---+ Package `TWiki::Store`

This module hosts the generic storage backend. This module provides the interface layer between the "real" store provider - which is hidden behind a handler - and the rest of the system. it is responsible for checking for topic existance, access permissions, and all the other general admin tasks that are common to all store implementations.

This module knows nothing about how the data is actually *stored* - that knowledge is entirely encapsulated in the handlers.

The general contract for methods in the class requires that errors are signalled using exceptions. TWiki::AccessControlException is used for access control exceptions, and Error::Simple for all other types of error.

ClassMethod new `($session)`

Construct a Store module, linking in the chosen sub-implementation.

ObjectMethod finish `()`

Break circular references.

ObjectMethod readTopic `($user,$web,$topic,$version) -> ($metaObject,$text)`

Reads the given version of a topic and it's meta-data. If the version is undef, then read the most recent version. The version number must be an integer, or undef for the latest version.

if $user is defined, view permission will be required for the topic read to be successful. Access control violations are flagged by a TWiki::AccessControlException. Permissions are checked for the user name passed in.

If the topic contains a web specification (is of the form Web.Topic) the web specification will override whatever is passed in $web.

The metadata and topic text are returned separately, with the metadata in a TWiki::Meta object. (The topic text is, as usual, just a string.)

ObjectMethod _findAttachments `($session,$web,$topic,$knownAttachments) -> @attachmentsFoundInPub`

Synchronise the attachment list with what's actually on disk Returns an ARRAY of FILEATTACHMENTs. These can be put in the new meta using meta->put('FILEATTACHMENTS', $tree)

This function is only called when the AutoAttachPubFiles configuration option is set.

IDEA On Windows machines where the underlying filesystem can store arbitary meta data against files, this might replace/fulfil the COMMENT purpose

TODO consider logging when things are added to metadata

ObjectMethod readTopicRaw `($user,$web,$topic,$version) -> $topicText`

Reads the given version of a topic, without separating out any embedded meta-data. If the version is undef, then read the most recent version. The version number must be an integer or undef.

If $user is defined, view permission will be required for the topic read to be successful. Access control violations are flagged by a TWiki::AccessControlException. Permissions are checked for the user name passed in.

If the topic contains a web specification (is of the form Web.Topic) the web specification will override whatever is passed in $web.

SMELL: DO NOT CALL THIS METHOD UNLESS YOU HAVE NO CHOICE. This method breaks encapsulation of the store, as it assumes meta is stored embedded in the text. Other implementors of store will be forced to insert meta-data to ensure correct operation of View raw=debug and the 'repRev' mode of Edit.

$web and $topic *must* be untainted.

ObjectMethod moveAttachment `($oldWeb,$oldTopic,$oldAttachment,$newWeb,$newTopic`

Move an attachment from one topic to another.

The caller to this routine should check that all topics are valid.

All parameters must be defined, and must be untainted.

ObjectMethod getAttachmentStream `($user,$web,$topic,$attName) -> \*STREAM`

- `$user` - the user doing the reading, or undef if no access checks
- `$web` - The web
- `$topic` - The topic
- `$attName` - Name of the attachment

Open a standard input stream from an attachment.

If $user is defined, view permission will be required for the topic read to be successful. Access control violations and errors will cause exceptions to be thrown.

Permissions are checked for the user name passed in.

ObjectMethod getAttachmentList `($web,$topic)`

returns @($attachmentName => [stat]) for any given web, topic

ObjectMethod attachmentExists `($web,$topic,$att) -> $boolean`

Determine if the attachment already exists on the given topic

ObjectMethod _removeAutoAttachmentsFromMeta

This is where we are going to remove from meta any entry that is marked as an automatic attachment.

ObjectMethod moveTopic `($oldWeb,$oldTopic,$newWeb,$newTopic,$user)`

All parameters must be defined and must be untainted.

ObjectMethod moveWeb `($oldWeb,$newWeb,$user)`

Move a web.

All parrameters must be defined and must be untainted.

ObjectMethod readAttachment `($user,$web,$topic,$attachment,$theRev) -> $text`

Read the given version of an attachment, returning the content.

View permission on the topic is required for the read to be successful. Access control violations are flagged by a TWiki::AccessControlException. Permissions are checked for the user passed in.

If $theRev is not given, the most recent rev is assumed.

ObjectMethod getRevisionNumber `($web,$topic,$attachment) -> $integer`

Get the revision number of the most recent revision. Returns the integer revision number or '' if the topic doesn't exist.

WORKS FOR ATTACHMENTS AS WELL AS TOPICS

ObjectMethod getWorkArea `($key) -> $directorypath`

Gets a private directory uniquely identified by $key. The directory is intended as a work area for plugins. The directory will exist.

ObjectMethod getRevisionDiff `($user,$web,$topic,$rev1,$rev2,$contextLines)`

-> \@diffArray

Return reference to an array of [diffType, $right, $left]

- `$user` - the user id, or undef to suppress access control checks
- `$web` - the web
- `$topic` - the topic
- `$rev1` Integer revision number
- `$rev2` Integer revision number
- `$contextLines` - number of lines of context required

ObjectMethod getRevisionInfo ($web,$topic,$rev,$attachment) -> ($date,$user,$rev,$comment)

Get revision info of a topic.

- `$web` Web name, optional, e.g. `'Main'`
- `$topic` Topic name, required, e.g. `'TokyoOffice'`
- `$rev` revision number. If 0, undef, or out-of-range, will get info about the most recent revision.
- `$attachment` attachment filename; undef for a topic

Return list with: (last update date, last user id, =

$date	in epochSec
$user	user **object**
$rev	the revision number
$comment	WHAT COMMENT?

e.g. =(1234561, 'phoeny', 5, 'no comment')

NOTE NOTE NOTE if you are working within the TWiki code DO NOT USE THIS FUNCTION FOR GETTING REVISION INFO OF TOPICS - use TWiki::Meta::getRevisionInfo instead. This is essential to allow clean transition to a topic object model later, and avoids the risk of confusion coming from meta and Store revision information being out of step. (it's OK to use it for attachments)

StaticMethod dataEncode ($uncoded) -> $coded

Encode meta-data fields, escaping out selected characters. The encoding is chosen to avoid problems with parsing the attribute values, while minimising the number of characters encoded so searches can still work (fairly) sensibly.

The encoding has to be exported because TWiki (and plugins) use encoded field data in other places e.g. RDiff, mainly as a shorthand for the properly parsed meta object. Some day we may be able to eliminate that....

StaticMethod dataDecode ($encoded) -> $decoded

Decode escapes in a string that was encoded using dataEncode

The encoding has to be exported because TWiki (and plugins) use encoded field data in other places e.g. RDiff, mainly as a shorthand for the properly parsed meta object. Some day we may be able to eliminate

that....

ObjectMethod saveTopic
($user,$web,$topic,$text,$meta,$options)

- $user - user doing the saving (object)
- $web - web for topic
- $topic - topic to atach to
- $text - topic text
- $meta - topic meta-data
- $options - Ref to hash of options

$options may include:

dontlog	don't log this change in twiki log
hide	if the attachment is to be hidden in normal topic view
comment	comment for save
file	Temporary file name to upload
minor	True if this is a minor change (used in log)
savecmd	Save command
forcedate	grr
unlock	

Save a new revision of the topic, calling plugins handlers as appropriate.

ObjectMethod saveAttachment
($web,$topic,$attachment,$user,$opts)

- $user - user doing the saving
- $web - web for topic
- $topic - topic to atach to
- $attachment - name of the attachment
- $opts - Ref to hash of options

$opts may include:

dontlog	don't log this change in twiki log
comment	comment for save
hide	if the attachment is to be hidden in normal topic view
stream	Stream of file to upload
file	Name of a file to use for the attachment data. ignored is stream is set.
filepath	Client path to file
filesize	Size of uploaded data
filedate	Date
tmpFilename	Pathname of the server file the stream is attached to. Required if stream is set.

Saves a new revision of the attachment, invoking plugin handlers as appropriate.

If file is not set, this is a properties-only save.

ObjectMethod repRev `($user,$web,$topic,$text,$meta,$options)`

Replace last (top) revision with different text.

Parameters and return value as saveTopic, except

- `$options` - as for saveTopic, with the extra option:
  - `timetravel` - if we want to force the deposited revision to look as much like the revision specified in `$rev` as possible.
  - `operation` - set to the name of the operation performing the save. This is used only in the log, and is normally `cmd` or `save`. It defaults to `save`.

Used to try to avoid the deposition of 'unecessary' revisions, for example where a user quickly goes back and fixes a spelling error.

Also provided as a means for administrators to rewrite history (timetravel).

It is up to the store implementation if this is different to a normal save or not.

ObjectMethod delRev `($user,$web,$topic,$text,$meta,$options)`

Parameters and return value as saveTopic.

Provided as a means for administrators to rewrite history.

Delete last entry in repository, restoring the previous revision.

It is up to the store implementation whether this actually does delete a revision or not; some implementations will simply promote the previous revision up to the head.

ObjectMethod lockTopic `($web,$topic)`

Grab a topic lock on the given topic. A topic lock will cause other processes that also try to claim a lock to block. A lock has a maximum lifetime of 2 minutes, so operations on a locked topic must be completed within that time. You cannot rely on the lock timeout clearing the lock, though; that should always be done by calling unlockTopic. The best thing to do is to guard the locked section with a try..finally clause. See man Error for more info.

Topic locks are used to make store operations atomic. They are *note* the locks used when a topic is edited; those are Leases (see `getLease`)

ObjectMethod unlockTopic `($user,$web,$topic)`

Release the topic lock on the given topic. A topic lock will cause other processes that also try to claim a lock to block. It is important to release a topic lock after a guard section is complete. This should normally be done in a 'finally' block. See man Error for more info.

Topic locks are used to make store operations atomic. They are *note* the locks used when a topic is edited; those are Leases (see `getLease`)

ObjectMethod webExists `($web) -> $boolean`

Test if web exists

- `$web` - Web name, required, e.g. `'Sandbox'`

A web *has* to have a preferences topic to be a web.

ObjectMethod topicExists `($web,$topic) -> $boolean`

Test if topic exists

- `$web` - Web name, optional, e.g. `'Main'`
- `$topic` - Topic name, required, e.g. `'TokyoOffice'`, or `"Main.TokyoOffice"`

Warning: topicExists does not call ($web, $topic) = $this->{session}->normalizeWebTopicName($web, $topic); for you (it'd make TWiki even slower) so make sure you do so.

ObjectMethod getTopicParent `($web,$topic) -> $string`

Get the name of the topic parent. Needs to be fast because of use by Render.pm.

ObjectMethod getTopicLatestRevTime `($web,$topic) -> $epochSecs`

Get an approximate rev time for the latest rev of the topic. This method is used to optimise searching. Needs to be as fast as possible.

ObjectMethod eachChange `($web,$time) -> $iterator`

Get an iterator over the list of all the changes in the given web between `$time` and now. $time is a time in seconds since 1st Jan 1970, and is not guaranteed to return any changes that occurred before (now - {Store}{RememberChangesFor}). Changes are returned in most-recent-first order.

ObjectMethod getTopicNames `($web) -> @topics`

Get list of all topics in a web

- `$web` - Web name, required, e.g. `'Sandbox'`

Return a topic list, e.g. `( 'WebChanges', 'WebHome', 'WebIndex', 'WebNotify' )`

ObjectMethod getListOfWebs `($filter) -> @webNames`

Gets a list of webs, filtered according to the spec in the $filter, which may include one of:

1. 'user' (for only user webs)
2. 'template' (for only template webs)

$filter may also contain the word 'public' which will further filter webs on whether NOSEARCHALL is specified for them or not. 'allowed' filters out webs that the user is denied access to by a *WEBVIEW.

If $TWiki::cfg{EnableHierarchicalWebs} is set, will also list sub-webs recursively.

ObjectMethod createWeb `($user,$newWeb,$baseWeb,$opts)`

$newWeb is the name of the new web.

$baseWeb is the name of an existing web (a template web). If the base web is a system web, all topics in it will be copied into the new web. If it is a normal web, only topics starting with 'Web' will be copied. If no base web is specified, an empty web (with no topics) will be created. If it is specified but does not exist, an error will be thrown.

$opts is a ref to a hash that contains settings to be modified in the web preferences topic in the new web.

ObjectMethod removeWeb `($user,$web)`

- `$user` - user doing the removing (for the history)
- `$web` - web being removed

Destroy a web, utterly. Removed the data and attachments in the web.

Use with great care!

The web must be a known web to be removed this way.

ObjectMethod getDebugText `($meta,$text) -> $text`

Generate a debug text form of the text/meta, for use in debug displays, by annotating the text with meta informtion.

ObjectMethod cleanUpRevID `($rev) -> $integer`

Cleans up (maps) a user-supplied revision ID and converts it to an integer number that can be incremented to create a new revision number.

This method should be used to sanitise user-provided revision IDs.

ObjectMethod copyTopic `($user,$fromweb,$fromtopic,$toweb,$totopic)`

Copy a topic and all it's attendant data from one web to another.

SMELL: Does not fix up meta-data!

ObjectMethod searchMetaData `($params) -> $text`

Search meta-data associated with topics. Parameters are passed in the $params hash, which may contain:

`type`	`topicmoved`, `parent` or `field`
`topic`	topic to search for, for `topicmoved` and `parent`
`name`	form field to search, for `field` type searches. May be a regex.
`value`	form field value. May be a regex.
`title`	Title prepended to the returned search results
`default`	defualt value if there are no results
`web`	web to search in, default is all webs
`format`	string for custom formatting results

The idea is that people can search for meta-data values without having to be aware of how or where meta-data is stored.

SMELL: should be replaced with a proper SQL-like search, c.f. DBCacheContrib.

ObjectMethod searchInWebMetaData `($query,$web,\@topics) -> \%matches`

Search for a meta-data expression in the content of a web. `$query` must be a `TWiki::Query` object.

Returns a reference to a hash that maps the names of topics that all matched to the result of the query expression (e.g. if the query expression is 'TOPICPARENT.name' then you will get back a hash that maps topic names to their parent.

ObjectMethod searchInWebContent `($searchString,$web,\@topics,\%options) -> \%map`

Search for a string in the content of a web. The search must be over all content and all formatted meta-data, though the latter search type is deprecated (use searchMetaData instead).

- `$searchString` - the search string, in egrep format if regex
- `$web` - The web to search in
- `\@topics` - reference to a list of topics to search
- `\%options` - reference to an options hash

The `\%options` hash may contain the following options:

- `type` - if `regex` will perform a egrep-syntax RE search (default '')
- `casesensitive` - false to ignore case (defaulkt true)
- `files_without_match` - true to return files only (default false)

The return value is a reference to a hash which maps each matching topic name to a list of the lines in that topic that matched the search, as would be returned by 'grep'. If `files_without_match` is specified, it will return on the first match in each topic (i.e. it will return only one match per topic, and will not return matching lines).

ObjectMethod getRevisionAtTime `($web,$topic,$time) -> $rev`

- `$web` - web for topic
- `$topic` - topic

- `$time` - time (in epoch secs) for the rev

Get the revision number of a topic at a specific time. Returns a single-digit rev number or undef if it couldn't be determined (either because the topic isn't that old, or there was a problem)

ObjectMethod getLease `($web,$topic) -> $lease`

- `$web` - web for topic
- `$topic` - topic

If there is an lease on the topic, return the lease, otherwise undef. A lease is a block of meta-information about a topic that can be recovered (this is a hash containing `user`, `taken` and `expires`). Leases are taken out when a topic is edited. Only one lease can be active on a topic at a time. Leases are used to warn if another user is already editing a topic.

ObjectMethod setLease `($web,$topic,$user,$length)`

Take out an lease on the given topic for this user for $length seconds.

See `getLease` for more details about Leases.

ObjectMethod clearLease `($web,$topic)`

Cancel the current lease.

See `getLease` for more details about Leases.

ObjectMethod removeSpuriousLeases `($web)`

Remove leases that are not related to a topic. These can get left behind in some store implementations when a topic is created, but never saved.

This topic: TWiki > TWikiStoreDotPm
Topic revision: r8 - 2010-05-29 - TWikiContributor

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