NSExceptionHandler Class Reference

Cocoa > Objective-C Language

2006-10-03

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

NSExceptionHandler Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/ExceptionHandling.framework
Availability	Mac OS X v10.0
Companion guide	Exception Programming Topics for Cocoa
Declared in	NSExceptionHandler.h

Overview

The `NSExceptionHandler` class provides facilities for monitoring and debugging exceptional conditions in Objective-C programs. It works by installing a special uncaught exception handler via the `NSSetUncaughtExceptionHandler` function. Consequently, to use the services of `NSExceptionHandler`, you must not install your own custom uncaught exception handler.

To use these services, you set a bit mask in the singleton `NSExceptionHandler` instance and, optionally, a delegate. The constants comprising the bit mask indicate the type of exception to be monitored and the behavior of the `NSExceptionHandler` object (or, simply, the exception handler). The delegate is asked to approve the logging and handling of each monitored `NSException` object.

The constants for configuring exception handler behavior can be categorized in several ways:

- Uncaught exceptions versus caught exceptions—or, more accurately, exceptions that would be caught (for example, by the top-level handler)
- Exception type or cause: system exceptions (such as invalid memory accesses), Objective-C runtime errors (such as messages sent to freed objects), and other exceptions
- Exception handler behavior: logging the exception (including a stack trace) to the console, handling the exception, and suspending program execution so the debugger can be attached

The way the exception handler handles an exception depends on the type of exception; the exception handler converts system exceptions and runtime errors into `NSException` objects with a stack trace embedded in their `userInfo` dictionary; for all other uncaught exceptions, it terminates the thread on which they occur . The constants used to configure an `NSExceptionHandler` object are described in `Logging and Handling Constants` (page 10) and `System Hang Constants` (page 12).

The `defaults` command-line system also allows you to set values corresponding to the `enum` constants used to configure the exception handler; see “Controlling Application Response to Exceptions" for details.

Tasks

Getting the Default Exception Handler

`+ defaultExceptionHandler` (page 6)
Returns the singleton `NSExceptionHandler` instance.

Getting and Setting Exception Masks

`- exceptionHandlingMask` (page 7)
Returns a bit mask representing the types of exceptions monitored by the receiver and its handling and logging behavior.

`- exceptionHangingMask` (page 7)
Returns a bit mask representing the types of exceptions that will halt execution for debugging.

`- setExceptionHandlingMask:` (page 8)
Sets the bit mask of constants specifying the types of exceptions monitored by the receiver and its handling and logging behavior.

`- setExceptionHangingMask:` (page 8)
Sets the bit mask of constants specifying the types of exceptions that will halt execution for debugging.

Getting and Setting the Delegate

`- delegate` (page 7)
Returns the delegate of the `NSExceptionHandler` object.

`- setDelegate:` (page 8)
Sets the delegate of the `NSExceptionHandler` object.

Logging and handling exceptions

`- exceptionHandler:shouldHandleException:mask:` (page 9) *delegate method*
Implemented by the delegate to evaluate whether the delegating `NSExceptionHandler` instance should handle a given exception.

`- exceptionHandler:shouldLogException:mask:` (page 9) *delegate method*
Implemented by the delegate to evaluate whether the delegating `NSExceptionHandler` instance should log a given exception.

Class Methods

defaultExceptionHandler

Returns the singleton `NSExceptionHandler` instance.

```
+ (NSExceptionHandler *)defaultExceptionHandler
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSExceptionHandler.h`

Instance Methods

delegate

Returns the delegate of the `NSExceptionHandler` object.

```
- (id)delegate
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSExceptionHandler.h`

exceptionHandlingMask

Returns a bit mask representing the types of exceptions monitored by the receiver and its handling and logging behavior.

```
- (unsigned int)exceptionHandlingMask
```

Return Value
A bit mask composed of one or more constants specifying the types of exceptions monitored and whether they are handled or logged (or both). See `Logging and Handling Constants` (page 10) for information about the constants.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSExceptionHandler.h`

exceptionHangingMask

Returns a bit mask representing the types of exceptions that will halt execution for debugging.

```
- (unsigned int)exceptionHangingMask
```

Return Value
A bit mask composed of one or more constants specifying the types of exceptions that will halt execution for debugging. See `System Hang Constants` (page 12) for information about the constants.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSExceptionHandler.h`

setDelegate:

Sets the delegate of the `NSExceptionHandler` object.

```
- (void)setDelegate:(id)anObject
```

Parameters

anObject

The object to receive the delegation messages described in "Logging and handling exceptions" (page 6)

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSExceptionHandler.h`

setExceptionHandlingMask:

Sets the bit mask of constants specifying the types of exceptions monitored by the receiver and its handling and logging behavior.

```
- (void)setExceptionHandlingMask:(unsigned int)aMask
```

Parameters

aMask

A bit mask composed of one or more constants specifying the types of exceptions monitored and whether they are handled or logged (or both). You specify multiple constants by performing a bitwise-OR operation. See `Logging and Handling Constants` (page 10) for information about the constants.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSExceptionHandler.h`

setExceptionHangingMask:

Sets the bit mask of constants specifying the types of exceptions that will halt execution for debugging.

```
- (void)setExceptionHangingMask:(unsigned int)aMask
```

Parameters

aMask

A bit mask composed of one or more constants specifying the types of exceptions that will halt execution for debugging. You specify multiple constants by performing a bitwise-OR operation. See `System Hang Constants` (page 12) for information about the constants.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSExceptionHandler.h`

Delegate Methods

exceptionHandler:shouldHandleException:mask:

Implemented by the delegate to evaluate whether the delegating `NSExceptionHandler` instance should handle a given exception.

```
- (BOOL)exceptionHandler:(NSExceptionHandler *)sender
    shouldHandleException:(NSException *)exception mask:(unsigned int)aMask
```

Parameters

sender

The `NSExceptionHandler` object sending the message.

exception

An `NSException` object describing the exception to be evaluated.

aMask

The bit mask indicating the types of exceptions handled by the `NSExceptionHandler` object. See `Logging and Handling Constants` (page 10) and `System Hang Constants` (page 12) for descriptions of the possible `enum` constants.

Return Value
`YES` to have the `NSExceptionHandler` object handle the exception, `NO` otherwise.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSExceptionHandler.h`

exceptionHandler:shouldLogException:mask:

Implemented by the delegate to evaluate whether the delegating `NSExceptionHandler` instance should log a given exception.

```
- (BOOL)exceptionHandler:(NSExceptionHandler *)sender shouldLogException:(NSException
    *)exception mask:(unsigned int)aMask
```

Parameters

sender

The `NSExceptionHandler` object sending the message.

exception

An `NSException` object describing the exception to be evaluated.

aMask

The bit mask indicating the types of exceptions logged by the `NSExceptionHandler` object. See `Logging and Handling Constants` (page 10) and `System Hang Constants` (page 12) for descriptions of the possible `enum` constants.

Return Value

`YES` to have the `NSExceptionHandler` object log the exception, `NO` otherwise.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSExceptionHandler.h`

Constants

Logging and Handling Constants

Use one or more of the following constants in the parameter of `setExceptionHandlingMask:` (page 8) to specify the types of exceptions that the exception handler should monitor and whether it should handle or log them.

```
enum {
    NSLogUncaughtExceptionMask              = 1 << 0,
    NSHandleUncaughtExceptionMask           = 1 << 1,
    NSLogUncaughtSystemExceptionMask        = 1 << 2,
    NSHandleUncaughtSystemExceptionMask     = 1 << 3,
    NSLogUncaughtRuntimeErrorMask           = 1 << 4,
    NSHandleUncaughtRuntimeErrorMask        = 1 << 5,
    NSLogTopLevelExceptionMask              = 1 << 6,
    NSHandleTopLevelExceptionMask           = 1 << 7,
    NSLogOtherExceptionMask                 = 1 << 8,
    NSHandleOtherExceptionMask              = 1 << 9
};
```

Constants

`NSLogUncaughtExceptionMask`

The exception handler logs uncaught exceptions.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHandleUncaughtExceptionMask`

The exception handler handles uncaught exceptions by terminating the thread in which they occur.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSLogUncaughtSystemExceptionMask`
The exception handler logs uncaught system exceptions.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHandleUncaughtSystemExceptionMask`
The exception handler handles uncaught system exceptions by converting them to `NSException` objects containing a stack trace.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSLogUncaughtRuntimeErrorMask`
The exception handler logs uncaught runtime errors.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHandleUncaughtRuntimeErrorMask`
The exception handler handles uncaught runtime errors by converting them to `NSException` objects containing a stack trace.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSLogTopLevelExceptionMask`
The exception handler logs exceptions that would be caught by the top-level handler.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHandleTopLevelExceptionMask`
The exception handler handles exceptions caught by the top-level handler by converting them to `NSException` objects containing a stack trace.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSLogOtherExceptionMask`
The exception handler logs exceptions caught by handlers lower than the top-level handler.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHandleOtherExceptionMask`
The exception handler handles exceptions caught by handlers lower than the top-level handler by converting them to `NSException` objects containing a stack trace.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

Discussion
When exception-handling domains are nested, `NSLogTopLevelExceptionMask` (page 11) and `NSHandleTopLevelExceptionMask` (page 11) deal with exceptions that would make it to the top two levels of exception handlers. In the main thread of a Cocoa application, the top-level handler is the global `NSApplication` instance.

Declared In
`ExceptionHandling/ExceptionHandler.h`

System Hang Constants

Use one or more of the following constants in the parameter of `setExceptionHangingMask:` (page 8) to specify the types of exceptions that cause the exception to halt execution so a debugger can be attached.

```
enum {
    NSHangOnUncaughtExceptionMask        = 1 << 0,
    NSHangOnUncaughtSystemExceptionMask        = 1 << 1,
    NSHangOnUncaughtRuntimeErrorMask        = 1 << 2,
    NSHangOnTopLevelExceptionMask        = 1 << 3,
    NSHangOnOtherExceptionMask            = 1 << 4
};
```

Constants

`NSHangOnUncaughtExceptionMask`

The exception handler suspends execution when it detects an uncaught exception (other than a system exception or runtime error).

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHangOnUncaughtSystemExceptionMask`

The exception handler suspends execution when it detects an uncaught system exception.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHangOnUncaughtRuntimeErrorMask`

The exception handler suspends execution when it detects an uncaught runtime error.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHangOnTopLevelExceptionMask`

The exception handler suspends execution when it detects an exception that would be handled by the top-level handler.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHangOnOtherExceptionMask`

The exception handler suspends execution when it detects an exception that would be handled by an object other than the top-level handler.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

Discussion

When exception-handling domains are nested, `NSHangOnTopLevelExceptionMask` (page 12) deals with exceptions that would make it to the top two levels of exception handlers. In the main thread of a Cocoa application, the top-level handler is the global `NSApplication` instance.

Declared In

`ExceptionHandling/ExceptionHandler.h`

Mask Definitions

The following `#define` constants are conveniences for specifying complete sets of exception-handling `enum` constants.

```
NSHangOnEveryExceptionMask
NSLogAndHandleEveryExceptionMask
```

Constants

`NSHangOnEveryExceptionMask`

Combines via bitwise-OR all the constants listed in `System Hang Constants` (page 12).

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSLogAndHandleEveryExceptionMask`

Combines via bitwise-OR all the constants listed in `Logging and Handling Constants` (page 10).

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

Declared In

`ExceptionHandling/ExceptionHandler.h`

Exception Global String Constants

Two of the following global string constants identify exceptions generated by the framework for Objective-C runtime errors and system exceptions such as invalid memory accesses. The other constant is used as a key to access the stack trace in the `userInfo` dictionary of an `NSException` object, when requested.

```
EXCEPTIONHANDLING_EXPORT NSString *NSUncaughtSystemExceptionException;
EXCEPTIONHANDLING_EXPORT NSString *NSUncaughtRuntimeErrorException;
EXCEPTIONHANDLING_EXPORT NSString *NSStackTraceKey;
```

Constants

`NSUncaughtSystemExceptionException`

Identifies an uncaught system exception.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSUncaughtRuntimeErrorException`

Identifies an Objective-C runtime error.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSStackTraceKey`

The key for fetching the stack trace (an `NSString` object) in the `userInfo` dictionary of the `NSException` object passed into one of the delegate methods described in "Logging and handling exceptions" (page 6).

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

Declared In

`ExceptionHandling/ExceptionHandler.h`

Document Revision History

This table describes the changes to *NSExceptionHandler Class Reference*.

Date	Notes
2006-10-03	New document that describes the class used for monitoring and debugging exceptional conditions in Objective-C code.

Document Revision History

Index