Package ‘XiMpLe’

November 24, 2015

Type Package

Title A Simple XML Tree Parser and Generator

Author Meik Michalke [aut, cre]

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

Depends R (>= 2.9.0),methods

Suggests testthat

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.

License GPL (>= 3)

Encoding UTF-8

LazyLoad yes

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

Version 0.03-23

Date 2015-11-24

RoxygenNote 5.0.1

Collate '00_class_01_XiMpLe.node.R'
'00_class_02_XiMpLe.doc.R'
'01_method_01_pasteXML.R'
'XiMpLe-internal.R'
'01_method_02_node.R'
'01_method_03_show.R'
'XMLNode.R'
'XMLTree.R'
'XiMpLe-package.R'
'parseXMLTree.R'
'pasteXMLTag.R'
'zzz_is_get_utils.R'

R topics documented:

XiMpLe-package 2
node 3
parseXMLTree 4
pasteXML 5
pasteXMLTag 6
show,XiMpLe.XML-method 7
XiMpLe.doc-class 8
XiMpLe.node-class 8
XMLName 9
XMLNode 12
XMLTree 13

Index **15**

`XiMpLe-package` *The XiMpLe Package*

Description

A Simple XML Tree Parser and Generator.

Details

Package:	XiMpLe
Type:	Package
Version:	0.03-23
Date:	2015-11-24
Depends:	R (>= 2.9.0),methods
Encoding:	UTF-8
License:	GPL (>= 3)
LazyLoad:	yes
URL:	http://reaktanz.de/?c=hacking&s=XiMpLe

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(s)

Meik Michalke

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.

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.
`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 = "\t", tidy = TRUE)

## S4 method for signature 'XiMpLe.doc'
pasteXML(obj, shine = 1, indent.by = "\t",
  tidy = TRUE)
```

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.

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 = "\t",
  tidy = TRUE)
```

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.

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)
```

---

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

XiMpLe.doc-class *S4 class XiMpLe.doc*

Description

Class XiMpLe.doc

Details

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

Slots

`file`: (`character`) Name of the file.

`xml`: (`list`) XML declaration of the file.

`dtd`: (`list`) Doctype definition of the file.

`children`: (`list`) A list of objects of class XiMpLe.node, representing the DOM structure of the XML document.

XiMpLe.node-class *S4 class XiMpLe.node*

Description

Class XiMpLe.node

Details

This class is used to create DOM trees of XML documents, like objects that are returned by `parseXMLTree`.

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.

Slots

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

`attributes`: (`list`) A list of named character values, representing the attributes of this node.

`children`: (`list`) A list of further objects of class XiMpLe.node, representing child nodes of this node.

`value`: (`character`) 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`.

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
```

`XMLNode` *Constructor function for XiMpLe.node objects*

Description

Can be used to create XML nodes.

Usage

```
XMLNode(name, ..., attrs = NULL, namespace = "",
  namespaceDefinitions = NULL, .children = list(...))

is.XiMpLe.node(x)
```

Arguments

`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 simple text values. If this 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.
`namespace`	Currently ignored.
`namespaceDefinitions`	Currently ignored.
`.children`	Alternative way of specifying children, if you have them already as a list.
`x`	An arbitrary `R` object.

Details

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

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!")
```

XMLTree — *Constructor function for XiMpLe.doc objects*

Description

Can be used to create full XML trees.

Usage

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

is.XiMpLe.doc(x)
```

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.
`x`	An arbitrary `R` object.

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

∗Topic **classes**
XiMpLe.doc-class, 8
XiMpLe.node-class, 8
∗Topic **methods**
pasteXML, 5
show,XiMpLe.XML-method, 7
XMLName, 9
∗Topic **package**
XiMpLe-package, 2

character, *8*

is.XiMpLe.doc *(XMLTree)*, 13
is.XiMpLe.node *(XMLNode)*, 12

list, *8*

node, 3, *12*
node,-methods *(node)*, 3
node,XiMpLe.doc-method *(node)*, 3
node,XiMpLe.node-method *(node)*, 3
node,XiMpLe.XML-method *(node)*, 3
node<- *(node)*, 3
node<-,-methods *(node)*, 3
node<-,XiMpLe.doc-method *(node)*, 3
node<-,XiMpLe.node-method *(node)*, 3
node<-,XiMpLe.XML-method *(node)*, 3

parseXMLTree, 4, *8*
pasteXML, 5, *7*, *13*, *14*
pasteXML,-methods *(pasteXML)*, 5
pasteXML,XiMpLe.doc-method *(pasteXML)*, 5
pasteXML,XiMpLe.node-method *(pasteXML)*, 5
pasteXMLNode, *8*
pasteXMLNode *(pasteXML)*, 5
pasteXMLTag, 6
pasteXMLTree *(pasteXML)*, 5

show,-methods *(show,XiMpLe.XML-method)*, 7
show,XiMpLe.doc-method *(show,XiMpLe.XML-method)*, 7
show,XiMpLe.node-method *(show,XiMpLe.XML-method)*, 7
show,XiMpLe.XML-method, 7

XiMpLe-package, 2
XiMpLe.doc, *3–5*, *7*, *9*, *12*, *13*
XiMpLe.doc,-class *(XiMpLe.doc-class)*, 8
XiMpLe.doc-class, 8
XiMpLe.node, *3–5*, *7*, *9*, *12*, *13*
XiMpLe.node,-class *(XiMpLe.node-class)*, 8
XiMpLe.node-class, 8
XMLAttrs *(XMLName)*, 9
XMLAttrs,-methods *(XMLName)*, 9
XMLAttrs,XiMpLe.node-method *(XMLName)*, 9
XMLAttrs<- *(XMLName)*, 9
XMLAttrs<-,-methods *(XMLName)*, 9
XMLAttrs<-,XiMpLe.node-method *(XMLName)*, 9
XMLChildren *(XMLName)*, 9
XMLChildren,-methods *(XMLName)*, 9
XMLChildren,XiMpLe.doc-method *(XMLName)*, 9
XMLChildren,XiMpLe.node-method *(XMLName)*, 9
XMLChildren<- *(XMLName)*, 9
XMLChildren<-,-methods *(XMLName)*, 9
XMLChildren<-,XiMpLe.doc-method *(XMLName)*, 9
XMLChildren<-,XiMpLe.node-method *(XMLName)*, 9
XMLDecl *(XMLName)*, 9
XMLDecl,-methods *(XMLName)*, 9
XMLDecl,XiMpLe.doc-method *(XMLName)*, 9
XMLDecl<- *(XMLName)*, 9
XMLDecl<-,-methods *(XMLName)*, 9
XMLDecl<-,XiMpLe.doc-method *(XMLName)*, 9
XMLDTD *(XMLName)*, 9

XMLDTD,-methods *(XMLName)*, 9
XMLDTD,XiMpLe.doc-method *(XMLName)*, 9
XMLDTD<- *(XMLName)*, 9
XMLDTD<-,-methods *(XMLName)*, 9
XMLDTD<-,XiMpLe.doc-method *(XMLName)*, 9
XMLFile *(XMLName)*, 9
XMLFile,-methods *(XMLName)*, 9
XMLFile,XiMpLe.doc-method *(XMLName)*, 9
XMLFile<- *(XMLName)*, 9
XMLFile<-,-methods *(XMLName)*, 9
XMLFile<-,XiMpLe.doc-method *(XMLName)*, 9
XMLName, 9
XMLName,-methods *(XMLName)*, 9
XMLName,XiMpLe.node-method *(XMLName)*, 9
XMLName<- *(XMLName)*, 9
XMLName<-,-methods *(XMLName)*, 9
XMLName<-,XiMpLe.node-method *(XMLName)*, 9
XMLNode, *7*, 12, *14*
XMLScan *(XMLName)*, 9
XMLScan,-methods *(XMLName)*, 9
XMLScan,XiMpLe.doc-method *(XMLName)*, 9
XMLScan,XiMpLe.node-method *(XMLName)*, 9
XMLScan<- *(XMLName)*, 9
XMLScan<-,-methods *(XMLName)*, 9
XMLScan<-,XiMpLe.doc-method *(XMLName)*, 9
XMLScan<-,XiMpLe.node-method *(XMLName)*, 9
XMLScanDeep *(XMLName)*, 9
XMLScanDeep,-methods *(XMLName)*, 9
XMLScanDeep,XiMpLe.doc-method *(XMLName)*, 9
XMLScanDeep,XiMpLe.node-method *(XMLName)*, 9
XMLTree, *7*, *13*, 13
XMLValue *(XMLName)*, 9
XMLValue,-methods *(XMLName)*, 9
XMLValue,XiMpLe.node-method *(XMLName)*, 9
XMLValue<- *(XMLName)*, 9
XMLValue<-,-methods *(XMLName)*, 9
XMLValue<-,XiMpLe.node-method *(XMLName)*, 9