NAME

HTML::LinkExtor − Extract links from an HTML document

SYNOPSIS

```
require HTML::LinkExtor;
$p = HTML::LinkExtor->new(\&cb, "http://www.perl.org/");
sub cb {
    my($tag, %links) = @_;
    print "$tag @{[%links]}\n";
}
$p->parse_file("index.html");
```

DESCRIPTION

HTML::LinkExtor is an HTML parser that extracts links from an HTML document. The *HTML::LinkExtor* is a subclass of *HTML::Parser*. This means that the document should be given to the parser by calling the `$p->`**parse()** or `$p->`**parse_file()** methods.

`$p` = HTML::LinkExtor->new
`$p` = HTML::LinkExtor->new(`$callback`)
`$p` = HTML::LinkExtor->new(`$callback`, `$base`)

The constructor takes two optional arguments. The first is a reference to a callback routine. It will be called as links are found. If a callback is not provided, then links are just accumulated internally and can be retrieved by calling the `$p->`**links()** method.

The `$base` argument is an optional base URL used to absolutize all URLs found. You need to have the *URI* module installed if you provide `$base`.

The callback is called with the lowercase tag name as first argument, and then all link attributes as separate key/value pairs. All non-link attributes are removed.

`$p->`links

Returns a list of all links found in the document. The returned values will be anonymous arrays with the following elements:

```
[$tag, $attr => $url1, $attr2 => $url2,...]
```

The `$p->`links method will also truncate the internal link list. This means that if the method is called twice without any parsing between them the second call will return an empty list.

Also note that `$p->`links will always be empty if a callback routine was provided when the *HTML::LinkExtor* was created.

EXAMPLE

This is an example showing how you can extract links from a document received using LWP:

```
use LWP::UserAgent;
use HTML::LinkExtor;
use URI::URL;

$url = "http://www.perl.org/";  # for instance
$ua = LWP::UserAgent->new;

# Set up a callback that collect image links
my @imgs = ();
sub callback {
   my($tag, %attr) = @_;
   return if $tag ne 'img';  # we only look closer at <img ...>
   push(@imgs, values %attr);
}

# Make the parser.  Unfortunately, we don't know the base yet
```

```
    # (it might be different from $url)
    $p = HTML::LinkExtor->new(\&callback);

    # Request document and parse it as it arrives
    $res = $ua->request(HTTP::Request->new(GET => $url),
                        sub {$p->parse($_[0])});

    # Expand all image URLs to absolute ones
    my $base = $res->base;
    @imgs = map { $_ = url($_, $base)->abs; } @imgs;

    # Print them out
    print join("\n", @imgs), "\n";
```

SEE ALSO

HTML::Parser, HTML::Tagset, LWP, URI::URL

COPYRIGHT