

URLSessionTask

A task, like downloading a specific resource, performed in a URL session.

Declaration

```
class URLSessionTask : NSObject
```

Overview

The `URLSessionTask` class is the base class for tasks in a URL session. Tasks are always part of a session; you create a task by calling one of the task creation methods on a `URLSession` instance. The method you call determines the type of task.

- Use `URLSession`’s `dataTask(with:)` and related methods to create `URLSessionDataTask` instances. Data tasks request a resource, returning the server’s response as one or more `NSData` objects in memory. They are supported in default, ephemeral, and shared sessions, but are not supported in background sessions.
- Use `URLSession`’s `uploadTask(with:from:)` and related methods to create `URLSessionUploadTask` instances. Upload tasks are like data tasks, except that they make it easier to provide a request body so you can upload data before retrieving the server’s response. Additionally, upload tasks are supported in background sessions.
- Use `URLSession`’s `downloadTask(with:)` and related methods to create `URLSessionDownloadTask` instances. Download tasks download a resource directly to a file on disk. Download tasks are supported in any type of session.
- Use `URLSession`’s `streamTask(withHostName:port:)` or `streamTask(with:)` to create `URLSessionStreamTask` instances. Stream tasks establish a TCP/IP connection from a host name and port or a net service object.

After you create a task, you start it by calling its `resume()` method. The session then maintains a strong reference to the task until the request finishes or fails; you don’t need to maintain a reference to the task unless it’s useful for your app’s internal bookkeeping.

Note

All task properties support key-value observing.

SDKs

iOS 7.0+

macOS 10.9+

tvOS 9.0+

watchOS 2.0+

Framework

Foundation

On This Page

- Declaration ☺
- Overview ☺
- Topics ☺
- Relationships ☺
- See Also ☺

Topics

Controlling the Task State

`func cancel()`

Cancels the task.

`func resume()`

Resumes the task, if it is suspended.

`func suspend()`

Temporarily suspends a task.

`var state: URLSessionTask.State`

The current state of the task—active, suspended, in the process of being canceled, or completed.

`enum URLSessionTask.State`

Constants for determining the current state of a task.

`var priority: Float`

The relative priority at which you'd like a host to handle the task, specified as a floating point value between 0.0 (lowest priority) and 1.0 (highest priority).

 `URLSessionTaskPriority`

Constants for providing task priority hints to a host, used with the `priority` property.

Obtaining Task Progress

`var progress: Progress`

A representation of the overall task progress.

`var countOfBytesExpectedToReceive: Int64`

The number of bytes that the task expects to receive in the response body.

`var countOfBytesReceived: Int64`

The number of bytes that the task has received from the server in the response body.

`var countOfBytesExpectedToSend: Int64`

The number of bytes that the task expects to send in the request body.

`var countOfBytesSent: Int64`

The number of bytes that the task has sent to the server in the request body.

`let NSURLSessionTransferSizeUnknown: Int64`

The total size of the transfer cannot be determined.

Obtaining General Task Information

`var currentRequest: URLRequest?`

The URL request object currently being handled by the task.

`var originalRequest: URLRequest?`

The original request object passed when the task was created.

`var response: URLResponse?`

The server's response to the currently active request.

`var taskDescription: String?`

An app-provided description of the current task.

`var taskIdentifier: Int`

An identifier uniquely identifying the task within a given session.

`var error: Error?`

An error object that indicates why the task failed.

Scheduling Tasks

`var countOfBytesClientExpectsToReceive: Int64`

A best-guess upper bound on the number of bytes the client expects to receive.

`var countOfBytesClientExpectsToSend: Int64`

A best-guess upper bound on the number of bytes the client expects to send.

`let NSURLSessionTransferSizeUnknown: Int64`

The total size of the transfer cannot be determined.

`var earliestBeginDate: Date?`

The earliest date at which the network load should begin.

Relationships

Inherits From

NSObject

Conforms To

CVarArg
Equatable
Hashable
NSCopying
ProgressReporting

See Also

First Steps

 [Fetching Website Data into Memory](#)

Receive data directly into memory by creating a data task from a URL session.

`class URLSession`

An object that coordinates a group of related network data transfer tasks.