---+ Package `TWiki`

TWiki operates by creating a singleton object (known as the Session object) that acts as a point of reference for all the different modules in the system. This package is the class for this singleton, and also contains the vast bulk of the basic constants and the per- site configuration mechanisms.

Global variables are avoided wherever possible to avoid problems with CGI accelerators such as mod_perl.

Public Data members

- `request` Pointer to the TWiki::Request
- `response` Pointer to the TWiki::Respose
- `context` Hash of context ids
- moved: `loginManager` TWiki::LoginManager singleton (moved to TWiki::Users)
- `plugins` TWiki::Plugins singleton
- `prefs` TWiki::Prefs singleton
- `remoteUser` Login ID when using ApacheLogin. Maintained for compatibility only, do not use.
- `requestedWebName` Name of web found in URL path or `web` URL parameter
- `sandbox` TWiki::Sandbox singleton
- `scriptUrlPath` URL path to the current script. May be dynamically extracted from the URL path if {GetScriptUrlFromCgi}. Only required to support {GetScriptUrlFromCgi} and not consistently used. Avoid.
- `security` TWiki::Access singleton
- `SESSION_TAGS` Hash of TWiki variables whose value is specific to the current request.
- `store` TWiki::Store singleton
- `topicName` Name of topic found in URL path or `topic` URL parameter
- `urlHost` Host part of the URL (including the protocol) determined during intialisation and defaulting to {DefaultUrlHost}
- `user` Unique user ID of logged-in user
- `users` TWiki::Users singleton
- `webName` Name of web found in URL path, or `web` URL parameter, or {UsersWebName}

StaticMethod getTWikiLibDir `() -> $path`

Returns the full path of the directory containing TWiki.pm

ObjectMethod UTF82SiteCharSet `($utf8) -> $ascii`

Auto-detect UTF-8 vs. site charset in string, and convert UTF-8 into site charset.

ObjectMethod writeCompletePage `($text,$pageType,$contentType)`

Write a complete HTML page with basic header to the browser.

- `$text` is the text of the page body (<html> to </html> if it's HTML)
- `$pageType` - May be "edit", which will cause headers to be generated that force caching for 24 hours, to prevent BackFromPreviewLosesText bug, which caused data loss with IE5 and IE6.
- `$contentType` - page content type | text/html

This method removes noautolink and nop tags before outputting the page unless $contentType is text/plain.

ObjectMethod generateHTTPHeaders `($pageType,$contentType,$contentLength) -> $header`

All parameters are optional.

- `$pageType` - May be "edit", which will cause headers to be generated that force caching for 24 hours, to prevent BackFromPreviewLosesText bug, which caused data loss with IE5 and IE6.
- `$contentType` - page content type | text/html
- `$contentLength` - content-length | no content-length will be set if this is undefined, as required by HTTP1.1

Implements the post-Dec2001 release plugin API, which requires the writeHeaderHandler in plugin to return a string of HTTP headers, CR/LF delimited. Filters any illegal headers. Plugin headers will override core settings.

Does **not** add a `Content-length` header.

StaticMethod isRedirectSafe `($redirect)=>$ok`

tests if the $redirect is an external URL, returning false if AllowRedirectUrl is denied

ObjectMethod redirect `($url,$passthrough,$action_redirectto)`

- $url - url or twikitopic to redirect to
- $passthrough - (optional) parameter to ***FILLMEIN***
- $action_redirectto - (optional) redirect to where ?redirectto= points to (if it's valid)

Redirects the request to `$url`, **unless**

1. It is overridden by a plugin declaring a `redirectCgiQueryHandler`.
2. `$session->{request}` is `undef` or
3. $query->param('noredirect') is set to a true value.

Thus a redirect is only generated when in a CGI context.

Normally this method will ignore parameters to the current query. Sometimes, for example when redirecting to a login page during authentication (and then again from the login page to the original requested URL), you want to make sure all parameters are passed on, and for this $passthrough should be set to true. In this case it will pass all parameters that were passed to the current query on to the redirect target. If the request_method for the current query was GET, then all parameters will be passed by encoding them in the URL (after ?). If the request_method was POST, then there is a risk the URL would be too big for the receiver, so it caches the form data and passes over a cache reference in the redirect GET.

NOTE: Passthrough is only meaningful if the redirect target is on the same server.

ObjectMethod cacheQuery `() -> $queryString`

Caches the current query in the params cache, and returns a rewritten query string for the cache to be picked up again on the other side of a redirect.

We can't encode post params into a redirect, because they may exceed the size of the GET request. So we cache the params, and reload them when the redirect target is reached.

StaticMethod isValidWikiWord `($name) -> $boolean`

Check for a valid WikiWord or WikiName

StaticMethod isValidTopicName `($name) -> $boolean`

Check for a valid topic name

StaticMethod isValidAbbrev `($name) -> $boolean`

Check for a valid ABBREV (acronym)

StaticMethod isValidWebName `($name,$system) -> $boolean`

STATIC Check for a valid web name. If $system is true, then system web names are considered valid (names starting with _) otherwise only user web names are valid

If $TWiki::cfg{EnableHierarchicalWebs} is off, it will also return false when a nested web name is passed to it.

ObjectMethod readOnlyMirrorWeb `($theWeb) -> ($mirrorSiteName,$mirrorViewURL,$mirrorLink,$mirrorl`

If this is a mirrored web, return information about the mirror. The info is returned in a quadruple:

site name	URL	link	note

ObjectMethod getSkin `() -> $string`

Get the currently requested skin path

ObjectMethod getScriptUrl `($absolute,$script,$web,$topic,...) -> $scriptURL`

Returns the URL to a TWiki script, providing the web and topic as "path info" parameters. The result looks something like this: "http://host/twiki/bin/$script/$web/$topic".

- `...` - an arbitrary number of name,value parameter pairs that will be url-encoded and added to the url. The special parameter name '#' is reserved for specifying an anchor. e.g. `getScriptUrl('x','y','view','#'=>'XXX',a=>1,b=>2)` will give `.../view/x/y?a=1&b=2#XXX`

If $absolute is set, generates an absolute URL. $absolute is advisory only; TWiki can decide to generate

absolute URLs (for example when run from the command-line) even when relative URLs have been requested.

The default script url is taken from {ScriptUrlPath}, unless there is an exception defined for the given script in {ScriptUrlPaths}. Both {ScriptUrlPath} and {ScriptUrlPaths} may be absolute or relative URIs. If they are absolute, then they will always generate absolute URLs. if they are relative, then they will be converted to absolute when required (e.g. when running from the command line, or when generating rss). If $script is not given, absolute URLs will always be generated.

If either the web or the topic is defined, will generate a full url (including web and topic). Otherwise will generate only up to the script name. An undefined web will default to the main web name.

ObjectMethod getPubUrl `($absolute,$web,$topic,$attachment) -> $url`

Composes a pub url. If $absolute is set, returns an absolute URL. If $absolute is set, generates an absolute URL. $absolute is advisory only; TWiki can decide to generate absolute URLs (for example when run from the command-line) even when relative URLs have been requested.

$web, $topic and $attachment are optional. A partial URL path will be generated if one or all is not given.

ObjectMethod cacheIconData `($action)`

Cache icon data based on action:

- 'delete' - delete cache file
- 'read' - read cache file
- 'expire' - expire (invalidate) cache if needed
- 'save' - save cache file

ObjectMethod formatIcon `($iconName,$format,$default) -> $icon`

Format an icon based on name and format parameter. The format parameter handles these variables (with example):

- $name: Name of icon ('home')
- $type: Type of icon ('gif')
- $filename: Icon filename ('home.gif')
- $web: Web where icon is located ('TWiki')
- $topic: Topic where icon is located ('TWikiDocGraphics')
- $description: Icon description ('Home')
- $width: Width of icon ('16')
- $height: Height of icon ('16')
- $img: Full img tag of icon ('')
- $url: URL of icon ('http://example.com/pub/TWiki/TWikiDocGraphics/home.gif')
- $urlpath: URL path of icon ('/pub/TWiki/TWikiDocGraphics/home.gif')

The optional default parameter specifies the icon name in case the icon is not defined. Leave empty if you assume icon files exist in the default location.

ObjectMethod normalizeWebTopicName `($theWeb,$theTopic) -> ($theWeb,$theTopic)`

Normalize a Web.TopicName

See TWikiFuncDotPm for a full specification of the expansion (not duplicated here)

WARNING if there is no web specification (in the web or topic parameters) the web defaults to $TWiki::cfg{UsersWebName}. If there is no topic specification, or the topic is '0', the topic defaults to the web home topic name.

ClassMethod new `($loginName,$query,\%initialContext)`

Constructs a new TWiki object. Parameters are taken from the query object.

- `$loginName` is the login username (**not** the wikiname) of the user you want to be logged-in if none is available from a session or browser. Used mainly for side scripts and debugging.
- `$query` the TWiki::Request query (may be undef, in which case an empty query is used)
- `\%initialContext` - reference to a hash containing context name=value pairs to be pre-installed in the context hash

ObjectMethod renderer `()`

Get a reference to the renderer object. Done lazily because not everyone needs the renderer.

ObjectMethod attach `()`

Get a reference to the attach object. Done lazily because not everyone needs the attach.

ObjectMethod templates `()`

Get a reference to the templates object. Done lazily because not everyone needs the templates.

ObjectMethod i18n `()`

Get a reference to the i18n object. Done lazily because not everyone needs the i18ner.

ObjectMethod search `()`

Get a reference to the search object. Done lazily because not everyone needs the searcher.

ObjectMethod security `()`

Get a reference to the security object. Done lazily because not everyone needs the security.

ObjectMethod net ()

Get a reference to the net object. Done lazily because not everyone needs the net.

ObjectMethod finish ()

Break circular references.

ObjectMethod writeLog ($action,$webTopic,$extra,$user)

- $action - what happened, e.g. view, save, rename
- $wbTopic - what it happened to
- $extra - extra info, such as minor flag
- $user - user who did the saving (user id)

Write the log for an event to the logfile

ObjectMethod writeWarning ($text)

Prints date, time, and contents $text to $TWiki::cfg{WarningFileName}, typically 'warnings.txt'. Use for warnings and errors that may require admin intervention. Use this for defensive programming warnings (e.g. assertions).

ObjectMethod writeDebug ($text)

Prints date, time, and contents of $text to $TWiki::cfg{DebugFileName}, typically 'debug.txt'. Use for debugging messages.

StaticMethod applyPatternToIncludedText ($text,$pattern) -> $text

Apply a pattern on included text to extract a subset

ObjectMethod inlineAlert ($template,$def,...) -> $string

Format an error for inline inclusion in rendered output. The message string is obtained from the template 'oops'.$template, and the DEF $def is selected. The parameters (...) are used to populate %PARAM1%..%PARAMn%

StaticMethod parseSections ($text) -> ($string,$sectionlistref)

Generic parser for sections within a topic. Sections are delimited by STARTSECTION and ENDSECTION, which may be nested, overlapped or otherwise abused. The parser builds an array of sections, which is ordered by the order of the STARTSECTION within the topic. It also removes all the SECTION tags from the text, and returns the text and the array of sections.

Each section is a `TWiki::Attrs` object, which contains the attributes {type, name, start, end} where start and end are character offsets in the string **after all section tags have been removed**. All sections are required to be uniquely named; if a section is unnamed, it will be given a generated name. Sections may overlap or nest.

See test/unit/Fn_SECTION.pm for detailed testcases that round out the spec.

ObjectMethod expandVariablesOnTopicCreation `($text,$user,$web,$topic) -> $text`

- `$text` - text to expand
- `$user` - This is the user expanded in e.g. %USERNAME. Optional, defaults to logged-in user.

Expand limited set of variables during topic creation. These are variables expected in templates that must be statically expanded in new content.

- `$web` - name of web
- `$topic` - name of topic

SMELL: no plugin handler

StaticMethod entityEncode `($text,$extras) -> $encodedText`

Escape special characters to HTML numeric entities. This is **not** a generic encoding, it is tuned specifically for use in TWiki.

HTML4.0 spec: "Certain characters in HTML are reserved for use as markup and must be escaped to appear literally. The "<" character may be represented with an *entity*, **<**. Similarly, ">" is escaped as **>**, and "&" is escaped as **&**. If an attribute value contains a double quotation mark and is delimited by double quotation marks, then the quote should be escaped as **"**.

Other entities exist for special characters that cannot easily be entered with some keyboards..."

This method encodes HTML special and any non-printable ascii characters (except for \n and \r) using numeric entities.

FURTHER this method also encodes characters that are special in TWiki meta-language.

$extras is an optional param that may be used to include **additional** characters in the set of encoded characters. It should be a string containing the additional chars.

StaticMethod entityDecode `($encodedText) -> $text`

Decodes all numeric entities (e.g. {). *Does not* decode named entities such as & (use HTML::Entities for that)

StaticMethod urlEncodeAttachment `($text)`

For attachments, URL-encode specially to 'freeze' any characters >127 in the site charset (e.g. ISO-8859-1 or KOI8-R), by doing URL encoding into native charset ($siteCharset) - used when generating attachment

URLs, to enable the web server to serve attachments, including images, directly.

This encoding is required to handle the cases of:

- browsers that generate UTF-8 URLs automatically from site charset URLs - now quite common - web servers that directly serve attachments, using the site charset for filenames, and cannot convert UTF-8 URLs into site charset filenames

The aim is to prevent the browser from converting a site charset URL in the web page to a UTF-8 URL, which is the default. Hence we 'freeze' the URL into the site character set through URL encoding.

In two cases, no URL encoding is needed: For EBCDIC mainframes, we assume that site charset URLs will be translated (outbound and inbound) by the web server to/from an EBCDIC character set. For sites running in UTF-8, there's no need for TWiki to do anything since all URLs and attachment filenames are already in UTF-8.

StaticMethod urlEncode `($string) -> encodedstring`

Encode by converting characters that are illegal in URLs to their %NN equivalents. This method is used for encoding strings that must be embedded *verbatim* in URLs; it cannot be applied to URLs themselves, as it escapes reserved characters such as = and ?.

RFC 1738, Dec. '94:

```
...Only alphanumerics [0-9a-zA-Z], the special
characters $-_.+!*'(), and reserved characters used for their
reserved purposes may be used unencoded within a URL.
```

Reserved characters are $&+,/:;=?@ - these are *also* encoded by this method.

This URL-encoding handles all character encodings including ISO-8859-*, KOI8-R, EUC-* and UTF-8.

This may not handle EBCDIC properly, as it generates an EBCDIC URL-encoded URL, but mainframe web servers seem to translate this outbound before it hits browser - see CGI::Util::escape for another approach.

StaticMethod urlDecode `($string) -> decodedstring`

Reverses the encoding done in urlEncode.

StaticMethod isTrue `($value,$default) -> $boolean`

Returns 1 if `$value` is true, and 0 otherwise. "true" means set to something with a Perl true value, with the special cases that "off", "false" and "no" (case insensitive) are forced to false. Leading and trailing spaces in `$value` are ignored.

If the value is undef, then `$default` is returned. If `$default` is not specified it is taken as 0.

StaticMethod spaceOutWikiWord `($word,$sep) -> $string`

Spaces out a wiki word by inserting a string (default: one space) between each word component. With

parameter $sep any string may be used as separator between the word components; if $sep is undefined it defaults to a space.

ObjectMethod expandAllTags `(\$text,$topic,$web,$meta)`

Expands variables by replacing the variables with their values. Some example variables: %TOPIC%, %SCRIPTURL%, %WIKINAME%, etc. $web and $incs are passed in for recursive include expansion. They can safely be undef. The rules for tag expansion are:

1. Tags are expanded left to right, in the order they are encountered.
2. Tags are recursively expanded as soon as they are encountered - the algorithm is inherently single-pass
3. A tag is not "encountered" until the matching }% has been seen, by which time all tags in parameters will have been expanded
4. Tag expansions that create new tags recursively are limited to a set number of hierarchical levels of expansion

ObjectMethod enterContext `($id,$val)`

Add the context id $id into the set of active contexts. The $val can be anything you like, but should always evaluate to boolean TRUE.

An example of the use of contexts is in the use of tag expansion. The commonTagsHandler in plugins is called every time tags need to be expanded, and the context of that expansion is signalled by the expanding module using a context id. So the forms module adds the context id "form" before invoking common tags expansion.

Contexts are not just useful for tag expansion; they are also relevant when rendering.

Contexts are intended for use mainly by plugins. Core modules can use $session->inContext($id) to determine if a context is active.

ObjectMethod leaveContext `($id)`

Remove the context id $id from the set of active contexts. (see `enterContext` for more information on contexts)

ObjectMethod inContext `($id)`

Return the value for the given context id (see `enterContext` for more information on contexts)

StaticMethod registerTagHandler `($tag,$fnref)`

STATIC Add a tag handler to the function tag handlers.

- `$tag` name of the tag e.g. MYTAG
- `$fnref` Function to execute. Will be passed ($session, \%params, $web, $topic)

StaticMethod registerRESTHandler `($subject,$verb,\&fn)`

Adds a function to the dispatch table of the REST interface for a given subject. See TWikiScripts#rest for more info.

- `$subject` - The subject under which the function will be registered.
- `$verb` - The verb under which the function will be registered.
- `\&fn` - Reference to the function.

The handler function must be of the form:

```
sub handler(\%session,$subject,$verb) -> $text
```

where:

- `\%session` - a reference to the TWiki session object (may be ignored)
- `$subject` - The invoked subject (may be ignored)
- `$verb` - The invoked verb (may be ignored)

Since: TWiki::Plugins::VERSION 1.1

ObjectMethod handleCommonTags `($text,$web,$topic,$meta) -> $text`

Processes %VARIABLE%, and %TOC% syntax; also includes 'commonTagsHandler' plugin hook.

Returns the text of the topic, after file inclusion, variable substitution, table-of-contents generation, and any plugin changes from commonTagsHandler.

$meta may be undef when, for example, expanding templates, or one-off strings at a time when meta isn't available.

ObjectMethod ADDTOHEAD `($args)`

Add `$html` to the HEAD tag of the page currently being generated.

Note that TWiki variables may be used in the HEAD. They will be expanded according to normal variable expansion rules.

%ADDTOHEAD%

You can write == in a topic or template. This variable accepts the following parameters:

- `_DEFAULT` optional, id of the head block. Used to generate a comment in the output HTML.
- `text` optional, text to use for the head block. Mutually exclusive with `topic`.
- `topic` optional, full TWiki path name of a topic that contains the full text to use for the head block. Mutually exclusive with `text`. Example: `topic="TWiki.MyTopic"`.
- `requires` optional, comma-separated list of id's of other head blocks this one depends on.

`%ADDTOHEAD%` expands in-place to the empty string, unless there is an error in which case the variable expands to an error string.

Use %RENDERHEAD% to generate the sorted head tags.

%<nop}RENDERHEAD%

... should be written where you want the sorted head tags to be generated. This will normally be in a template. The variable expands to a sorted list of the head blocks added up to the point the RENDERHEAD variable is expanded. Each expanded head block is preceded by an HTML comment that records the ID of the head block.

Head blocks are sorted to satisfy all their requires constraints. The output order of blocks with no requires value is undefined. If cycles exist in the dependency order, the cycles will be broken but the resulting order of blocks in the cycle is undefined.

StaticMethod initialize ($pathInfo,$remoteUser,$topic,$url,$query) -> ($topicName,$webName,$scriptUrlPath,$userName,$datal

Return value: ($topicName, $webName, $TWiki::cfg{ScriptUrlPath}, $userName, $TWiki::cfg{DataDir})

Static method to construct a new singleton session instance. It creates a new TWiki and sets the Plugins $SESSION variable to point to it, so that TWiki::Func methods will work.

This method is **DEPRECATED** but is maintained for script compatibility.

Note that $theUrl, if specified, must be identical to $query->url()

StaticMethod readFile ($filename) -> $text

Returns the entire contents of the given file, which can be specified in any format acceptable to the Perl open() function. Fast, but inherently unsafe.

WARNING: Never, ever use this for accessing topics or attachments! Use the Store API for that. This is for global control files only, and should be used **only** if there is **absolutely no alternative**.

StaticMethod expandStandardEscapes ($str) -> $unescapedStr

Expands standard escapes used in parameter values to block evaluation. The following escapes are handled:

Escape:	Expands To:
$n or $n()	New line. Use $n() if followed by alphanumeric character, e.g. write Foo$n()Bar instead of Foo$nBar
$nop or $nop()	Is a "no operation".
$quot	Double quote (")
$percnt	Percent sign (%)
$dollar	Dollar sign ($)
$lt	Less than sign (<)
$gt	Greater than sign (>)

This topic: TWiki > TWikiDotPm
Topic revision: r6 - 2011-04-12 - TWikiContributor

Note: Please contribute updates to this topic on TWiki.org at TWiki:TWiki.TWikiDotPm.