---+ Package `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**.

PROTECTED ClassMethod new ($session, $mapping_id)

Construct a user mapping object, using the given mapping id.

ObjectMethod finish `()`

Break circular references.

ObjectMethod loginTemplateName `() -> $templateFile`

Allows UserMappings to come with customised login screens - that should preferably only over-ride the UI function

Default is "login"

ObjectMethod supportsRegistration `() -> $boolean`

Return true if the UserMapper supports registration (ie can create new users)

Default is **false**

ObjectMethod handlesUser `($cUID,$login,$wikiname) -> $boolean`

Called by the TWiki::Users object to determine which loaded mapping to use for a given user (must be fast).

The user can be identified by any of $cUID, $login or $wikiname. Any of these parameters may be undef, and they should be tested in order; cUID first, then login, then wikiname.

ObjectMethod login2cUID `($login,$dontcheck) -> cUID`

Convert a login name to the corresponding canonical user name. The canonical name can be any string of 7-bit alphanumeric and underscore characters, and must map 1:1 to the login name. (undef on failure)

(if $dontcheck is true, return a cUID for a nonexistant user too. This is used for registration)

Subclasses **must** implement this method.

Note: This method was previously (in TWiki 4.2.0) known as getCanonicalUserID. The name was changed to avoid confusion with TWiki::Users::getCanonicalUserID, which has a more generic function. However to support older user mappers, getCanonicalUserID will still be called if login2cUID is not defined.

ObjectMethod getLoginName `($cUID) -> login`

Converts an internal cUID to that user's login (undef on failure)

Subclasses **must** implement this method.

ObjectMethod addUser `($login,$wikiname,$password,$emails,$mcp) -> $cUID`

Add a user to the persistant mapping that maps from usernames to wikinames and vice-versa.

$login and $wikiname must be acceptable to $TWiki::cfg{NameFilter}. $login must **always** be specified. $wikiname may be undef, in which case the user mapper should make one up.

This function must return a canonical user id that it uses to uniquely identify the user. This can be the login name, or the wikiname if they are all guaranteed unigue, or some other string consisting only of 7-bit alphanumerics and underscores.

If you fail to create a new user (for eg your Mapper has read only access),

```
    throw Error::Simple('Failed to add user: '.$error);
```

where $error is a descriptive string.

Throws an Error::Simple if user adding is not supported (the default).

ObjectMethod removeUser `($cUID) -> $boolean`

Delete the users entry from this mapper. Throws an Error::Simple if user removal is not supported (the default).

ObjectMethod getWikiName `($cUID) -> $wikiname`

Map a canonical user name to a wikiname.

Returns the $cUID by default.

ObjectMethod userExists `($cUID) -> $boolean`

Determine if the user already exists or not. Whether a user exists or not is determined by the password manager.

Subclasses **must** implement this method.

ObjectMethod eachUser `() -> TWiki::ListIteratorofcUIDs`

Get an iterator over the list of all the registered users **not** including groups.

Subclasses **must** implement this method.

ObjectMethod eachGroupMember `($group) -> TWiki::ListIteratorofcUIDs`

Return a iterator over the canonical user ids of users that are members of this group. Should only be called on groups.

Note that groups may be defined recursively, so a group may contain other groups. This method should **only** return users i.e. all contained groups should be fully expanded.

Subclasses **must** implement this method.

ObjectMethod isGroup `($name) -> boolean`

Establish if a user refers to a group or not. If $name is not a group name it will probably be a canonical user id, though that should not be assumed.

Subclasses **must** implement this method.

ObjectMethod eachGroup `() -> TWiki::ListIteratorofgroupnames`

Get an iterator over the list of all the groups.

Subclasses **must** implement this method.

ObjectMethod eachMembership `($cUID) -> TWiki::ListIteratorofgroupsthisuserisin`

Return an iterator over the names of groups that $cUID is a member of.

Subclasses **must** implement this method.

ObjectMethod isAdmin `($cUID) -> $boolean`

True if the user is an administrator.

ObjectMethod isInGroup `($cUID,$group) -> $bool`

Test if the user identified by $cUID is in the given group. The default implementation iterates over all the members of $group, which is rather inefficient.

ObjectMethod findUserByEmail `($email) -> \@users`

- `$email` - email address to look up

Return a list of canonical user names for the users that have this email registered with the password manager or the user mapping manager.

ObjectMethod getEmails `($name) -> @emailAddress`

If $name is a cUID, return that user's email addresses. If it is a group, return the addresses of everyone in the group.

Duplicates should be removed from the list.

ObjectMethod setEmails `($cUID,@emails)`

Set the email address(es) for the given user.

ObjectMethod getMustChangePassword `($cUID) -> $flag`

Returns 1 if the $cUID must change the password, else 0. Returns undef if $cUID not found.

ObjectMethod getUserData `($cUID) -> $dataRef`

Return a reference to an array of hashes with user data, used to manage users. Each item is a hash with:

- `{name}` - name of field, such as "email"
- `{title}` - title of field, such as "E-mail"
- `{value}` - value of field, such as "jimmy@example.com"
- `{type}` - type of field: `text`, `password`, `checkbox`, `label`
- `{size}` - size of field, such as `40`
- `{note}` - comment note, if any

User management forms can be build dynamically from this data structure. Each password manager may return a different set of fields.

ObjectMethod setUserData `($cUID,$dataRef)`

Set the user data of a user. Same array of hashes as getUserData is assumed, although only `{name}` and `{value}` are used.

ObjectMethod findUserByWikiName `($wikiname) -> listofcUIDsassociatedwiththatwikiname`

- `$wikiname` - wikiname to look up

Return a list of canonical user names for the users that have this wikiname. Since a single wikiname might be used by multiple login ids, we need a list.

Note that if $wikiname is the name of a group, the group will **not** be expanded.

Subclasses **must** implement this method.

ObjectMethod checkPassword `($login,$passwordU) -> $boolean`

Finds if the password is valid for the given login. This is called using a login name rather than a cUID because the user may not have been mapped at the time it is called.

Returns 1 on success, undef on failure.

Default behaviour is to return 1.

ObjectMethod setPassword `($cUID,$newPassU,$oldPassU) -> $boolean`

If the $oldPassU matches matches the user's password, then it will replace it with $newPassU.

If $oldPassU is not correct and not 1, will return 0.

If $oldPassU is 1, will force the change irrespective of the existing password, adding the user if necessary.

Otherwise returns 1 on success, undef on failure.

Default behaviour is to fail.

ObjectMethod passwordError `() -> $string`

Returns a string indicating the error that happened in the password handlers TODO: these delayed errors should be replaced with Exceptions.

returns undef if no error (the default)

This topic: TWiki > TWikiUserMappingDotPm
Topic revision: r3 - 2011-08-21 - TWikiContributor

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