Class

UIImagePickerController

A view controller that manages the system interfaces for taking pictures, recording movies, and choosing items from the user's media library.

Declaration

```
class UIImagePickerController : UINavigationController
```

Overview

An image picker controller manages user interactions and delivers the results of those interactions to a delegate object. The role and appearance of an image picker controller depend on the *source type* you assign to it before you present it.

- A `sourceType` of `UIImagePickerController.SourceType.camera` provides a user interface for taking a new picture or movie (on devices that support media capture).
- A `sourceType` of `UIImagePickerController.SourceType.photoLibrary` or `UIImagePickerController.SourceType.savedPhotosAlbum` provides a user interface for choosing among saved pictures and movies.

To use an image picker controller containing its default controls, perform these steps:

1. Verify that the device is capable of picking content from the desired source. Do this by calling the `isSourceTypeAvailable(_:)` class method, providing a constant from the `UIImagePickerController.SourceType` enumeration.
2. Check which media types are available for the source type you're using, by calling the `availableMediaTypes(for:)` class method. This lets you distinguish between a camera that can be used for video recording and one that can be used only for still images.
3. Tell the image picker controller to adjust the UI according to the media types you want to make available—still images, movies, or both—by setting the `mediaTypes` property.
4. Present the user interface. On iPhone or iPod touch, do this modally (full-screen) by calling the `present(_:animated:completion:)` method of the currently active view controller, passing your configured image picker controller as the new view controller.

   On iPad, the correct way to present an image picker depends on its source type, as summarized in this table:

Camera	Photo Library	Saved Photos Album
Use full screen	Must use a popover	Must use a popover

   The table indicates that on iPad, if you specify a source type of `UIImagePickerController.SourceType.photoLibrary` or `UIImagePickerController.SourceType.savedPhotosAlbum`, you must present the image picker using a popover controller (to learn how to do this, see `UIPopoverPresentationController`). If you attempt to present an image picker modally (full-screen) for choosing among saved pictures and movies, the system raises an exception.

On iPad, if you specify a source type of `UIImagePickerController.SourceType.camera`, you can present the image picker modally (full-screen) or by using a popover. However, Apple recommends that you present the camera interface only full-screen.

5. When the user taps a button to pick a newly-captured or saved image or movie, or cancels the operation, dismiss the image picker using your delegate object. For newly-captured media, your delegate can then save it to the Camera Roll on the device. For previously-saved media, your delegate can then use the image data according to the purpose of your app.

For details on these steps, refer to Taking Pictures and Movies.

You can customize an image picker controller to manage user interactions yourself. To do this, provide an overlay view containing the controls you want to display, and use the methods described in Capturing Still Images or Movies. You can display your custom overlay view in addition to, or instead of, the default controls. Custom overlay views for the `UIImagePickerController` class are available in iOS 3.1 and later by way of the `cameraOverlayView`property. For a code example, see the PhotoPicker: Using UIImagePickerController to Select Pictures and Take Photos sample code project.

Important

The `UIImagePickerController` class supports portrait mode only. This class is intended to be used as-is and does not support subclassing. The view hierarchy for this class is private and must not be modified, with one exception. You can assign a custom view to the `cameraOverlayView` property and use that view to present additional information or manage the interactions between the camera interface and your code.

Providing a Delegate Object

To use an image picker controller, you must provide a delegate that conforms to the `UIImagePickerControllerDelegate` protocol. Starting in iOS 4.1, you can use the delegate to save still-image metadata to the Camera Roll along with the image. See `UIImagePickerControllerDelegate`.

Adjusting Flash Mode

In iOS 4.0 and later, you can provide custom controls to let the user adjust flash mode (on devices that have a flash LED), pick which camera to use (on devices that have a front and rear camera), and switch between still image and movie capture. You can also manage these settings programmatically. You can also manipulate the flash directly to provide effects such as a strobe light. Present a picker interface set to use video capture mode. Then, turn the flash LED on or off by setting the `cameraFlashMode` property to `UIImagePickerController.CameraFlashMode.on` or `UIImagePickerController.CameraFlashMode.off`.

Working with Movies

Movie capture has a default duration limit of 10 minutes but can be adjusted using the `videoMaximumDuration` property. When a user taps the Share button to send a movie to MMS, MobileMe, YouTube, or another destination, an appropriate duration limit and an appropriate video quality are enforced.

The default camera interface supports editing of previously-saved movies. Editing involves trimming from the start or end of the movie, then saving the trimmed movie. To display an interface dedicated to movie editing, rather than one that also supports recording new movies, use the `UIVideoEditorController` class instead of this one. See `UIVideoEditorController`.

Working with Live Photos

Live Photos is a Camera app feature on supported devices, enabling a picture to be not just a single moment in time but to include motion and sound from the moments just before and after its capture. A `PHLivePhoto` object represents a Live Photo, and the `PHLivePhotoView` class provides a system-standard, interactive user interface for displaying a Live Photo and playing back its content.

Although Live Photos include sound and motion, they remain photos. When you use an image picker controller to capture or choose still images (by including only the `kUTTypeImage` type in the `mediaTypes` array), assets that were captured as Live Photos continue to appear in the picker. However, when the user chooses an asset, your `delegate` object receives only a `UIImage` object containing a still-image representation of the Live Photo.

To obtain the full motion and sound content when the user chooses a Live Photo with the image picker, you must include *both* the `kUTTypeImage` and `kUTTypeLivePhoto` types in the `mediaTypes` array. For more information, see `livePhoto` in `UIImagePickerControllerDelegate`.

Fully-Customized Media Capture and Browsing

To perform fully-customized image or movie capture, instead use the AVFoundation framework as described in Still and Video Media Capture. Camera access using the AVFoundation framework is available starting in iOS 4.0.

To create a fully-customized image picker for browsing the photo library, use classes from the Photos framework. For example, you could create a custom image picker that displays larger thumbnail images generated and cached by iOS, that makes use of image metadata including timestamp and location information, or that integrates with other features such as MapKit and iCloud Photo Sharing. For more information, see *Photos*. Media browsing using the Photos framework is available starting in iOS 8.0.

Topics

Responding to Interactions with the Picker

`var delegate: (UIImagePickerControllerDelegate & UINavigationControllerDelegate)?`

The image picker's delegate object.

`protocol UIImagePickerControllerDelegate`

A set of methods that your delegate object must implement to interact with the image picker interface.

Setting the Picker Source

`class func availableMediaTypes(for: UIImagePickerController.SourceType) -> [String]?`

Returns an array of the available media types for the specified source type.

`class func isSourceTypeAvailable(UIImagePickerController.SourceType) -> Bool`

Returns a Boolean value indicating whether the device supports picking media using the specified source type.

`var sourceType: UIImagePickerController.SourceType`

The type of picker interface to be displayed by the controller.

`enum UIImagePickerController.SourceType`

The source to use when picking an image or when determining available media types.

Configuring the Picker

`var mediaTypes: [String]`

An array indicating the media types to be accessed by the media picker controller.

`var allowsEditing: Bool`

A Boolean value indicating whether the user is allowed to edit a selected still image or movie.

Configuring the Video Capture Options

`var videoQuality: UIImagePickerController.QualityType`

The video recording and transcoding quality.

`enum UIImagePickerController.QualityType`

Video quality settings for movies recorded with the built-in camera, or transcoded by displaying in the image picker.

`var videoMaximumDuration: TimeInterval`

The maximum duration, in seconds, for a video recording.

Customizing the Camera Controls

`var showsCameraControls: Bool`

Indicates whether the image picker displays the default camera controls.

`var cameraOverlayView: UIView?`

The view to display on top of the default image picker interface.

`var cameraViewTransform: CGAffineTransform`

The transform to apply to the camera's preview image.

Capturing Still Images or Movies

`func takePicture()`

Captures a still image using the camera.

`func startVideoCapture() -> Bool`

Starts video capture using the camera specified by the `UIImagePickerController.CameraDevice` property.

`func stopVideoCapture()`

Stops video capture.

Configuring the Camera to Use

`class func isCameraDeviceAvailable(UIImagePickerController.CameraDevice) -> Bool`

Returns a Boolean value that indicates whether a given camera is available.

`var cameraDevice: UIImagePickerController.CameraDevice`

The camera used by the image picker controller.

`enum UIImagePickerController.CameraDevice`

The camera to use for image or movie capture.

Configuring the Camera Capture Mode

`class func availableCaptureModes(for: UIImagePickerController.CameraDevice) -> [NSNumber]?`

Returns an array of `NSNumber` objects indicating the capture modes supported by a given camera device.

`var cameraCaptureMode: UIImagePickerController.CameraCaptureMode`

The capture mode used by the camera.

`enum UIImagePickerController.CameraCaptureMode`

The category of media for the camera to capture.

Configuring the Flash Behavior

`class func isFlashAvailable(for: UIImagePickerController.CameraDevice) -> Bool`

Indicates whether a given camera has flash illumination capability.

`var cameraFlashMode: UIImagePickerController.CameraFlashMode`

The flash mode used by the active camera.

`enum UIImagePickerController.CameraFlashMode`

The flash mode to use with the active camera.

Configuring the Export Presets

`var imageExportPreset: UIImagePickerController.ImageURLExportPreset`

The preset to use when preparing images for export to your app.

`enum UIImagePickerController.ImageURLExportPreset`

Constants indicating how to export images to the client app.

`var videoExportPreset: String`

The preset to use when preparing video for export to your app.