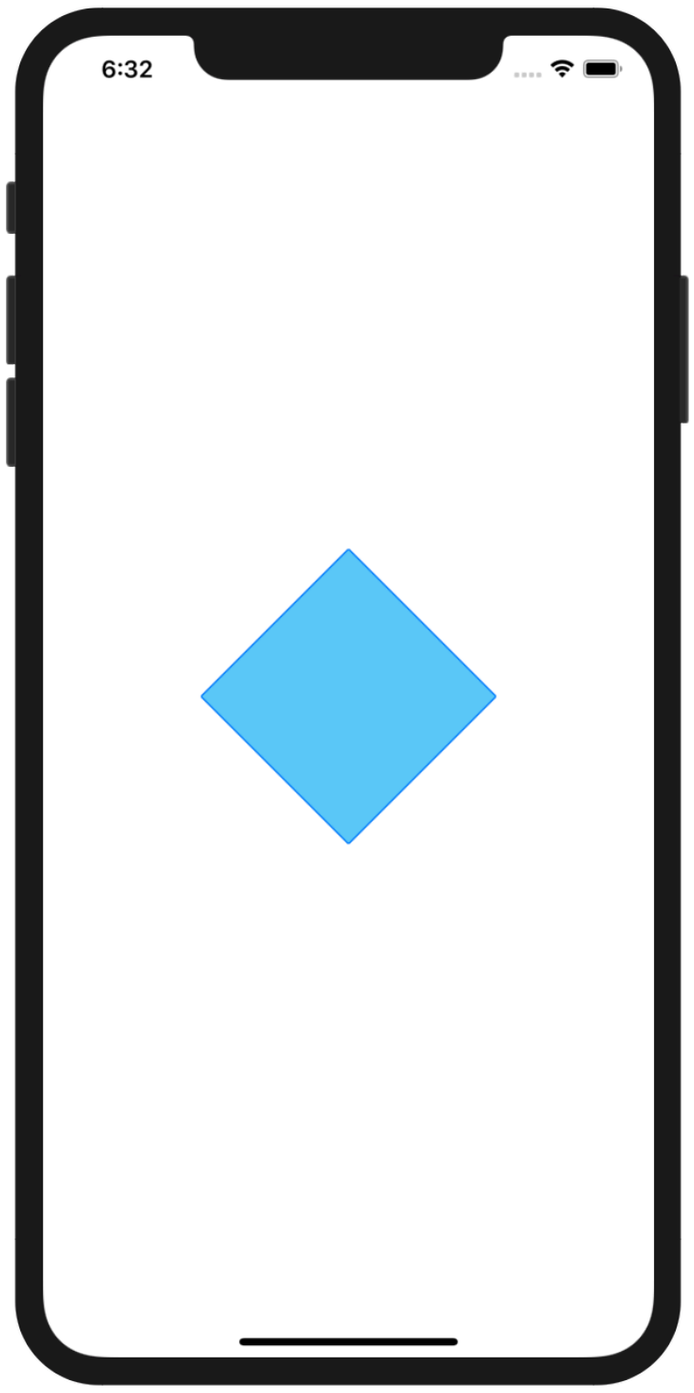
Core Graphics

```swift
let path = UIBezierPath()

// Origine du tracé
path.move(to: CGPoint(x: 100, y: 0))

// Ajout de lignes au tracé
path.addLine(to: CGPoint(x: 200, y: 100))
path.addLine(to: CGPoint(x: 100, y: 200))
path.addLine(to: CGPoint(x: 0, y: 100))
path.close()

// Couleurs de remplissage (fill) et bordure (stroke)
UIColor.systemTeal.setFill()
UIColor.systemBlue.setStroke()

path.fill()
path.stroke()
```

Core Graphics

```
class Figure : UIView {
    init() {
        super.init(frame: UIScreen.main.bounds)

        self.backgroundColor = UIColor.white
    }

    required init?(coder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    override func draw(_ rect: CGRect) {
        // ...
    }
}
```

Core Graphics



Médias

- Différents kits existent également pour la gestion des médias :
    - AVFoundation : gestion d’éléments audio-visuels
    - Core MIDI : interaction avec des appareils MIDI (claviers, synthétiseurs, etc.)
    - PhotoKit : gestion des photos et vidéos de l’utilisateur
    - HTTP Live Streaming : distribution d’audio et vidéos vers appareils iOS, tvOS ou macOS

Prise de photos / vidéos

- `UIImagePickerController` :
    - Très simple, standard
    - Aucune personnalisation
- AVFoundation :
    - Plus compliqué
    - Beaucoup plus de possibilités (personnalisation de l’interface, traitement de l’image, etc.)

Image picker controller

```
class ViewController : UIViewController {
    @IBOutlet var bouton: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()

        if !UIImagePickerController.isSourceTypeAvailable(.camera) {
            bouton.isHidden = true
        } else {
            bouton.addTarget(self, action: #selector(prendrePhoto), for: .touchDown)
        }
    }

    @objc prendrePhoto() {
        let imagePicker = UIImagePickerController()
        imagePicker.sourceType = .camera
        imagePicker.delegate = self

        present(imagePicker, animated: true, completion: nil)
    }
}
```

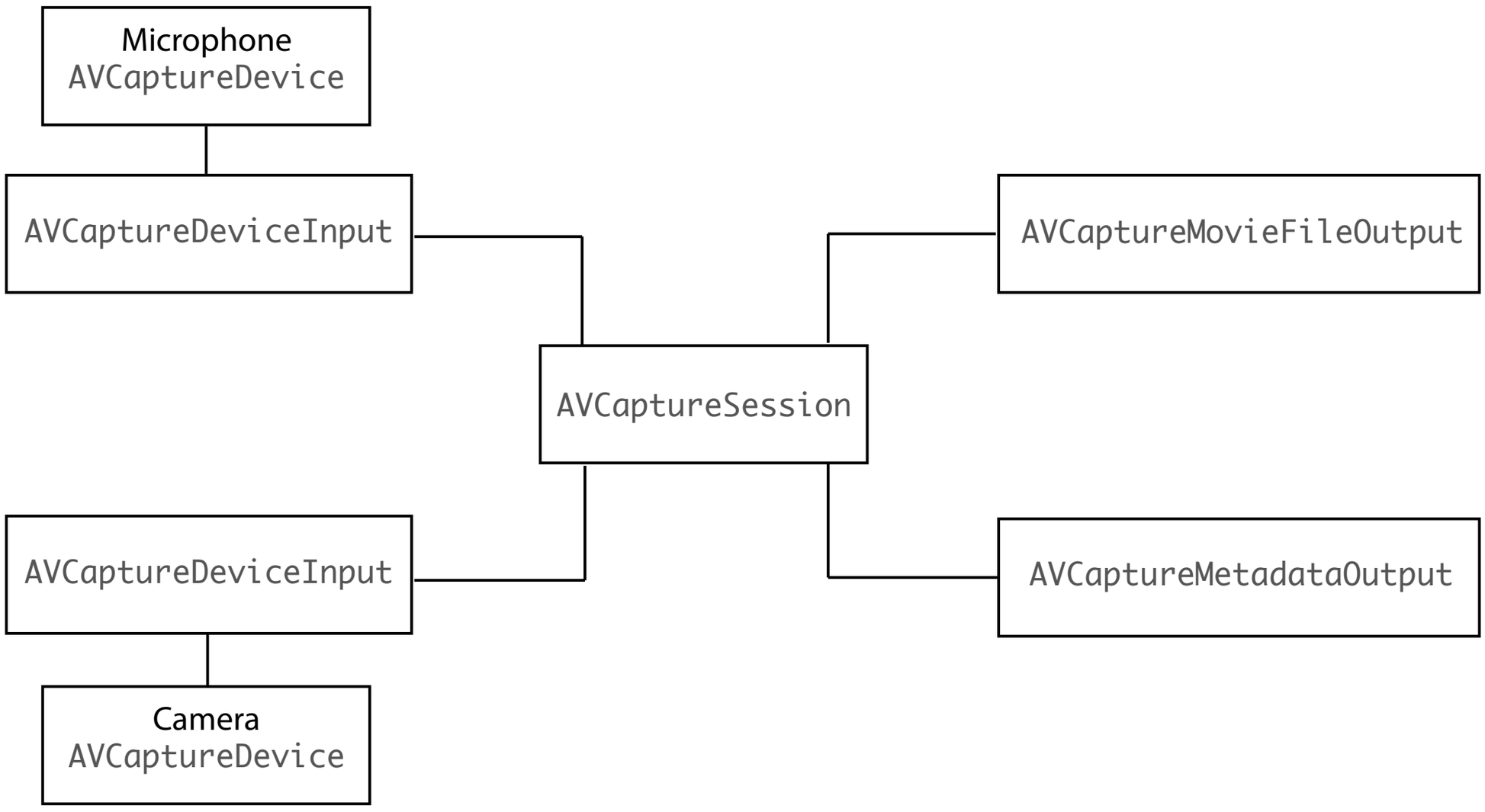
Image picker controller

```
extension ViewController : UIImagePickerControllerDelegate, UINavigationControllerDelegate {

    func imagePickerController(_ picker: UIImagePickerController,
            didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey : Any]) {
        dismiss(animated: true, completion: nil)

        let image = info[.originalImage] as? UIImage

        imageView.image = image
    }
}
```

AVFoundation

- C’est à vous de designer l’interface de capture dans le storyboard
    - Les boutons, contrôles, icônes, etc. sont à votre guise
    - Une vue (de type `UIView`) est à prévoir pour accueillir le rendu de l’appareil photo
- Vous devez gérer le cas où aucun dispositif de capture n’est disponible

AVFoundation



AVFoundation

- Différents éléments en entrée :
    - Caméra frontale
    - Caméra principale
    - Microphone
- Différentes fonctionnalités :
    - Caméra grand-angle
    - Longue focale
    - Triple appareil photo
- Vous pouvez utiliser `AVCaptureDevice.DiscoverySession` pour détecter les entrées disponibles

AVFoundation

- Différents types de sortie existent :
    - `AVCaptureMovieFileOutput` : fichier vidéo
    - `AVCaptureAudioFileOutput` : fichier audio
    - `AVCapturePhotoOutput` : fichier image ou Live Photo
    - `AVCaptureMetadataOutput` : détection d’objets ou de visages

AVFoundation

```
class VideoViewController : UIViewController {
    @IBOutlet var button: UIButton!
    @IBOutlet var preview: UIView!

    var session: AVCaptureSession!
    var previewLayer: AVCaptureVideoPreviewLayer!

    var started = false

    override func viewDidLoad() {
        bouton.addTarget(self, action: #selector(toggle),
            for: .touchDown)

        self.session = AVCaptureSession()
        self.previewLayer = AVCaptureVideoPreviewLayer(session: session)

        self.previewLayer.frame = preview.layer.frame
        self.previewLayer.videoGravity = .resizeAspectFill
    }

    @objc toggle() {
        if !started {
            startVideo()
        } else {
            stopVideo()
        }
    }
}
```

AVFoundation

```
func startVideo() {
    self.started = true

    let camera = AVCaptureDevice.default(for: .video)
    let micro  = AVCaptureDevice.default(for: .audio)

    do {
        let videoInput = try AVCaptureDeviceInput(device: camera!)
        let audioInput = try AVCaptureDeviceInput(device: micro!)

        session.addInput(videoInput)
        session.addInput(audioInput)

        let videoOutput = AVCaptureMovieFileOutput()

        session.addOutput(videoOutput)

        session.startRunning()
    } catch {
        // Une erreur est survenue lors de la création de la prise
    }
}
```

AVFoundation

```
func stopVideo() {
    self.started = false
    session.stopRunning()

    // Redirection de l'utilisateur vers un lecteur
    // de la vidéo par exemple.
}
```

Lecture d’audio

- Utilisation de la classe `AVAudioPlayer`
- L’interface est à la charge du développeur également
- Les données audio peuvent être fournies par un fichier ou chargées en mémoire (sous forme d’objet `Data`)

Lecture d’audio

```
guard let url = Bundle.main.url(forResource: "musique",
    withExtension: "mp3") else {
    // Aucun fichier trouvé
    return
}

do {
    let audioPlayer = try AVAudioPlayer(contentsOf: url)

    audioPlayer.volume = 0.5
    audioPlayer.play()
} catch {
    // Erreur de création du lecteur
}
```

Lecture vidéo

- Utilisation de la classe `AVPlayer` avec les classes `AVPlayerLayer` et `AVPlayerItem`

- Interface également à la charge du développeur

- Mêmes formats de données supportés que pour l’audio (fichier ou données en mémoire)

Localisation

- Internationalisation : processus rendant un code adapté à plusieurs langues — déjà fournie par Apple dans notre cas

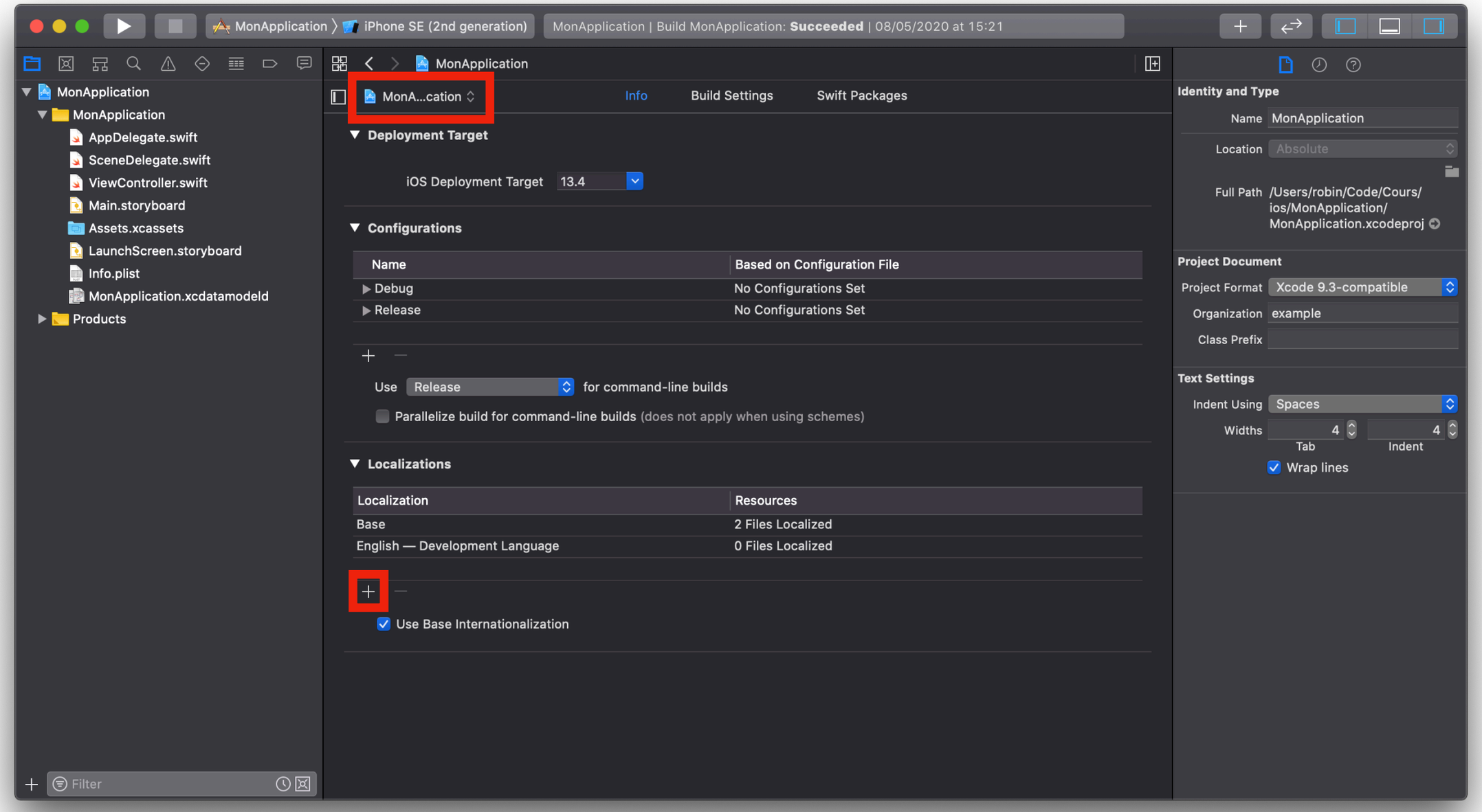
- Localisation : processus se basant sur l’internationalisation pour fournir l’application dans différentes langues — ici, ça sera la traduction à proprement parler

- Peuvent être trouvés sous leurs abréviations « i18n » et « l10n »

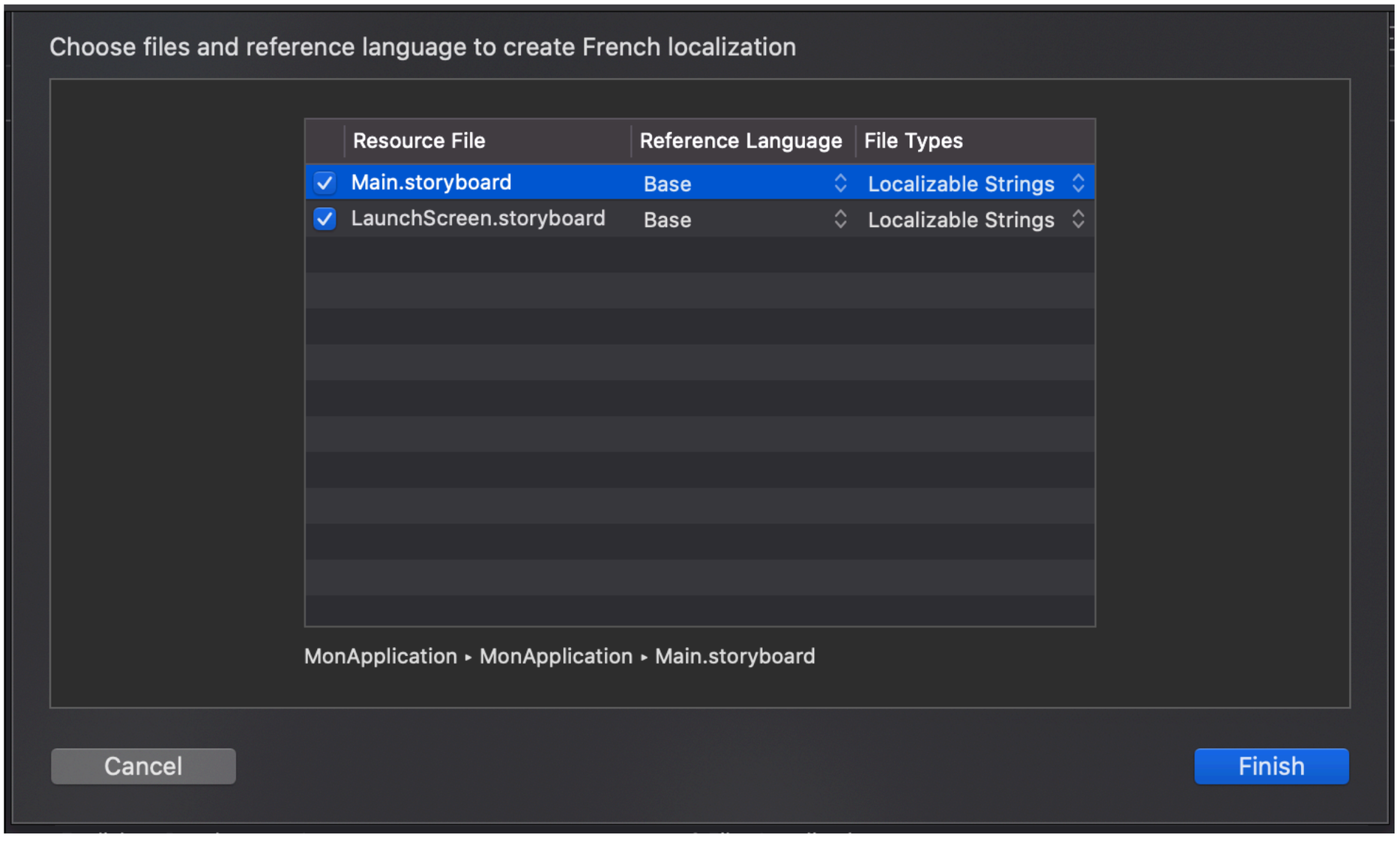
Localisation

- Possibilité de s’adapter selon la langue pour :
    - Les textes du storyboard
    - Les ressources
        - Certaines chaînes de caractères
- Rappel : les contraintes « leading » et « trailing » dans les storyboards, s’adaptent selon la langue
    - leading = gauche en anglais, français, etc
    - leading = droite en arabe, hébreux, etc.

Localisation

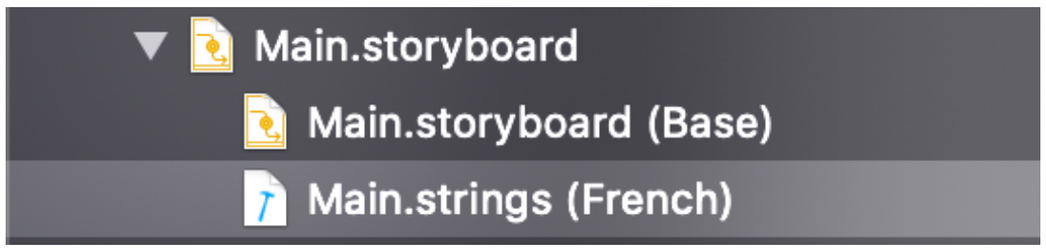


Localisation

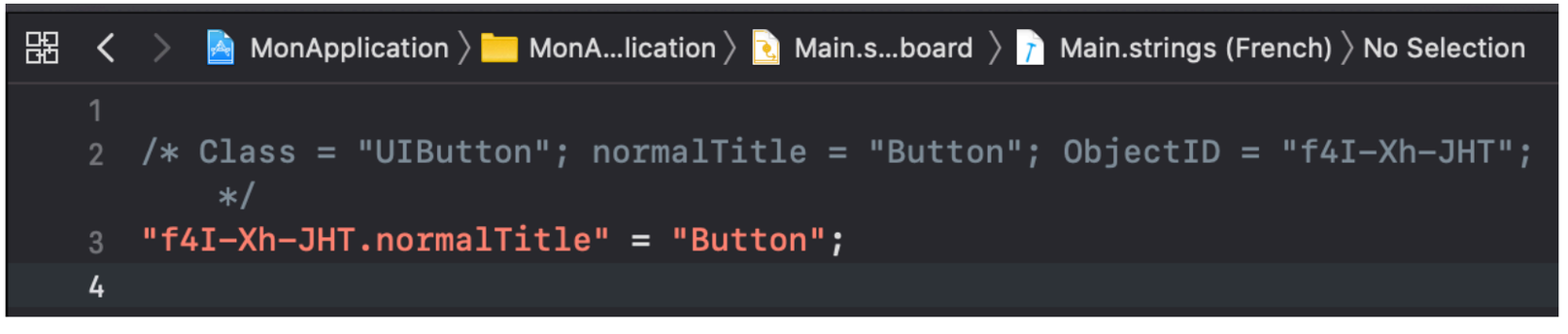


Localisation

Chaque ressource a ensuite un fichier de traduction associé :



Les fichiers de traduction contiennent les IDs des éléments associé à leur texte :



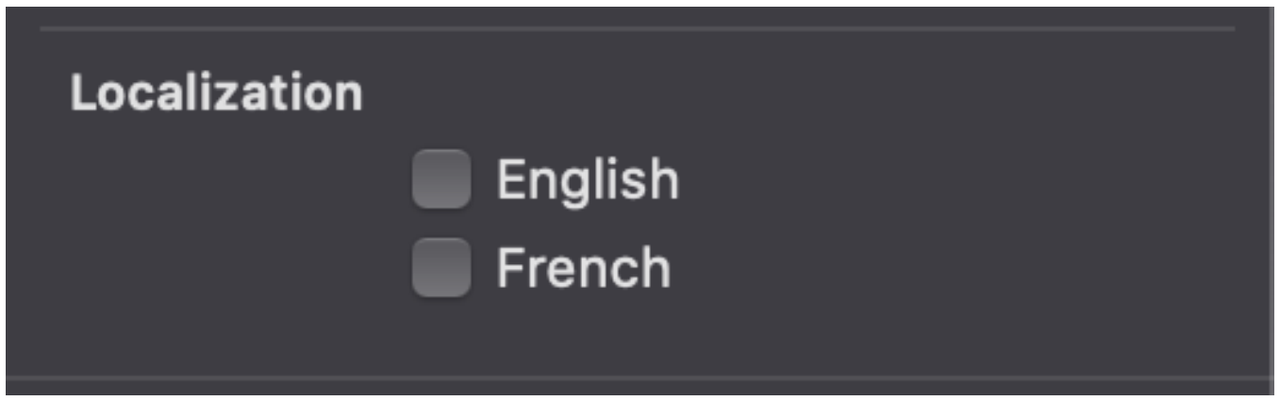
Localisation

- Problème : les fichiers générés ne sont pas automatiquement mis à jour

- Vous pouvez soit :
  - Copier vos précédentes traductions, régénérer les fichiers de strings et importer les anciennes traductions

  - Utiliser un outil externe (e.g. https://github.com/Flinesoft/BartyCrouch)

Localisation



Il est possible de localiser vos ressources dans le catalogue des assets
en spécifiant depuis l’inspecteur d’attributs les langues cibles

Localisation

- Il est également possible de créer des chaînes dynamiques par langage

- Simple système de clé / valeur par langage

- Si vous utilisez une clé qui n’est pas traduite, iOS affiche cette clé à l’utilisateur

- À utiliser avec parcimonie

    - Préférer les méthodes vues précédemment

Localisation

```
"buttonLabel" = "Appuyez sur ce bouton";
"labelDynamique" = "Vous avez %d nouveaux messages";
```

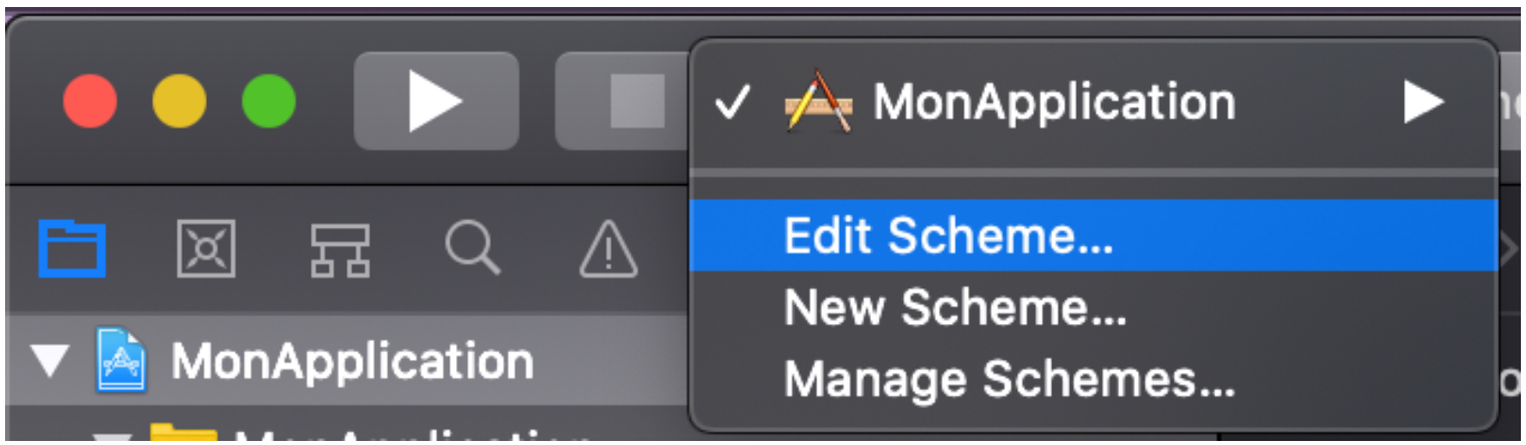
Il suffit de créer un nouveau fichier « Strings file » dans votre projet, appelé `Localizable.strings`

Une fois le fichier ouvert, dans le « File inspector » (1ère icône du panneau droit), il faut ensuite cliquer sur « Localize » et cocher les langues souhaitées

Localisation

```
// Le paramètre `comment` est à simple titre informatif
// pour le développeur

NSLocalizedString("buttonLabel", comment: "")
// => "Appuyez sur ce bouton"

String(format: NSLocalizedString("labelDynamique", comment: ""), 4)
// => "Vous avez 4 nouveaux messages"
```

Localisation



Pour changer la langue de votre application sur le simulateur ou dans le cadre des tests, il est possible de définir la langue au niveau de Xcode

Localisation