Package 'XiMpLe'

July 21, 2025

Type Package

Title A Simple XML Tree Parser and Generator

Description Provides a simple XML tree parser/generator. It includes functions to read XML files into R objects, get information out of and into nodes, and write R objects back to XML code. It's not as powerful as the 'XML' package and doesn't aim to be, but for simple XML handling it could be useful. It was originally developed for the R GUI and IDE 'RKWard' <https://rkward.kde.org>, to make plugin development easier.

Author Meik Michalke [aut, cre]

Maintainer Meik Michalke <meik.michalke@hhu.de>

Depends R (>= 3.5.0)

Imports methods

Suggests testthat,knitr,rmarkdown

VignetteBuilder knitr

URL https://reaktanz.de/?c=hacking&s=XiMpLe

BugReports https://github.com/rkward-community/XiMpLe/issues

License GPL (>= 3)

Encoding UTF-8

LazyLoad yes

Version 0.11-3

Date 2024-07-18

RoxygenNote 7.3.1

Collate '00_class_01_XiMpLe.node.R' '00_class_02_XiMpLe.doc.R' '00_class_03_XiMpLe.validity.R' '01_method_01_pasteXML.R' 'XiMpLe-internal.R' '01_method_02_node.R' '01_method_03_show.R' '01_method_04_validXML.R' '01_method_05_XMLgenerators.R' 'XMLNode.R' 'XMLTree.R' 'XiMpLe-package.R' 'gen_tag_functions.R' 'parseXMLTree.R' 'pasteXMLTag.R' 'provide_file.R' 'zzz_is_get_utils.R' 'zzz_is_get_utils_deprecated.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2024-07-23 10:20:01 UTC

Contents

`XiMpLe-package` *A Simple XML Tree Parser and Generator*

Description

Provides a simple XML tree parser/generator. It includes functions to read XML files into R objects, get information out of and into nodes, and write R objects back to XML code. It's not as powerful as the 'XML' package and doesn't aim to be, but for simple XML handling it could be useful. It was originally developed for the R GUI and IDE 'RKWard' <https://rkward.kde.org>, to make plugin development easier.

Details

The DESCRIPTION file:

Package:	XiMpLe
Type:	Package
Version:	0.11-3
Date:	2024-07-18
Depends:	R (>= 3.0.0)
Encoding:	UTF-8

License:	GPL (>= 3)
LazyLoad:	yes
URL:	https://reaktanz.de/?c=hacking&s=XiMpLe

Author(s)

Meik Michalke [aut, cre]

Maintainer: Meik Michalke <meik.michalke@hhu.de>

See Also

Useful links:

- https://reaktanz.de/?c=hacking&s=XiMpLe
- Report bugs at https://github.com/rkward-community/XiMpLe/issues

---

gen_tag_functions *Function generator to simplify generation of* XiMpLe_node *objects*

Description

Takes a vector of character strings and turns them into functions in the defined environment which in turn will generate XiMpLe_node objects with the string values as tag names.

Usage

```
gen_tag_functions(
  tags,
  func_names = paste0(tags, "_"),
  envir = .GlobalEnv,
  replace = FALSE,
  func_rename = c(`?xml_` = "xml_", `!--_` = "comment_", `![CDATA[_` = "CDATA_",
    `!DOCTYPE_` = "DOCTYPE_")
)
```

Arguments

tags	A character vector defining the tags the generated functions should produce.
func_names	A character vector the same length as tags, defining the names of the functions to generate.
envir	The environment where all generated functions should appear.
replace	Logical, whether objects by the same name already present in envir should be preserved or replaced/overwritten.
func_rename	Named character vector defining which tags' functions should get a different name. This makes it easier to get functions with valid names that generate special tag nodes.

Details

The generated functions will be named according to `func_names` and only have a dots argument that is given to `XMLNode`. See the examples to understand how it's supposed to work.

Value

As many functions as specified by `tags`/`func_names`.

See Also

`XMLNode`,

Examples

```
# Say we would like to generate an HTML website and want to use
# <a>, <div> and <p> tags.
# The standard way of creating a <div> node would be this:
(my_node <- XMLNode("div", "some content", class="important"))

# By using gen_tag_functions(), we can create some shortcut functions
# to get better readability for our code and save some typing:
gen_tag_functions(tags=c("a", "div", "p"))
# We can now use div_() instead of XMLNode("div"):
(my_node2 <- div_("some content", class="important"))

# It also works for nested tags:
(my_node3 <- div_(a_(href="foo", "some content")))

# If you don't want these functions filling up your .GlobalEnv,
# you can also put them in an attached environment, e.g.
attach(list(), name="XiMpLe_wrappers")
gen_tag_functions(tags=c("head", "body"), envir=as.environment("XiMpLe_wrappers"))
```

---

node *Extract/manipulate a node or parts of it from an XML tree*

Description

This method can be used to get parts of a parsed XML tree object, or to fill it with new values.

`XiMpLe.XML` is a class union for objects of classes `XiMpLe_node` and `XiMpLe_doc`.

Usage

```
node(
  obj,
  node = list(),
  what = NULL,
  cond.attr = NULL,
```

```
  cond.value = NULL,
  element = NULL
)

## S4 method for signature 'XiMpLe.XML'
node(
  obj,
  node = list(),
  what = NULL,
  cond.attr = NULL,
  cond.value = NULL,
  element = NULL
)

node(
  obj,
  node = list(),
  what = NULL,
  cond.attr = NULL,
  cond.value = NULL,
  element = NULL
) <- value

## S4 replacement method for signature 'XiMpLe.XML'
node(
  obj,
  node = list(),
  what = NULL,
  cond.attr = NULL,
  cond.value = NULL,
  element = NULL
) <- value
```

Arguments

obj	An object of class XiMpLe_doc or XiMpLe_node.
node	A list of node names (or their numeric values), where each element is the child of its previous element. duplicate matches will be returned as a list.
what	A character string, must be a valid slot name of class XiMpLe_node, like "attributes" or "value". If not NULL, only that part of a node will be returned. There's also two special properties for this option: what="@path" will not return the node or it's contents, but a character string with the "path" to it in the object; what="obj@path" is the same but won't have obj substituted with the object's name.
cond.attr	A named character string, to further filter the returned results. If not NULL, only nodes with fully matching attributes will be considered.
cond.value	A character string, similar to cond.attr, but is matched against the value between a pair of tags.

element	A character string naming one list element of the node slot. If NULL, all elements will be returned.
value	The value to set.

Examples

```
## Not run:
node(my.xml.tree, node=list("html","body"), what="attributes")
node(my.xml.tree, node=list("html","head","title"), what="value") <- "foobar"

## End(Not run)
```

---

parseXMLTree *Read an XML file into an R object*

Description

Read an XML file into an R object

Usage

```
parseXMLTree(file, drop = NULL, object = FALSE)
```

Arguments

file	Character string, valid path to the XML file which should be parsed.
drop	Character vector with the possible values "comments", "cdata" "declarations", and "doctype", defining element classes to be dropped from the resulting object, or "empty_attributes", in case you would like to omit empty attributes (as used in HTML).
object	Logical, if TRUE, file will not be treated as a path name but as a character vector to be parsed as XML directly.

Value

An object of class XiMpLe_doc with four slots:

file: Full path to the parsed file, or "object" if object=TRUE.

xml: XML declaration, if found.

dtd: Doctype definition, if found.

children: A list of objects of class XiMpLe_node, with the elements "name" (the node name), "attributes" (list of attributes, if found), "children" (list of XiMpLe_node object, if found) and "value" (text value between a pair of start/end tags, if found).

See Also

XiMpLe_node, XiMpLe_doc

Examples

```
## Not run:
sample.XML.object <- parseXMLTree("~/data/sample_file.xml")

## End(Not run)
```

---

pasteXML *Paste methods for XiMpLe XML objects*

Description

These methods can be used to paste objects if class XiMpLe_node or XiMpLe_doc.

Usage

```
pasteXML(obj, ...)

## S4 method for signature 'XiMpLe_node'
pasteXML(
  obj,
  level = 1,
  shine = 1,
  indent.by = getOption("XiMpLe_indent", "\t"),
  tidy = TRUE,
  tidy.omit = c("![CDATA[", "*![CDATA["),
  as_script = FALSE,
  func_rename = c(`?xml_` = "xml_", `!--_` = "comment_", `![CDATA[_` = "CDATA_",
    `!DOCTYPE_` = "DOCTYPE_")
)

## S4 method for signature 'XiMpLe_doc'
pasteXML(
  obj,
  shine = 1,
  indent.by = getOption("XiMpLe_indent", "\t"),
  tidy = TRUE,
  tidy.omit = c("![CDATA[", "*![CDATA["),
  as_script = FALSE,
  func_rename = c(`?xml_` = "xml_", `!--_` = "comment_", `![CDATA[_` = "CDATA_",
    `!DOCTYPE_` = "DOCTYPE_")
)
```

Arguments

obj	An object of class XiMpLe_node or XiMpLe_doc.
...	Additional options for the generic method, see options for a specific method, respectively.

level	Indentation level.
shine	Integer, controlling if the output should be formatted for better readability. Possible values: **0** No formatting. **1** Nodes will be indented. **2** Nodes will be indented and each attribute gets a new line.
indent.by	A charachter string defining how indentation should be done. Defaults to tab.
tidy	Logical, if TRUE the special characters "<" and ">" will be replaced with the entities "<" and "gt;" in attributes and text values.
tidy.omit	A character vector with node names that should be excluded from tidy.
as_script	Logical, if TRUE, tags will be pasted as a sketch for a script to be run in conjunction with functions generated by gen_tag_functions. This script code will most likely not run without adjustments, but is perhaps a good start anyway.
func_rename	Named character vector defining which tags' functions should get a different name. This makes it easier to get functions with valid names that generate special tag nodes. Only used when as_script=TRUE. Use the same names and values as you used in gen_tag_functions.

Note

The functions pasteXMLNode() and pasteXMLTree() have been replaced by the pasteXML methods. They should no longer be used.

See Also

XiMpLe_node, XiMpLe_doc

---

pasteXMLTag *Write an XML tag*

Description

Creates a whole XML tag with attributes and, if it is a pair of start and end tags, also one object as child. This can be used recursively to create whole XML tree structures with this one function.

Usage

```
pasteXMLTag(
  tag,
  attr = NULL,
  child = NULL,
  empty = TRUE,
  level = 1,
  allow.empty = FALSE,
  rename = NULL,
```

```
  shine = 2,
  indent.by = getOption("XiMpLe_indent", "\t"),
  tidy = TRUE,
  as_script = FALSE,
  func_name = paste0(tag, "_"),
  func_rename = c(`?xml_` = "xml_", `!--_` = "comment_", `![CDATA[_` = "CDATA_",
    `!DOCTYPE_` = "DOCTYPE_")
)
```

Arguments

tag	Character string, name of the XML tag.
attr	A list of attributes for the tag.
child	If empty=FALSE, a character string to be pasted as a child node between start and end tag.
empty	Logical, <true /> or <false></false>
level	Indentation level.
allow.empty	Logical, if FALSE, tags without attributes will not be returned.
rename	An optional named list if the attributes in XML need to be renamed from their list names in attr. This list must in turn have a list element named after tag, containing named character elements, where the names represent the element names in attr and their values the names the XML attribute should get.
shine	Integer, controlling if the output should be formatted for better readability. Possible values: **0** No formatting. **1** Nodes will be indented. **2** Nodes will be indented and each attribute gets a new line.
indent.by	A charachter string defining how indentation should be done. Defaults to tab.
tidy	Logical, if TRUE the special characters "<", ">" and "&" will be replaced with the entities "<", ">" and "&" in attribute values. For comment or CDATA tags, if the text includes newline characters they will also be indented.
as_script	Logical, if TRUE, tags will be pasted as a sketch for a script to be run in conjunction with functions generated by gen_tag_functions. This script code will most likely not run without adjustments, but is perhaps a good start anyway.
func_name	A character string, defining a function name for tag. Only used when as_script=TRUE.
func_rename	Named character vector defining which tags' functions should get a different name. This makes it easier to get functions with valid names that generate special tag nodes. Only used when as_script=TRUE. Use the same names and values as you used in gen_tag_functions.

Value

A character string.

Note

However, you will probably not want to use this function at all, as it is much more comfortable to create XML nodes or even nested trees with XMLNode and XMLTree, and then feed the result to pasteXML.

See Also

XMLNode, XMLTree, pasteXML

Examples

```
sample.XML.tag <- pasteXMLTag("a",
  attr=list(href="http://example.com", target="_blank"),
  child="klick here!",
  empty=FALSE)
```

---

provide_file *Manage static files in project directory*

Description

Copies or overwrites files from a source directory to your project directory. Can be used to make sure that files you are referencing in your generated XML code are present and up to date.

Usage

```
provide_file(rel, to, from, overwrite = TRUE, mode = "0777", quiet = FALSE)
```

Arguments

rel	Relative path of file as to be used in HTML.
to	Full path to the project directory where files should be copied to.
from	Full path to the directory where the file can be found under its rel_path.
overwrite	Logical, whether existing files should be re-written or kept in place.
mode	Permissions for newly created directories below to.
quiet	Logical, whether you would like to see a message when files are copied or already exist.

Details

The function returns the relative path that was given as its first argument, e.g. it can be used inside XMLNode to add relative paths to arguments while also copying the referenced file to the given output directory, keeping the relative path.

It can be useful to write a simple wrapper around this function to set the relevant from and to paths for a project (see examples).

Value

When called, the file is copied from the `from` to the `to` directory, including the relative path given by `rel`. Missing target directories below `to` are created on-the-fly. If successful, the function finally returns an invisible character string identical to `rel`.

Examples

```
## Not run:
# a direct call that would copy the file ~/webpage/v1/static/css/bootstrap.min.css
# to the project directory as /tmp/static/css/bootstrap.min.css
# and include "static/css/bootstrap.min.css" in the <link> tag
my_HTML <- XMLNode(
  "link",
  rel="stylesheet",
  type="text/css",
  href=provide_file(
    rel="static/css/bootstrap.min.css",
    to="/tmp",
    from="~/webpage/v1"
  )
)

# for larger projects, a wrapper function might become handy
prov <- function(
  rel,
  to="/tmp",
  from="~/webpage/v1",
  overwrite=TRUE,
  mode="0777"
){
  provide_file(rel=rel, to=to, from=from, overwrite=overwrite, mode=mode)
}
# let's combine it with a shortcut function for <link>
gen_tag_functions("link")
# now this code produces the same result as the direct call above
my_HTML2 <- link_(
  rel="stylesheet",
  type="text/css",
  href=prov("static/css/bootstrap.min.css")
)

## End(Not run)
```

`show,XiMpLe.XML-method`

Show method for S4 objects of XiMpLe XML classes

Description

Used to display objects of class `XiMpLe_doc` and `XiMpLe_node`

Usage

```
## S4 method for signature 'XiMpLe.XML'
show(object)
```

Arguments

object	An object of class XiMpLe_doc or XiMpLe_node

See Also

XiMpLe_doc XiMpLe_node

---

validXML *Validate S4 objects of XiMpLe XML classes*

Description

Checks whether objects of class XiMpLe_doc or XiMpLe_node have valid child nodes.

Usage

```
validXML(
  obj,
  validity = XMLValidity(),
  parent = NULL,
  children = TRUE,
  attributes = TRUE,
  warn = FALSE,
  section = parent,
  caseSens = TRUE
)

## S4 method for signature 'XiMpLe.XML'
validXML(
  obj,
  validity = XMLValidity(),
  parent = NULL,
  children = TRUE,
  attributes = TRUE,
  warn = FALSE,
  section = parent,
  caseSens = TRUE
)
```

Arguments

obj	An object of class XiMpLe_doc or XiMpLe_node. If parent=NULL, this object will be checked for validity, including its child nodes. If parent is either a character string or another XiMpLe node, it will be checked whether obj is a valid child node of parent.
validity	An object of class XiMpLe.validity, see XMLValidity.
parent	Either a character string (name of the parent node) or a XiMpLe node, whose name will be used as name of the parent node.
children	Logical, whether child node names should be checked for validity.
attributes	Logical, whether attributes should be checked for validity.
warn	Logical, whether invalid objects should cause a warning or stop with an error.
section	Either a character string (name of the section) or a XiMpLe node, whose name will be used as name of the XML section this check refers to. This is only relevant for warnings and error messages, in case you want to use something different than the actual parent node name.
caseSens	Logical, whether checks should be case sensitive or not.

Details

XiMpLe can't handle DOM specifications yet, but this method can be used to construct validation schemes.

Value

Returns TRUE if tests pass, and depending on the setting of warn either FALSE or an error if a test fails.

Note

: If no parent is specified, obj will be checked recursively.

See Also

validXML, XMLValidity, XiMpLe_doc, and XiMpLe_node

Examples

```
HTMLish <- XMLValidity(
   children=list(
     body=c("a", "p", "ol", "ul", "strong"),
     head=c("title"),
     html=c("head", "body"),
     li=c("a", "br", "strong"),
     ol=c("li"),
     p=c("a", "br", "ol", "ul", "strong"),
     ul=c("li")
   ),
   attrs=list(
```

```
      a=c("href", "name"),
      p=c("align")
    ),
    allChildren=c("!--"),
    allAttrs=c("id", "class"),
    empty=c("br")
)
# make XML object
validChildNodes <- XMLNode("html",
  XMLNode("head",
    XMLNode("!--", "comment always passes"),
    XMLNode("title", "test")
  ),
  XMLNode("body",
    XMLNode("p",
      XMLNode("a", "my link"),
      XMLNode("br"),
      "text goes on"
    )
  )
)
invalidChildNodes <- XMLNode("html",
  XMLNode("head",
    XMLNode("title",
      XMLNode("body", "test")
    )
  )
)

# do validity checks
# the first should pass
validXML(
  validChildNodes,
  validity=HTMLish
)

# now this one should cause a warning and return FALSE
validXML(
  invalidChildNodes,
  validity=HTMLish,
  warn=TRUE
)
```

XiMpLe.validity,-class

Class XiMpLe.validity

Description

Used for objects that describe valid child nodes and attributes of XiMpLe_nodes.

Usage

```
is.XiMpLe.validity(x)
```

Arguments

x	An arbitrary R object.

Details

A contructor function `XMLValidity(...)` is available to be used instead of `new("XiMpLe.validity", ...)`.

Slots

`children` Named list of vectors or XiMpLe.validity objects. The element name defines the parent node name and each character string a valid child node name. If a value is in turn of class XiMpLe.validity, this object will be used for recursive validation of deeper nodes.

`attrs` Named list of character vectors. The element name defines the parent node name and each character string a valid attribute name.

`allChildren` Character vector, names of globally valid child nodes for all nodes, if any.

`allAttrs` Character vector, names of globally valid attributes for all nodes, if any.

`empty` Character vector, names of nodes that must be empty nodes (i.e., no closing tag), if any.

`ignore` Character vector, names of nodes that should be ignored, if any.

See Also

validXML

Examples

```
HTMLish <- XMLValidity(
   children=list(
     body=c("a", "p", "ol", "ul", "strong"),
     head=c("title"),
     html=c("head", "body"),
     li=c("a", "br", "strong"),
     ol=c("li"),
     p=c("a", "br", "ol", "ul", "strong"),
     ul=c("li")
   ),
   attrs=list(
     a=c("href", "name"),
     p=c("align")
   ),
   allChildren=c("!--"),
   allAttrs=c("id", "class"),
   empty=c("br")
)
```

```
# this example uses recursion: the <b> node can have the "foo"
# attribute only below an <a> node; the <d> node is also only valid
# in an <a> node
XMLRecursion <- XMLValidity(
   children=list(
     a=XMLValidity(
       children=list(
         b=c("c")
       ),
       attrs=list(
         b=c("foo")
       ),
       allChildren=c("d")
     )
   ),
   attrs=list(
     b=c("bar")
   )
 )
```

`XiMpLe_doc,-class` *Classes XiMpLe_doc and XiMpLe.doc (old)*

Description

This class is used for objects that are returned by `parseXMLTree`.

Usage

```
is.XiMpLe.doc(x)

as_XiMpLe_doc(obj, extra = list(), version = 2)

## S4 method for signature 'XiMpLe.doc'
as_XiMpLe_doc(obj, extra = list(), version = 2)
```

Arguments

`x`	An arbitrary `R` object.
`obj`	An object of old class `XiMpLe.doc`.
`extra`	A list of values to set the `extra` slot. Note that this will be applied recursively on child nodes also.
`version`	Integer numeric, to set the `version` slot. Note that this will be applied recursively on child nodes also.

Details

Class XiMpLe.doc is the older one, XiMpLe_doc was introduced with XiMpLe 0.11-1. It has two new slots (extra and version) that would have made it impossible to load old objects without issues. XiMpLe_doc inherits from XiMpLe.doc. You can convert old objects into valid new ones using the as_XiMpLe_doc method and are also advised to do so, as the XiMpLe.doc class might become deprecated in future releases.

A contructor function XiMpLe_doc(...) is available to be used instead of new("XiMpLe_doc", ...).

There's also XiMpLe_doc_old(...) to be used instead of new("XiMpLe.doc", ...), but you should not use that any longer.

Slots

file Character string, Name of the file.

xml Either a named list of character values (attributes for the XML declaration of the file), or a list of XiMpLe_nodes with tags whose names must start with a "?".

dtd A named list, attributes for the doctype definition of the file.

children A list of objects of class XiMpLe_node (XiMpLe_doc only), or XiMpLe.node (old, XiMpLe.doc only), representing the DOM structure of the XML document.

extra A named list that can be used to store additional information on a document (XiMpLe_doc only).

version A numeric integer, currently defaults to 2 (XiMpLe_doc only). This is a version number that can be used in the future in combination with the added extra slot. Should that get sime supported values that are interpreted by package methods, the version number will be increased and the differences documented here. You shouldn't set it manually.

XiMpLe_node,-class *Classes XiMpLe_node and XiMpLe.node (old)*

Description

These classes are used to create DOM trees of XML documents, like objects that are returned by parseXMLTree.

Usage

```
is.XiMpLe.node(x)

is.XiMpLe_node(x)

as_XiMpLe_node(obj, extra = list(), version = 2)

## S4 method for signature 'XiMpLe.node'
as_XiMpLe_node(obj, extra = list(), version = 2)
```

Arguments

x	An arbitrary R object.
obj	An object of old class XiMpLe.node.
extra	A list of values to set the extra slot. Note that this will be applied recursively on child nodes also.
version	Integer numeric, to set the version slot. Note that this will be applied recursively on child nodes also.

Details

Class XiMpLe.node is the older one, XiMpLe_node was introduced with XiMpLe 0.11-1. It has two new slots (extra and version) that would have made it impossible to load old objects without issues. XiMpLe_node inherits from XiMpLe.node. You can convert old objects into valid new ones using the as_XiMpLe_node method and are also advised to do so, as the XiMpLe.node class might become deprecated in future releases.

There are certain special values predefined for the name slot to easily create special XML elements:

name="" If the name is an empty character string, a pseudo node is created, pasteXMLNode will paste its value as plain text.

name="!-" Creates a comment tag, i.e., this will comment out all its children.

name="![CDATA[" Creates a CDATA section and places all its children in it.

name="*![CDATA[" Creates a CDATA section and places all its children in it, where the CDATA markers are commented out by /* */, as is used for JavaScript in XHTML.

A contructor function XiMpLe_node(...) is available to be used instead of new("XiMpLe_node", ...).

There's also XiMpLe_node_old(...) to be used instead of new("XiMpLe.node", ...), but you should not use that any longer.

Slots

name Name of the node (i.e., the XML tag identifier). For special names see details.

attributes A list of named character values, representing the attributes of this node. Use character() as value for empty attributes.

children A list of further objects of class XiMpLe.node, representing child nodes of this node.

value Plain text to be used as the enclosed value of this node. Set to value="" if you want a childless node to be forced into an non-empty pair of start and end tags by pasteXMLNode.

extra A named list that can be used to store additional information on a node (XiMpLe_node only). The only value with a noticeable effect as of now (version 2) is:

shine: A numeric integer value between 0 and 2, overwriting the shine value of, e.g., pasteXML for this particular node.

version A numeric integer, currently defaults to 2 (XiMpLe_node only). This is a version number that can be used in the future in combination with the added extra slot. Should that get more supported values like shine that are interpreted by package methods, the version number will be increased and the differences documented here. You shouldn't set it manually.

XMLgenerators *Generate XML generator functions from XiMpLe.validity object*

Description

Takes an object of class `XiMpLe.validity` and turns it into a character vector of generator functions for each XML node that was defined.

Usage

```
XMLgenerators(
  validity,
  prefix = "XML",
  checkValidity = TRUE,
  indent.by = getOption("XiMpLe_indent", "\t"),
  roxygenDocs = FALSE,
  valParam = "validity",
  replaceChar = "_",
  dir = NULL,
  overwrite = FALSE,
  oneFile = NULL
)

## S4 method for signature 'XiMpLe.validity'
XMLgenerators(
  validity,
  prefix = "XML",
  checkValidity = TRUE,
  indent.by = "\t",
  roxygenDocs = FALSE,
  valParam = "validity",
  replaceChar = "_",
  dir = NULL,
  overwrite = FALSE,
  oneFile = NULL
)
```

Arguments

`validity`	An dobject of class `XiMpLe.validity`.
`prefix`	A character string to be used as a prefix for the resulting function names.
`checkValidity`	Logical, whether all functions should include a check for valid XML.
`indent.by`	A charachter string defining how indentation should be done.
`roxygenDocs`	Logical, whether a skeleton for roxygen2-ish documentation should be added.
`valParam`	A charachter string, name of the additional parameter to use for validation if `checkValidity=TRUE`.

replaceChar	A (single) character to be used as an replacement for invalid characters for R parameter names.
dir	A charachter string, path to write files to. If dir=NULL, no files are being written, but the results returned in form of a character vector. If dir is set and the directory does not yet exist, it will be created.
overwrite	Logical, whether existing files should be replaced when dir is set.
oneFile	A charachter string. If set, all functions are to be documented in one single *.Rd file, named like the string.

Details

The resulting code follows these rules:

- Each child node gets its own argument, except if there is only one valid child node. It will use the dots element instead then.
- Each attribute will also get its own argument.
- If CheckValidity=TRUE, one extra argument named after the value of valParam will be added.
- All arguments are set to NULL by default.
- Only the main level of "allAttrs" will be taken into account, there's no recursion for this slot.

Value

If dir=NULL a named vector of character strings. Otherwise one or more files are written do the location specified via dir.

See Also

XMLValidity and XiMpLe.validity

Examples

```
HTMLish <- XMLValidity(
   children=list(
     body=c("a", "p", "ol", "ul", "strong"),
     head=c("title"),
     html=c("head", "body"),
     li=c("a", "br", "strong"),
     ol=c("li"),
     p=c("a", "br", "ol", "ul", "strong"),
     ul=c("li")
   ),
   attrs=list(
     a=c("href", "name"),
     p=c("align")
   ),
   allChildren=c("!--"),
   allAttrs=c("id", "class"),
```

```
   empty=c("br")
)
XMLgenerators(HTMLish)
```

XMLName *Getter/setter methods for S4 objects of XiMpLe XML classes*

Description

Used to get/set certain slots from objects of class XiMpLe_doc and XiMpLe_node.

Usage

```
XMLName(obj)

## S4 method for signature 'XiMpLe_node'
XMLName(obj)

XMLName(obj) <- value

## S4 replacement method for signature 'XiMpLe_node'
XMLName(obj) <- value

XMLAttrs(obj)

## S4 method for signature 'XiMpLe_node'
XMLAttrs(obj)

XMLAttrs(obj) <- value

## S4 replacement method for signature 'XiMpLe_node'
XMLAttrs(obj) <- value

XMLChildren(obj)

## S4 method for signature 'XiMpLe_node'
XMLChildren(obj)

## S4 method for signature 'XiMpLe_doc'
XMLChildren(obj)

XMLChildren(obj) <- value

## S4 replacement method for signature 'XiMpLe_node'
XMLChildren(obj) <- value

## S4 replacement method for signature 'XiMpLe_doc'
```

```
XMLChildren(obj) <- value

XMLValue(obj)

## S4 method for signature 'XiMpLe_node'
XMLValue(obj)

XMLValue(obj) <- value

## S4 replacement method for signature 'XiMpLe_node'
XMLValue(obj) <- value

XMLFile(obj)

## S4 method for signature 'XiMpLe_doc'
XMLFile(obj)

XMLFile(obj) <- value

## S4 replacement method for signature 'XiMpLe_doc'
XMLFile(obj) <- value

XMLDecl(obj)

## S4 method for signature 'XiMpLe_doc'
XMLDecl(obj)

XMLDecl(obj) <- value

## S4 replacement method for signature 'XiMpLe_doc'
XMLDecl(obj) <- value

XMLDTD(obj)

## S4 method for signature 'XiMpLe_doc'
XMLDTD(obj)

XMLDTD(obj) <- value

## S4 replacement method for signature 'XiMpLe_doc'
XMLDTD(obj) <- value

XMLScan(obj, name, as.list = FALSE)

## S4 method for signature 'XiMpLe_node'
XMLScan(obj, name, as.list = FALSE)

## S4 method for signature 'XiMpLe_doc'
```

```
XMLScan(obj, name, as.list = FALSE)

XMLScan(obj, name) <- value

## S4 replacement method for signature 'XiMpLe_node'
XMLScan(obj, name) <- value

## S4 replacement method for signature 'XiMpLe_doc'
XMLScan(obj, name) <- value

XMLScanDeep(obj, find = NULL, search = "attributes")

## S4 method for signature 'XiMpLe_node'
XMLScanDeep(obj, find = NULL, search = "attributes")

## S4 method for signature 'XiMpLe_doc'
XMLScanDeep(obj, find = NULL, search = "attributes")
```

Arguments

obj	An object of class XiMpLe_node or XiMpLe_doc
value	The new value to set.
name	Character, name of nodes to scan for.
as.list	Logical, if TRUE allways returns a list (or NULL), otherwise if exactly one result is found, it will be returned as as single XiMpLe_node.
find	Character, name of element to scan for.
search	Character, name of the slot to scan, one of "attributes", "name", or "value" for nodes.

Details

These are convenience methods to get or set slots from XML objects without using the @ operator.

XMLName(): get/set the XML node name (slot name of class XiMpLe_node)

XMLAttrs(): get/set the XML node attributes (slot attrs of class XiMpLe_node)

XMLValue(): get/set the XML node value (slot value of class XiMpLe_node)

XMLChildren(): get/set the XML child nodes (slot children of both classes XiMpLe_node and XiMpLe_doc)

XMLFile(): get/set the XML document file name (slot file of class XiMpLe_doc)

XMLDecl(): get/set the XML document declaration (slot xml of class XiMpLe_doc)

XMLDTD(): get/set the XML document doctype definition (slot dtd of class XiMpLe_doc)

Another special method can scan a node/document tree object for appearances of nodes with a particular name:

`XMLScan(obj, name, as.list=FALSE)`: get/set the XML nodes by name (recursively searches slot `name` of both classes `XiMpLe_node` and `XiMpLe_doc`). If `as.list=TRUE` allways returns a list (or NULL), otherwise if exactly one result is found, it will be returned as as single `XiMpLe_node`.

Finally, there is a method to scan for certain values in XiMpLe objects and just list them. For instance, it can be used to list all instances of a certain attribute type in a document tree:

`XMLScanDeep(obj, find, search="attributes")`: returns all found instances of `find` in all slots defined by `search`.

See Also

node, XiMpLe_doc, XiMpLe_node

Examples

```
xmlTestNode <- XMLNode("foo", XMLNode("testchild"))
XMLName(xmlTestNode) # returns "foo"
XMLName(xmlTestNode) <- "bar"
XMLName(xmlTestNode) # now returns "bar"

# search for a child node
XMLScan(xmlTestNode, "testchild")
# remove nodes of that name
XMLScan(xmlTestNode, "testchild") <- NULL
```

XMLName,XiMpLe.node-method

Deprecated functions and methods

Description

These functions were used in earlier versions of the package but are now either replaced or removed.

Usage

```
## S4 method for signature 'XiMpLe.node'
XMLName(obj)

## S4 replacement method for signature 'XiMpLe.node'
XMLName(obj) <- value

## S4 method for signature 'XiMpLe.node'
XMLAttrs(obj)

## S4 replacement method for signature 'XiMpLe.node'
XMLAttrs(obj) <- value
```

```
## S4 method for signature 'XiMpLe.node'
XMLChildren(obj)

## S4 method for signature 'XiMpLe.doc'
XMLChildren(obj)

## S4 replacement method for signature 'XiMpLe.node'
XMLChildren(obj) <- value

## S4 replacement method for signature 'XiMpLe.doc'
XMLChildren(obj) <- value

## S4 method for signature 'XiMpLe.node'
XMLValue(obj)

## S4 replacement method for signature 'XiMpLe.node'
XMLValue(obj) <- value

## S4 method for signature 'XiMpLe.doc'
XMLFile(obj)

## S4 replacement method for signature 'XiMpLe.doc'
XMLFile(obj) <- value

## S4 method for signature 'XiMpLe.doc'
XMLDecl(obj)

## S4 replacement method for signature 'XiMpLe.doc'
XMLDecl(obj) <- value

## S4 method for signature 'XiMpLe.doc'
XMLDTD(obj)

## S4 replacement method for signature 'XiMpLe.doc'
XMLDTD(obj) <- value

## S4 method for signature 'XiMpLe.node'
XMLScan(obj, name, as.list = FALSE)

## S4 method for signature 'XiMpLe.doc'
XMLScan(obj, name, as.list = FALSE)

## S4 replacement method for signature 'XiMpLe.node'
XMLScan(obj, name) <- value

## S4 replacement method for signature 'XiMpLe.doc'
XMLScan(obj, name) <- value
```

```
## S4 method for signature 'XiMpLe.node'
XMLScanDeep(obj, find = NULL, search = "attributes")

## S4 method for signature 'XiMpLe.doc'
XMLScanDeep(obj, find = NULL, search = "attributes")

## S4 method for signature 'XiMpLe.node'
pasteXML(obj, ...)

## S4 method for signature 'XiMpLe.doc'
pasteXML(obj, ...)
```

Arguments

obj	An object of deprecated classes XiMpLe.node or XiMpLe.doc.
value	No longer used.
name	No longer used.
as.list	No longer used.
find	No longer used.
search	No longer used.
...	No longer used.

Details

For methods this can also mean that the object class is deprecated and you are asked to update old objects via as_XiMpLe_node or as_XiMpLe_doc.

XMLNode *Constructor function for* XiMpLe_node *objects*

Description

Can be used to create XML nodes.

Usage

```
XMLNode(
  tag_name,
  ...,
  attrs,
  shine,
  namespace,
  namespaceDefinitions,
  .children = list(...)
)
```

Arguments

tag_name	Character string, the tag name.
...	Optional children for the tag. Must be either objects of class XiMpLe_node or character strings, which are treated as attributes if they are named, and as simple text values otherwise. Use a named character() value for empty attributes. If this argument is empty, the tag will be treated as an empty tag. To force a closing tag, supply an empty string, i.e. "".
attrs	An optional named list of attributes. Will be appended to attributes already defined in the ... argument.
shine	A numeric integer value between 0 and 2, overwriting the shine value of, e.g., pasteXML for this particular node.
namespace	Currently ignored.
namespaceDefinitions	Currently ignored.
.children	Alternative way of specifying children, if you have them already as a list. This argument completely replaces values defined in the ... argument.

Details

To generate a CDATA node, set tag_name="![CDATA[", to create a comment, set tag_name="!--".

Note that all defined attributes will silently be dropped if a text node, CDATA node or comment is generated.

Value

An object of class XiMpLe_node.

See Also

XMLTree, pasteXML

Examples

```
(sample.XML.node <- XMLNode("a",
  attrs=list(
    href="http://example.com",
    target="_blank"
  ),
  .children="klick here!"
))
# This is equivalent
(sample.XML.node2 <- XMLNode("a",
  "klick here!",
  href="http://example.com",
  target="_blank"
))
# As is this
(sample.XML.node3 <- XMLNode("a",
```

```
  .children=list(
    "klick here!",
    href="http://example.com",
    target="_blank"
  )
))
```

`XMLTree` *Constructor function for XiMpLe.doc objects*

Description

Can be used to create full XML trees.

Usage

```
XMLTree(..., xml = NULL, dtd = NULL, .children = list(...))
```

Arguments

`...`	Optional children for the XML tree. Must be either objects of class `XiMpLe_node` or character strings, which are treated as simple text values.
`xml`	A named list, XML declaration of the XML tree. Currently just pasted, no checking is done.
`dtd`	A named list, doctype definition of the XML tree. Valid elements are `doctype` (root element), `decl` ("PUBLIC" or "SYSTEM"), `id` (the identifier) and `refer` (URI to .dtd). Currently just pasted, no checking is done.
`.children`	Alternative way of specifying children, if you have them already as a list.

Value

An object of class `XiMpLe_doc`

See Also

`XMLNode`, `pasteXML`

Examples

```
sample.XML.a <- XMLNode("a",
  attrs=list(href="http://example.com", target="_blank"),
  .children="klick here!")
sample.XML.body <- XMLNode("body", .children=list(sample.XML.a))
sample.XML.html <- XMLNode("html", .children=list(XMLNode("head", ""),
  sample.XML.body))
sample.XML.tree <- XMLTree(sample.XML.html,
  xml=list(version="1.0", encoding="UTF-8"),
  dtd=list(doctype="html", decl="PUBLIC",
    id="-//W3C//DTD XHTML 1.0 Transitional//EN",
    refer="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"))
```

Index