Instance Method

performSelectorOnMainThread:withObject:waitUntilDone:modes:

Invokes a method of the receiver on the main thread using the specified modes.

Language
Swift | Objective-C

SDKs
iOS 2.0+
macOS 10.2+
tvOS 9.0+
watchOS 2.0+

On This Page
Declaration
Parameters
Discussion
See Also

Declaration

```
- (void)performSelectorOnMainThread:(SEL)aSelector
                         withObject:(id)arg
                      waitUntilDone:(BOOL)wait
                              modes:(NSArray<NSString *> *)array;
```

Parameters

aSelector
A Selector that identifies the method to invoke. The method should not have a significant return value and should take a single argument of type id, or no arguments.

arg
The argument to pass to the method when it is invoked. Pass `nil` if the method does not take an argument.

wait
A Boolean that specifies whether the current thread blocks until after the specified selector is performed on the receiver on the main thread. Specify `YES` to block this thread; otherwise, specify `NO` to have this method return immediately.

If the current thread is also the main thread, and you pass `YES`, the message is performed immediately, otherwise the perform is queued to run the next time through the run loop.

array
An array of strings that identifies the modes in which it is permissible to perform the specified selector. This array must contain at least one string. If you specify `nil` or an empty array for this parameter, this method returns without performing the specified selector. For information about run loop modes, see Run Loops in Threading Programming Guide.

Discussion

You can use this method to deliver messages to the main thread of your application. The main thread encompasses the application's main run loop, and is where the `NSApplication` object receives events. The message in this case is a method of the current object that you want to execute on the thread.

This method queues the message on the run loop of the main thread using the run loop modes specified in the `array` parameter. As part of its normal run loop processing, the main thread dequeues the message (assuming it is running in one of the specified modes) and invokes the desired method. Multiple calls to this method from the same thread cause the corresponding selectors to be queued and performed in the same same order in which the calls were made, assuming the associated run loop modes for each selector are the same. If you specify different modes for each selector, any selectors whose associated mode does not match the current run loop mode are skipped until the run loop subsequently executes in that mode.

You cannot cancel messages queued using this method. If you want the option of canceling a message on the current thread, you must use either the `performSelector:withObject:afterDelay:` or `performSelector:withObject:afterDelay:inModes:` method.

Special Considerations

This method registers with the runloop of its current context, and depends on that runloop being run on a regular basis to perform correctly. One common context where you might call this method and end up registering with a runloop that is not automatically run on a regular basis is when being invoked by a dispatch queue. If you need this type of functionality when running on a dispatch queue, you should use `dispatch_after` and related methods to get the behavior you want.

See Also

Related Symbols

- `performSelector:withObject:afterDelay:inModes:`
  Invokes a method of the receiver on the current thread using the specified modes after a delay.
- `performSelector:withObject:afterDelay:`
  Invokes a method of the receiver on the current thread using the default mode after a delay.
- `performSelector:onThread:withObject:waitUntilDone:modes:`
  Invokes a method of the receiver on the specified thread using the specified modes.
- `performSelectorOnMainThread:withObject:waitUntilDone:`
  Invokes a method of the receiver on the main thread using the default mode.