NAME

XML::LibXML::Devel – makes functions from LibXML.xs available

SYNOPSIS

```
/*******************************************
 * C functions you want to access
 */
xmlNode *return_node();
void receive_node(xmlNode *);

###############################################
# XS Code
void *
  xs_return_node
  CODE:
      RETVAL = return_node();
  OUTPUT:
      RETVAL

void
  xs_receive_node
      void *n
  CODE:
      receive_node(n);

###############################################
# Perl code
use XML::LibXML::Devel;

sub return_node
{
  my $raw_node = xs_return_node();
  my $node = XML::LibXML::Devel::node_to_perl($raw_node);
  XML::LibXML::Devel::refcnt_inc($raw_node);
  return $node;
}

sub receive_node
{
  my ($node) = @_;
  my $raw_node = XML::LibXML::Devel::node_from_perl($node);
  xs_receive_node($raw_node);
  XML::LibXML::Devel::refcnt_inc($raw_node);
}
```

DESCRIPTION

`XML::LibXML::Devel` makes functions from LibXML.xs available that are needed to wrap libxml2 nodes in and out of XML::LibXML::Nodes. This gives cleaner dependencies than using LibXML.so directly.

To XS a library that uses libxml2 nodes the first step is to do this so that xmlNodePtr is passed as void *. These raw nodes are then turned into libxml nodes by using this `Devel` functions.

Be aware that this module is currently rather experimental. The function names may change if I XS more functions and introduce a reasonable naming convention.

Be also aware that this module is a great tool to cause segfaults and introduce memory leaks. It does

however provide a partial cure by making `xmlMemUsed` available as `mem_used`.

FUNCTIONS

NODE MANAGEMENT

node_to_perl

```
node_to_perl($raw_node);
```

Returns a LibXML::Node object. This has a proxy node with a reference counter and an owner attached. The raw node will be deleted as soon as the reference counter reaches zero. If the C library is keeping a pointer to the raw node, you need to call refcnt_inc immediately. You also need to replace xmlFreeNode by a call to refcnt_dec.

node_to_perl

```
node_from_perl($node);
```

Returns a raw node. This is a void * pointer and you can do nothing but passing it to functions that treat it as an xmlNodePtr. The raw node will be freed as soon as its reference counter reaches zero. If the C library is keeping a pointer to the raw node, you need to call refcnt_inc immediately. You also need to replace xmlFreeNode by a call to refcnt_dec.

refcnt_inc

```
refcnt_inc($raw_node);
```

Increments the raw nodes reference counter. The raw node must already be known to perl to have a reference counter.

refcnt_dec

```
refcnt_dec($raw_node);
```

Decrements the raw nodes reference counter and returns the value it had before. if the counter becomes zero or less, this method will free the proxy node holding the reference counter. If the node is part of a subtree, refcnt_dec will fix the reference counts and delete the subtree if it is not required any more.

refcnt

```
refcnt($raw_node);
```

Returns the value of the reference counter.

fix_owner

```
fix_owner($raw_node, $raw_parent);
```

This functions fixes the reference counts for an entire subtree. it is very important to fix an entire subtree after node operations where the documents or the owner node may get changed. this method is aware about nodes that already belong to a certain owner node.

MEMORY DEBUGGING

$ENV{DEBUG_MEMORY}

```
BEGIN {$ENV{DEBUG_MEMORY} = 1;};
use XML::LibXML;
```

This turns on libxml2 memory debugging. It must be set before XML::LibXML is loaded.

mem_used

```
mem_used();
```

Returns the number of bytes currently allocated.

EXPORT

None by default.

SEE ALSO

This was created to support the needs of Apache2::ModXml2. So this can serve as an example.

AUTHOR

Joachim Zobel <jz−2011@heute−morgen.de>

COPYRIGHT AND LICENSE