Class

SFSafariViewController

An object that provides a visible standard interface for browsing the web.

iOS 9.0+ | iPadOS 9.0+ | Mac Catalyst 13.1+ | visionOS 1.0+

```
@MainActor
class SFSafariViewController
```

Overview

An `SFSafariViewController` object presents a self-contained web interface inside your app. Present this view controller to let people view websites from anywhere on the internet without leaving your app. The web interface supports Safari features such as Reader, AutoFill, Fraudulent Website Warning, and content blocking. Interactions with the web interface aren't visible to your app, and you can't access AutoFill data, browsing history, or website data. You don't need to secure data between your app and Safari. To share data between your app and Safari, use `ASWebAuthenticationSession` instead.

Present an `SFSafariViewController` when you don't need to customize or interact with the web content. After you present the content, interactions with the web content occur solely within the view controller. When the person dismisses the view controller, control returns to your app's interface. If you need to customize the controls of the web interface, or you want to interact with content in that interface, display the content using a `WKWebView` object instead.

Important

In accordance with App Store Review Guidelines, when you present this view controller, it must visibly present information to users. You may not hide or obscure the view controller behind other views or layers. Additionally, you may not use `SFSafariViewController` to track users without their knowledge and consent.

UI features include the following:

- A read-only address field with a security indicator and a Reader button
- An Action button that invokes an activity view controller offering custom services from your app and activities, such as messaging, from the system and other extensions
- A Done button, back and forward navigation buttons, and a button to open the page directly in Safari
- Peek and Pop for links and detected data using 3D Touch

When a person peeks and pops a link in `SFSafariViewController`, the view controller loads and displays the link destination. When a person peeks and pops a link in a `WKWebView` class, the web view opens the link in Safari by default.

Presenting the view controller in your interface

Present a `SFSafariViewController` modally using the `present(_:animated:completion:)` method and the default presentation style. The view controller animates into position and provides an optimized full-screen experience for web browsing. It also supports intuitive and expected gestures for returning to your app's content. For example, the view controller's leading-edge swipe gesture automatically switches between navigating to the previous page in the browsing history and dismissing the view controller.

To create a more lightweight browsing experience, change the presentation style to `UIModalPresentationStyle.formSheet` or `UIModalPresentationStyle.pageSheet`. You might use this approach to display terms of service or support documentation. Don't use these presentation styles to display content from websites.

> **Important**
>
> Don't embed an `SFSafariViewController` as a child view controller in your app, or include it in your view controller hierarchy. Always present it modally using the `present(_:animated:completion:)` method.

`SFSafariViewController` supplies a custom transitioning delegate to manage its presentation and transition animations. The custom delegate is recommended, but you can remove it if you prefer the default modal transition behavior. To remove the delegate, assign a new value to the view controller's `transitioningDelegate` property. The object you assign to this property must adopt the `UIViewControllerTransitioningDelegate` protocol but doesn't need to implement any methods of that protocol.

When building apps to support multiple platforms, be aware of behavior differences when presenting an `SFSafariViewController` from your UI. In Mac apps built with Mac Catalyst,

and in compatible iPad and iPhone apps running in visionOS, showing an SFSafariViewController opens the web content in the default web browser instead. In visionOS, you can use SFSafariViewController to support context menu previews for links, but don't use it to open a URL. The view controller displays a preview of the link content but redirects the URL to the default web browser if someone opens it. Instead of showing the view controller in visionOS, modify your experience to open the URL using an OpenURLAction in SwiftUI or the open(_:options:completionHandler:) method of UIApplication instead.

Measuring ad taps with Private Click Measurement (PCM)

When you navigate to a webpage in SFSafariViewController after a person taps an ad in your app, measure conversions on the website in a privacy-preserving way with Private Click Measurement (PCM).

Add a UIEventAttributionView subview to the ad view or control in order to measure taps. When a person taps the ad, follow these steps to configure an SFSafariViewController instance to use Private Click Measurement for the tap:

1. Create and configure a UIEventAttribution object with attribution data.
2. Assign the object to the eventAttribution property of an SFSafariViewController.Configuration object.
3. Initialize an SFSafariViewController object with the configuration object, and present it. SFSafariViewController validates that a tap on a UIEventAttributionView initiated the navigation to the webpage. If not, it discards the attribution data.

If the external website reports a conversion, SFSafariViewController forwards the attribution report to the specified remote server.

For more information on the proposed PCM web standard, see Introducing Private Click Measurement and Private Click Measurement Draft Community Group Report.

Topics

Creating a View Controller

`init(url: URL, configuration: SFSafariViewController.Configuration)`

Initializes and configures a Safari view controller that loads the specified URL.

`class Configuration`

A configuration object that defines how a Safari view controller should be initialized.

`convenience init(url: URL)`

Initializes a Safari view controller that loads the specified URL.

~~`init(url: URL, entersReaderIfAvailable: Bool)`~~

Initializes a Safari view controller that will load the specified URL, entering Reader mode if Reader mode is requested and available.

Deprecated

Responding to View Controller Interaction

`var delegate: (any SFSafariViewControllerDelegate)?`

An object that provides behavior for the Safari view controller's Done and Action buttons.

`protocol SFSafariViewControllerDelegate`

A protocol used to implement custom event handling for a Safari view controller.

Configuring the View Controller

`var configuration: SFSafariViewController.Configuration`

A copy of the Safari view controller's initialized configuration.

`var dismissButtonStyle: SFSafariViewController.DismissButtonStyle`

The style of dismiss button to use in the navigation bar to close the Safari view controller.

`enum DismissButtonStyle`

~~`var preferredBarTintColor: UIColor?`~~

The color to tint the background of the navigation bar and the toolbar.

Deprecated

~~`var preferredControlTintColor: UIColor?`~~

The color to tint the control buttons on the navigation bar and the toolbar.

Deprecated

Type Methods

`class func prewarmConnections(to: [URL]) -> SFSafariViewController.PrewarmingToken`

Classes

```
class ActivityButton

class DataStore

class PrewarmingToken
```

Relationships

Inherits From

UIViewController

Conforms To

CVarArg
CustomDebugStringConvertible
CustomStringConvertible
Equatable
Hashable
NSCoding
NSExtensionRequestHandling
NSObjectProtocol
NSTouchBarProvider
UIActivityItemsConfigurationProviding
UIAppearanceContainer
UIContentContainer
UIFocusEnvironment
UIPasteConfigurationSupporting
UIResponderStandardEditActions
UIStateRestoring
UITraitChangeObservable
UITraitEnvironment
UIUserActivityRestoring

See Also

Safari content in your app

`typealias` `CompletionHandler`

The completion handler for an authentication session when the user cancels or finishes the login.

Importing data exported from Safari

Transfer bookmarks, saved passwords, and other information between browsers.