---+ Package `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.

`twiki`

The TWiki object this login manager is attached to.

StaticMethod makeLoginManager `($twiki) -> $TWiki::LoginManager`

Factory method, used to generate a new TWiki::LoginManager object for the given session.

ClassMethod new `($session,$impl)`

Construct the user management object

ObjectMethod finish ()

Break circular references.

ClassMethod _real_trace ($session,$impl)

Construct the user management object

ClassMethod _IP2SID ($session,$impl)

read/write IP to SID map, return SID

ObjectMethod loadSession ($defaultUser) -> $login

Get the client session data, using the cookie and/or the request URL. Set up appropriate session variables in the twiki object and return the login name.

$defaultUser is a username to use if one is not available from other sources. The username passed when you create a TWiki instance is passed in here.

ObjectMethod checkAccess ()

Check if the script being run in this session is authorised for execution. If not, throw an access control exception.

ObjectMethod complete ()

Complete processing after the client's HTTP request has been responded to. Flush the user's session (if any) to disk.

StaticMethod expireDeadSessions ()

Delete sessions and passthrough files that are sitting around but are really expired. This **assumes** that the sessions are stored as files.

This is a static method, but requires TWiki::cfg. It is designed to be run from a session or from a cron job.

ObjectMethod userLoggedIn ($login,$wikiname)

Called when the user is known. It's invoked from TWiki::UI::Register::finish for instance,

1. when the user follows the link in their verification email message
2. or when the session store is read
3. when the user authenticates (via templatelogin / sudo)

- `$login` - string login name
- `$wikiname` - string wikiname

ObjectMethod _myScriptURLRE `($thisl)`

ObjectMethod _rewriteURL `($thisl)`

ObjectMethod _rewriteFORM `($thisl)`

ObjectMethod endRenderingHandler `()`

This handler is called by getRenderedVersion just before the plugins postRenderingHandler. So it is passed all HTML text just before it is printed.

DEPRECATED Use postRenderingHandler instead.

ObjectMethod _pushCookie `($thisl)`

ObjectMethod addCookie `($c)`

Add a cookie to the list of cookies for this session.

- `$c` - a CGI::Cookie

ObjectMethod modifyHeader `(\%header)`

Modify a HTTP header

- `\%header` - header entries

ObjectMethod redirectCgiQuery `($url)`

Generate an HTTP redirect on STDOUT, if you can. Return 1 if you did.

- `$url` - target of the redirection.

ObjectMethod getSessionValues `() -> \%values`

Get a name->value hash of all the defined session variables

ObjectMethod getSessionValue `($name) -> $value`

Get the value of a session variable.

ObjectMethod setSessionValue `($name,$value)`

Set the value of a session variable. We do not allow setting of AUTHUSER and SESSION_REQUEST_NUMBER.

ObjectMethod clearSessionValue `($name) -> $boolean`

Clear the value of a session variable. We do not allow setting of AUTHUSER.

ObjectMethod forceAuthentication `() -> boolean`

VIRTUAL METHOD implemented by subclasses

Triggered by an access control violation, this method tests to see if the current session is authenticated or not. If not, it does whatever is needed so that the user can log in, and returns 1.

If the user has an existing authenticated session, the function simply drops though and returns 0.

ObjectMethod loginUrl `(...) -> $url`

VIRTUAL METHOD implemented by subclasses

Return a full URL suitable for logging in.

- . . . - url parameters to be added to the URL, in the format required by TWiki::getScriptUrl()

ObjectMethod getUser `()`

VIRTUAL METHOD implemented by subclasses

If there is some other means of getting a username - for example, Apache has remote_user() - then return it. Otherwise, return undef and the username stored in the session will be used.

ObjectMethod _LOGIN `($thisl)`

ObjectMethod _LOGOUTURL `($thisl)`

ObjectMethod _LOGOUT `($thisl)`

ObjectMethod _AUTHENTICATED `($thisl)`

ObjectMethod _CANLOGIN `($thisl)`

ObjectMethod _SESSION_VARIABLE `($thisl)`

ObjectMethod _LOGINURL `($thisl)`

ObjectMethod _dispLogon `($thisl)`

_skinSelect ()

Internal use only TODO: what does it do?

sub createCryptToken ($session)-> $token Takes the input as session and returns the MD5 hash string. This subroutine is responsible for updating the token database

The tokens solve the CSRF issue

sub cleanCryptTokens($session, $token)

This subroutine takes care of cleaning used tokens Usually called from token verification subroutines.

sub addCryptTokeninForm ( )-> returns the form with "crypttoken" html input hidden field

If TWiki Application developer has added "crypttoken" then the current subroutine returns the form without performing any parsing. If the form with method - POST do not have any "crypttoken", this subroutine adds the token.

This topic: TWiki21Nov > TWikiLoginManagerDotPm
Topic revision: r4 - 2011-08-21 - TWikiContributor