NSUserDefaults

Inherits From: NSObject

Conforms To: NSObject (NSObject)

Declared In: Foundation/NSUserDefaults.h

Class Description

The NSUserDefaults class allows an application to query and manipulate a user's defaults settings.

Defaults are grouped in domains. For example, there's a domain for application-specific defaults and another for global defaults. Each domain has a name and stores defaults as key-value pairs in an NSDictionary object. A default is identified by a string key, and its value can be any property-list object (NSData, NSString, NSArray, or NSDictionary). The standard domains are:

Domain	Identifier
Argument	NSArgumentDomain
Application	Identified by the application's name
Global	NSGlobalDomain
Preferred languages	Identified by the language names
Registration	NSRegistrationDomain

The identifiers starting with "NS" above are global constants.

The argument domain is composed of defaults parsed from the application's arguments. The application domain contains the defaults set by the application. It's identified by the name of the application, as returned by this message:

```
NSString *applicationName = [[NSProcessInfo processInfo] processName];
```

The global domain contains defaults that are meant to be seen by all applications. The registration domain is a set of temporary defaults whose values can be set by the application to ensure that searches for default values will always be successful. Applications can create additional domains as needed.

A search for the value of a given default proceeds through the domains listed in an NSUserDefault object's *search list*. *Only* domains in the search list are searched. The standard search list contains the domains from the table above, in the order listed. A search ends when the default is found. Thus, if multiple domains contain the same default, only the domain nearest the beginning of the search list provides the default's value. Using the **searchList** method, you can reorder the default search list or set up one that is a subset of all the user's domains.

Typically, you use this class by invoking the **standardUserDefaults** class method to get an NSUserDefaults object. This method returns a global NSUserDefaults object with a search list already initialized. Then use the **setObject:forKey:** and **objectForKey:** methods to set and access user defaults.

The rest of the methods allow more complex defaults management. You can create your own domains, modify any domain, set up a custom search list, and even control the synchronization of the in-memory and on-disk defaults representations. The **synchronize** method saves any modifications to the persistent domains and updates all persistent domains that were not modified to what is on disk. **synchronize** is automatically invoked at periodic intervals.

You can create either persistent or volatile domains. Persistent domains are permanent and last past the life of the NSUserDefaults object. Any changes to the persistent domains are committed to disk. Volatile domains last only last as long as the NSUserDefaults object exists. The standard domains have the following states:

Domain	State
NSArgumentDomain	volatile
Application	persistent
NSGlobalDomain	persistent
Preferred languages	volatile
NSRegistrationDomain	volatile

Warning:

- User defaults are not thread safe.
- Automatic saving of changes to disk (through **synchronize**) depends on a run-loop being present.
- You should synchronize any domain you have altered before exiting a process.

Method Types

Getting the shared instance	+ standardUserDefaults
Initializing an NSUserDefaults	– init – initWithUser:
Getting a default	– arrayForKey: – boolForKey: – dataForKey: – dictionaryForKey: – floatForKey: – integerForKey: – objectForKey: – stringArrayForKey: – stringForKey:

Setting and removing defaults	– removeObjectForKey: – setBool:forKey: – setFloat:forKey: – setInteger:forKey: – setObject:forKey:
Getting the search list	– searchList – dictionaryRepresentation
Maintaining persistent domains	– persistentDomainForName: – persistentDomainNames – removePersistentDomainForName: – setPersistentDomain:forName: – synchronize
Maintaining volatile domains	– removeVolatileDomainForName: – setVolatileDomain:forName: – volatileDomainForName: – volatileDomainNames
Registering defaults	– registerDefaults:

Class Methods

standardUserDefaults

+ (NSUserDefaults *)**standardUserDefaults**

Returns the shared defaults object. If it doesn't exist yet, it's created with a search list containing the names of the following domains, in this order:

- NSArgumentDomain, consisting of defaults parsed from the application's arguments
- A domain identified by the process (application) name
- NSGlobalDomain, consisting of defaults meant to be seen by all applications
- Separate domains for each of the user's preferred languages
- NSRegistrationDomain, a set of temporary defaults whose values can be set by the application to ensure that searches will always be successful

The defaults are initialized for the current user. Subsequent modifications to the standard search list remain in effect even when this method is invoked again—the search list is guaranteed to be standard only the first time this method is invoked. The shared instance is provided as a convenience; you may create custom instances with **initWithUser:** and **init**.

See also: – **init**, – **initWithUser:**

Instance Methods

arrayForKey:

– (NSArray *)**arrayForKey:**(NSString *)*defaultName*

Invokes **objectForKey:** with key *defaultName*. Returns the value associated with *defaultName* if it's an NSArray object and **nil** otherwise.

See also: –**boolForKey:**, – **dataForKey:**, – **dictionaryForKey:**, – **floatForKey:**, – **integerForKey:**, – **objectForKey:**, **– stringArrayForKey:**, **– stringForKey:**

boolForKey:

– (BOOL)**boolForKey:**(NSString *)*defaultName*

Invokes **stringForKey**: with key *defaultName*. Returns YES if the value associated with *defaultName* is an NSString containing uppercase or lowercase "YES" or responds to the **intValue** message by returning a non-zero value. Otherwise, returns NO.

See also: –**arrayForKey:**, – **dataForKey:**, – **dictionaryForKey:**, – **floatForKey:**, – **integerForKey:**, – **objectForKey:**, **– stringArrayForKey:**, **– stringForKey:**

dataForKey:

– (NSData *)**dataForKey:**(NSString *)*defaultName*

Invokes **objectForKey:** with key *defaultName*. Returns the corresponding value if it's an NSData object and nil otherwise.

See also: – **arrayForKey:**, –**boolForKey:**, – **dictionaryForKey:**, – **floatForKey:**, – **integerForKey:**, – **objectForKey:**, **– stringArrayForKey:**, **– stringForKey:**

dictionaryForKey:

– (NSDictionary *)**dictionaryForKey:**(NSString *)*defaultName*

Invokes **objectForKey:** with key *defaultName*. Returns the corresponding value if it's an NSDictionary object and **nil** otherwise.

See also: – **arrayForKey:**, **–boolForKey:, – dataForKey:**, – **floatForKey:**, – **integerForKey:**, – **objectForKey:**, **– stringArrayForKey:**, **– stringForKey:**

dictionaryRepresentation

– (NSDictionary *)**dictionaryRepresentation**

Returns a dictionary that contains a union of all key-value pairs in the domains in the search list. As with **objectForKey:**, key-value pairs in domains that are earlier in the search list take precedence. The combined result doesn't preserve information about which domain each entry came from.

See also: – **searchList**

floatForKey:

– (float)**floatForKey:**(NSString *)*defaultName*

Invokes **stringForKey:** with key *defaultName*. Returns 0 if no string is returned. Otherwise, the resulting string is sent a **floatValue** message, which provides this method's return value.

See also: – **arrayForKey:**, **–boolForKey:, – dataForKey:**, – **dictionaryForKey:**, – **integerForKey:**, – **objectForKey:**, **– stringArrayForKey:**, **– stringForKey:**

init

– (id)**init**

Initializes defaults for the current user account and returns an NSUserDefaults instance with the argument and registration domains set up. This method doesn't put anything in the search list. Invoke it only if you've allocated your own NSUserDefaults instance instead of using the shared one.

See also: + **standardUserDefaults**

initWithUser:

– (id)**initWithUser:**(NSString *)*username*

Initializes defaults for the user account identified by *username* and returns an NSUserDefaults instance with the argument and registration domains set up. This method doesn't put anything in the search list. Invoke it only if you've allocated your own NSUserDefaults instance instead of using the shared one.

See also: + **standardUserDefaults**

integerForKey:

– (int)**integerForKey:**(NSString *)*defaultName*

Invokes **stringForKey:** with key *defaultName*. Returns 0 if no string is returned. Otherwise, the resulting string is sent a **intValue** message, which provides this method's return value.

See also: – **arrayForKey:**, **–boolForKey:, – dataForKey:**, – **dictionaryForKey:**, – **floatForKey:**, – **objectForKe**y:, **– stringArrayForKey:**, **– stringForKey:**

objectForKey:

– (id)**objectForKey:**(NSString *)*defaultName*

Returns the value of the first occurrence of the default identified by *defaultName*, searching the domains included in the search list in the order they're listed. Returns **nil** if the default isn't found.

See also: – **arrayForKey:**, **–boolForKey:, – dataForKey:**, – **dictionaryForKey:**, – **floatForKey:**, – **integerForKey:**:, **– stringArrayForKey:**, **– stringForKey:**

persistentDomainForName:

– (NSDictionary *)**persistentDomainForName:**(NSString *)*domainName*

Returns a dictionary representing the persistent domain identified by *domainName*. The keys in the dictionary are names of defaults, and the value corresponding to each key is a property-list object (NSString, NSDictionary, NSArray, or NSData).

See also: – **removePersistentDomainForName:**, – **setPersistentDomainForName:**

persistentDomainNames

– (NSArray *)**persistentDomainNames**

Returns an array containing the names of the current persistent domains. You can get each domain by using the domain names in the array as arguments to **persistentDomainForName:**.

See also: – **removePersistentDomainForName:**, – **setPersistentDomainForName:**

registerDefaults:

– (void)**registerDefaults:**(NSDictionary *)*dictionary*

Adds the contents of *dictionary* to the registration domain. If there is no registration domain, it's created using dictionary, and NSRegistrationDomain is added to the end of the search list

removeObjectForKey:

– (void)**removeObjectForKey:**(NSString *)*defaultName*

Removes the value for the default identified by *defaultName* in the standard application domain. Removing a default has no effect on the value returned by the **objectForKey:** method if the same key exists in a domain that precedes the standard application domain in the search list.

See also: – setObject:forKey:

removePersistentDomainForName:

– (void)**removePersistentDomainForName:**(NSString *)*domainName*

Removes the persistent domain identified by *domainName* from the user's defaults. The first time that a persistent domain is changed after **synchronize**, an NSUserDefaultsChanged notification is posted.

See also: – **setPersistentDomainForName:**

removeVolatileDomainForName:

– (void)**removeVolatileDomainForName:**(NSString *)*domainName*

Removes the volatile domain identified by *domainName* from the user's defaults.

See also: – **setVolatileDomainForName:**

searchList

– (NSMutableArray *)**searchList**

Returns a mutable array of domain names, identifying the domains that **objectForKey:** will search. You can customize the search list by modifying the array that's returned. Domain names in the list with no corresponding domain are ignored.

See also: – **dictionaryRepresentation**

setBool:forKey:

– (void)**setBool:**(BOOL)*value* **forKey:**(NSString *)*defaultName*

Sets the value of the default identified by *defaultName* to a string representation of YES or NO, depending on value. Invokes **setObject:forKey:** as part of its implementation.

See also: – **boolForKey:**

setFloat:forKey:

– (void)**setFloat:**(float)*value* **forKey:**(NSString *)*defaultName*

Sets the value of the default identified by *defaultName* to a string representation of value. Invokes **setObject:forKey:** as part of its implementation.

See also: – **floatForKey:**

setInteger:forKey:

– (void)**setInteger:**(int)*value* **forKey:**(NSString *)*defaultName*

Sets the value of the default identified by *defaultName* to a string representation of value. Invokes **setObject:forKey:** as part of its implementation.

See also: – **integerForKey:**

setObject:forKey:

– (void)**setObject:**(id)*value* **forKey:**(NSString *)*defaultName*

Sets the value of the default identified by *defaultName* in the standard application domain. Setting a default has no effect on the value returned by the **objectForKey:** method if the same key exists in a domain that precedes the application domain in the search list.

See also: – **objectForKey:**, – **removeObjectForKey:**

setPersistentDomain:forName:

– (void)**setPersistentDomain:**(NSDictionary *)*domain* **forName:**(NSString *)*domainName*

Sets the dictionary for the persistent domain named *domainName*; raises an NSInvalidArgumentException if a persistent domain with *domainName* already exists. The first time that a persistent domain is changed after synchronize, an NSUserDefaultsChanged notification is posted.

See also: – **persistentDomainForName:**, – **persistentDomainNames**

setVolatileDomain:forName:

– (void)**setVolatileDomain:**(NSDictionary *)*domain* **forName:**(NSString *)*domainName*

Sets the dictionary to *domain* for the volatile domain named *domainName*. This method raises an NSInvalidArgumentException if a volatile domain with *domainName* already exists.

See also: – **volatileDomainForName:**, – **volatileDomainNames**

stringArrayForKey:

– (NSArray *)**stringArrayForKey:**(NSString *)*defaultName*

Invokes **objectForKey:** with key *defaultName*. Returns the corresponding value if it's an NSArray object containing NSStrings, and **nil** otherwise.

See also: – **arrayForKey:**, **–boolForKey:, – dataForKey:**, – **dictionaryForKey:**, – **floatForKey:**, – **integerForKey:**, – **objectForKey:**, **– stringForKey:**

stringForKey:

– (NSString *)**stringForKey:**(NSString *)*defaultName*

Invokes **objectForKey:** with key *defaultName*. Returns the corresponding value if it's an NSString object and **nil** otherwise.

See also: – **arrayForKey:**, **–boolForKey:, – dataForKey:**, – **dictionaryForKey:**, – **floatForKey:**, – **integerForKey:**, – **objectForKey:**, **– stringArrayForKey:**

synchronize

– (BOOL)**synchronize**

Saves any modifications to the persistent domains and updates all persistent domains that were not modified to what is on disk. Returns NO if it could not save data to disk. Since the **synchronize** method is automatically invoked at periodic intervals, use this method only if you cannot wait for the automatic synchronization (for example if your application is about to exit), or if you want to update user defaults to what is on disk even though you have not made any changes.

See also: – **persistentDomainForName:**, – **persistentDomainName**s, – **removePersistentDomainForName:**, – **setPersistentDomainForName:**

volatileDomainForName:

– (NSDictionary *)**volatileDomainForName:**(NSString *)*domainName*

Returns a dictionary representing the volatile domain identified by *domainName*. The keys in the dictionary are names of defaults, and the value corresponding to each key is a property-list object (NSString, NSData, NSDictionary, NSArray).

See also: – **removeVolatileDomainForName:**, – **setVolatileDomainForName:**

volatileDomainNames

– (NSArray *)**volatileDomainNames**

Returns an array containing the names of the current volatile domains. You can get each domain by using the domain names in the array as arguments to **volatileDomainForName:**.

See also: – **removeVolatileDomainForName:**, – **setVolatileDomainForName:**

Notifications

NSUserDefaultsDidChangeNotification

Notification Object	The NSUserDefaults instance
UserInfo	None

This notification is posted the first time after a **synchronize** when a change is made to defaults in a persistent domain.