

NAME

HTTP::Request - HTTP style request message

SYNOPSIS

```
require HTTP::Request;
$request = HTTP::Request->new(GET => 'http://www.example.com/');
```

and usually used like this:

```
$ua = LWP::UserAgent->new;
$response = $ua->request($request);
```

DESCRIPTION

[HTTP::Request](#) is a class encapsulating HTTP style requests, consisting of a request line, some headers, and a content body. Note that the LWP library uses HTTP style requests even for non-HTTP protocols. Instances of this class are usually passed to the *request()* method of an [LWP::UserAgent](#) object.

[HTTP::Request](#) is a subclass of [HTTP::Message](#) and therefore inherits its methods. The following additional methods are available:

```
$r = HTTP::Request->new( $method, $uri )
```

```
$r = HTTP::Request->new( $method, $uri, $header )
```

```
$r = HTTP::Request->new( $method, $uri, $header, $content )
```

Constructs a new [HTTP::Request](#) object describing a request on the object *\$uri* using method *\$method*. The *\$method* argument must be a string. The *\$uri* argument can be either a string, or a reference to a URI object. The optional *\$header* argument should be a reference to an [HTTP::Headers](#) object or a plain array reference of key/value pairs. The optional *\$content* argument should be a string of bytes.

```
$r = HTTP::Request->parse( $str )
```

This constructs a new request object by parsing the given string.

```
$r->method
```

```
$r->method( $val )
```

This is used to get/set the method attribute. The method should be a short string like "GET", "HEAD", "PUT" or "POST".

```
$r->uri
```

```
$r->uri( $val )
```

This is used to get/set the uri attribute. The *\$val* can be a reference to a URI object or a plain string. If a string is given, then it should be parseable as an absolute URI.

```
$r->header( $field )
```

```
$r->header( $field => $value )
```

This is used to get/set header values and it is inherited from [HTTP::Headers](#) via [HTTP::Message](#). See [HTTP::Headers](#) for details and other similar methods that can be used to access the headers.

```
$r->accept_decodable
```

This will set the **Accept-Encoding** header to the list of encodings that *decoded_content()* can decode.

```
$r->content
```

```
$r->content( $bytes )
```

This is used to get/set the content and it is inherited from the [HTTP::Message](#) base class. See [HTTP::Message](#) for details and other methods that can be used to access the content.

Note that the content should be a string of bytes. Strings in perl can contain characters outside the range of a byte. The `Encode` module can be used to turn such strings into a string of bytes.

```
$r->as_string
```

```
$r->as_string( $eol )
```

Method returning a textual representation of the request.

SEE ALSO

[HTTP::Headers](#), [HTTP::Message](#), [HTTP::Request::Common](#), [HTTP::Response](#)

COPYRIGHT

Copyright 1995-2004 Gisle Aas.

This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself.