


---+!! TWiki Source Code Packages

 This documentation is automatically generated from the pod, so it always matches the running code

[[CGISessionDotPm]]

This package doesn't smell

[[CGISessionDriverDBIDotPm]]

This package doesn't smell

[[CGISessionDriverDb_fileDotPm]]

This package doesn't smell

[[CGISessionDriverDotPm]]

This package doesn't smell

[[CGISessionDriverFileDotPm]]

This package doesn't smell

[[CGISessionDriverMysqlDotPm]]

This package doesn't smell

[[CGISessionDriverPostgresqlDotPm]]

This package doesn't smell

[[CGISessionDriverSqliteDotPm]]

This package doesn't smell

[[CGISessionErrorHandlerDotPm]]

This package doesn't smell

[[CGISessionIDIncrDotPm]]

This package doesn't smell

[[CGISessionIDMd5DotPm]]

This package doesn't smell

[[CGISessionSerializeDefaultDotPm]]

This package doesn't smell

[[CGISessionSerializeFreezethawDotPm]]

This package doesn't smell

[[CGISessionSerializeJsonDotPm]]

This package doesn't smell

[[CGISessionSerializeStorableDotPm]]

This package doesn't smell

[[CGISessionSerializeYamlDotPm]]

This package doesn't smell

[[CGISessionTutorialDotPm]]

This package doesn't smell

[[MonitorDotPm]]

This package doesn't smell

TWiki::AccessControlException

Exception used raise an access control violation. This exception has the following fields:

- `web` - the web which was being accessed
- `topic` - the topic being accessed (if any)
- `user` - canonical username of the person doing the accessing. Use the methods of the `TWiki::Users` class to get more information about the user.
- `mode` - the access mode e.g. `CHANGE`, `VIEW` etc
- `reason` a text string giving the reason for the refusal.

The exception may be thrown by plugins. If a plugin throws the exception, it will normally be caught and the browser redirected to a login screen (if the user is not logged in) or reported (if they are and just don't have access).

This package doesn't smell

TWiki::Access

A singleton object of this class manages the access control database.

This package doesn't smell

TWiki::Aggregatelterator

combine multiple iterators

This package doesn't smell

TWiki::Attach

A singleton object of this class is used to deal with attachments to topics.

This package doesn't smell

TWiki::Attrs

Class of attribute sets, designed for parsing and storing attribute values from a TWiki tag e.g. %TAG{ "joe" fred="bad" joe="mad" }%

An attribute set is a hash containing an entry for each parameter. The default parameter (unnamed quoted string) is named `_DEFAULT` in the hash.

Attributes declared later in the string will override those of the same name defined earlier. The one exception to this is the `_DEFAULT` key, where the *first* instance is always taken.

As well as the default TWiki syntax (parameter values double-quoted) this class also parses single-quoted values, unquoted spaceless values, spaces around the `=`, and commas as well as spaces separating values. The extended syntax has to be enabled by passing the `$friendly` parameter to `new`.

This package doesn't smell

TWiki::Compatibility

Support for compatibility with old TWiki versions. Packaged separately because 99.999999% of the time this won't be needed.

This package has smell factor of 2

TWiki::Configure::Load

Purpose

This module consists of just a single subroutine `readConfig`. It allows to safely modify configuration variables *for one single run* without affecting normal TWiki operation.

This package doesn't smell

[[TWikiConfigureUlsEXTENDDotPm]]

This package has smell factor of 2

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}`

This package has smell factor of 38

[[TWikiEngineDotPm]]

This package has smell factor of 1

TWiki::EngineException

Exception used to raise an engine related error. This exception has the following fields:

- `status` - status code to send to client
- `reason` a text string giving the reason for the refusal.

This package doesn't smell

TWiki::Form

Object representing a single form definition.

Form definitions are mainly used to control rendering of a form for editing, though there is some application login there that handles transferring values between edits and saves.

A form definition consists of a `TWiki::Form` object, which has a list of field definitions. Each field definition is an object of a type derived from `TWiki::Form::FieldDefinition`. These objects are responsible for the actual syntax and semantics of the field type. Form definitions are parsed from TWiki tables, and the types are mapped by name to a class declared in `TWiki::Form::*` - for example, the `text` type is mapped to `TWiki::Form::Text` and the `checkbox` type to `TWiki::Form::Checkbox`.

The `TWiki::Form::FieldDefinition` class declares default behaviours for types that accept a single value in their definitions. The `TWiki::Form::ListFieldDefinition` extends this for types that have lists of possible values.

This package has smell factor of **4**

TWiki::Form::FieldDefinition

Base class of all field definition classes.

Type-specific classes are derived from this class to define specific per-type behaviours. This class also provides default behaviours for when a specific type cannot be loaded.

This package doesn't smell

TWiki::Form::ListFieldDefinition

Form field definitions that accept lists of values in the field definition. This is different to being multi-valued, which means the field type can **store** multiple values.

This package has smell factor of **1**

[[TWikiFormSelectDotPm]]

This package doesn't smell

TWiki::Func

Official list of stable TWiki functions for Plugin developers

This module defines official functions that TWiki plugins can use to interact with the TWiki engine and content.

Refer to `EmptyPlugin` and `lib/TWiki/Plugins/EmptyPlugin.pm` for a template plugin and documentation on how to write a plugin.

Plugins should **only** use functions published in this module. If you use functions in other TWiki libraries you might create a security hole and you will probably need to change your plugin when you upgrade TWiki.

Deprecated functions will still work in older code, though they should *not* be called in new plugins and should be replaced in older plugins as soon as possible.

The version of the TWiki::Func module is defined by the VERSION number of the TWiki::Plugins module, currently 6.00. This can be shown by the %PLUGINVERSION% TWiki variable, and accessed in code using \$TWiki::Plugins::VERSION. The 'Since' field in the function documentation refers to \$TWiki::Plugins::VERSION.

Notes on use of \$TWiki::Plugins::VERSION (from 1.2 forwards):

- If the **major** version (e.g. 1.) is the same then any plugin coded to use any **earlier** revision of the 1. API will still work. No function has been removed from the interface, nor has any API published in that version changed in such a way as to **require** plugins to be recoded.
- If the **minor** version (e.g. 1.1) is incremented there may be changes in the API that may help improve the coding of some plugins - for example, new interfaces giving access to previously hidden core functions. In addition, **deprecation** of functions in the interface trigger a minor version increment. Note that deprecated functions are not *removed*, they are merely frozen, and plugin authors are recommended to stop using them.
- Any additional digits in the version number relate to minor changes, such as the addition of parameters to the existing functions, or addition of utility functions that are unlikely to require significant changes to existing plugins.
- TWiki::Plugins::VERSION also applies to the plugin handlers. The handlers are documented in the EmptyPlugin, and that module indicates what version of TWiki::Plugins::VERSION it relates to.

A full history of the changes to this API can be found at the end of this topic.

This package has smell factor of 1

TWiki::I18N

Support for strings translation and language detection.

This package has smell factor of 2

TWiki::I18N::Extract

Support translatable strings extraction from TWiki topics and templates. Depends on Locale::Maketext::Extract (part of CPAN::Locale::Maketext::Lexicon).

This package has smell factor of 1

TWiki::If::Node

Node class for the result of an If statement parse

This package doesn't smell

TWiki::If::OP_isweb

This package doesn't smell

TWiki::If::Parser

Support for the conditions in %IF{ } statements.

This package doesn't smell

TWiki::Infix::Error

Class of errors used with TWiki::Infix::Parser

This package doesn't smell

TWiki::Infix::Node

Base class for node types generated by Infix::Parser. You don't **have** to use it, but it may be useful.

This package doesn't smell

TWiki::Infix::Parser

A simple stack-based parser that parses infix expressions with nonary, unary and binary operators specified using an operator table.

Escapes are supported in strings, using backslash.

This package doesn't smell

TWiki::LineIterator

Iterator over the lines in a file

This package doesn't smell

TWiki::ListIterator

Iterator over a list

This package doesn't smell

TWiki::LoginManager::ApacheLogin

This is login manager that you can specify in the security setup section of configure. It instructs TWiki to cooperate with your web server (typically Apache) to require authentication information (username & password) from users. It requires that you configure your web server to demand authentication for scripts named "login" and anything ending in "auth". The latter should be symlinks to existing scripts; e.g.,
viewauth -> view, editauth -> edit, and so on.

See also TWikiUserAuthentication.

Subclass of TWiki::LoginManager; see that class for documentation of the methods of this class.

This package has smell factor of **1**

TWiki::LoginManager

The package is also a Factory for login managers and also the base class for all login managers.

On it's own, an object of this class is used when you specify 'none' in the security setup section of configure. When it is used, logins are not supported. If you want to authenticate users then you should consider TemplateLogin or ApacheLogin, which are subclasses of this class.

If you are building a new login manager, then you should write a new subclass of this class, implementing the methods marked as **VIRTUAL**. There are already examples in the `lib/TWiki/LoginManager` directory.

The class has extensive tracing, which is enabled by `$TWiki::cfg{Trace}{LoginManager.pm}`. The tracing is done in such a way as to let the perl optimiser optimise out the trace function as a no-op if tracing is disabled.

Here's an overview of how it works:

Early in `TWiki::new`, the login manager is created. The creation of the login manager does two things:

1. If sessions are in use, it loads `CGI::Session` but doesn't initialise the session yet.
2. Creates the login manager object

Slightly later in `TWiki::new`, `loginManager->loadSession` is called.

1. Calls `loginManager->getUser` to get the username **before** the session is created
 - ◆ `TWiki::LoginManager::ApacheLogin` looks at `REMOTE_USER` (only for authenticated scripts)
 - ◆ `TWiki::LoginManager::TemplateLogin` just returns `undef`
2. reads the `TWIKISID` cookie to get the `SID` (or the `TWIKISID` parameters in the CGI query if cookies aren't available, or `IP2SID` mapping if that's enabled).
3. Creates the `CGI::Session` object, and the session is thereby read.
4. If the username still isn't known, reads it from the cookie. Thus `TWiki::LoginManager::ApacheLogin` overrides the cookie using `REMOTE_USER`, and `TWiki::LoginManager::TemplateLogin` **always** uses the session.

Later again in `TWiki::new`, plugins are given a chance to **override** the username found from the `loginManager`.

The last step in `TWiki::new` is to find the user, using whatever user mapping manager is in place.

ObjectData twiki

The TWiki object this login manager is attached to.

This package has smell factor of **8**

[[TWikiLoginManagerSessionDotPm]]

This package doesn't smell

TWiki::LoginManager::TemplateLogin

This is a login manager that you can specify in the security setup section of configure. It provides users with a template-based form to enter usernames and passwords, and works with the PasswordManager that you specify to verify those passwords.

Subclass of TWiki::LoginManager; see that class for documentation of the methods of this class.

This package has smell factor of **2**

TWiki::Merge

Support for merging strings

This package has smell factor of **1**

TWiki::Meta

All TWiki topics have **data** (text) and **meta-data** (information about the topic). Meta-data includes information such as file attachments, form fields, topic parentage etc. When TWiki loads a topic from the store, it represents the meta-data in the topic using an object of this class.

A meta-data object is a hash of different types of meta-data (keyed on the type, such as 'FIELD' and 'TOPICINFO').

Each entry in the hash is an array, where each entry in the array contains another hash of the key=value pairs, corresponding to a single meta-datum.

If there may be multiple entries of the same top-level type (i.e. for FIELD and FILEATTACHMENT) then the array has multiple entries. These types are referred to as "keyed" types. The array entries are keyed with the attribute 'name' which must be in each entry in the array.

For unkeyed types, the array has only one entry.

Pictorially,

- TOPICINFO
 - ◆ author => '...'
 - ◆ date => '...'
 - ◆ ...
- FILEATTACHMENT
 - ◆ [0] -> { name => '...' ... }
 - ◆ [1] -> { name => '...' ... }
- FIELD
 - ◆ [0] -> { name => '...' ... }
 - ◆ [1] -> { name => '...' ... }

As well as the meta-data, the object also stores the web name, topic name and remaining text after meta-data extraction.

This package has smell factor of **2**

TWiki::Net

Object that brokers access to network resources.

This package has smell factor of **3**

TWiki::Net::HTTPResponse

Fakeup of HTTP::Response for use when LWP is not available. Only implements a small subset of the HTTP::Response methods:

code()
message()
header(\$field)
content()
is_error()
is_redirect()

See the documentation of HTTP::Response for information about the methods.

This package doesn't smell

TWiki::OopsException

Exception used to raise a request to redirect to an Oops URL.

An OopsException thrown anywhere in the code will redirect the browser to a url based on the `oops` script. `oops` requires the name of an oops template file from the `templates` directory. This file will be expanded and the parameter values passed to the exception instantiated. The result will be shown in the browser.

Plugins may throw TWiki::OopsException. For example:

```
use Error;

...

throw TWiki::OopsException( 'bathplugin',
                           def => 'toestuck',
                           web => $web,
                           topic => $topic,
                           params => [ 'bigtoe', 'hot tap' ] );
```

This package doesn't smell

```
----+ [[TWikiPluginDotPm]]
```

This package has smell factor of ***2***

```
----+ [[TWikiPluginsDotPm][TWiki::Plugins]]
```

This module defines the singleton object that handles Plugins loading, initialization and execution.

This class uses Chain of Responsibility (GOF) pattern to dispatch handler calls to registered plugins.

This package doesn't smell

```
---++ [[TWikiPluralsDotPm][TWiki::Plurals]]
```

Handle conversion of plural topic names to singular form.

This package doesn't smell

```
---++ [[TWikiPrefsDotPm][TWiki::Prefs]]
```

The Prefs class is a singleton that implements management of preferences. It uses a stack of TWiki::Prefs::PrefsCache objects to store the preferences for global, web, user and topic contexts, and provides the means to look up preferences in these.

Preferences from different places stack on top of each other, so there are global preferences, then site, then web (and subweb and subsubweb), then topic, included topic and so on. Each level of the stack is tagged with a type identifier.

The module also maintains a separate of the preferences found in every topic and web it reads. This supports the lookup of preferences for webs and topics that are not on the stack, and must not be chained in (you can't allow a user to override protections from their home topic!)

This package doesn't smell

```
---++ [[TWikiPrefsParserDotPm][TWiki::Prefs::Parser]]
```

This Prefs-internal class is used to parse * Set and * Local statements from arbitrary text, and extract settings from meta objects. It is used by TopicPrefs to parse preference settings from topics.

This class does no validation or duplicate-checking on the settings; it simply returns the recognized settings in the order it sees them in.

This package has smell factor of *1*

```
---++ [[TWikiPrefsPrefsCacheDotPm][TWiki::Prefs::PrefsCache]]
```

The PrefsCache package holds a cache of topics that have been read in, using the TopicPrefs class. These functions manage that cache.

We maintain 2 hashes of values:

- * {locals} Contains all locals at this level. Locals are values that only apply when the current topic is the topic where the local is defined. The variable names are decorated with the locality where they apply.
- * {values} contains all sets, locals, and all values inherited from the parent level

As each cache level is built, the values are copied down from the parent cache level. This sounds monstrously inefficient, but in fact perl does this a lot better than doing a multi-level lookup when a value is referenced. This is especially important when many prefs lookups may be done in a session, for example when searching.

SourceCode < TWiki < TWiki

This package doesn't smell

```
---++ [[TWikiQueryHoistREsDotPm][TWiki::Query::HoistREs]]
```

Static functions to extract regular expressions from queries. The REs can be used in caching stores that use the TWiki standard inline meta-data representation to pre-filter topic lists for more efficient query matching.

See =Store/RcsFile.pm= for an example of usage.

This package doesn't smell

```
---++ [[TWikiQueryNodeDotPm][TWiki::Query]]
```

A Query object is a representation of a query over the TWiki database.

Fields are given by name, and values by strings or numbers. Strings should always be surrounded by double quotes.

See TWiki.QuerySearch for details of the query language. At the time of writing only a subset of the entire query language is supported, for use in searching.

A query object implements the =evaluate= method as its general contract with the rest of the world. This method does a "hard work" evaluation of the parser tree. Of course, smarter Store implementations should be able to do it better....

This package has smell factor of *2*

```
---++ [[TWikiQueryParserDotPm][TWiki::Query::Parser]]
```

Parser for queries

This package doesn't smell

```
---++ [[TWikiRenderDotPm][TWiki::Render]]
```

This module provides most of the actual HTML rendering code in TWiki.

This package has smell factor of *22*

```
---++ [[TWikiRequestDotPm][[]]]
```

This package has smell factor of *1*

```
---++ [[TWikiResponseDotPm][[]]]
```

This package doesn't smell

```
---++ [[TWikiSandboxDotPm][TWiki::Sandbox]]
```

This object provides an interface to the outside world. All calls to system functions, or handling of file names, should be brokered by this object.

NOTE: TWiki creates a singleton sandbox that is *shared* by all TWiki runs under a single mod_perl instance. If any TWiki run modifies the

SourceCode < TWiki < TWiki

sandbox, that modification will carry over in to subsequent runs.
Be very, very careful!

This package has smell factor of *4*

```
----+ [[TWikiSearchDotPm][TWiki::Search]]
```

This module implements all the search functionality.

This package has smell factor of *13*

```
----+ [[TWikiStoreDotPm][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 existence, 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.

This package has smell factor of *15*

```
----+ [[TWikiStoreQueryAlgorithmsBruteForceDotPm][TWiki::Store::QueryAlgorithms::BruteForce]]
```

Default brute-force query algorithm

Has some basic optimisation: it hoists regular expressions out of the query to use with `grep`, so we can narrow down the set of topics that we have to evaluate the query on.

Not sure exactly where the breakpoint is between the costs of hoisting and the advantages of hoisting. Benchmarks suggest that it's around 6 topics, though this may vary depending on disk speed and memory size. It also depends on the complexity of the query.

This package doesn't smell

```
----+ [[TWikiStoreRcsFileDotPm][TWiki::Store::RcsFile]]
```

This class is `PACKAGE PRIVATE` to `Store`, and should never be used from anywhere else. It is the base class of implementations of stores that manipulate RCS format files.

The general contract of the methods on this class and its subclasses calls for errors to be signalled by `Error::Simple` exceptions.

Refer to `Store.pm` for models of usage.

This package has smell factor of *11*

SourceCode < TWiki < TWiki

```
---++ [[TWikiStoreRcsLiteDotPm][TWiki::Store::RcsLite]]
```

This package does not publish any methods. It implements the virtual methods of the [[TWikiStoreRcsFileDotPm][TWiki::Store::RcsFile]] superclass.

Simple replacement for RCS. Doesn't support:

- * branches
- * locking

Neither of which are used (or needed) by TWiki.

This module doesn't know anything about the content of the topic

There is one of these object for each file stored under RcsLite.

This object is PACKAGE PRIVATE to Store, and should NEVER be used from anywhere else.

FIXME:

- * need to tidy up dealing with \n for differences
- * still have difficulty on line ending at end of sequences, consequence of doing a line based

```
---++ File format
```

<verbatim>

```
rcstext    ::= admin {delta}* desc {deltatext}*
admin      ::= head {num};
              { branch {num}; }
              access {id}*;
              symbols {sym : num}*;
              locks {id : num}*; {strict ;}
              { comment {string}; }
              { expand {string}; }
              { newphrase }*
delta      ::= num
              date num;
              author id;
              state {id};
              branches {num}*;
              next {num};
              { newphrase }*
desc       ::= desc string
deltatext  ::= num
              log string
              { newphrase }*
              text string
num        ::= {digit | .}+
digit      ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
id         ::= {num} idchar {idchar | num }*
sym        ::= {digit}* idchar {idchar | digit }*
idchar     ::= any visible graphic character except special
special    ::= $ | , | . | : | ; | @
string     ::= @{any character, with @ doubled}*@
newphrase  ::= id word* ;
word       ::= id | num | string | :
```

</verbatim>

Identifiers are case sensitive. Keywords are in lower case only. The sets of keywords and identifiers can overlap. In most environments RCS uses the ISO 8859/1 encoding: visible graphic characters are codes 041-176 and 240-377, and white space characters are codes 010-015 and 040.

Dates, which appear after the date keyword, are of the form Y.mm.dd.hh.mm.ss, where Y is the year, mm the month (01-12), dd the day (01-31), hh the hour (00-23), mm the minute (00-59), and ss the second (00-60). Y contains just the last two digits of the year for years from 1900 through 1999, and all the digits of years thereafter. Dates use the Gregorian calendar; times use UTC.

SourceCode < TWiki < TWiki

The newphrase productions in the grammar are reserved for future extensions to the format of RCS files. No newphrase will begin with any keyword already in use.

Revisions consist of a sequence of 'a' and 'd' edits that need to be applied to rev N+1 to get rev N. Each edit has an offset (number of lines from start) and length (number of lines). For 'a', the edit is followed by length lines (the lines to be inserted in the text). For example:

```
d1 3      means "delete three lines starting with line 1
a4 2      means "insert two lines at line 4"
xxxxxx    is the new line 4
yyyyyy    is the new line 5
```

This package has smell factor of *2*

```
---++ [[TWikiStoreRcsWrapDotPm][TWiki::Store::RcsWrap]]
```

This package does not publish any methods. It implements the virtual methods of the [[TWikiStoreRcsFileDotPm][TWiki::Store::RcsFile]] superclass.

Wrapper around the RCS commands required by TWiki.
There is one of these object for each file stored under RCS.

This package has smell factor of *3*

```
---++ [[TWikiStoreSearchAlgorithmsForkingDotPm][TWiki::Store::SearchAlgorithms::Forking]]
```

Forking implementation of the RCS cache search.

```
---++ search($searchString, $topics, $options, $sDir) -> \%seen
Search .txt files in $dir for $searchString. See RcsFile::searchInWebContent
for details.
```

This package has smell factor of *3*

```
---++ [[TWikiStoreSearchAlgorithmsPurePerlDotPm][TWiki::Store::SearchAlgorithms::PurePerl]]
```

Pure perl implementation of the RCS cache search.

```
---++ search($searchString, $topics, $options, $sDir) -> \%seen
Search .txt files in $dir for $string. See RcsFile::searchInWebContent
for details.
```

This package doesn't smell

```
---++ [[TWikiTemplatesDotPm][TWiki::Templates]]
```

Support for the TWiki template language.

This package has smell factor of *2*

```
---++ [[TWikiTimeDotPm][TWiki::Time]]
```

Time handling functions.

This package has smell factor of *5*

---++ [[TWikiUIChangeFormDotPm][TWiki::UI::ChangeForm]]

Service functions used by the UI packages

This package doesn't smell

----+ [[TWikiUIDotPm][]]

This package doesn't smell

---++ [[TWikiUIEditDotPm][TWiki::UI::Edit]]

Edit command handler

This package has smell factor of *1*

---++ [[TWikiUIManageDotPm][TWiki::UI::Manage]]

UI functions for web, topic and user management

This package has smell factor of *4*

---++ [[TWikiUIOopsDotPm][TWiki::UI::Oops]]

UI delegate for oops function

This package doesn't smell

----+ [[TWikiUIRDiffDotPm][TWiki::UI::RDiff]]

UI functions for diffing.

This package has smell factor of *12*

---++ [[TWikiUIRegisterDotPm][TWiki::UI::Register]]

User registration handling.

This package has smell factor of *8*

---++ [[TWikiUIRestDotPm][TWiki::UI::Rest]]

UI delegate for REST interface

This package has smell factor of *1*

---++ [[TWikiUISaveDotPm][TWiki::UI::Save]]

UI delegate for save function

This package has smell factor of *1*

```
---++ [[TWikiUISearchDotPm][TWiki::UI::Search]]
```

UI functions for searching.

This package has smell factor of *3*

```
---++ [[TWikiUIStatisticsDotPm][TWiki::UI::Statistics]]
```

Statistics extraction and presentation

This package has smell factor of *1*

```
---++ [[TWikiUIUploadDotPm][TWiki::UI::Upload]]
```

UI delegate for attachment management functions

This package has smell factor of *2*

```
---++ [[TWikiUIViewDotPm][TWiki::UI::View]]
```

UI delegate for view function

This package has smell factor of *4*

```
---++ [[TWikiUserMappingDotPm][TWiki::UserMapping]]
```

This is a virtual base class (a.k.a an interface) for all user mappers. It is **not** useable as a mapping in TWiki - use the BaseUserMapping for default behaviour.

User mapping is the process by which TWiki maps from a username (a login name) to a display name and back. It is also where groups are maintained.

See TWiki::Users::BaseUserMapping and TWiki::Users::TWikiUserMapping for the default implementations of this interface.

If you want to write a user mapper, you will need to implement the methods described in this class.

User mappings work by mapping both login names and display names to a `_canonical user id_`. This user id is composed from a prefix that defines the mapper in use (something like 'BaseUserMapping_' or 'LdapUserMapping_') and a unique user id that the mapper uses to identify the user.

The null prefix is reserver for the TWikiUserMapping for compatibility with old TWiki releases.

Note: in all the following documentation, `=${cUID=}` refers to a **canonical user id**.

This package has smell factor of *1*

SourceCode < TWiki < TWiki

```
---++ [[TWikiUsersApacheHtpasswdUserDotPm][TWiki::Users::ApacheHtpasswdUser]]
```

Password manager that uses Apache::Htpasswd to manage users and passwords.

Subclass of [[TWikiUsersPasswordDotPm][=TWiki::Users::Password=]].

See documentation of that class for descriptions of the methods of this class.

Duplicates functionality of

```
[[TWikiUsersHtpasswdUserDotPm][=TWiki::Users::HtpasswdUser=]];
```

provided mainly as an example of how to write a new password manager.

This package has smell factor of *1*

```
---++ [[TWikiUsersBaseUserMappingDotPm][TWiki::Users::BaseUserMapping]]
```

User mapping is the process by which TWiki maps from a username (a login name) to a display name and back. It is also where groups are maintained.

The BaseMapper provides support for a small number of predefined users.

No registration - this is a read only usermapper. It uses the mapper prefix 'BaseUserMapping_'.

```
----++ Users
```

- * TWikiAdmin - uses the password that was set in Configure (IF its not null)
- * TWikiGuest
- * UnknownUser
- * TWikiContributor - 1 Jan 2005
- * TWikiRegistrationAgent - 1 Jan 2005

```
----++ Groups
```

- * \$TWiki::cfg{SuperAdminGroup}
- * TWikiBaseGroup

This package has smell factor of *2*

```
---++ [[TWikiUsersDotPm][TWiki::Users]]
```

This package provides services for the lookup and manipulation of login and wiki names of users, and their authentication.

It is a Facade that presents a common interface to the User Mapping and Password modules. The rest of the core should *only* use the methods of this package, and should *never* call the mapping or password managers directly.

TWiki uses the concept of a `_login name_` which is used to authenticate a user. A login name maps to a `_wiki name_` that is used to identify the user for display. Each login name is unique to a single user, though several login names may map to the same wiki name.

Using this module (and the associated plug-in user mapper) TWiki supports the concept of `_groups_`. Groups are sets of login names that are treated equally for the purposes of access control. Group names do not have to be wiki names, though it is helpful for display if they are.

Internally in the code TWiki uses something referred to as a `_canonical user id_` or just `_user id_`. The user id is also used externally to uniquely identify the user when (for example) recording topic histories. The user id is *usually* just the login name, but it doesn't need to be. It just has to be a unique 7-bit alphanumeric and underscore string that can be mapped to/from login and wiki names by the user mapper.

The canonical user id should *never* be seen by a user. On the other hand, core code should never use anything *but* a canonical user id to refer

to a user.

Terminology

- * A *login name* is the name used to log in to TWiki. Each login name is assumed to be unique to a human. The Password module is responsible for authenticating and manipulating login names.
- * A *canonical user id* is an internal TWiki representation of a user. Each canonical user id maps 1:1 to a login name.
- * A *wikiname* is how a user is displayed. Many user ids may map to a single wikiname. The user mapping module is responsible for mapping the user id to a wikiname.
- * A *group id* represents a group of users and other groups. The user mapping module is responsible for mapping from a group id to a list of canonical user ids for the users in that group.
- * An *email* is an email address associated with a *login name*. A single login name may have many emails.

NOTE:

- * wherever the code references \$cUID, its a canonical_id
- * wherever the code references \$group, its a group_name
- * \$name may be a group or a cUID

This package has smell factor of *6*

```
---++ [[TWikiUsersHtPasswdUserDotPm][TWiki::Users::HtPasswdUser]]
```

Support for htpasswd and htdigest format password files.

Subclass of [[TWikiUsersPasswordDotPm][=TWiki::Users::Password=]].
See documentation of that class for descriptions of the methods of this class.

This package has smell factor of *3*

```
---++ [[TWikiUsersPasswordDotPm][TWiki::Users::Password]]
```

Base class of all password handlers. Default behaviour is no passwords, so anyone can be anyone they like.

The methods of this class should be overridden by subclasses that want to implement other password handling methods.

This package doesn't smell

```
---++ [[TWikiUsersTWikiUserMappingDotPm][TWiki::Users::TWikiUserMapping]]
```

The User mapping is the process by which TWiki maps from a username (a login name) to a wikiname and back. It is also where groups are defined.

By default TWiki maintains user topics and group topics in the %USERSWEB% that define users and group. These topics are

- * !TWikiUsers - stores a mapping from usernames to TWiki names
- * !WikiName - for each user, stores info about the user
- * !GroupNameGroup - for each group, a topic ending with "Group" stores a list of users who are

Many sites will want to override this behaviour, for example to get users and groups from a corpora

This class implements the basic TWiki behaviour using topics to store users, but is also designed to be subclassed so that other services can be used.

Subclasses should be named 'XxxxUserMapping' so that configure can find them.

This package has smell factor of *14*

There were a total of *219* smells
 </div><!-- /patternTopic-->

%META{"form"}%

%META{"attachments"}%</div><!-- /patternContent-->

%MAKETEXT{"This topic:"}% <nop>%WEB%%META{"parent" prefix=" >

%MAKETEXT{"Topic revision:"}% %REVINFO{format="r\$rev - \$date - <nop>\$wikiname"}%

</div><!-- /patternMainContents-->

</div><!-- /patternMain-->

</div><!-- /patternFloatWrap-->

<div class="clear"> </div>

</div><!-- /patternOuter--><div id="patternBottomBar"><div id="patternBottomBarContents"><div id=

</div><!-- /patternPage-->

</div><!-- /patternPageShadow-->

</div><!-- /patternScreen-->

</body></html>