NSXMLDocument Class Reference

Cocoa > Data Management

2007-02-27

Contents

NSXMLDocument Class Reference

Inherits from	NSXMLNode : NSObject
Conforms to	NSCopying (NSXMLNode) NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.4 and later.
Companion guide	Tree-Based XML Programming Guide for Cocoa
Declared in	NSXMLDocument.h NSXMLNodeOptions.h
Related sample code	AlbumToSlideshow CocoaSOAP Core Data HTML Store TimelineToTC

Overview

An instance of `NSXMLDocument` represents an XML document as internalized into a logical tree structure. An `NSXMLDocument` object can have multiple child nodes but only one element, the root element. Any other node must be a `NSXMLNode` object representing a comment or a processing instruction. If you attempt to add any other kind of child node to an `NSXMLDocument` object, such as an attribute, namespace, another document object, or an element other than the root, `NSXMLDocument` raises an exception. If you add a valid child node and that object already has a parent, `NSXMLDocument` raises an exception. An `NSXMLDocument` object may also have document-global attributes, such as XML version, character encoding, referenced DTD, and MIME type.

The initializers of the `NSXMLDocument` class read an external source of XML, whether it be a local file or remote website, parse it, and process it into the tree representation. You can also construct an `NSXMLDocument` programmatically. There are accessor methods for getting and setting document attributes, methods for transforming documents using XSLT, a method for dynamically validating a document, and methods for printing out the content of an `NSXMLDocument` as XML, XHTML, HTML, or plain text.

Subclassing Notes

Methods to Override

To subclass `NSXMLDocument` you need to override the primary initializer, `initWithData:options:error:` (page 12), and the methods listed below. In most cases, you need only invoke the superclass implementation, adding any subclass-specific code before or after the invocation, as necessary.

- `rootElement` (page 19)
- `setChildren:` (page 20)
- `removeChildAtIndex:` (page 18)
- `insertChild:atIndex:` (page 14)
- `characterEncoding` (page 10)
- `setCharacterEncoding:` (page 19)
- `documentContentKind` (page 11)
- `setDocumentContentKind:` (page 20)
- `DTD` (page 11)
- `setDTD:` (page 21)
- `MIMEType` (page 15)
- `setMIMEType:` (page 21)
- `isStandalone` (page 15)
- `setStandalone:` (page 22)
- `version` (page 24)
- `setURI:` (page 22)
- `setVersion:` (page 23)

By default `NSXMLDocument` implements the `NSObject` `isEqual:` method to perform a deep comparison: two `NSXMLDocument` objects are not considered equal unless they have the same name, same child nodes, same attributes, and so on. The comparison does not consider the parent node (and hence the node's location). If you want a different standard of comparison, override `isEqual:`.

Special Considerations

Because of the architecture and data model of NSXML, when it parses and processes a source of XML it cannot know about your subclass unless you override the class method `replacementClassForClass:` (page 9) to return your custom class in place of an `NSXML` class. If your custom class has no direct `NSXML` counterpart—for example, it is a subclass of `NSXMLNode` that represents CDATA sections—then you can walk the tree after it has been created and insert the new node where appropriate.

Tasks

Initializing NSXMLDocument Objects

`– initWithContentsOfURL:options:error:` (page 11)
: Initializes and returns an NSXMLDocument object created from the XML or HTML contents of a URL-referenced source

`– initWithData:options:error:` (page 12)
: Initializes and returns an `NSXMLDocument` object created from an `NSData` object.

`– initWithRootElement:` (page 13)
: Returns an `NSXMLDocument` object initialized with a single child, the root element.

`– initWithXMLString:options:error:` (page 13)
: Initializes and returns an `NSXMLDocument` object created from a string containing XML markup text.

`+ replacementClassForClass:` (page 9)
: Overridden by subclasses to substitute a custom class for an NSXML class that the parser uses to create node instances.

Managing Document Attributes

`– characterEncoding` (page 10)
: Returns the character encoding used for the XML.

`– setCharacterEncoding:` (page 19)
: Sets the character encoding of the receiver to *`encoding`*,

`– documentContentKind` (page 11)
: Returns the kind of document content for output.

`– setDocumentContentKind:` (page 20)
: Sets the kind of output content for the receiver.

`– DTD` (page 11)
: Returns an `NSXMLDTD` object representing the internal DTD associated with the receiver.

`– setDTD:` (page 21)
: Sets the internal DTD to be associated with the receiver.

`– isStandalone` (page 15)
: Returns whether the receiver represents a standalone XML document—that is, one without an external DTD.

`– setStandalone:` (page 22)
: Sets a Boolean value that specifies whether the receiver represents a standalone XML document.

`– MIMEType` (page 15)
: Returns the MIME type for the receiver.

`– setMIMEType:` (page 21)
: Sets the MIME type of the receiver.

`– URI` (page 23)
: Returns the URI identifying the source of this document.

- `setURI:` (page 22)
  Sets the URI identifying the source of this document.
- `version` (page 24)
  Returns the version of the receiver's XML.
- `setVersion:` (page 23)
  Sets the version of the receiver's XML.

Managing the Root Element

- `rootElement` (page 19)
  Returns the root element of the receiver.
- `setRootElement:` (page 21)
  Set the root element of the receiver.

Adding and Removing Child Nodes

- `addChild:` (page 10)
  Adds a child node after the last of the receiver's existing children.
- `insertChild:atIndex:` (page 14)
  Inserts a node object at specified position in the receiver's array of children.
- `insertChildren:atIndex:` (page 14)
  Inserts an array of children at a specified position in the receiver's array of children.
- `removeChildAtIndex:` (page 18)
  Removes the child node of the receiver located at a specified position in its array of children.
- `replaceChildAtIndex:withNode:` (page 18)
  Replaces the child node of the receiver located at a specified position in its array of children with another node.
- `setChildren:` (page 20)
  Sets the child nodes of the receiver.

Transforming a Document Using XSLT

- `objectByApplyingXSLT:arguments:error:` (page 16)
  Applies the XSLT pattern rules and templates (specified as a data object) to the receiver and returns a document object containing transformed XML or HTML markup.
- `objectByApplyingXSLTString:arguments:error:` (page 17)
  Applies the XSLT pattern rules and templates (specified as a string) to the receiver and returns a document object containing transformed XML or HTML markup.
- `objectByApplyingXSLTAtURL:arguments:error:` (page 16)
  Applies the XSLT pattern rules and templates located at a specified URL to the receiver and returns a document object containing transformed XML markup or an `NSData` object containing plain text, RTF text, and so on.

Writing a Document as XML Data

- `XMLData` (page 24)
  Returns the XML string representation of the receiver—that is, the entire document—encapsulated in a data object.
- `XMLDataWithOptions:` (page 25)
  Returns the XML string representation of the receiver—that is, the entire document—encapsulated in a data object.

Validating a Document

- `validateAndReturnError:` (page 23)
  Validates the document against the governing schema and returns whether the document conforms to the schema.

Class Methods

replacementClassForClass:

Overridden by subclasses to substitute a custom class for an NSXML class that the parser uses to create node instances.

```
+ (Class)replacementClassForClass:(Class)class
```

Parameters

`class`
A `Class` object identifying an NSXML class that is to be replaced by your custom class.

Return Value
The substituted class.

Discussion
For example, if you have a custom subclass of `NSXMLElement` that you want to be used in place of `NSXMLElement`, you would make the following override:

```
+ (Class)replacementClassForClass:(Class)currentClass {
    if ( currentClass == [NSXMLElement class] ) {
        return [MyCustomElementClass class];
    }
}
```

This method is invoked before a document is parsed. The substituted class must be a subclass of `NSXMLNode`, `NSXMLDocument`, `NSXMLElement`, `NSXMLDTD`, or `NSXMLDTDNode`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setRootElement:` (page 21)

Declared In
`NSXMLDocument.h`

Instance Methods

addChild:

Adds a child node after the last of the receiver's existing children.

```
- (void)addChild:(NSXMLNode *)child
```

Parameters

child
The `NSXMLNode` object to be added.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- insertChild:atIndex:` (page 14)
`- removeChildAtIndex:` (page 18)
`- setChildren:` (page 20)

Declared In
`NSXMLDocument.h`

characterEncoding

Returns the character encoding used for the XML.

```
- (NSString *)characterEncoding
```

Return Value
The character encoding used for the XML, or `nil` if no encoding is specified.

Discussion
Typically the encoding is specified in the XML declaration of a document that is processed, but it can be set at any time. If the specified encoding does not match the actual encoding, parsing of the document may fail.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setCharacterEncoding:` (page 19)

Declared In
`NSXMLDocument.h`

documentContentKind

Returns the kind of document content for output.

```
- (NSXMLDocumentContentKind)documentContentKind
```

Discussion
Most of the differences among content kind have to do with the handling of content-less tags such as `<br>`. The valid `NSXMLDocumentContentKind` constants are `NSXMLDocumentXMLKind`, `NSXMLDocumentXHTMLKind`, `NSXMLDocumentHTMLKind`, and `NSXMLDocumentTextKind`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setDocumentContentKind:` (page 20)

Declared In
`NSXMLDocument.h`

DTD

Returns an `NSXMLDTD` object representing the internal DTD associated with the receiver.

```
- (NSXMLDTD *)DTD
```

Return Value
An `NSXMLDTD` object representing the internal DTD associated with the receiver or `nil` if no DTD has been associated.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setDTD:` (page 21)

Declared In
`NSXMLDocument.h`

initWithContentsOfURL:options:error:

Initializes and returns an NSXMLDocument object created from the XML or HTML contents of a URL-referenced source

```
- (id)initWithContentsOfURL:(NSURL *)url options:(NSUInteger)mask error:(NSError
    **)error
```

Parameters

`url`

An `NSURL` object specifying a URL source.

`mask`

A bit mask for input options. You can specify multiple options by bit-OR'ing them. See "Constants" (page 25) for a list of valid input options.

error

An error object that, on return, identifies any parsing errors and warnings or connection problems.

Return Value

An initialized `NSXMLDocument` object, or `nil` if initialization fails because of parsing errors or other reasons.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `initWithData:options:error:` (page 12)
- `initWithRootElement:` (page 13)
- `initWithXMLString:options:error:` (page 13)

Declared In

`NSXMLDocument.h`

initWithData:options:error:

Initializes and returns an `NSXMLDocument` object created from an `NSData` object.

```
- (id)initWithData:(NSData *)data options:(NSUInteger)mask error:(NSError **)error
```

Parameters

data

A data object with XML content.

mask

A bit mask for input options. You can specify multiple options by bit-OR'ing them. See "Constants" (page 25) for a list of valid input options.

error

An error object that, on return, identifies any parsing errors and warnings or connection problems.

Return Value

An initialized `NSXMLDocument` object, or `nil` if initialization fails because of parsing errors or other reasons.

Discussion

This method is the designated initializer for the `NSXMLDocument` class.

If you specify `NSXMLDocumentTidyXML` as one of the options, NSXMLDocument performs several clean-up operations on the document XML (such as removing leading tabs). It does however, respect the `xmlns:space="preserve"` attribute when it attempts to tidy the XML.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `initWithContentsOfURL:options:error:` (page 11)
- `initWithRootElement:` (page 13)
- `initWithXMLString:options:error:` (page 13)

Declared In

`NSXMLDocument.h`

initWithRootElement:

Returns an `NSXMLDocument` object initialized with a single child, the root element.

```
- (id)initWithRootElement:(NSXMLElement *)root
```

Parameters

root

An `NSXMLElement` object representing an XML element.

Return Value

An initialized `NSXMLDocument` object, or `nil` if initialization fails for any reason.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- initWithContentsOfURL:options:error:` (page 11)
`- initWithData:options:error:` (page 12)
`- initWithXMLString:options:error:` (page 13)

Related Sample Code

AlbumToSlideshow

Declared In

`NSXMLDocument.h`

initWithXMLString:options:error:

Initializes and returns an `NSXMLDocument` object created from a string containing XML markup text.

```
- (id)initWithXMLString:(NSString *)string options:(NSUInteger)mask error:(NSError
    **)error
```

Parameters

string

A string object containing XML markup text.

mask

A bit mask for input options. You can specify multiple options by bit-OR'ing them. See "Constants" (page 25) for a list of valid input options.

error

An error object that, on return, identifies any parsing errors and warnings or connection problems.

Return Value

An initialized `NSXMLDocument` object, or `nil` if initialization fails because of parsing errors or other reasons.

Discussion

The encoding of the document is set to UTF-8.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- initWithContentsOfURL:options:error:` (page 11)

`- initWithData:options:error:` (page 12)
`- initWithRootElement:` (page 13)

Related Sample Code
CocoaSOAP

Declared In
`NSXMLDocument.h`

insertChild:atIndex:

Inserts a node object at specified position in the receiver's array of children.

```
- (void)insertChild:(NSXMLNode *)child atIndex:(NSUInteger)index
```

Parameters

child
The `NSXMLNode` object to be inserted. The added node must be an `NSXMLNode` object representing a comment, processing instruction, or the root element.

index
An integer specifying the index of the children array to insert *child*. The indexes of children after the new child are incremented. If *index* is less than zero or greater than the number of children, an out-of-bounds exception is raised.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- addChild:` (page 10)
`- insertChildren:atIndex:` (page 14)
`- removeChildAtIndex:` (page 18)
`- replaceChildAtIndex:withNode:` (page 18)

Declared In
`NSXMLDocument.h`

insertChildren:atIndex:

Inserts an array of children at a specified position in the receiver's array of children.

```
- (void)insertChildren:(NSArray *)children atIndex:(NSUInteger)index
```

Parameters

children
An array of `NSXMLNode` objects representing comments, processing instructions, or the root element.

index
An integer identifying the location in the receiver's children array for insertion. The indexes of children after the new child are increased by `[children count]`. If *index* is less than zero or greater than the number of children, an out-of-bounds exception is raised.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `addChild:` (page 10)
- `removeChildAtIndex:` (page 18)
- `replaceChildAtIndex:withNode:` (page 18)
- `setChildren:` (page 20)

Declared In
`NSXMLDocument.h`

isStandalone

Returns whether the receiver represents a standalone XML document—that is, one without an external DTD.

```
- (BOOL)isStandalone
```

Return Value
`YES` if the receiver represents a standalone XML document, `NO` if the "standalone" declaration was not present in the original document and hasn't been set since.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setStandalone:` (page 22)

Declared In
`NSXMLDocument.h`

MIMEType

Returns the MIME type for the receiver.

```
- (NSString *)MIMEType
```

Return Value
The MIME type for the receiver (for example, "text/xml").

Discussion
MIME types are assigned by IANA (see http://www.iana.org/assignments/media-types/index.html).

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setMIMEType:` (page 21)

Declared In
`NSXMLDocument.h`

objectByApplyingXSLT:arguments:error:

Applies the XSLT pattern rules and templates (specified as a data object) to the receiver and returns a document object containing transformed XML or HTML markup.

```
- (id)objectByApplyingXSLT:(NSData *)xslt arguments:(NSDictionary *)arguments
    error:(NSError **)error
```

Parameters

xslt

A data object containing the XSLT pattern rules and templates.

arguments

A dictionary containing `NSString` key-value pairs that are passed as runtime parameters to the XSLT processor. Pass in `nil` if you have no parameters to pass.

Note: Several XML websites discuss XSLT parameters, including O'Reilly Media's http://www.xml.com.

error

If an error occurs, indirectly returns an `NSError` object encapsulating error or warning messages generated by XSLT processing.

Return Value

Depending on intended output, the method returns an `NSXMLDocument` object *or* an `NSData` data containing transformed XML or HTML markup. If the message is supposed to create plain text or RTF, then an `NSData` object is returned, otherwise an XML document object. The method returns `nil` if XSLT processing did not succeed.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `objectByApplyingXSLTAtURL:arguments:error:` (page 16)
- `objectByApplyingXSLTString:arguments:error:` (page 17)

Declared In

`NSXMLDocument.h`

objectByApplyingXSLTAtURL:arguments:error:

Applies the XSLT pattern rules and templates located at a specified URL to the receiver and returns a document object containing transformed XML markup or an `NSData` object containing plain text, RTF text, and so on.

```
- (id)objectByApplyingXSLTAtURL:(NSURL *)xsltURL arguments:(NSDictionary *)arguments
    error:(NSError **)error
```

Parameters

xsltURL

An `NSURL` object specifying a valid URL.

arguments

A dictionary containing `NSString` key-value pairs that are passed as runtime parameters to the XSLT processor. Pass in `nil` if you have no parameters to pass.

Note: Several XML websites discuss XSLT parameters, including O'Reilly Media's http://www.xml.com.

error

If an error occurs, indirectly returns an `NSError` object encapsulating error or warning messages generated by XSLT processing or from an attempt to connect to a website identified by the URL.

Return Value

Depending on intended output, the returns an `NSXMLDocument` object *or* an `NSData` data containing transformed XML or HTML markup. If the message is supposed to create plain text or RTF, then an `NSData` object is returned, otherwise an XML document object. The method returns `nil` if XSLT processing did not succeed.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `objectByApplyingXSLT:arguments:error:` (page 16)
- `objectByApplyingXSLTString:arguments:error:` (page 17)

Declared In

`NSXMLDocument.h`

objectByApplyingXSLTString:arguments:error:

Applies the XSLT pattern rules and templates (specified as a string) to the receiver and returns a document object containing transformed XML or HTML markup.

```
- (id)objectByApplyingXSLTString:(NSString *)xsltarguments:(NSDictionary
    *)argumentserror:(NSError **)error
```

Parameters

xslt

A string object containing the XSLT pattern rules and templates.

arguments

A dictionary containing `NSString` key-value pairs that are passed as runtime parameters to the XSLT processor. Pass in `nil` if you have no parameters to pass.

Note: Several XML websites discuss XSLT parameters, including O'Reilly Media's http://www.xml.com.

error

If an error occurs, indirectly returns an `NSError` object encapsulating error or warning messages generated by XSLT processing.

Return Value
Depending on intended output, the method returns an `NSXMLDocument` object *or* an `NSData` data containing transformed XML or HTML markup. If the message is supposed to create plain text or RTF, then an `NSData` object is returned, otherwise an XML document object. The method returns `nil` if XSLT processing did not succeed.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- objectByApplyingXSLT:arguments:error:` (page 16)
`- objectByApplyingXSLTAtURL:arguments:error:` (page 16)

Declared In
`NSXMLDocument.h`

removeChildAtIndex:

Removes the child node of the receiver located at a specified position in its array of children.

```
- (void)removeChildAtIndex:(NSUInteger)index
```

Parameters

`index`
An integer identifying the position of an child in the receiver's array. If *`index`* is less than zero or greater than the number of children minus one, an out-of-bounds exception is raised.

Discussion
Subsequent children have their indexes decreased by one. The removed `NSXMLNode` object is autoreleased.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- insertChild:atIndex:` (page 14)
`- replaceChildAtIndex:withNode:` (page 18)

Declared In
`NSXMLDocument.h`

replaceChildAtIndex:withNode:

Replaces the child node of the receiver located at a specified position in its array of children with another node.

```
- (void)replaceChildAtIndex:(NSUInteger)index withNode:(NSXMLNode *)node
```

Parameters

`index`
An integer identifying a position in the receiver's array of children. If *`index`* is less than zero or greater than the number of children minus one, an out-of-bounds exception is raised.

node
An `NSXMLNode` object to replace the one at *index*; it must represent a comment, a processing instruction, or the root element.

Discussion
The removed `NSXMLNode` object is autoreleased.

Availability
Available in Mac OS X v10.4 and later.

See Also
`– insertChild:atIndex:` (page 14)
`– removeChildAtIndex:` (page 18)

Declared In
`NSXMLDocument.h`

rootElement

Returns the root element of the receiver.

```
- (NSXMLElement *)rootElement
```

Return Value
The root element of the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
`– setRootElement:` (page 21)

Declared In
`NSXMLDocument.h`

setCharacterEncoding:

Sets the character encoding of the receiver to *encoding*,

```
- (void)setCharacterEncoding:(NSString *)encoding
```

Parameters

encoding
A string that specifies an encoding; it must match the name of an IANA character set. See http://www.iana.org/assignments/character-sets for a list of valid encoding specifiers.

Discussion
Typically the encoding is specified in the XML declaration of a document that is processed, but it can be set at any time. If the specified encoding does not match the actual encoding, parsing of the document might fail.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `characterEncoding` (page 10)

Related Sample Code
AlbumToSlideshow

Declared In
`NSXMLDocument.h`

setChildren:

Sets the child nodes of the receiver.

```
- (void)setChildren:(NSArray *)children
```

Parameters

`children`
An array of `NSXMLNode` objects. Each of these objects must represent comments, processing instructions, or the root element; otherwise, an exception is raised. Pass in `nil` to remove all children.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `addChild:` (page 10)
- `insertChildren:atIndex:` (page 14)

Declared In
`NSXMLDocument.h`

setDocumentContentKind:

Sets the kind of output content for the receiver.

```
- (void)setDocumentContentKind:(NSXMLDocumentContentKind)kind
```

Parameters

`kind`
An `enum` constant identifying a kind of document content. The valid `NSXMLDocumentContentKind` constants are `NSXMLDocumentXMLKind`, `NSXMLDocumentXHTMLKind`, `NSXMLDocumentHTMLKind`, and `NSXMLDocumentTextKind`.

Discussion
Most of the differences among document-content kind have to do with the handling of content-less tags such as `<br>`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `documentContentKind` (page 11)

Declared In
`NSXMLDocument.h`

setDTD:

Sets the internal DTD to be associated with the receiver.

```
- (void)setDTD:(NSXMLDTD *)documentTypeDeclaration
```

Parameters

documentTypeDeclaration
An `NSXMLDTD` object representing the internal DTD to be associated with the receiver.

Discussion
When the receiver is written out, this document type declaration appears in the output, just after the XML declaration.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- DTD` (page 11)

Declared In
`NSXMLDocument.h`

setMIMEType:

Sets the MIME type of the receiver.

```
- (void)setMIMEType:(NSString *)MIMEType
```

Parameters

MIMEType
A string object identifying a MIME type, for example, "text/xml". MIME types are assigned by IANA (see http://www.iana.org/assignments/media-types/index.html).

Availability
Available in Mac OS X v10.4 and later.

See Also
`- MIMEType` (page 15)

Declared In
`NSXMLDocument.h`

setRootElement:

Set the root element of the receiver.

```
- (void)setRootElement:(NSXMLNode *)root
```

Parameters

root

An `NSXMLNode` object that is to be the root element.

Discussion

As a side effect, this method removes all other children, including `NSXMLNode` objects representing comments and processing-instructions.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- rootElement` (page 19)

Declared In

`NSXMLDocument.h`

setStandalone:

Sets a Boolean value that specifies whether the receiver represents a standalone XML document.

```
- (void)setStandalone:(BOOL)standalone
```

Parameters

standalone

`YES` if the receiver represents a standalone XML document, `NO` otherwise.

Discussion

A standalone document does not have an external DTD associated with it.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- isStandalone` (page 15)

Declared In

`NSXMLDocument.h`

setURI:

Sets the URI identifying the source of this document.

```
- (void)setURI:(NSString *)URI
```

Parameters

URI

A string object representing a URI source, or `nil` to remove the current URI.

Discussion

This attribute is automatically set when the receiver is initialized using `initWithContentsOfURL:options:error:` (page 11).

See Also
`- URI` (page 23)

setVersion:

Sets the version of the receiver's XML.

```
- (void)setVersion:(NSString *)version
```

Parameters

version
A string object identifying the version of the XML.

Discussion
Currently, the version should be either "1.0 "or "1.1".

Availability
Available in Mac OS X v10.4 and later.

See Also
`- version` (page 24)

Related Sample Code
AlbumToSlideshow

Declared In
`NSXMLDocument.h`

URI

Returns the URI identifying the source of this document.

```
- (NSString *)URI
```

Return Value
The URI identifying the source of this document or `nil` if this attribute has not been set.

See Also
`- setURI:` (page 22)

validateAndReturnError:

Validates the document against the governing schema and returns whether the document conforms to the schema.

```
- (BOOL)validateAndReturnError:(NSError **)error
```

Parameters

error
If validation fails, on return contains an `NSError` object describing the reason or reasons for failure.

Return Value
`YES` if the validation operation succeeded, otherwise `NO`.

Discussion
The constants indicating the kind of validation errors are emitted by the underlying parser; see `NSXMLParser.h` for most of these constants. If the schema is defined with a DTD, this method uses the `NSXMLDTD` object set for the receiver for validation. If the schema is based on XML Schema, the method uses the URL specified as the value of the `xsi:schemaLocation` attribute of the root element.

You can validate an XML document when it is first processed by specifying the `NSXMLDocumentValidate` option when you initialize an `NSXMLDocument` object with the `initWithContentsOfURL:options:error:` (page 11), `initWithData:options:error:` (page 12), or `initWithXMLString:options:error:` (page 13) methods.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setDTD:` (page 21)

Declared In
`NSXMLDocument.h`

version

Returns the version of the receiver's XML.

```
- (NSString *)version
```

Return Value
The version of the receiver's XML or `nil` if the version has not be set.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setVersion:` (page 23)

Declared In
`NSXMLDocument.h`

XMLData

Returns the XML string representation of the receiver—that is, the entire document—encapsulated in a data object.

```
- (NSData *)XMLData
```

Discussion
This method invokes `XMLDataWithOptions:` with an option of `NSXMLNodeOptionsNone`. The encoding used is based on the value returned from `characterEncoding` (page 10) or UTF-8 if no valid encoding is returned by that method.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- XMLDataWithOptions:` (page 25)

Related Sample Code
CocoaSOAP

Declared In
`NSXMLDocument.h`

XMLDataWithOptions:

Returns the XML string representation of the receiver—that is, the entire document—encapsulated in a data object.

```
- (NSData *)XMLDataWithOptions:(NSUInteger)options
```

Parameters

`options`
: One or more options (bit-OR'd if multiple) to affect the output of the document; see "Constants" (page 25) for the valid output options.

Discussion
The encoding used is based on the value returned from `characterEncoding` (page 10).

Availability
Available in Mac OS X v10.4 and later.

See Also
`- XMLData` (page 24)

Related Sample Code
AlbumToSlideshow

Declared In
`NSXMLDocument.h`

Constants

Input and Output Options

Input and output options specifically intended for `NSXMLDocument` objects.

```
NSXMLDocumentTidyHTML = 1 << 9,
NSXMLDocumentTidyXML = 1 << 10,
NSXMLDocumentValidate = 1 << 13,
NSXMLDocumentXInclude = 1 << 16,
NSXMLDocumentIncludeContentTypeDeclaration = 1 << 18,
```

Constants

`NSXMLDocumentTidyHTML`

Formats HTML into valid XHTML during processing of the document.

When tidying, `NSXMLDocument` adds a line break before the close tag of a block-level element (`<p>`, `<div>`, `<h1>`, and so on); it also makes the string value of `<br>` or `<hr>` a line break. These operations make the string value of the HTML `<body>` more readable. After using this option, avoid outputting the document as anything other than the default kind, `NSXMLDocumentXHTMLKind`.

(Input)

Available in Mac OS X v10.4 and later.

Declared in `NSXMLNodeOptions.h`.

`NSXMLDocumentTidyXML`

Changes malformed XML into valid XML during processing of the document.

It also eliminates “pretty-printing” formatting, such as leading tab characters. However, it respects the `xmlns:space="preserve"` attribute.

(Input)

Available in Mac OS X v10.4 and later.

Declared in `NSXMLNodeOptions.h`.

`NSXMLDocumentValidate`

Validates this document against its DTD (internal or external) or XML Schema.

(Input)

Available in Mac OS X v10.4 and later.

Declared in `NSXMLNodeOptions.h`.

`NSXMLDocumentXInclude`

Replaces all XInclude nodes in the document with the nodes referred to.

XInclude allows clients to include parts of another XML document within a document.

(Input)

Available in Mac OS X v10.4 and later.

Declared in `NSXMLNodeOptions.h`.

`NSXMLDocumentIncludeContentTypeDeclaration`

Includes a content type declaration for HTML or XHTML in the output of the document.

(Output)

Available in Mac OS X v10.4 and later.

Declared in `NSXMLNodeOptions.h`.

Discussion

Because `NSXMLDocument` is a subclass of `NSXMLNode`, you can also use the relevant input and output options described in “Constants” in the `NSXMLNode` class reference. You can specify input options in the `NSXMLDocument` methods `initWithContentsOfURL:options:error:` (page 11), `initWithData:options:error:` (page 12), `initWithXMLString:options:error:` (page 13). The `XMLDataWithOptions:` (page 25) method takes output options.

Declared In
`NSXMLNodeOptions.h`

NSXMLDocumentContentKind

Type used to define the kind of document content.

```
typedef NSUInteger NSXMLDocumentContentKind;
```

Discussion
For possible values, see "Document Content Types" (page 27).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSXMLDocument.h`

Document Content Types

Define document types.

```
enum {
    NSXMLDocumentXMLKind = 0,
    NSXMLDocumentXHTMLKind,
    NSXMLDocumentHTMLKind,
    NSXMLDocumentTextKind
};
```

Constants

`NSXMLDocumentXMLKind`

The default type of document content type, which is XML.

Available in Mac OS X v10.4 and later.

Declared in `NSXMLDocument.h`.

`NSXMLDocumentXHTMLKind`

The document output is XHTML.

This is set automatically if the `NSXMLDocumentTidyHTML` option is set and NSXML detects HTML.

Available in Mac OS X v10.4 and later.

Declared in `NSXMLDocument.h`.

`NSXMLDocumentHTMLKind`

Outputs empty tags in HTML without a close tag, such as `<br>`.

Available in Mac OS X v10.4 and later.

Declared in `NSXMLDocument.h`.

`NSXMLDocumentTextKind`

Outputs the string value of the document by extracting the string values from all text nodes.

Available in Mac OS X v10.4 and later.

Declared in `NSXMLDocument.h`.

Discussion

You specify one of the `NSXMLDocumentContentKind` constants in `setDocumentContentKind:` (page 20) to indicate the kind of content required for document output.

Declared In

`NSXMLDocument.h`

Document Revision History

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

Date	Notes
2007-02-27	Added descriptions of the objectByApplyingXSLTString:arguments:error: method and the NSXMLDocumentContentKind type.
2007-02-08	Made formatting changes to conform to style guide.
2006-11-07	Clarified discussion of arguments parameter of objectByApplyingXSLT...:arguments:error: methods
2006-05-23	First publication of this content as a separate document.

Index