

Trac

version 1.0

The Trac Team

September 08, 2012

Contents

Trac 1.0 API Documentation	1
Content	1
API Reference	1
trac.attachment -- Attachments for Trac resources	1
Interfaces	1
Classes	2
Components	2
trac.cache -- Control of cached data coherency	3
Public API	3
Internal API	4
trac.core -- the Trac "kernel"	5
Component model	5
More on components	6
Miscellaneous	7
trac.env -- Trac Environment model and APIs	7
Interfaces	7
Components	8
Functions	12
trac.mimeview.api -- Trac content transformation APIs	13
File metadata management	13
Interfaces	13
Components	15
Helper classes	16
Functions	18
trac.ticket.roadmap -- The Roadmap and Milestone modules	19
Interfaces	19
Components	19
Helper Functions	20
trac.util -- General purpose utilities	20
trac.util.datefmt -- Date and Time manipulation	20
Conversion	20
Parsing	21
Formatting	21
jQuery UI datepicker helpers	21
Timezone utilities	22
trac.util.html -- HTML transformations	22
Building HTML programmatically	22
HTML clean-up and sanitization	22
Misc. HTML processing	24
trac.util.presentation -- Utilities for dynamic content generation	24

trac.util.text -- Text manipulation	26
The Unicode toolbox	26
Web utilities	26
Console and file system	27
Miscellaneous	27
Text formatting	27
Conversion utilities	29
Web related utilities	29
OS related utilities	29
Python "system" utilities	30
Setuptools utilities	31
Cryptographic related utilities	31
Data structures which don't fit anywhere else	31
Algorithmic utilities	33
trac.util.datefmt -- Date and Time manipulation	33
Conversion	33
Parsing	34
Formatting	34
jQuery UI datepicker helpers	35
Timezone utilities	35
trac.util.html -- HTML transformations	35
Building HTML programmatically	35
HTML clean-up and sanitization	36
Misc. HTML processing	37
trac.util.presentation -- Utilities for dynamic content generation	37
trac.util.text -- Text manipulation	39
The Unicode toolbox	39
Web utilities	40
Console and file system	40
Miscellaneous	41
Text formatting	41
Conversion utilities	42
trac.versioncontrol.api -- Trac Version Control APIs	42
Interfaces	42
Components	43
Exceptions	45
Abstract classes	45
Helper Functions	49
trac.versioncontrol.diff -- Utilities for generation of diffs	49
Synopsis	49
Function Reference	49
trac.versioncontrol.svn_fs -- Subversion backend for Trac	50

Note about Unicode	50
Components	51
Concrete classes	51
trac.web.api -- Trac Web Request Handling	53
Interfaces	53
Classes	54
Helper Functions	55
trac.web.auth -- Trac Authentication	56
Component	56
Support Classes	56
trac.web.chrome -- Trac content generation for the Web	56
Interfaces	57
Components	57
Functions	60
Web resources	60
Page admonitions	61
Contextual Navigation	61
Miscellaneous	61
trac.web.href -- Creation of URLs	61
trac.web.main -- Trac Web Entry Point	63
trac.web.dispatch_request	63
Components	63
Classes	64
Helper Functions	64
Miscellaneous	64
trac.wiki.api -- The Wiki API	64
Interfaces	64
The Wiki System	67
Other Functions	67
trac.wiki.macros -- The standard set of Wiki macros	68
tracopt.mimeview -- Optional content generation modules	68
Syntax Highlighters	68
Testing in Trac	69
Running the tests	69
Prerequisites	69
Invoking the tests	69
Understanding failures	70
Prerequisites on Windows	70
Writing Tests for Core	70
Where tests belong	70
Using Twill	70
Example	71

Test Environment Helpers	71
Functional Test Environment	71
Functional Tester	72
Using an alternate database backend	74
Postgres	74
MySQL	75
Troubleshooting	75
Writing Tests for Plugins	75
Testing a VCS backend	76
Glossary	77
Documentation TODO	77
Indices and tables	77
Index	79
Python Module Index	91

Trac 1.0 API Documentation

Release: tags-trac-1.0

Date: September 08, 2012

This is work in progress. The API is not yet fully covered, but what you'll find here should be accurate, otherwise it's a bug and you're welcome to open a ticket for reporting the problem.

Content

API Reference

trac.attachment -- Attachments for Trac resources

This module contains the **Attachment** model class and the **AttachmentModule** component which manages file attachments for any kind of Trac resources. Currently, the wiki pages, tickets and milestones all support file attachments. You can use the same utility methods from the **AttachmentModule** as they do for easily adding attachments to other kinds of resources.

See also the `attach_file_form.html` and `attachment.html` templates which can be used to display the attachments.

Interfaces

class `trac.attachment.IAttachmentChangeListener`

Extension point interface for components that require notification when attachments are created or deleted.

See also [trac.attachment.IAttachmentChangeListener extension points](#)

attachment_added (*attachment*)

Called when an attachment is added.

attachment_deleted (*attachment*)

Called when an attachment is deleted.

attachment_reparented (*attachment, old_parent_realm, old_parent_id*)

Called when an attachment is reparented.

class `trac.attachment.IAttachmentManipulator`

Extension point interface for components that need to manipulate attachments.

Unlike change listeners, a manipulator can reject changes being committed to the database.

See also [trac.attachment.IAttachmentManipulator extension points](#)

prepare_attachment (*req, attachment, fields*)

Not currently called, but should be provided for future compatibility.

validate_attachment (*req, attachment*)

Validate an attachment after upload but before being stored in Trac environment.

Must return a list of (*field*, *message*) tuples, one for each problem detected. *field* can be any of *description*, *username*, *filename*, *content*, or **None** to indicate an overall problem with the attachment.

Therefore, a return value of `[]` means everything is OK.

class `trac.attachment.ILegacyAttachmentPolicyDelegate`

Interface that can be used by plugins to seamlessly participate to the legacy way of checking for attachment permissions.

This should no longer be necessary once it becomes easier to setup fine-grained permissions in the default permission store.

See also [trac.attachment.ILegacyAttachmentPolicyDelegate extension points](#)

check_attachment_permission (action, username, resource, perm)

Return the usual **True/False/None** security policy decision appropriate for the requested action on an attachment.

param action: one of ATTACHMENT_VIEW, ATTACHMENT_CREATE, ATTACHMENT_DELETE
param username: the user string
param resource: the **Resource** for the attachment. Note that when ATTACHMENT_CREATE is checked, the resource **.id** will be **None**.
param perm: the permission cache for that username and resource

Classes

```
class trac.attachment.Attachment (env, parent_realm_or_attachment_resource,
parent_id=None, filename=None, db=None)
```

delete (db=None)

Delete the attachment, both the record in the database and the file itself.

Changed in version 1.0: the **db** parameter is no longer needed (will be removed in version 1.1.1)

insert (filename, fileobj, size, t=None, db=None)

Create a new Attachment record and save the file content.

Changed in version 1.0: the **db** parameter is no longer needed (will be removed in version 1.1.1)

classmethod select (env, parent_realm, parent_id, db=None)

Iterator yielding all **Attachment** instances attached to resource identified by **parent_realm** and **parent_id**.

Changed in version 1.0: the **db** parameter is no longer needed (will be removed in version 1.1.1)

classmethod delete_all (env, parent_realm, parent_id, db=None)

Delete all attachments of a given resource.

Changed in version 1.0: the **db** parameter is no longer needed (will be removed in version 1.1.1)

classmethod reparent_all (env, parent_realm, parent_id, new_realm, new_id)

Reparent all attachments of a given resource to another resource.

```
class trac.attachment.InvalidAttachment (message, title=None, show_traceback=False)
```

Exception raised when attachment validation fails.

If message is a genshi.builder.tag object, everything up to the first <p> will be displayed in the red box, and everything after will be displayed below the red box. If title is given, it will be displayed as the large header above the error message.

Components

```
class trac.attachment.AttachmentModule
```

change_listeners

List of components that implement **IAttachmentChangeListener**

manipulators

List of components that implement **IAttachmentManipulator**

max_size

Maximum allowed file size (in bytes) for ticket and wiki attachments.

max_zip_size

Maximum allowed total size (in bytes) for an attachment list to be downloadable as a **zip**. Set this to -1 to disable download as **zip**. ("since 1.0")

render_unsafe_content

Whether attachments should be rendered in the browser, or only made downloadable.

Pretty much any file may be interpreted as HTML by the browser, which allows a malicious user to attach a file containing cross-site scripting attacks.

For public sites where anonymous users can create attachments it is recommended to leave this option disabled (which is the default).

viewable_attachments (context)

Return the list of viewable attachments in the given context.

Parameters: **context** -- the **RenderingContext** corresponding to the parent **Resource** for the attachments

attachment_data (context)

Return a data dictionary describing the list of viewable attachments in the current context.

get_history (start, stop, realm)

Return an iterable of tuples describing changes to attachments on a particular object realm.

The tuples are in the form (change, realm, id, filename, time, description, author). **change** can currently only be **created**.

FIXME: no iterator

get_timeline_events (req, resource_realm, start, stop)

Return an event generator suitable for **ITimelineEventProvider**.

Events are changes to attachments on resources of the given **resource_realm.realm**.

get_search_results (req, resource_realm, terms)

Return a search result generator suitable for **ISearchSource**.

Search results are attachments on resources of the given **resource_realm.realm** whose filename, description or author match the given terms.

get_resource_url (resource, href, **kwargs)

Return an URL to the attachment itself.

A **format** keyword argument equal to **'raw'** will be converted to the raw-attachment prefix.

`class trac.attachment.AttachmentAdmin`

trac-admin command provider for attachment administration.

trac.cache -- Control of cached data coherency

Trac is a server application which may involve multiple concurrent processes. The coherency of the data presented to the clients is ensured by the underlying database and its transaction handling. However, a server process will not systematically retrieve data from the database, as various in-memory caches are used for performance reasons. We could ensure the integrity of those caches in a single process in presence of multiple threads by the appropriate use of locking and by updating the caches as needed, but we also need a mechanism for invalidating the caches in the *other* processes.

The purpose of this module is to provide a **cached decorator** which can annotate a data *retriever* method of a class for turning it into an attribute working like a cache. This means that an access to this attribute will only call the underlying retriever method once on first access, or only once after the cache has been invalidated, even if this invalidation happened in another process.

Public API

`trac.cache.cached (fn_or_attr=None)`

Method decorator creating a cached attribute from a data retrieval method.

Accessing the cached attribute gives back the cached value. The data retrieval method is transparently called by the **CacheManager** on first use after the program start or after the cache has been invalidated. Invalidating the cache for this value is done by deleting the attribute.

Note that the cache validity is maintained using the **cache** table in the database. Cache invalidation is performed within a transaction block, and can be nested within another transaction block.

When the decorator is used in a class for which instances behave as singletons within the scope of a given **Environment** (typically **Component** classes), the key used to identify the attribute in the database is constructed from the names of the containing module, class and retriever method:

```
class WikiSystem(Component):
    @cached
    def pages(self):
        return set(name for name, in self.env.db_query(
            "SELECT DISTINCT name FROM wiki"))
```

Otherwise, when the decorator is used in non-"singleton" objects, a string specifying the name of an attribute containing a string unique to the instance must be passed to the decorator. This value will be appended to the key constructed from module, class and method name:

```
class SomeClass(object):
    def __init__(self, env, name):
        self.env = env
        self.name = name
        self._metadata_id = name

    @cached('_metadata_id')
    def metadata(self):
        ...
```

Note that in this case the key attribute is overwritten with a hash of the key on first access, so it should not be used for any other purpose.

In either case, this decorator requires that the object on which it is used has an `env` attribute containing the application **Environment**.

Changed in version 1.0: The data retrieval method used to be called with a single argument `db` containing a reference to a database connection. This is the same connection that can be retrieved via the normal `db_query` or `db_transaction`, so this is no longer needed, though methods supporting that argument are still supported (but will be removed in version 1.1.1).

Internal API

```
class trac.cache.CacheManager
```

Cache manager.

```
reset_metadata ()
```

Reset per-request cache metadata.

```
get (id, retriever, instance)
```

Get cached or fresh data for the given id.

```
invalidate (id)
```

Invalidate cached data for the given id.

The following classes are the **descriptors** created by the **cached** decorator:

```
class trac.cache.CachedSingletonProperty (retriever)
```

Cached property descriptor for classes behaving as singletons in the scope of one **Environment** instance.

This means there will be no more than one cache to monitor in the database for this kind of cache. Therefore, using only "static" information for the key is enough. For the same reason it is also safe to store the corresponding id as a property of the descriptor instance.

```
class trac.cache.CachedProperty (retriever, key_attr)
```

Cached property descriptor for classes having potentially multiple instances associated to a single **Environment** instance.

As we'll have potentially many different caches to monitor for this kind of cache, the key needs to be augmented by a string unique to each instance of the owner class. As the resulting id will be different for each instance of the

owner class, we can't store it as a property of the descriptor class, so we store it back in the attribute used for augmenting the key (`key_attr`).

Both classes inherit from a common base:

```
class trac.cache.CachedPropertyBase (retriever)
    Base class for cached property descriptors
```

trac.core -- the Trac "kernel"

Component model

The Trac component model is very simple, it is based on **Interface** classes that are used to document a particular set of methods and properties that must be defined by a **Component** subclass when it declares it *implements* that interface.

```
class trac.core.Interface
    Marker base class for extension point interfaces.
```

```
class trac.core.Component
    Base class for components.
    Every component can declare what extension points it provides, as well as what extension points of other
    components it extends.

    static implements (*interfaces)
        Can be used in the class definition of Component subclasses to declare the extension points that are
        extended.
```

The static method **Component.implements** is never used as such, but rather via the global **implements** function. This globally registers that particular component subclass as an implementation of the listed interfaces.

```
trac.core.implements (*interfaces)
    Can be used in the class definition of Component subclasses to declare the extension points that are extended.
```

For example:

```
class IStuffProvider(Interface):
    """All interfaces start by convention with an "I" and even if it's
    not a convention, in practice most interfaces are "Provider" of
    something ;-)
    """

    def get_stuff(color=None):
        """We usually don't specify "self" here, but try to describe
        as precisely as possible how the method might be called and
        what is the expected return type."""

class ComponentA(Component):

    implements(IStuffProvider)

    # IStuffProvider methods

    def get_stuff(self, color=None):
        if not color or color == 'yellow':
            yield ('duck', "the regular waterproof plastic duck")
```

The benefit of implementing an interface is to possibility to define an **ExtensionPoint** property for an **Interface**, in a **Component** subclass. Such a property provides a convenient way to retrieve *all* registered and enabled component instances for that interface. The enabling of components is the responsibility of the **ComponentManager**, see **is_component_enabled** below.

```
class trac.core.ExtensionPoint (interface)
```

Marker class for extension points in components.
Create the extension point.

Parameters: **interface** -- the **Interface** subclass that defines the protocol for the extension point

extensions (component)

Return a list of components that declare to implement the extension point interface.

Continuing the example:

```
class StuffModule(Component):

    stuff_providers = ExtensionPoint(ISTuffProvider)

    def get_all_stuff(self, color=None):
        stuff = {}
        for provider in self.stuff_provider:
            for name, descr in provider.get_stuff(color) or []:
                stuff[name] = descr
        return stuff
```

Note that besides going through an extension point, **Component** subclass instances can alternatively be retrieved directly by using the instantiation syntax. This is not an usual instantiation though, as this will always return the same instance in the given **ComponentManager** "scope" passed to the constructor:

```
>>> a1 = ComponentA(mgr)
>>> a2 = ComponentA(mgr)
>>> a1 is a2
True
```

The same thing happens when retrieving components via an extension point, the retrieved instances belong to the same "scope" as the instance used to access the extension point:

```
>>> b = StuffModule(mgr)
>>> any(a is a1 for a in b.stuff_providers)
True
```

class trac.core.ComponentManager

The component manager keeps a pool of active components.
Initialize the component manager.

is_enabled (cls)

Return whether the given component class is enabled.

disable_component (component)

Force a component to be disabled.

Parameters: **component** -- can be a class or an instance.

component_activated (component)

Can be overridden by sub-classes so that special initialization for components can be provided.

is_component_enabled (cls)

Can be overridden by sub-classes to veto the activation of a component.

If this method returns **False**, the component was disabled explicitly. If it returns **None**, the component was neither enabled nor disabled explicitly. In both cases, the component with the given class will not be available.

In practice, there's only one kind of **ComponentManager** in the Trac application itself, the **trac.env.Environment**.

More on components

We have seen above that one way to retrieve a `Component` instance is to call the constructor on a `ComponentManager` instance `mgr`:

```
a1 = ComponentA(mgr)
```

This will eventually trigger the creation of a new `ComponentA` instance if there wasn't already one created for `mgr`¹. At this unique occasion, the constructor of the component subclass will be called *without arguments*, so if you define a constructor it must have the following signature:

```
def __init__(self):
    self.all_colors = set()
```

Note that one should try to do as little as possible in a `Component` constructor. The most complex operation could be for example the allocation of a lock to control the concurrent access to some data members and guarantee thread-safe initialization of more costly resources on first use. Never do such costly initializations in the constructor itself.

Miscellaneous

```
class trac.core.TracError (message, title=None, show_traceback=False)
```

Exception base class for errors in Trac.

If message is a `genshi.builder.tag` object, everything up to the first `<p>` will be displayed in the red box, and everything after will be displayed below the red box. If title is given, it will be displayed as the large header above the error message.

trac.env -- Trac Environment model and APIs

Interfaces

```
class trac.env.IEnvironmentSetupParticipant
```

Extension point interface for components that need to participate in the creation and upgrading of Trac environments, for example to create additional database tables.

Please note that `IEnvironmentSetupParticipant` instances are called in arbitrary order. If your upgrades must be ordered consistently, please implement the ordering in a single `IEnvironmentSetupParticipant`. See the database upgrade infrastructure in Trac core for an example.

See also [trac.env.IEnvironmentSetupParticipant extension points](#)

environment_created ()

Called when a new Trac environment is created.

environment_needs_upgrade (db)

Called when Trac checks whether the environment needs to be upgraded.

Should return `True` if this participant needs an upgrade to be performed, `False` otherwise.

upgrade_environment (db)

Actually perform an environment upgrade.

Implementations of this method don't need to commit any database transactions. This is done implicitly for each participant if the upgrade succeeds without an error being raised.

However, if the `upgrade_environment` consists of small, restartable, steps of upgrade, it can decide to commit on its own after each successful step.

```
class trac.env.ISystemInfoProvider
```

Provider of system information, displayed in the "About Trac" page and in internal error reports.

See also [trac.env.ISystemInfoProvider extension points](#)

¹

Ok, it *might* happen that more than one component instance get created due to a race condition. This is usually harmless, see [#9418](#).

get_system_info ()

Yield a sequence of (**name**, **version**) tuples describing the name and version information of external packages used by a component.

Components

The **Environment** is special in the sense it is not only a **Component**, but also a **trac.core.ComponentManager**.

```
class trac.env.Environment (path, create=False, options=[], ])
```

Trac environment manager.

Trac stores project information in a Trac environment. It consists of a directory structure containing among other things:

- a configuration file,
- project-specific templates and plugins,
- the wiki and ticket attachments files,
- the SQLite database file (stores tickets, wiki pages...) in case the database backend is sqlite

Initialize the Trac environment.

Parameters:

- **path** -- the absolute path to the Trac environment
- **create** -- if **True**, the environment is created and populated with default data; otherwise, the environment is expected to already exist.
- **options** -- A list of (**section**, **name**, **value**) tuples that define configuration options

system_info_providers

List of components that implement **ISystemInfoProvider**

setup_participants

List of components that implement **IEnvironmentSetupParticipant**

components_section

This section is used to enable or disable components provided by plugins, as well as by Trac itself. The component to enable/disable is specified via the name of the option. Whether its enabled is determined by the option value; setting the value to **enabled** or **on** will enable the component, any other value (typically **disabled** or **off**) will disable the component.

The option name is either the fully qualified name of the components or the module/package prefix of the component. The former enables/disables a specific component, while the latter enables/disables any component in the specified package/module.

Consider the following configuration snippet: {{{ [components] trac.ticket.report.ReportModule = disabled webadmin.* = enabled }}}}

The first option tells Trac to disable the [wiki:TracReports report module]. The second option instructs Trac to enable all components in the **webadmin** package. Note that the trailing wildcard is required for module/package matching.

To view the list of active components, go to the "Plugins" page on "About Trac" (requires **CONFIG_VIEW** [wiki:TracPermissions permissions]).

See also: TracPlugins

shared_plugins_dir

Path to the //shared plugins directory//.

Plugins in that directory are loaded in addition to those in the directory of the environment **plugins**, with this one taking precedence.

("since 0.11")

base_url

Reference URL for the Trac deployment.

This is the base URL that will be used when producing documents that will be used outside of the web browsing context, like for example when inserting URLs pointing to Trac resources in notification e-mails.

base_url_for_redirect

Optionally use `[trac] base_url` for redirects.

In some configurations, usually involving running Trac behind a HTTP proxy, Trac can't automatically reconstruct the URL that is used to access it. You may need to use this option to force Trac to use the `base_url` setting also for redirects. This introduces the obvious limitation that this environment will only be usable when accessible from that URL, as redirects are frequently used. ("since 0.10.5")

secure_cookies

Restrict cookies to HTTPS connections.

When true, set the `secure` flag on all cookies so that they are only sent to the server on HTTPS connections. Use this if your Trac instance is only accessible through HTTPS. ("since 0.11.2")

project_name

Name of the project.

project_description

Short description of the project.

project_url

URL of the main project web site, usually the website in which the `base_url` resides. This is used in notification e-mails.

project_admin

E-Mail address of the project's administrator.

project_admin_trac_url

Base URL of a Trac instance where errors in this Trac should be reported.

This can be an absolute or relative URL, or `'.'` to reference this Trac instance. An empty value will disable the reporting buttons. ("since 0.11.3")

project_footer

Page footer text (right-aligned).

project_icon

URL of the icon of the project.

log_type

Logging facility to use.

Should be one of (`none`, `file`, `stderr`, `syslog`, `winlog`).

log_file

If `log_type` is `file`, this should be a path to the log-file. Relative paths are resolved relative to the `log` directory of the environment.

log_level

Level of verbosity in log.

Should be one of (`CRITICAL`, `ERROR`, `WARN`, `INFO`, `DEBUG`).

log_format

Custom logging format.

If nothing is set, the following will be used:

Trac[\$(module)s] \$(levelname)s: \$(message)s

In addition to regular key names supported by the Python logger library (see <http://docs.python.org/library/logging.html>), one could use:

- `$(path)s` the path for the current environment
- `$(basename)s` the last path component of the current environment

- `$(project)s` the project name

Note the usage of `$(...)s` instead of `%(...)s` as the latter form would be interpreted by the `ConfigParser` itself.

Example: `$(thread)d Trac[$(basename)s:$(module)s] $(levelname)s: $(message)s`
 "(since 0.10.5)"

`get_systeminfo ()`

Return a list of `(name, version)` tuples describing the name and version information of external packages used by Trac and plugins.

`component_activated (component)`

Initialize additional member variables for components.

Every component activated through the `Environment` object gets three member variables: `env` (the environment object), `config` (the environment configuration) and `log` (a logger object).

`is_component_enabled (cls)`

Implemented to only allow activation of components that are not disabled in the configuration.

This is called by the `ComponentManager` base class when a component is about to be activated. If this method returns `False`, the component does not get activated. If it returns `None`, the component only gets activated if it is located in the `plugins` directory of the environment.

`enable_component (cls)`

Enable a component or module.

`verify ()`

Verify that the provided path points to a valid Trac environment directory.

`get_db_cnx ()`

Return a database connection from the connection pool

Deprecated: Use `db_transaction()` or `db_query()` instead

`db_transaction` for obtaining the `db` database connection which can be used for performing any query (SELECT/INSERT/UPDATE/DELETE):

```
with env.db_transaction as db:
    ...
```

`db_query` for obtaining a `db` database connection which can be used for performing SELECT queries only:

```
with env.db_query as db:
    ...
```

`with_transaction (db=None)`

Decorator for transaction functions :deprecated:

`get_read_db ()`

Return a database connection for read purposes :deprecated:

See `trac.db.api.get_read_db` for detailed documentation.

`db_query`

Return a context manager which can be used to obtain a read-only database connection.

Example:

```
with env.db_query as db:
    cursor = db.cursor()
    cursor.execute("SELECT ...")
    for row in cursor.fetchall():
        ...
```

Note that a connection retrieved this way can be "called" directly in order to execute a query:


```
with env.db_query as db:
    for row in db("SELECT ..."):
        ...
```

If you don't need to manipulate the connection itself, this can even be simplified to:

```
for row in env.db_query("SELECT ..."):
    ...
```

Warning: after a `with env.db_query as db` block, though the `db` variable is still available, you shouldn't use it as it might have been closed when exiting the context, if this context was the outermost context (`db_query` or `db_transaction`).

`db_transaction`

Return a context manager which can be used to obtain a writable database connection.

Example:

```
with env.db_transaction as db:
    cursor = db.cursor()
    cursor.execute("UPDATE ...")
```

Upon successful exit of the context, the context manager will commit the transaction. In case of nested contexts, only the outermost context performs a commit. However, should an exception happen, any context manager will perform a rollback.

Like for its read-only counterpart, you can directly execute a DML query on the `db`:

```
with env.db_transaction as db:
    db("UPDATE ...")
```

If you don't need to manipulate the connection itself, this can also be simplified to:

```
env.db_transaction("UPDATE ...")
```

Warning: after a `with env.db_transaction as db` block, though the `db` variable is still available, you shouldn't use it as it might have been closed when exiting the context, if this context was the outermost context (`db_query` or `db_transaction`).

`shutdown (tid=None)`

Close the environment.

`get_repository (reponame=None, authname=None)`

Return the version control repository with the given name, or the default repository if `None`.

The standard way of retrieving repositories is to use the methods of `RepositoryManager`. This method is retained here for backward compatibility.

Parameters:

- **reponame** -- the name of the repository
- **authname** -- the user name for authorization (not used anymore, left here for compatibility with 0.11)

`create (options=[,])`

Create the basic directory structure of the environment, initialize the database and populate the configuration file with default values.

If options contains ('inherit', 'file'), default values will not be loaded; they are expected to be provided by that file or other options.

`get_version (db=None, initial=False)`

Return the current version of the database. If the optional argument `initial` is set to `True`, the version of the database used at the time of creation will be returned.

In practice, for database created before 0.11, this will return **False** which is "older" than any db version number.

Since : 0.11

Since 1.0: deprecation warning: the **db** parameter is no longer used and will be removed in version 1.1.1

setup_config ()

Load the configuration file.

get_templates_dir ()

Return absolute path to the templates directory.

get_htdocs_dir ()

Return absolute path to the htdocs directory.

get_log_dir ()

Return absolute path to the log directory.

setup_log ()

Initialize the logging sub-system.

get_known_users (cnx=None)

Generator that yields information about all known users, i.e. users that have logged in to this Trac environment and possibly set their name and email.

This function generates one tuple for every user, of the form (username, name, email) ordered alpha-numerically by username.

Parameters: **cnx** -- the database connection; if omitted, a new connection is retrieved

Since 1.0: deprecation warning: the **cnx** parameter is no longer used and will be removed in version 1.1.1

backup (dest=None)

Create a backup of the database.

Parameters: **dest** -- Destination file; if not specified, the backup is stored in a file called db_name.trac_version.bak

needs_upgrade ()

Return whether the environment needs to be upgraded.

upgrade (backup=False, backup_dest=None)

Upgrade database.

Parameters:

- **backup** -- whether or not to backup before upgrading

- **backup_dest** -- name of the backup file

Returns: whether the upgrade was performed

href

The application root path

abs_href

The application URL

Functions

trac.env.open_environment (env_path=None, use_cache=False)

Open an existing environment object, and verify that the database is up to date.

Parameters:

- **env_path** -- absolute path to the environment directory; if omitted, the value of the **TRAC_ENV** environment variable is used
- **use_cache** -- whether the environment should be cached for subsequent invocations of this function

Returns: the **Environment** object

trac.mimeview.api -- Trac content transformation APIs

File metadata management

The **trac.mimeview** package centralizes the intelligence related to file metadata, principally concerning the **type** (MIME type) of the content and, if relevant, concerning the text encoding (charset) used by the content.

There are primarily two approaches for getting the MIME type of a given file, either taking advantage of existing conventions for the file name, or examining the file content and applying various heuristics.

The module also knows how to convert the file content from one type to another type.

In some cases, only the **url** pointing to the file's content is actually needed, that's why we avoid to read the file's content when it's not needed.

The actual **content** to be converted might be a **unicode** object, but it can also be the raw byte string (**str**) object, or simply an object that can be **read()**.

Note

(for plugin developers)

The Mimeview API is quite complex and many things there are currently a bit difficult to work with (e.g. what an actual **content** might be, see the last paragraph of this description).

So this area is mainly in a "work in progress" state, which will be improved along the lines described in [#3332](#).

In particular, if you are interested in writing **IContentConverter** and **IHTMLPreviewRenderer** components, note that those interfaces will be merged into a new style **IContentConverter**. Feel free to contribute remarks and suggestions for improvements to the corresponding ticket ([#3332](#) as well).

Interfaces

class `trac.mimeview.api.IHTMLPreviewRenderer`

Extension point interface for components that add HTML renderers of specific content types to the **Mimeview** component.

Note

This interface will be merged with **IContentConverter**, as conversion to text/html will simply be a particular content conversion.

Note however that the **IHTMLPreviewRenderer** will still be supported for a while through an adapter, whereas the **IContentConverter** interface itself will be changed.

So if all you want to do is convert to HTML and don't feel like following the API changes, you should rather implement this interface for the time being.

See also [trac.mimeview.api.IHTMLPreviewRenderer extension points](#)

expand_tabs = *False*

implementing classes should set this property to **True** if they support text content where Trac should expand tabs into spaces

returns_source = *False*

indicate whether the output of this renderer is source code that can be decorated with annotations

get_extra_mimetypes ()

Augment the Mimeview system with new mimetypes associations.

This is an optional method. Not implementing the method or returning nothing is fine, the component will still be asked via `get_quality_ratio` if it supports a known mimetype. But implementing it can be useful when the component knows about additional mimetypes which may augment the list of already mimetype to keywords associations.

Generate `(mimetype, keywords)` pairs for each additional mimetype, with `keywords` being a list of keywords or extensions that can be used as aliases for the mimetype (typically file suffixes or Wiki processor keys).

New in version 1.0.

get_quality_ratio (mimetype)

Return the level of support this renderer provides for the `content` of the specified MIME type. The return value must be a number between 0 and 9, where 0 means no support and 9 means "perfect" support.

render (context, mimetype, content, filename=None, url=None)

Render an XHTML preview of the raw `content` in a `RenderingContext`.

The `content` might be:

- a `str` object
- an `unicode` string
- any object with a `read` method, returning one of the above

It is assumed that the content will correspond to the given `mimetype`.

Besides the `content` value, the same content may eventually be available through the `filename` or `url` parameters. This is useful for renderers that embed objects, using `<object>` or `` instead of including the content inline.

Can return the generated XHTML text as a single string or as an iterable that yields strings. In the latter case, the list will be considered to correspond to lines of text in the original content.

class trac.mimeview.api.IHTMLPreviewAnnotator

Extension point interface for components that can annotate an XHTML representation of file contents with additional information.

See also [trac.mimeview.api.IHTMLPreviewAnnotator extension points](#)

get_annotation_type ()

Return a `(type, label, description)` tuple that defines the type of annotation and provides human readable names. The `type` element should be unique to the annotator. The `label` element is used as column heading for the table, while `description` is used as a display name to let the user toggle the appearance of the annotation type.

get_annotation_data (context)

Return some metadata to be used by the `annotate_row` method below.

This will be called only once, before lines are processed. If this raises an error, that annotator won't be used.

annotate_row (context, row, number, line, data)

Return the XHTML markup for the table cell that contains the annotation data.

`context` is the context corresponding to the content being annotated, `row` is the `tr` Element being built, `number` is the line number being processed and `line` is the line's actual content. `data` is whatever additional data the `get_annotation_data` method decided to provide.

class trac.mimeview.api.IContentConverter

An extension point interface for generic MIME based content conversion.

Note

This api will likely change in the future (see [#3332](#))

See also [trac.mimeview.api.IContentConverter extension points](#)

get_supported_conversions ()

Return an iterable of tuples in the form (key, name, extension, in_mimetype, out_mimetype, quality) representing the MIME conversions supported and the quality ratio of the conversion in the range 0 to 9, where 0 means no support and 9 means "perfect" support. eg. ('latex', 'LaTeX', 'tex', 'text/x-trac-wiki', 'text/plain', 8)

convert_content (req, mimetype, content, key)

Convert the given content from mimetype to the output MIME type represented by key. Returns a tuple in the form (content, output_mime_type) or None if conversion is not possible.

Components

class trac.mimeview.api.Mimeview

Generic HTML renderer for data, typically source code.

renderers

List of components that implement `IHTMLPreviewRenderer`

annotators

List of components that implement `IHTMLPreviewAnnotator`

converters

List of components that implement `IContentConverter`

default_charset

Charset to be used when in doubt.

tab_width

Displayed tab width in file preview. ("since 0.9")

max_preview_size

Maximum file size for HTML preview. ("since 0.9")

treat_as_binary

Comma-separated list of MIME types that should be treated as binary data. ("since 0.11.5")

get_supported_conversions (mimetype)

Return a list of target MIME types in same form as `IContentConverter.get_supported_conversions()`, but with the converter component appended. Output is ordered from best to worst quality.

convert_content (req, mimetype, content, key, filename=None, url=None)

Convert the given content to the target MIME type represented by `key`, which can be either a MIME type or a key. Returns a tuple of (content, output_mime_type, extension).

get_annotation_types ()

Generator that returns all available annotation types.

render (context, mimetype, content, filename=None, url=None, annotations=None, force_source=False)

Render an XHTML preview of the given `content`.

`content` is the same as an `IHTMLPreviewRenderer.render`'s `content` argument.

The specified `mimetype` will be used to select the most appropriate `IHTMLPreviewRenderer` implementation available for this MIME type. If not given, the MIME type will be inferred from the filename or the content.

Return a string containing the XHTML text.

When rendering with an `IHTMLPreviewRenderer` fails, a warning is added to the request associated with the context (if any), unless the `disable_warnings` hint is set to `True`.

`get_max_preview_size ()`

Deprecated : use `max_preview_size` attribute directly.

`get_charset (content='', mimetype=None)`

Infer the character encoding from the `content` or the `mimetype`.
`content` is either a `str` or an `unicode` object.

The charset will be determined using this order:

- from the charset information present in the `mimetype` argument
- auto-detection of the charset from the `content`
- the configured `default_charset`

`get_mimetype (filename, content=None)`

Infer the MIME type from the `filename` or the `content`.

`content` is either a `str` or an `unicode` object.

Return the detected MIME type, augmented by the charset information (i.e. "<mimetype>; charset=..."), or `None` if detection failed.

`is_binary (mimetype=None, filename=None, content=None)`

Check if a file must be considered as binary.

`to_utf8 (content, mimetype=None)`

Convert an encoded `content` to utf-8.

Deprecated : since 0.10, you should use `unicode` strings only.

`to_unicode (content, mimetype=None, charset=None)`

Convert `content` (an encoded `str` object) to an `unicode` object.

This calls `trac.util.to_unicode` with the `charset` provided, or the one obtained by `Mimeview.get_charset()`.

`configured_modes_mapping (renderer)`

Return a MIME type to (`mode`,`quality`) mapping for given `option`

`preview_data (context, content, length, mimetype, filename, url=None, annotations=None, force_source=False)`

Prepares a rendered preview of the given `content`.

Note: `content` will usually be an object with a `read` method.

`send_converted (req, in_type, content, selector, filename='file')`

Helper method for converting `content` and sending it directly.

`selector` can be either a key or a MIME Type.

Helper classes

`class trac.mimeview.api.RenderingContext (resource, href=None, perm=None)`

A rendering context specifies "how" the content should be rendered.

It holds together all the needed contextual information that will be needed by individual renderer components.

To that end, a context keeps track of the `Href` instance (`href`) which should be used as a base for building URLs.

It also provides a `PermissionCache` (`perm`) which can be used to restrict the output so that only the authorized information is shown.

A rendering context may also be associated to some Trac resource which will be used as the implicit reference when rendering relative links or for retrieving relative content and can be used to retrieve related metadata.

Rendering contexts can be nested, and a new context can be created from an existing context using the call syntax. The previous context can be retrieved using the `parent` attribute.

For example, when rendering a wiki text of a wiki page, the context will be associated to a resource identifying that wiki page.

If that wiki text contains a `[[TicketQuery]]` wiki macro, the macro will set up nested contexts for each matching ticket that will be used for rendering the ticket descriptions.

Since : version 0.11

Directly create a `RenderingContext`.

Parameters:

- **resource** (`Resource`) -- the associated resource
- **href** -- an `Href` object suitable for creating URLs
- **perm** -- a `PermissionCache` object used for restricting the generated output to "authorized" information only.

The actual `perm` attribute of the rendering context will be bound to the given `resource` so that fine-grained permission checks will apply to that.

`parent` = `None`

The parent context, if any

`static from_request` (`*args`, `**kwargs`)

Deprecated : since 1.0, use `web_context` instead.

`child (resource=None, id=False, version=False, parent=False)`

Create a nested rendering context.

`self` will be the parent for the new nested context.

Parameters:

- **resource** -- either a `Resource` object or the realm string for a resource specification to be associated to the new context. If `None`, the resource will be the same as the resource of the parent context.
- **id** -- the identifier part of the resource specification
- **version** -- the version of the resource specification

Returns: the new context object

Return type: `RenderingContext`

```
>>> context = RenderingContext('wiki', 'WikiStart')
>>> ticket1 = Resource('ticket', 1)
>>> context.child('ticket', 1).resource == ticket1
True
>>> context.child(ticket1).resource is ticket1
True
>>> context.child(ticket1)().resource is ticket1
True
```

`set_hints (**keyvalues)`

Set rendering hints for this rendering context.

```
>>> ctx = RenderingContext('timeline')
>>> ctx.set_hints(wiki_flavor='onliner', shorten_lines=True)
>>> t_ctx = ctx('ticket', 1)
>>> t_ctx.set_hints(wiki_flavor='html', preserve_newlines=True)
>>> (t_ctx.get_hint('wiki_flavor'), t_ctx.get_hint('shorten_lines'), t_ctx.get_hint('preserve_newlines'))
('html', True, True)
>>> (ctx.get_hint('wiki_flavor'), ctx.get_hint('shorten_lines'), ctx.get_hint('preserve_newlines'))
('onliner', True, None)
```

`get_hint (hint, default=None)`

Retrieve a rendering hint from this context or an ancestor context.

```
>>> ctx = RenderingContext('timeline')
>>> ctx.set_hints(wiki_flavor='onliner')
>>> t_ctx = ctx('ticket', 1)
```

```
>>> t_ctx.get_hint('wiki_flavor')
'oneliner'
>>> t_ctx.get_hint('preserve_newlines', True)
True
```

has_hint (hint)

Test whether a rendering hint is defined in this context or in some ancestor context.

```
>>> ctx = RenderingContext('timeline')
>>> ctx.set_hints(wiki_flavor='oneliner')
>>> t_ctx = ctx('ticket', 1)
>>> t_ctx.has_hint('wiki_flavor')
True
>>> t_ctx.has_hint('preserve_newlines')
False
```

class trac.mimeview.api.**Context** (resource, href=None, perm=None)

Deprecated : old name kept for compatibility, use **RenderingContext**.

Directly create a **RenderingContext**.

Parameters:

- **resource** (**Resource**) -- the associated resource
- **href** -- an **Href** object suitable for creating URLs
- **perm** -- a **PermissionCache** object used for restricting the generated output to "authorized" information only.

The actual **perm** attribute of the rendering context will be bound to the given **resource** so that fine-grained permission checks will apply to that.

class trac.mimeview.api.**Content** (input, max_size)

A lazy file-like object that only reads **input** if necessary.

Functions

trac.mimeview.api.get_mimetype (filename, content=None, mime_map=MIME_MAP)

Guess the most probable MIME type of a file with the given name.

Parameters:

- **filename** -- is either a filename (the lookup will then use the suffix) or some arbitrary keyword.
- **content** -- is either a **str** or an **unicode** string.

trac.mimeview.api.ct_mimetype (content_type)

Return the mimetype part of a content type.

trac.mimeview.api.is_binary (data)

Detect binary content by checking the first thousand bytes for zeroes.

Operate on either **str** or **unicode** strings.

trac.mimeview.api.detect_unicode (data)

Detect different unicode charsets by looking for BOMs (Byte Order Mark).

Operate obviously only on **str** objects.

trac.mimeview.api.content_to_unicode (env, content, mimetype)

Retrieve an **unicode** object from a **content** to be previewed.

In case the raw content had an unicode BOM, we remove it.

```
>>> from trac.test import EnvironmentStub
>>> env = EnvironmentStub()
```



```
>>> content_to_unicode(env, u"\uffeffNo BOM! h\u00e9 !", '')
u'No BOM! h\xe9 !'
>>> content_to_unicode(env, "■No BOM! hé !", '')
u'No BOM! h\xe9 !'
```

trac.ticket.roadmap -- The Roadmap and Milestone modules

The **component** responsible for the *Roadmap* feature in Trac is the **RoadmapModule**. It provides an overview of the milestones and the progress in each of these milestones.

The **component** responsible for interacting with each milestone is the **MilestoneModule**. A milestone also provides an overview of the progress in terms of tickets processed.

The grouping of tickets in each progress bar is governed by the use of another component implementing the **ITicketGroupStatsProvider** interface. By default, this is the **DefaultTicketGroupStatsProvider** (for both the **RoadmapModule** and the **MilestoneModule**), which provides a configurable way to specify how tickets are grouped.

Interfaces

class trac.ticket.roadmap.**ITicketGroupStatsProvider**

See also [trac.ticket.roadmap.ITicketGroupStatsProvider extension points](#)

get_ticket_group_stats (*ticket_ids*)

Gather statistics on a group of tickets.

This method returns a valid **TicketGroupStats** object.

class trac.ticket.roadmap.**TicketGroupStats** (*title, unit*)

Encapsulates statistics on a group of tickets.

Parameters:

- **title** -- the display name of this group of stats (e.g. 'ticket status')
- **unit** -- is the units for these stats in plural form, e.g. _('hours')

add_interval (*title, count, qry_args, css_class, overall_completion=None*)

Adds a division to this stats' group's progress bar.

Parameters:

- **title** -- the display name (e.g. 'closed', 'spent effort') of this interval that will be displayed in front of the unit name
- **count** -- the number of units in the interval
- **qry_args** -- a dict of extra params that will yield the subset of tickets in this interval on a query.
- **css_class** -- is the css class that will be used to display the division
- **overall_completion** -- can be set to true to make this interval count towards overall completion of this group of tickets.

Changed in version 0.12: deprecated **countsToProg** argument was removed, use **overall_completion** instead

Components

class trac.ticket.roadmap.**MilestoneModule**

View and edit individual milestones.

stats_provider

Name of the component implementing **ITicketGroupStatsProvider**, which is used to collect statistics on groups of tickets for display in the milestone views.

```
class trac.ticket.roadmap.RoadmapModule
```

Give an overview over all the milestones.

```
stats_provider
```

Name of the component implementing `ITicketGroupStatsProvider`, which is used to collect statistics on groups of tickets for display in the roadmap views.

```
class trac.ticket.roadmap.DefaultTicketGroupStatsProvider
```

Configurable ticket group statistics provider.

See [TracIni#milestone-groups-section](#) for a detailed example configuration.

Helper Functions

```
trac.ticket.roadmap.apply_ticket_permissions (env, req, tickets)
```

Apply permissions to a set of milestone tickets as returned by `get_tickets_for_milestone()`.

```
trac.ticket.roadmap.get_tickets_for_milestone (env, db=None, milestone=None,
field='component')
```

Retrieve all tickets associated with the given `milestone`.

Changed in version 1.0: the `db` parameter is no longer needed and will be removed in version 1.1.1

```
trac.ticket.roadmap.grouped_stats_data (env, stats_provider, tickets, by,
per_group_stats_data)
```

Get the `tickets` stats data grouped by ticket field `by`.

`per_group_stats_data(gstat, group_name)` should return a data dict to include for the group with field value `group_name`.

trac.util -- General purpose utilities

The `trac.util` package is a hodgepodge of various categories of utilities. If a category contains enough code in itself, it earns a sub-module on its own, like the following ones:

trac.util.datefmt -- Date and Time manipulation

Since version 0.10, Trac mainly uses `datetime.datetime` objects for handling date and time values. This enables us to properly deal with timezones so that time can be shown in the user's own local time.

Conversion

From "anything" to a `datetime`:

```
trac.util.datefmt.to_datetime (t, tzinfo=None)
```

Convert `t` into a `datetime` object, using the following rules:

- If `t` is already a `datetime` object and `tzinfo` is `None`, it is simply returned.
- If `t` is already a `datetime` object and `tzinfo` is not `None`, it will adjust `t` to `tzinfo` timezone.
- If `t` is `None`, the current time will be used.
- If `t` is a number, it is interpreted as a timestamp.

If no `tzinfo` is given, the local timezone will be used.

Any other input will trigger a `TypeError`.

A `datetime` can be converted to milliseconds and microseconds timestamps. The latter is the preferred representation for dates and times values for storing them in the database, since Trac 0.12.

```
trac.util.datefmt.to_timestamp (dt)
```

Return the corresponding POSIX timestamp

```
trac.util.datefmt.to_utimestamp (dt)
```

Return a microsecond POSIX timestamp for the given `datetime`.

Besides `to_datetime`, there's a specialized conversion from microseconds timestamps to `datetime`:

```
trac.util.datefmt.from_utimestamp (ts)
```

Return the **datetime** for the given microsecond POSIX timestamp.

Parsing

```
trac.util.datefmt.parse_date (text, tzinfo=None, locale=None, hint='date')
```

Formatting

```
trac.util.datefmt.pretty_timedelta (time1, time2=None, resolution=None)
```

Calculate time delta between two **datetime** objects. (the result is somewhat imprecise, only use for prettyprinting).

If either **time1** or **time2** is None, the current time will be used instead.

```
trac.util.datefmt.format_datetime (t=None, format='%x %X', tzinfo=None, locale=None)
```

Format the **datetime** object **t** into an **unicode** string

If **t** is None, the current time will be used.

The formatting will be done using the given **format**, which consist of conventional **strftime** keys. In addition the format can be 'iso8601' to specify the international date format (compliant with RFC 3339).

tzinfo will default to the local timezone if left to **None**.

Derivatives:

```
trac.util.datefmt.format_date (t=None, format='%x', tzinfo=None, locale=None)
```

Convenience method for formatting the date part of a **datetime** object. See **format_datetime** for more details.

```
trac.util.datefmt.format_time (t=None, format='%X', tzinfo=None, locale=None)
```

Convenience method for formatting the time part of a **datetime** object. See **format_datetime** for more details.

Propose suggestion for date/time input format:

```
trac.util.datefmt.get_date_format_hint (locale=None)
```

Present the default format used by **format_date** in a human readable form. This is a format that will be recognized by **parse_date** when reading a date.

```
trac.util.datefmt.get_datetime_format_hint (locale=None)
```

Present the default format used by **format_datetime** in a human readable form. This is a format that will be recognized by **parse_date** when reading a date.

```
trac.util.datefmt.http_date (t=None)
```

Format **datetime** object **t** as a rfc822 timestamp

```
trac.util.datefmt.is_24_hours (locale)
```

Returns **True** for 24 hour time formats.

Formatting and parsing according to user preferences:

```
trac.util.datefmt.user_time (req, func, *args, **kwargs)
```

A helper function which passes to **tzinfo** and **locale** keyword arguments of **func** using **req** parameter. It is expected to be used with **format_*** and **parse_date** methods in **trac.util.datefmt** package.

Parameters:

- **req** -- a instance of **Request**
- **func** -- a function which must accept **tzinfo** and **locale** keyword arguments
- **args** -- arguments which pass to **func** function
- **kwargs** -- keyword arguments which pass to **func** function

jQuery UI datepicker helpers

```
trac.util.datefmt.get_date_format_jquery_ui (locale)
```

Get the date format for the jQuery UI datepicker library.

```
trac.util.datefmt.get_time_format_jquery_ui (locale)
```

Get the time format for the jQuery UI timepicker addon.

```
trac.util.datefmt.get_day_names_jquery_ui (req)
```

Get the day names for the jQuery UI datepicker library

```
trac.util.datefmt.get_first_week_day_jquery_ui (req)
```

Get first week day for jQuery date picker

```
trac.util.datefmt.get_month_names_jquery_ui (req)
```

Get the month names for the jQuery UI datepicker library

```
trac.util.datefmt.get_timezone_list_jquery_ui (t=None)
```

Get timezone list for jQuery timepicker addon

Timezone utilities

```
trac.util.datefmt.localtz
```

A global **LocalTimezone** instance.

```
class trac.util.datefmt.LocalTimezone
```

A 'local' time zone implementation

```
trac.util.datefmt.all_timezones
```

List of all available timezones. If **pytz** is installed, this corresponds to a rich variety of "official" timezones, otherwise this corresponds to **FixedOffset** instances, ranging from GMT -12:00 to GMT +13:00.

```
trac.util.datefmt.timezone (tzname)
```

Fetch timezone instance by name or raise **KeyError**

```
trac.util.datefmt.get_timezone (tzname)
```

Fetch timezone instance by name or return **None**

```
class trac.util.datefmt.FixedOffset (offset, name)
```

Fixed offset in minutes east from UTC.

trac.util.html -- HTML transformations

Building HTML programmatically

With the introduction of the **Genshi** template engine in Trac 0.11, most of the (X)HTML content is produced directly using Genshi facilities, like the **builder** or snippet templates. The old **html** tag building facility is now not much more than an alias to the tag **ElementFactory**, and most of the code uses directly the latter.

```
trac.util.html.html
```

A **TransposingElementFactory** using **str.lower** transformation.

```
class trac.util.html.TransposingElementFactory (func, namespace=None)
```

A **genshi.builder.ElementFactory** which applies **func** to the named attributes before creating a **genshi.builder.Element**.

HTML clean-up and sanitization

```
class trac.util.html.TracHTMLSanitizer (safe_schemes=frozenset(['mailto', 'ftp', 'http',
'file', 'https', None]), safe_css=frozenset(['counter-reset', 'counter-increment',
'min-height', 'quotes', 'border-top', 'font', 'list-style-image', 'outline-width',
'border-right', 'border-bottom', 'border-spacing', 'background', 'list-style-type',
'text-align', 'page-break-inside', 'orphans', 'page-break-before', 'text-transform',
'line-height', 'padding-left', 'font-size', 'right', 'word-spacing', 'padding-top',
'outline-style', 'bottom', 'content', 'border-right-style', 'padding-right',
'border-left-style', 'background-color', 'border-bottom-color', 'outline-color',
'unicode-bidi', 'max-width', 'font-family', 'caption-side', 'border-right-width',
'border-top-style', 'color', 'border-collapse', 'border-bottom-width', 'float', 'height',
'max-height', 'margin-right', 'border-top-width', 'top', 'border-width', 'min-width',
```

```
'width', 'font-variant', 'border-top-color', 'background-position', 'empty-cells',
'direction', 'border-left', 'visibility', 'padding', 'border-style',
'background-attachment', 'overflow', 'border-bottom-style', 'cursor', 'margin', 'display',
'border-left-width', 'letter-spacing', 'vertical-align', 'clip', 'border-color',
'list-style', 'padding-bottom', 'margin-left', 'widows', 'border', 'font-style',
'border-left-color', 'background-repeat', 'table-layout', 'margin-bottom', 'font-weight',
'opacity', 'border-right-color', 'page-break-after', 'white-space', 'text-indent',
'background-image', 'outline', 'clear', 'z-index', 'text-decoration', 'margin-top',
'position', 'left', 'list-style-position']))
```

Sanitize HTML constructions which are potentially vector of phishing or XSS attacks, in user-supplied HTML.

See also [genshi.HTMLSanitizer](#).

class `trac.util.html.Deuglifier`

Help base class used for cleaning up HTML riddled with `` tags and replace them with appropriate ``.

The subclass must define a **rules()** static method returning a list of regular expression fragments, each defining a capture group in which the name will be reused for the span's class. Two special group names, `font` and `endfont` are used to emit `` and ``, respectively.

See some usage examples in `tracopt.mimeview.enscript.EnscriptDeuglifier` and `tracopt.mimeview.php.PhpDeuglifier`.

trac.util.html.escape (*cls*, *text*, *quotes=True*)

Create a Markup instance from a string and escape special characters it may contain (<, >, & and ").

```
>>> escape('1 < 2')
<Markup u'&#34;1 &lt; 2&#34;'>
```

If the **quotes** parameter is set to **False**, the " character is left as is. Escaping quotes is generally only required for strings that are to be used in attribute values.

```
>>> escape('1 < 2', quotes=False)
<Markup u'"1 &lt; 2"'>
```

Parameters:

- **text** -- the text to escape
- **quotes** -- if **True**, double quote characters are escaped in addition to the other special characters

Returns: the escaped **Markup** string

Return type: **Markup**

trac.util.html.unescape (*text*)

Reverse-escapes &, <, >, and " and returns a **unicode** object.

```
>>> unescape(Markup('1 &lt; 2'))
u'1 < 2'
```

If the provided **text** object is not a **Markup** instance, it is returned unchanged.

```
>>> unescape('1 &lt; 2')
'1 &lt; 2'
```

Parameters: **text** -- the text to unescape

Returns: the unescaped string

Return type: **unicode**

class `trac.util.html.FormTokenInjector` (*form_token*, *out*)

Identify and protect forms from CSRF attacks.

This filter works by adding a input type=hidden field to POST forms.

Misc. HTML processing

`trac.util.html.expand_markup (stream, ctxt=None)`

A Genshi stream filter for expanding `genshi.Markup` events.

Note: Expansion may not be possible if the fragment is badly formed, or partial.

`trac.util.html.find_element (frag, attr=None, cls=None)`

Return the first element in the fragment having the given attribute or class, using a preorder depth-first search.

`trac.util.html.plaintext (text, keeplinebreaks=True)`

Extract the text elements from (X)HTML content

Parameters:

- **text** -- unicode or `genshi.builder.Fragment`
- **keeplinebreaks** -- optionally keep linebreaks

trac.util.presentation -- Utilities for dynamic content generation

The following utilities are all available within Genshi templates.

`trac.util.presentation.captions_button (req, symbol, text)`

Return symbol and text or only symbol, according to user preferences.

`trac.util.presentation.classes (*args, **kwargs)`

Helper function for dynamically assembling a list of CSS class names in templates.

Any positional arguments are added to the list of class names. All positional arguments must be strings:

```
>>> classes('foo', 'bar')
u'foo bar'
```

In addition, the names of any supplied keyword arguments are added if they have a truth value:

```
>>> classes('foo', bar=True)
u'foo bar'
>>> classes('foo', bar=False)
u'foo'
```

If none of the arguments are added to the list, this function returns **None**:

```
>>> classes(bar=False)
```

`trac.util.presentation.first_last (idx, seq)`

Generate first or last or both, according to the position **idx** in sequence **seq**.

`trac.util.presentation.group (iterable, num, predicate=None)`

Combines the elements produced by the given iterable so that every **n** items are returned as a tuple.

```
>>> items = [1, 2, 3, 4]
>>> for item in group(items, 2):
...     print item
(1, 2)
(3, 4)
```

The last tuple is padded with **None** values if its' length is smaller than **num**.

```
>>> items = [1, 2, 3, 4, 5]
>>> for item in group(items, 2):
...     print item
(1, 2)
(3, 4)
(5, None)
```

The optional **predicate** parameter can be used to flag elements that should not be packed together with other items. Only those elements where the predicate function returns True are grouped with other elements, otherwise they are returned as a tuple of length 1:

```
>>> items = [1, 2, 3, 4]
>>> for item in group(items, 2, lambda x: x != 3):
...     print item
(1, 2)
(3,)
(4, None)
```

`trac.util.presentation.istext (text)`

True for text (**unicode** and **str**), but **False** for **Markup**.

`trac.util.presentation.paginate (items, page=0, max_per_page=10)`

Simple generic pagination.

Given an iterable, this function returns:

- the slice of objects on the requested page,
- the total number of items, and
- the total number of pages.

The **items** parameter can be a list, tuple, or iterator:

```
>>> items = range(12)
>>> items
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
>>> paginate(items)
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 12, 2)
>>> paginate(items, page=1)
([10, 11], 12, 2)
>>> paginate(iter(items))
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 12, 2)
>>> paginate(iter(items), page=1)
([10, 11], 12, 2)
```

This function also works with generators:

```
>>> def generate():
...     for idx in range(12):
...         yield idx
>>> paginate(generate())
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 12, 2)
>>> paginate(generate(), page=1)
([10, 11], 12, 2)
```

The **max_per_page** parameter can be used to set the number of items that should be displayed per page:

```
>>> items = range(12)
>>> paginate(items, page=0, max_per_page=6)
([0, 1, 2, 3, 4, 5], 12, 2)
>>> paginate(items, page=1, max_per_page=6)
([6, 7, 8, 9, 10, 11], 12, 2)
```

`trac.util.presentation.separated (items, sep=', ')`

Yield (**item**, **sep**) tuples, one for each element in **items**.

sep will be **None** for the last item.

```
>>> list(separated([1, 2]))
[(1, ','), (2, None)]
```

```
>>> list(separated([1]))
[(1, None)]
```

```
>>> list(separated("abc", ':'))
[('a', ':'), ('b', ':'), ('c', None)]
```

`trac.util.presentation.to_json` (*value*)

Encode **value** to JSON.

Modules generating paginated output will be happy to use a rich pagination controller. See *Query*, *Report* and *Search* modules for example usage.

`class trac.util.presentation.Paginator` (*items*, *page*=0, *max_per_page*=10, *num_items*=None)

Pagination controller

trac.util.text -- Text manipulation

The Unicode toolbox

Trac internals are almost exclusively dealing with Unicode text, represented by **unicode** instances. The main advantage of using **unicode** over UTF-8 encoded **str** (as this used to be the case before version 0.10), is that text transformation functions in the present module will operate in a safe way on individual characters, and won't risk to eventually cut a multi-byte sequence in the middle. Similar issues with Python string handling routines are avoided as well, like surprising results when splitting text in lines. For example, did you know that "Priorità" is encoded as 'Priorit\xc3\x0a' in UTF-8? Calling **strip()** on this value in some locales can cut away the trailing `\x0a` and it's no longer valid UTF-8...

The drawback is that most of the outside world, while eventually "Unicode", is definitely not **unicode**. This is why we need to convert back and forth between **str** and **unicode** at the boundaries of the system. And more often than not we even have to guess which encoding is used in the incoming **str** strings.

Encoding **unicode** to **str** is usually directly performed by calling **encode()** on the **unicode** instance, while decoding is preferably left to the **to_unicode** helper function, which converts **str** to **unicode** in a robust and guaranteed successful way.

`trac.util.text.to_unicode` (*text*, *charset*=None)

Convert input to an **unicode** object.

For a **str** object, we'll first try to decode the bytes using the given **charset** encoding (or UTF-8 if none is specified), then we fall back to the latin1 encoding which might be correct or not, but at least preserves the original byte sequence by mapping each byte to the corresponding unicode code point in the range U+0000 to U+00FF.

For anything else, a simple **unicode()** conversion is attempted, with special care taken with **Exception** objects.

`trac.util.text.exception_to_unicode` (*e*, *traceback*=False)

Convert an **Exception** to an **unicode** object.

In addition to **to_unicode**, this representation of the exception also contains the class name and optionally the traceback.

Web utilities

`trac.util.text.unicode_quote` (*value*, *safe*='/')

A unicode aware version of **urllib.quote**

Parameters:

- **value** -- anything that converts to a **str**. If **unicode** input is given, it will be UTF-8 encoded.
- **safe** -- as in **quote**, the characters that would otherwise be quoted but shouldn't here (defaults to '/')

`trac.util.text.unicode_quote_plus` (*value*, *safe*='')

A unicode aware version of **urllib.quote_plus**.

Parameters:

- **value** -- anything that converts to a **str**. If **unicode** input is given, it will be UTF-8 encoded.
- **safe** -- as in **quote_plus**, the characters that would otherwise be quoted but shouldn't here (defaults to '/')

`trac.util.text.unicode_unquote (value)`

A unicode aware version of **urllib.unquote**.

Parameters: **str** -- UTF-8 encoded **str** value (for example, as obtained by **unicode_quote**).

Return type: **unicode**

`trac.util.text.unicode_urlencode (params, safe='')`

A unicode aware version of **urllib.urlencode**.

Values set to **empty** are converted to the key alone, without the equal sign.

`trac.util.text.quote_query_string (text)`

Quote strings for query string

`trac.util.text.javascript_quote (text)`

Quote strings for inclusion in single or double quote delimited Javascript strings

`trac.util.text.to_js_string (text)`

Embed the given string in a double quote delimited Javascript string (conform to the JSON spec)

Console and file system

`trac.util.text.path_to_unicode (path)`

Convert a filesystem path to unicode, using the filesystem encoding.

`trac.util.text.stream_encoding (stream)`

Return the appropriate encoding for the given stream.

`trac.util.text.console_print (out, *args, **kwargs)`

Output the given arguments to the console, encoding the output as appropriate.

Parameters: **kwargs** -- **newline** controls whether a newline will be appended (defaults to **True**)

`trac.util.text.printout (*args, **kwargs)`

Do a **console_print** on **sys.stdout**.

`trac.util.text.printerr (*args, **kwargs)`

Do a **console_print** on **sys.stderr**.

`trac.util.text.raw_input (prompt)`

Input one line from the console and converts it to unicode as appropriate.

Miscellaneous

`trac.util.text.empty`

A special tag object evaluating to the empty string, used as marker for missing value (as opposed to a present but empty value).

`class trac.util.text.unicode_passwd`

Conceal the actual content of the string when **repr** is called.

`trac.util.text.levenshtein_distance (lhs, rhs)`

Return the Levenshtein distance between two strings.

Text formatting

`trac.util.text.pretty_size (size, format='%.1f')`

Pretty print content size information with appropriate unit.

Parameters:

- **size** -- number of bytes
- **format** -- can be used to adjust the precision shown

`trac.util.text.breakable_path (path)`

Make a path breakable after path separators, and conversely, avoid breaking at spaces.

`trac.util.text.normalize_whitespace (text, to_space=u'\xa0', remove=u'\u200b')`

Normalize whitespace in a string, by replacing special spaces by normal spaces and removing zero-width spaces.

`trac.util.text.unquote_label (txt)`

Remove (one level of) enclosing single or double quotes.

New in version 1.0.

`trac.util.text.fix_eol (text, eol)`

Fix end-of-lines in a text.

`trac.util.text.expandtabs (s, tabstop=8, ignoring=None)`

Expand tab characters '\t' into spaces.

Parameters:

- **tabstop** -- number of space characters per tab (defaults to the canonical 8)
- **ignoring** -- if not **None**, the expansion will be "smart" and go from one tabstop to the next. In addition, this parameter lists characters which can be ignored when computing the indent.

`trac.util.text.obfuscate_email_address (address)`

Replace anything looking like an e-mail address ('@something') with a trailing ellipsis ('@...')

`trac.util.text.text_width (text, ambiwidth=1)`

Determine the column width of `text` in Unicode characters.

The characters in the East Asian Fullwidth (F) or East Asian Wide (W) have a column width of 2. The other characters in the East Asian Halfwidth (H) or East Asian Narrow (Na) have a column width of 1.

That `ambiwidth` parameter is used for the column width of the East Asian Ambiguous (A). If 1, the same width as characters in US-ASCII. This is expected by most users. If 2, twice the width of US-ASCII characters. This is expected by CJK users.

cf. <http://www.unicode.org/reports/tr11/>.

`trac.util.text.print_table (data, headers=None, sep=' ', out=None, ambiwidth=None)`

Print data according to a tabular layout.

Parameters:

- **data** -- a sequence of rows; assume all rows are of equal length.
- **headers** -- an optional row containing column headers; must be of the same length as each row in `data`.
- **sep** -- column separator
- **out** -- output file descriptor (**None** means use `sys.stdout`)
- **ambiwidth** -- column width of the East Asian Ambiguous (A). If **None**, detect `ambiwidth` with the locale settings. If others, pass to the `ambiwidth` parameter of `text_width`.

`trac.util.text.shorten_line (text, maxlen=75)`

Truncates content to at most `maxlen` characters.

This tries to be (a bit) clever and attempts to find a proper word boundary for doing so.

`trac.util.text.stripws (text, leading=True, trailing=True)`

Strips unicode white-spaces and ZWSPs from `text`.

Parameters:

- **leading** -- strips leading spaces from `text` unless `leading` is **False**.
- **trailing** -- strips trailing spaces from `text` unless `trailing` is **False**.

`trac.util.text.wrap (t, cols=75, initial_indent='', subsequent_indent='', linesep='\n', ambiwidth=1)`

Wraps the single paragraph in `t`, which contains unicode characters. The every line is at most `cols` characters long.

That `ambiwidth` parameter is used for the column width of the East Asian Ambiguous (A). If `1`, the same width as characters in US-ASCII. This is expected by most users. If `2`, twice the width of US-ASCII characters. This is expected by CJK users.

Conversion utilities

`trac.util.text.unicode_to_base64 (text, strip_newlines=True)`

Safe conversion of `text` to base64 representation using utf-8 bytes.

Strips newlines from output unless `strip_newlines` is `False`.

`trac.util.text.unicode_from_base64 (text)`

Safe conversion of `text` to unicode based on utf-8 bytes.

`trac.util.text.to_utf8 (text, charset='latin1')`

Convert a string to UTF-8, assuming the encoding is either UTF-8, ISO Latin-1, or as specified by the optional `charset` parameter.

Deprecated since version 0.10: You should use `unicode` strings only.

Otherwise, the functions are direct members of the `trac.util` package (i.e. placed in the "`__init__.py`" file).

Web related utilities

`trac.util.get_reporter_id (req, arg_name=None)`

Get most informative "reporter" identity out of a request.

That's the `Request`'s `authname` if not 'anonymous', or a `Request` argument, or the session name and e-mail, or only the name or only the e-mail, or 'anonymous' as last resort.

Parameters:

- `req` -- a `trac.web.api.Request`
- `arg_name` -- if given, a `Request` argument which may contain the id for non-authenticated users

`trac.util.content_disposition (type=None, filename=None)`

Generate a properly escaped Content-Disposition header.

OS related utilities

`trac.util.copytree (src, dst, symlinks=False, skip=[], overwrite=False)`

Recursively copy a directory tree using `copy2()` (from `shutil.copytree`.)

Added a `skip` parameter consisting of absolute paths which we don't want to copy.

`trac.util.create_file (path, data='', mode='w')`

Create a new file with the given data.

`trac.util.create_unique_file (path)`

Create a new file. An index is added if the path exists

`trac.util.getuser ()`

Retrieve the identity of the process owner

`trac.util.is_path_below (path, parent)`

Return `True` iff `path` is equal to `parent` or is located below `parent` at any level.

`trac.util.makedirs (path, overwrite=False)`

Create as many directories as necessary to make `path` exist.

If `overwrite` is `True`, don't raise an exception in case `path` already exists.

`trac.util.read_file (path, mode='r')`

Read a file and return its content.

`trac.util.rename (old, new)`

Rename a file or directory.

`class trac.util.AtomicFile (path, mode='w', bufsize=-1)`

A file that appears atomically with its full content.

This file-like object writes to a temporary file in the same directory as the final file. If the file is committed, the temporary file is renamed atomically (on Unix, at least) to its final name. If it is rolled back, the temporary file is removed.

`class trac.util.NaivePopen (command, input=None, capturestderr=None)`

This is a deadlock-safe version of `popen` that returns an object with `errorlevel`, `out` (a string) and `err` (a string).

The optional `input`, which must be a `str` object, is first written to a temporary file from which the process will read.

(`capturestderr` may not work under Windows 9x.)

Example:

```
print Popen3('grep spam', '\n\nhere spam\n\n').out
```

`class trac.util.WindowsError`

Dummy exception replacing `WindowsError` on non-Windows platforms

Also defined on non-Windows systems (by a dummy `OSError` subclass).

`class trac.util.file_or_std (filename, mode='r', bufsize=-1)`

Context manager for opening a file or using a standard stream

If `filename` is non-empty, open the file and close it when exiting the block. Otherwise, use `sys.stdin` if opening for reading, or `sys.stdout` if opening for writing or appending.

`trac.util.urandom`

The standard `os.urandom` when available, otherwise a reasonable replacement.

Python "system" utilities

Complements the `inspect`, `traceback` and `sys` modules.

`trac.util.fq_class_name (obj)`

Return the fully qualified class name of given object.

`trac.util.arity (f)`

Return the number of arguments expected by the given function, unbound or bound method.

`trac.util.get_last_traceback ()`

Retrieve the last traceback as an `unicode` string.

`trac.util.get_lines_from_file (filename, lineno, context=0, globals=None)`

Return `content` number of lines before and after the specified `lineno` from the (source code) file identified by `filename`.

Returns a `(lines_before, line, lines_after)` tuple.

`trac.util.get_frame_info (tb)`

Return frame information for a traceback.

`trac.util.import_namespace (globals_dict, module_name)`

Import the namespace of a module into a `globals` dict.

This function is used in stub modules to import all symbols defined in another module into the global namespace of the stub, usually for backward compatibility.

`trac.util.safe_import__ (module_name)`

Safe imports: rollback after a failed import.

Initially inspired from the `RollbackImporter` in `PyUnit`, but it's now much simpler and works better for our needs.

See <http://pyunit.sourceforge.net/notes/reloading.html>

`trac.util.safe_repr (x)`

`repr` replacement which "never" breaks.

Make sure we always get a representation of the input `x` without risking to trigger an exception (e.g. from a buggy `x.__repr__`).

New in version 1.0.

`trac.util.get_doc (obj)`

Return the docstring of an object as a tuple (**summary**, **description**), where **summary** is the first paragraph and **description** is the remaining text.

Setuptools utilities

`trac.util.get_module_path (module)`

Return the base path the given module is imported from

`trac.util.get_sources (path)`

Return a dictionary mapping Python module source paths to the distributions that contain them.

`trac.util.get_pkginfo (dist)`

Get a dictionary containing package information for a package

dist can be either a Distribution instance or, as a shortcut, directly the module instance, if one can safely infer a Distribution instance from it.

Always returns a dictionary but it will be empty if no Distribution instance can be created for the given module.

Cryptographic related utilities

`trac.util.hex_entropy (digits=32)`

Generate **digits** number of hex digits of entropy.

`trac.util.md5crypt (password, salt, magic='1')`

Based on FreeBSD src/lib/libcrypt/crypt.c 1.2

Parameters:

- **password** -- the plain text password to crypt
- **salt** -- the raw salt
- **magic** -- our magic string

Data structures which don't fit anywhere else

`class trac.util.Ranges (r=None, reorder=False)`

Holds information about ranges parsed from a string

Author : Tim Hatch

```
>>> x = Ranges("1,2,9-15")
>>> 1 in x
True
>>> 5 in x
False
>>> 10 in x
True
>>> 16 in x
False
>>> [i for i in range(20) if i in x]
[1, 2, 9, 10, 11, 12, 13, 14, 15]
```

Also supports iteration, which makes that last example a bit simpler:

```
>>> list(x)
[1, 2, 9, 10, 11, 12, 13, 14, 15]
```

Note that it automatically reduces the list and short-circuits when the desired ranges are a relatively small portion of the entire set:

```
>>> x = Ranges("99")
>>> 1 in x # really fast
```

```
False
>>> x = Ranges("1, 2, 1-2, 2") # reduces this to 1-2
>>> x.pairs
[(1, 2)]
>>> x = Ranges("1-9,2-4") # handle ranges that completely overlap
>>> list(x)
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

The members 'a' and 'b' refer to the min and max value of the range, and are None if the range is empty:

```
>>> x.a
1
>>> x.b
9
>>> e = Ranges()
>>> e.a, e.b
(None, None)
```

Empty ranges are ok, and ranges can be constructed in pieces, if you so choose:

```
>>> x = Ranges()
>>> x.appendrange("1, 2, 3")
>>> x.appendrange("5-9")
>>> x.appendrange("2-3") # reduce'd away
>>> list(x)
[1, 2, 3, 5, 6, 7, 8, 9]
```

Reversed ranges are ignored, unless the Ranges has the **reorder** property set.

```
>>> str(Ranges("20-10"))
''
>>> str(Ranges("20-10", reorder=True))
'10-20'
```

As rendered ranges are often using u',u200b' (comma + Zero-width space) to enable wrapping, we also support reading such ranges, as they can be copy/pasted back.

```
>>> str(Ranges(u'1,\u200b3,\u200b5,\u200b6,\u200b7,\u200b9'))
'1,3,5-7,9'
```

appendrange (r)

Add ranges to the current one.

A range is specified as a string of the form "low-high", and **r** can be a list of such strings, a string containing comma-separated ranges, or **None**.

truncate (max)

Truncate the Ranges by setting a maximal allowed value.

Note that this **max** can be a value in a gap, so the only guarantee is that **self.b** will be lesser than or equal to **max**.

```
>>> r = Ranges("10-20,25-45")
>>> str(r.truncate(30))
'10-20,25-30'
```

```
>>> str(r.truncate(22))
'10-20'
```

```
>>> str(r.truncate(10))
'10'
```

`trac.util.to_ranges` (*revs*)

Converts a list of revisions to a minimal set of ranges.

```
>>> to_ranges([2, 12, 3, 6, 9, 1, 5, 11])
'1-3,5-6,9,11-12'
>>> to_ranges([])
''
```

`class trac.util.lazy` (*fn*)

A lazily-evaluated attribute

Algorithmic utilities

`trac.util.embedded_numbers` (*s*)

Comparison function for natural order sorting based on
<http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/214202>.

`trac.util.pairwise` (*iterable*)

```
>>> list(pairwise([0, 1, 2, 3]))
[(0, 1), (1, 2), (2, 3)]
```

Deprecated since version 0.11: if this really needs to be used, rewrite it without `izip`

`trac.util.partition` (*iterable*, *order=None*)

```
>>> partition([(1, "a"), (2, "b"), (3, "a")])
{'a': [1, 3], 'b': [2]}
>>> partition([(1, "a"), (2, "b"), (3, "a")], "ab")
[[1, 3], [2]]
```

`trac.util.as_int` (*s*, *default*, *min=None*, *max=None*)

Convert *s* to an int and limit it to the given range, or return default if unsuccessful.

`trac.util.as_bool` (*value*)

Convert the given value to a **bool**.

If **value** is a string, return **True** for any of "yes", "true", "enabled", "on" or non-zero numbers, ignoring case. For non-string arguments, return the argument converted to a **bool**, or **False** if the conversion fails.

`trac.util.pathjoin` (**args*)

Strip / from the arguments and join them with a single /.

`trac.util.datefmt` -- Date and Time manipulation

Since version 0.10, Trac mainly uses `datetime.datetime` objects for handling date and time values. This enables us to properly deal with timezones so that time can be shown in the user's own local time.

Conversion

From "anything" to a `datetime`:

`trac.util.datefmt.to_datetime` (*t*, *tzinfo=None*)

Convert *t* into a `datetime` object, using the following rules:

- If *t* is already a `datetime` object and *tzinfo* is `None`, it is simply returned.
- If *t* is already a `datetime` object and *tzinfo* is not `None`, it will adjust *t* to *tzinfo* timezone.

- If `t` is `None`, the current time will be used.
- If `t` is a number, it is interpreted as a timestamp.

If no `tzinfo` is given, the local timezone will be used.

Any other input will trigger a `TypeError`.

A `datetime` can be converted to milliseconds and microseconds timestamps. The latter is the preferred representation for dates and times values for storing them in the database, since Trac 0.12.

`trac.util.datefmt.to_timestamp(dt)`

Return the corresponding POSIX timestamp

`trac.util.datefmt.to_utimestamp(dt)`

Return a microsecond POSIX timestamp for the given `datetime`.

Besides `to_datetime`, there's a specialized conversion from microseconds timestamps to `datetime`:

`trac.util.datefmt.from_utimestamp(ts)`

Return the `datetime` for the given microsecond POSIX timestamp.

Parsing

`trac.util.datefmt.parse_date(text, tzinfo=None, locale=None, hint='date')`

Formatting

`trac.util.datefmt.pretty_timedelta(time1, time2=None, resolution=None)`

Calculate time delta between two `datetime` objects. (the result is somewhat imprecise, only use for prettyprinting).

If either `time1` or `time2` is `None`, the current time will be used instead.

`trac.util.datefmt.format_datetime(t=None, format='%x %X', tzinfo=None, locale=None)`

Format the `datetime` object `t` into an `unicode` string

If `t` is `None`, the current time will be used.

The formatting will be done using the given `format`, which consist of conventional `strftime` keys. In addition the format can be 'iso8601' to specify the international date format (compliant with RFC 3339).

`tzinfo` will default to the local timezone if left to `None`.

Derivatives:

`trac.util.datefmt.format_date(t=None, format='%x', tzinfo=None, locale=None)`

Convenience method for formatting the date part of a `datetime` object. See `format_datetime` for more details.

`trac.util.datefmt.format_time(t=None, format='%X', tzinfo=None, locale=None)`

Convenience method for formatting the time part of a `datetime` object. See `format_datetime` for more details.

Propose suggestion for date/time input format:

`trac.util.datefmt.get_date_format_hint(locale=None)`

Present the default format used by `format_date` in a human readable form. This is a format that will be recognized by `parse_date` when reading a date.

`trac.util.datefmt.get_datetime_format_hint(locale=None)`

Present the default format used by `format_datetime` in a human readable form. This is a format that will be recognized by `parse_date` when reading a date.

`trac.util.datefmt.http_date(t=None)`

Format `datetime` object `t` as a rfc822 timestamp

`trac.util.datefmt.is_24_hours(locale)`

Returns `True` for 24 hour time formats.

Formatting and parsing according to user preferences:

`trac.util.datefmt.user_time(req, func, *args, **kwargs)`

A helper function which passes to `tzinfo` and `locale` keyword arguments of `func` using `req` parameter. It is expected to be used with `format_*` and `parse_date` methods in `trac.util.datefmt` package.

Parameters:

- `req` -- a instance of `Request`
- `func` -- a function which must accept `tzinfo` and `locale` keyword arguments
- `args` -- arguments which pass to `func` function
- `kwargs` -- keyword arguments which pass to `func` function

jQuery UI datepicker helpers

```
trac.util.datefmt.get_date_format_jquery_ui (locale)
    Get the date format for the jQuery UI datepicker library.

trac.util.datefmt.get_time_format_jquery_ui (locale)
    Get the time format for the jQuery UI timepicker addon.

trac.util.datefmt.get_day_names_jquery_ui (req)
    Get the day names for the jQuery UI datepicker library

trac.util.datefmt.get_first_week_day_jquery_ui (req)
    Get first week day for jQuery date picker

trac.util.datefmt.get_month_names_jquery_ui (req)
    Get the month names for the jQuery UI datepicker library

trac.util.datefmt.get_timezone_list_jquery_ui (t=None)
    Get timezone list for jQuery timepicker addon
```

Timezone utilities

```
trac.util.datefmt.localtz
    A global LocalTimezone instance.

class trac.util.datefmt.LocalTimezone
    A 'local' time zone implementation

trac.util.datefmt.all_timezones
    List of all available timezones. If pytz is installed, this corresponds to a rich variety of "official" timezones, otherwise this corresponds to FixedOffset instances, ranging from GMT -12:00 to GMT +13:00.

trac.util.datefmt.timezone (tzname)
    Fetch timezone instance by name or raise KeyError

trac.util.datefmt.get_timezone (tzname)
    Fetch timezone instance by name or return None

class trac.util.datefmt.FixedOffset (offset, name)
    Fixed offset in minutes east from UTC.
```

trac.util.html -- HTML transformations

Building HTML programmatically

With the introduction of the `Genshi` template engine in Trac 0.11, most of the (X)HTML content is produced directly using `Genshi` facilities, like the `builder` or snippet templates. The old `html` tag building facility is now not much more than an alias to the tag `ElementFactory`, and most of the code uses directly the latter.

```
trac.util.html.html
    A TransposingElementFactory using str.lower transformation.
```

```
class trac.util.html.TransposingElementFactory (func, namespace=None)
```

A `genshi.builder.ElementFactory` which applies `func` to the named attributes before creating a `genshi.builder.Element`.

HTML clean-up and sanitization

```
class trac.util.html.TracHTMLSanitizer (safe_schemes=frozenset(['mailto', 'ftp', 'http',
'file', 'https', None]), safe_css=frozenset(['counter-reset', 'counter-increment',
'min-height', 'quotes', 'border-top', 'font', 'list-style-image', 'outline-width',
'border-right', 'border-bottom', 'border-spacing', 'background', 'list-style-type',
'text-align', 'page-break-inside', 'orphans', 'page-break-before', 'text-transform',
'line-height', 'padding-left', 'font-size', 'right', 'word-spacing', 'padding-top',
'outline-style', 'bottom', 'content', 'border-right-style', 'padding-right',
'border-left-style', 'background-color', 'border-bottom-color', 'outline-color',
'unicode-bidi', 'max-width', 'font-family', 'caption-side', 'border-right-width',
'border-top-style', 'color', 'border-collapse', 'border-bottom-width', 'float', 'height',
'max-height', 'margin-right', 'border-top-width', 'top', 'border-width', 'min-width',
'width', 'font-variant', 'border-top-color', 'background-position', 'empty-cells',
'direction', 'border-left', 'visibility', 'padding', 'border-style',
'background-attachment', 'overflow', 'border-bottom-style', 'cursor', 'margin', 'display',
'border-left-width', 'letter-spacing', 'vertical-align', 'clip', 'border-color',
'list-style', 'padding-bottom', 'margin-left', 'widows', 'border', 'font-style',
'border-left-color', 'background-repeat', 'table-layout', 'margin-bottom', 'font-weight',
'opacity', 'border-right-color', 'page-break-after', 'white-space', 'text-indent',
'background-image', 'outline', 'clear', 'z-index', 'text-decoration', 'margin-top',
'position', 'left', 'list-style-position']))
```

Sanitize HTML constructions which are potentially vector of phishing or XSS attacks, in user-supplied HTML.

See also [genshi.HTMLSanitizer](#).

```
class trac.util.html.Deuglifier
```

Help base class used for cleaning up HTML riddled with `` tags and replace them with appropriate ``.

The subclass must define a `rules()` static method returning a list of regular expression fragments, each defining a capture group in which the name will be reused for the span's class. Two special group names, `font` and `endfont` are used to emit `` and ``, respectively.

See some usage examples in `tracopt.mimeview.enscript.EnscriptDeuglifier` and `tracopt.mimeview.php.PhpDeuglifier`.

```
trac.util.html.escape (cls, text, quotes=True)
```

Create a Markup instance from a string and escape special characters it may contain (<, >, & and ").

```
>>> escape('"1 < 2"')
<Markup u'&#34;1 &lt; 2&#34;'>
```

If the `quotes` parameter is set to `False`, the `"` character is left as is. Escaping quotes is generally only required for strings that are to be used in attribute values.

```
>>> escape('"1 < 2"', quotes=False)
<Markup u'"1 &lt; 2"'>
```

Parameters:

- **text** -- the text to escape
- **quotes** -- if `True`, double quote characters are escaped in addition to the other special characters

Returns: the escaped `Markup` string

Return type: `Markup`

```
trac.util.html.unescape (text)
```

Reverse-escapes `&`, `<`, `>`, and `"` and returns a `unicode` object.

```
>>> unescape(Markup('1 &lt; 2'))
u'1 < 2'
```

If the provided **text** object is not a **Markup** instance, it is returned unchanged.

```
>>> unescape('1 &lt; 2')
'1 &lt; 2'
```

Parameters: **text** -- the text to unescape

Returns: the unescaped string

Return type: **unicode**

`class trac.util.html.FormTokenInjector (form_token, out)`

Identify and protect forms from CSRF attacks.

This filter works by adding a input type=hidden field to POST forms.

Misc. HTML processing

`trac.util.html.expand_markup (stream, ctxt=None)`

A Genshi stream filter for expanding **genshi.Markup** events.

Note: Expansion may not be possible if the fragment is badly formed, or partial.

`trac.util.html.find_element (frag, attr=None, cls=None)`

Return the first element in the fragment having the given attribute or class, using a preorder depth-first search.

`trac.util.html.plaintext (text, keeplinebreaks=True)`

Extract the text elements from (X)HTML content

Parameters:

- **text** -- **unicode** or **genshi.builder.Fragment**
- **keeplinebreaks** -- optionally keep linebreaks

trac.util.presentation -- Utilities for dynamic content generation

The following utilities are all available within Genshi templates.

`trac.util.presentation.captioned_button (req, symbol, text)`

Return symbol and text or only symbol, according to user preferences.

`trac.util.presentation.classes (*args, **kwargs)`

Helper function for dynamically assembling a list of CSS class names in templates.

Any positional arguments are added to the list of class names. All positional arguments must be strings:

```
>>> classes('foo', 'bar')
u'foo bar'
```

In addition, the names of any supplied keyword arguments are added if they have a truth value:

```
>>> classes('foo', bar=True)
u'foo bar'
>>> classes('foo', bar=False)
u'foo'
```

If none of the arguments are added to the list, this function returns **None**:

```
>>> classes(bar=False)
```

`trac.util.presentation.first_last (idx, seq)`

Generate first or last or both, according to the position **idx** in sequence **seq**.

`trac.util.presentation.group (iterable, num, predicate=None)`

Combines the elements produced by the given iterable so that every **n** items are returned as a tuple.

```
>>> items = [1, 2, 3, 4]
>>> for item in group(items, 2):
...     print item
(1, 2)
(3, 4)
```

The last tuple is padded with **None** values if its' length is smaller than **num**.

```
>>> items = [1, 2, 3, 4, 5]
>>> for item in group(items, 2):
...     print item
(1, 2)
(3, 4)
(5, None)
```

The optional **predicate** parameter can be used to flag elements that should not be packed together with other items. Only those elements where the predicate function returns True are grouped with other elements, otherwise they are returned as a tuple of length 1:

```
>>> items = [1, 2, 3, 4]
>>> for item in group(items, 2, lambda x: x != 3):
...     print item
(1, 2)
(3,)
(4, None)
```

trac.util.presentation.istext (*text*)

True for text (**unicode** and **str**), but **False** for **Markup**.

trac.util.presentation.paginate (*items*, *page=0*, *max_per_page=10*)

Simple generic pagination.

Given an iterable, this function returns:

- the slice of objects on the requested page,
- the total number of items, and
- the total number of pages.

The **items** parameter can be a list, tuple, or iterator:

```
>>> items = range(12)
>>> items
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
>>> paginate(items)
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 12, 2)
>>> paginate(items, page=1)
([10, 11], 12, 2)
>>> paginate(iter(items))
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 12, 2)
>>> paginate(iter(items), page=1)
([10, 11], 12, 2)
```

This function also works with generators:

```
>>> def generate():
...     for idx in range(12):
...         yield idx
>>> paginate(generate())
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 12, 2)
>>> paginate(generate(), page=1)
```

```
([10, 11], 12, 2)
```

The `max_per_page` parameter can be used to set the number of items that should be displayed per page:

```
>>> items = range(12)
>>> paginate(items, page=0, max_per_page=6)
([0, 1, 2, 3, 4, 5], 12, 2)
>>> paginate(items, page=1, max_per_page=6)
([6, 7, 8, 9, 10, 11], 12, 2)
```

`trac.util.presentation.separated (items, sep=', ')`
 Yield `(item, sep)` tuples, one for each element in `items`.
`sep` will be `None` for the last item.

```
>>> list(separated([1, 2]))
[(1, ','), (2, None)]
```

```
>>> list(separated([1]))
[(1, None)]
```

```
>>> list(separated("abc", ':'))
[('a', ':'), ('b', ':'), ('c', None)]
```

`trac.util.presentation.to_json (value)`
 Encode `value` to JSON.

Modules generating paginated output will be happy to use a rich pagination controller. See *Query*, *Report* and *Search* modules for example usage.

`class trac.util.presentation.Paginator (items, page=0, max_per_page=10, num_items=None)`
 Pagination controller

trac.util.text -- Text manipulation

The Unicode toolbox

Trac internals are almost exclusively dealing with Unicode text, represented by `unicode` instances. The main advantage of using `unicode` over UTF-8 encoded `str` (as this used to be the case before version 0.10), is that text transformation functions in the present module will operate in a safe way on individual characters, and won't risk to eventually cut a multi-byte sequence in the middle. Similar issues with Python string handling routines are avoided as well, like surprising results when splitting text in lines. For example, did you know that "Priorità" is encoded as 'Priorit\xc3\x0a' in UTF-8? Calling `strip()` on this value in some locales can cut away the trailing `\x0a` and it's no longer valid UTF-8...

The drawback is that most of the outside world, while eventually "Unicode", is definitely not `unicode`. This is why we need to convert back and forth between `str` and `unicode` at the boundaries of the system. And more often than not we even have to guess which encoding is used in the incoming `str` strings.

Encoding `unicode` to `str` is usually directly performed by calling `encode()` on the `unicode` instance, while decoding is preferably left to the `to_unicode` helper function, which converts `str` to `unicode` in a robust and guaranteed successful way.

`trac.util.text.to_unicode (text, charset=None)`
 Convert input to an `unicode` object.

For a `str` object, we'll first try to decode the bytes using the given `charset` encoding (or UTF-8 if none is specified), then we fall back to the latin1 encoding which might be correct or not, but at least preserves the original byte sequence by mapping each byte to the corresponding unicode code point in the range U+0000 to U+00FF. For anything else, a simple `unicode()` conversion is attempted, with special care taken with `Exception` objects.

```
trac.util.text.exception_to_unicode (e, traceback=False)
```

Convert an **Exception** to an **unicode** object.

In addition to **to_unicode**, this representation of the exception also contains the class name and optionally the traceback.

Web utilities

```
trac.util.text.unicode_quote (value, safe='/')
```

A unicode aware version of **urllib.quote**

Parameters:

- **value** -- anything that converts to a **str**. If **unicode** input is given, it will be UTF-8 encoded.
- **safe** -- as in **quote**, the characters that would otherwise be quoted but shouldn't here (defaults to '/')

```
trac.util.text.unicode_quote_plus (value, safe='')
```

A unicode aware version of **urllib.quote_plus**.

Parameters:

- **value** -- anything that converts to a **str**. If **unicode** input is given, it will be UTF-8 encoded.
- **safe** -- as in **quote_plus**, the characters that would otherwise be quoted but shouldn't here (defaults to '/')

```
trac.util.text.unicode_unquote (value)
```

A unicode aware version of **urllib.unquote**.

Parameters: **str** -- UTF-8 encoded **str** value (for example, as obtained by **unicode_quote**).

Return type: **unicode**

```
trac.util.text.unicode_urlencode (params, safe='')
```

A unicode aware version of **urllib.urlencode**.

Values set to **empty** are converted to the key alone, without the equal sign.

```
trac.util.text.quote_query_string (text)
```

Quote strings for query string

```
trac.util.text.javascript_quote (text)
```

Quote strings for inclusion in single or double quote delimited Javascript strings

```
trac.util.text.to_js_string (text)
```

Embed the given string in a double quote delimited Javascript string (conform to the JSON spec)

Console and file system

```
trac.util.text.path_to_unicode (path)
```

Convert a filesystem path to unicode, using the filesystem encoding.

```
trac.util.text.stream_encoding (stream)
```

Return the appropriate encoding for the given stream.

```
trac.util.text.console_print (out, *args, **kwargs)
```

Output the given arguments to the console, encoding the output as appropriate.

Parameters: **kwargs** -- **newline** controls whether a newline will be appended (defaults to **True**)

```
trac.util.text.printout (*args, **kwargs)
```

Do a **console_print** on **sys.stdout**.

```
trac.util.text.printerr (*args, **kwargs)
```

Do a **console_print** on **sys.stderr**.

```
trac.util.text.raw_input (prompt)
```

Input one line from the console and converts it to unicode as appropriate.

Miscellaneous

`trac.util.text.empty`

A special tag object evaluating to the empty string, used as marker for missing value (as opposed to a present but empty value).

`class trac.util.text.unicode_passwd`

Conceal the actual content of the string when `repr` is called.

`trac.util.text.levenshtein_distance (lhs, rhs)`

Return the Levenshtein distance between two strings.

Text formatting

`trac.util.text.pretty_size (size, format='%.1f')`

Pretty print content size information with appropriate unit.

Parameters:

- **size** -- number of bytes
- **format** -- can be used to adjust the precision shown

`trac.util.text.breakable_path (path)`

Make a path breakable after path separators, and conversely, avoid breaking at spaces.

`trac.util.text.normalize_whitespace (text, to_space=u'\xa0', remove=u'\u200b')`

Normalize whitespace in a string, by replacing special spaces by normal spaces and removing zero-width spaces.

`trac.util.text.unquote_label (txt)`

Remove (one level of) enclosing single or double quotes.

New in version 1.0.

`trac.util.text.fix_eol (text, eol)`

Fix end-of-lines in a text.

`trac.util.text.expandtabs (s, tabstop=8, ignoring=None)`

Expand tab characters `'\t'` into spaces.

Parameters:

- **tabstop** -- number of space characters per tab (defaults to the canonical 8)
- **ignoring** -- if not `None`, the expansion will be "smart" and go from one tabstop to the next. In addition, this parameter lists characters which can be ignored when computing the indent.

`trac.util.text.obfuscate_email_address (address)`

Replace anything looking like an e-mail address (`'@something'`) with a trailing ellipsis (`'@...'`)

`trac.util.text.text_width (text, ambiwidth=1)`

Determine the column width of `text` in Unicode characters.

The characters in the East Asian Fullwidth (F) or East Asian Wide (W) have a column width of 2. The other characters in the East Asian Halfwidth (H) or East Asian Narrow (Na) have a column width of 1.

That `ambiwidth` parameter is used for the column width of the East Asian Ambiguous (A). If `1`, the same width as characters in US-ASCII. This is expected by most users. If `2`, twice the width of US-ASCII characters. This is expected by CJK users.

cf. <http://www.unicode.org/reports/tr11/>.

`trac.util.text.print_table (data, headers=None, sep=' ', out=None, ambiwidth=None)`

Print data according to a tabular layout.

Parameters:

- **data** -- a sequence of rows; assume all rows are of equal length.
- **headers** -- an optional row containing column headers; must be of the same length as each row in **data**.
- **sep** -- column separator
- **out** -- output file descriptor (**None** means use **sys.stdout**)
- **ambiwidth** -- column width of the East Asian Ambiguous (A). If **None**, detect **ambiwidth** with the locale settings. If others, pass to the **ambiwidth** parameter of **text_width**.

`trac.util.text.shorten_line (text, maxlen=75)`

Truncates content to at most **maxlen** characters.

This tries to be (a bit) clever and attempts to find a proper word boundary for doing so.

`trac.util.text.stripws (text, leading=True, trailing=True)`

Strips unicode white-spaces and ZWSPs from **text**.

Parameters:

- **leading** -- strips leading spaces from **text** unless **leading** is **False**.
- **trailing** -- strips trailing spaces from **text** unless **trailing** is **False**.

`trac.util.text.wrap (t, cols=75, initial_indent='', subsequent_indent='', linesep='\n', ambiwidth=1)`

Wraps the single paragraph in **t**, which contains unicode characters. The every line is at most **cols** characters long.

That **ambiwidth** parameter is used for the column width of the East Asian Ambiguous (A). If **1**, the same width as characters in US-ASCII. This is expected by most users. If **2**, twice the width of US-ASCII characters. This is expected by CJK users.

Conversion utilities

`trac.util.text.unicode_to_base64 (text, strip_newlines=True)`

Safe conversion of **text** to base64 representation using utf-8 bytes.

Strips newlines from output unless **strip_newlines** is **False**.

`trac.util.text.unicode_from_base64 (text)`

Safe conversion of **text** to unicode based on utf-8 bytes.

`trac.util.text.to_utf8 (text, charset='latin1')`

Convert a string to UTF-8, assuming the encoding is either UTF-8, ISO Latin-1, or as specified by the optional **charset** parameter.

Deprecated since version 0.10: You should use **unicode** strings only.

trac.versioncontrol.api -- Trac Version Control APIs

This module implements an abstraction layer over different kind of version control systems and the mechanism to access several heterogeneous repositories under a single "virtual" hierarchy.

This abstraction was derived from the original model built around the Subversion system (versioned tree, changesets). It gradually became more general, now aiming at supporting distributed version control systems (DVCS).

Interfaces

`class trac.versioncontrol.api.IRepositoryConnector`

Provide support for a specific version control system.

See also [trac.versioncontrol.api.IRepositoryConnector](#) extension points

`get_supported_types ()`

Return the types of version control systems that are supported.

Yields (**reptype**, **priority**) pairs, where **reptype** is used to match against the configured [**trac**] **repository_type** value in TracIni.

If multiple provider match a given type, the **priority** is used to choose between them (highest number is highest priority).

If the **priority** returned is negative, this indicates that the connector for the given **reptype** indeed exists but can't be used for some reason. The **error** property can then be used to store an error message or exception relevant to the problem detected.

get_repository (repos_type, repos_dir, params)

Return a Repository instance for the given repository type and dir.

`class trac.versioncontrol.api.IRepositoryProvider`

Provide known named instances of Repository.

See also [trac.versioncontrol.api.IRepositoryProvider extension points](#)

get_repositories ()

Generate repository information for known repositories.

Repository information is a key,value pair, where the value is a dictionary which must contain at the very least either of the following entries:

- **'dir': the repository directory which can be used by the**
connector to create a **Repository** instance. This defines a "real" repository.
- **'alias': the name of another repository. This defines an**
alias to another (real) repository.

Optional entries:

- **'type': the type of the repository (if not given, the**
default repository type will be used).
- **'description': a description of the repository (can**
contain WikiFormatting).
- **'hidden': if set to 'true', the repository is hidden**
from the repository index.
- **'url': the base URL for checking out the repository.**

`class trac.versioncontrol.api.IRepositoryChangeListener`

Listen for changes in repositories.

See also [trac.versioncontrol.api.IRepositoryChangeListener extension points](#)

changeset_added (repos, changeset)

Called after a changeset has been added to a repository.

changeset_modified (repos, changeset, old_changeset)

Called after a changeset has been modified in a repository.

The **old_changeset** argument contains the metadata of the changeset prior to the modification. It is **None** if the old metadata cannot be retrieved.

Components

`class trac.versioncontrol.api.RepositoryManager`

Version control system manager.

connectors

List of components that implement **IRepositoryConnector**

providers

List of components that implement **IRepositoryProvider**

change_listeners

List of components that implement `IRepositoryChangeListener`

repositories_section

One of the alternatives for registering new repositories is to populate the `[repositories]` section of the `trac.ini`.

This is especially suited for setting up convenience aliases, short-lived repositories, or during the initial phases of an installation.

See [TracRepositoryAdmin#Intrac.ini TracRepositoryAdmin] for details about the format adopted for this section and the rest of that page for the other alternatives.

("since 0.12")

repository_type

Default repository connector type. ("since 0.10")

This is also used as the default repository type for repositories defined in `[[TracIni#repositories-section repositories]]` or using the "Repositories" admin panel. ("since 0.12")

repository_dir

Path to the default repository. This can also be a relative path ("since 0.11").

This option is deprecated, and repositories should be defined in the `[TracIni#repositories-section repositories]` section, or using the "Repositories" admin panel. ("since 0.12")

repository_sync_per_request

List of repositories that should be synchronized on every page request.

Leave this option empty if you have set up post-commit hooks calling `trac-admin $ENV changeset added` on all your repositories (recommended). Otherwise, set it to a comma-separated list of repository names.

Note that this will negatively affect performance, and will prevent changeset listeners from receiving events from the repositories specified here. The default is to synchronize the default repository, for backward compatibility.

("since 0.12")

get_repositories ()

Retrieve repositories specified in `TracIni`.

The `[repositories]` section can be used to specify a list of repositories.

get_supported_types ()

Return the list of supported repository types.

get_repositories_by_dir (directory)

Retrieve the repositories based on the given directory.

Parameters: `directory` -- the key for identifying the repositories.

Returns: list of `Repository` instances.

get_repository_id (reponame)

Return a unique id for the given repository name.

This will create and save a new id if none is found.

ote: this should probably be renamed as we're dealing

exclusively with *db* repository ids here.

get_repository (reponame)

Retrieve the appropriate `Repository` for the given repository name.

Parameters: `reponame` -- the key for specifying the repository. If no name is given, take the default repository.

Returns: if no corresponding repository was defined, simply return `None`.

get_repository_by_path (path)

Retrieve a matching `Repository` for the given `path`.

Parameters: `path` -- the eventually scoped repository-scoped path

Returns: a (**reponame**, **repos**, **path**) triple, where **path** is the remaining part of **path** once the **reponame** has been truncated, if needed.

get_default_repository (context)

Recover the appropriate repository from the current context.

Lookup the closest source or changeset resource in the context hierarchy and return the name of its associated repository.

get_all_repositories ()

Return a dictionary of repository information, indexed by name.

get_real_repositories ()

Return a set of all real repositories (i.e. excluding aliases).

reload_repositories ()

Reload the repositories from the providers.

notify (event, reponame, revs)

Notify repositories and change listeners about repository events.

The supported events are the names of the methods defined in the **IRepositoryChangeListener** interface.

shutdown (tid=None)

Free **Repository** instances bound to a given thread identifier

class trac.versioncontrol.api.**DbRepositoryProvider**

Component providing repositories registered in the DB.

get_repositories ()

Retrieve repositories specified in the repository DB table.

add_repository (reponame, dir, type_=None)

Add a repository.

add_alias (reponame, target)

Create an alias repository.

remove_repository (reponame)

Remove a repository.

modify_repository (reponame, changes)

Modify attributes of a repository.

Exceptions

Subclasses of **ResourceNotFound**.

class trac.versioncontrol.api.**NoSuchChangeset** (rev)

class trac.versioncontrol.api.**NoSuchNode** (path, rev, msg=None)

Abstract classes

class trac.versioncontrol.api.**Repository** (name, params, log)

Base class for a repository provided by a version control system.

Initialize a repository.

Parameters:

- **name** -- a unique name identifying the repository, usually a type-specific prefix followed by the path to the repository.
- **params** -- a **dict** of parameters for the repository. Contains the name of the repository under the key "name" and the surrogate key that identifies the repository in the database under the key "id".
- **log** -- a logger instance.

close ()

Close the connection to the repository.

get_base ()

Return the name of the base repository for this repository.

This function returns the name of the base repository to which scoped repositories belong. For non-scoped repositories, it returns the repository name.

clear (youngest_rev=None)

Clear any data that may have been cached in instance properties.

youngest_rev can be specified as a way to force the value of the **youngest_rev** property ("will change in 0.12").

sync (rev_callback=None, clean=False)

Perform a sync of the repository cache, if relevant.

If given, **rev_callback** must be a callable taking a **rev** parameter. The backend will call this function for each **rev** it decided to synchronize, once the synchronization changes are committed to the cache. When **clean** is **True**, the cache is cleaned first.

sync_changeset (rev)

Resync the repository cache for the given **rev**, if relevant.

Returns a "metadata-only" changeset containing the metadata prior to the resync, or **None** if the old values cannot be retrieved (typically when the repository is not cached).

get_quickjump_entries (rev)

Generate a list of interesting places in the repository.

rev might be used to restrict the list of available locations, but in general it's best to produce all known locations.

The generated results must be of the form (category, name, path, rev).

get_path_url (path, rev)

Return the repository URL for the given path and revision.

The returned URL can be **None**, meaning that no URL has been specified for the repository, an absolute URL, or a scheme-relative URL starting with `//`, in which case the scheme of the request should be prepended.

get_changeset (rev)

Retrieve a Changeset corresponding to the given revision **rev**.

get_changeset_uid (rev)

Return a globally unique identifier for the "rev" changeset.

Two changesets from different repositories can sometimes refer to the "very same" changeset (e.g. the repositories are clones).

get_changesets (start, stop)

Generate Changeset belonging to the given time period (start, stop).

has_node (path, rev=None)

Tell if there's a node at the specified (path,rev) combination.

When **rev** is **None**, the latest revision is implied.

get_node (path, rev=None)

Retrieve a Node from the repository at the given path.

A Node represents a directory or a file at a given revision in the repository. If the `rev` parameter is specified, the Node corresponding to that revision is returned, otherwise the Node corresponding to the youngest revision is returned.

`get_oldest_rev ()`

Return the oldest revision stored in the repository.

`get_youngest_rev ()`

Return the youngest revision in the repository.

`previous_rev (rev, path='')`

Return the revision immediately preceding the specified revision.

If `path` is given, filter out ancestor revisions having no changes below `path`.

In presence of multiple parents, this follows the first parent.

`next_rev (rev, path='')`

Return the revision immediately following the specified revision.

If `path` is given, filter out descendant revisions having no changes below `path`.

In presence of multiple children, this follows the first child.

`parent_revs (rev)`

Return a list of parents of the specified revision.

`rev_older_than (rev1, rev2)`

Provides a total order over revisions.

Return `True` if `rev1` is an ancestor of `rev2`.

`get_path_history (path, rev=None, limit=None)`

Retrieve all the revisions containing this path.

If given, `rev` is used as a starting point (i.e. no revision "newer" than `rev` should be returned). The result format should be the same as the one of `Node.get_history()`

`normalize_path (path)`

Return a canonical representation of path in the repos.

`normalize_rev (rev)`

Return a (unique) canonical representation of a revision.

It's up to the backend to decide which string values of `rev` (usually provided by the user) should be accepted, and how they should be normalized. Some backends may for instance want to match against known tags or branch names.

In addition, if `rev` is `None` or "", the youngest revision should be returned.

`short_rev (rev)`

Return a compact representation of a revision in the repos.

`display_rev (rev)`

Return a representation of a revision in the repos for displaying to the user.

This can be a shortened revision string, e.g. for repositories using long hashes.

`get_changes (old_path, old_rev, new_path, new_rev, ignore_ancestry=1)`

Generates changes corresponding to generalized diffs.

Generator that yields change tuples (old_node, new_node, kind, change) for each node change between the two arbitrary (path,rev) pairs.

The old_node is assumed to be None when the change is an ADD, the new_node is assumed to be None when the change is a DELETE.

`is_viewable (perm)`

Return True if view permission is granted on the repository.

`can_view (perm)`

Return True if view permission is granted on the repository.

```
class trac.versioncontrol.api.Node (repos, path, rev, kind)
```

Represents a directory or file in the repository at a given revision.

resource
source

get_content ()

Return a stream for reading the content of the node.

This method will return **None** for directories. The returned object must support a **read([len])** method.

get_entries ()

Generator that yields the immediate child entries of a directory.

The entries are returned in no particular order. If the node is a file, this method returns **None**.

get_history (limit=None)

Provide backward history for this Node.

Generator that yields (**path**, **rev**, **chg**) tuples, one for each revision in which the node was changed. This generator will follow copies and moves of a node (if the underlying version control system supports that), which will be indicated by the first element of the tuple (i.e. the path) changing. Starts with an entry for the current revision.

Parameters: **limit** -- if given, yield at most **limit** results.

get_previous ()

Return the change event corresponding to the previous revision.

This returns a (**path**, **rev**, **chg**) tuple.

get_annotations ()

Provide detailed backward history for the content of this Node.

Retrieve an array of revisions, one **rev** for each line of content for that node. Only expected to work on (text) FILE nodes, of course.

get_properties ()

Returns the properties (meta-data) of the node, as a dictionary.

The set of properties depends on the version control system.

get_content_length ()

The length in bytes of the content.

Will be **None** for a directory.

get_content_type ()

The MIME type corresponding to the content, if known.

Will be **None** for a directory.

is_viewable (perm)

Return True if view permission is granted on the node.

can_view (perm)

Return True if view permission is granted on the node.

```
class trac.versioncontrol.api.Changeset (repos, rev, message, author, date)
```

Represents a set of changes committed at once in a repository.

resource
changeset

get_properties ()

Returns the properties (meta-data) of the node, as a dictionary.

The set of properties depends on the version control system.

Warning: this used to yield 4-elements tuple (besides **name** and **text**, there were **wikiflag** and **htmlclass** values). This is now replaced by the usage of `IPropertyRenderer` (see #1601).

get_changes ()

Generator that produces a tuple for every change in the changeset.

The tuple will contain (**path**, **kind**, **change**, **base_path**, **base_rev**), where **change** can be one of `Changeset.ADD`, `Changeset.COPY`, `Changeset.DELETE`, `Changeset.EDIT` or `Changeset.MOVE`, and **kind** is one of `Node.FILE` or `Node.DIRECTORY`. The **path** is the targeted path for the **change** (which is the "deleted" path for a `DELETE` change). The **base_path** and **base_rev** are the source path and rev for the action (`None` and `-1` in the case of an `ADD` change).

get_branches ()

Yield branches to which this changeset belong. Each branch is given as a pair (**name**, **head**), where **name** is the branch name and **head** a flag set if the changeset is a head for this branch (i.e. if it has no children changeset).

get_tags ()

Yield tags associated with this changeset.
New in version 1.0.

is_viewable (perm)

Return True if view permission is granted on the changeset.

can_view (perm)

Return True if view permission is granted on the changeset.

Helper Functions

`trac.versioncontrol.api.is_default (reponame)`

Check whether **reponame** is the default repository.

trac.versioncontrol.diff -- Utilities for generation of diffs**Synopsis**

get_filtered_hunks, **get_hunks** are low-level wrappers for Python's `difflib.SequenceMatcher`, and they generate groups of opcodes corresponding to diff "hunks".

get_change_extent is a low-level utility used when marking intra-lines differences.

diff_blocks is used at a higher-level to fill the template data needed by the "diff_div.html" template.

unified_diff is also a higher-level function returning differences following the **unified diff** file format.

Finally, **get_diff_options** is an utility for retrieving user diff preferences from a **Request**.

Function Reference

`trac.versioncontrol.diff.get_change_extent (str1, str2)`

Determines the extent of differences between two strings.

Returns a pair containing the offset at which the changes start, and the negative offset at which the changes end. If the two strings have neither a common prefix nor a common suffix, `(0, 0)` is returned.

`trac.versioncontrol.diff.get_filtered_hunks (fromlines, tolines, context=None, ignore_blank_lines=False, ignore_case=False, ignore_space_changes=False)`

Retrieve differences in the form of `difflib.SequenceMatcher` opcodes, grouped according to the **context** and **ignore_*** parameters.

Parameters:

- **fromlines** -- list of lines corresponding to the old content
- **tolines** -- list of lines corresponding to the new content
- **ignore_blank_lines** -- differences about empty lines only are ignored
- **ignore_case** -- upper case / lower case only differences are ignored
- **ignore_space_changes** -- differences in amount of spaces are ignored
- **context** -- the number of "equal" lines kept for representing the context of the change

Returns: generator of grouped `diff.lib.SequenceMatcher` opcodes

If none of the `ignore_*` parameters is `True`, there's nothing to filter out the results will come straight from the `SequenceMatcher`.

```
trac.versioncontrol.diff.get_hunks (fromlines, tolines, context=None)
```

Generator yielding grouped opcodes describing differences .

See `get_filtered_hunks` for the parameter descriptions.

```
trac.versioncontrol.diff.hdf_diff (*args, **kwargs)
```

Deprecated : use `diff_blocks` (will be removed in 1.1.1)

```
trac.versioncontrol.diff.diff_blocks (fromlines, tolines, context=None, tabwidth=8,
ignore_blank_lines=0, ignore_case=0, ignore_space_changes=0)
```

Return an array that is adequate for adding to the data dictionary

See `get_filtered_hunks` for the parameter descriptions.

See also the `diff_div.html` template.

```
trac.versioncontrol.diff.unified_diff (fromlines, tolines, context=None,
ignore_blank_lines=0, ignore_case=0, ignore_space_changes=0)
```

Generator producing lines corresponding to a textual diff.

See `get_filtered_hunks` for the parameter descriptions.

```
trac.versioncontrol.diff.get_diff_options (req)
```

Retrieve user preferences for diffs.

Returns: (style, options, data) triple. style can be 'inline' or 'sidebyside', options a sequence of "diff" flags, data the style and options information represented as key/value pairs in dictionaries, for example: `{ 'style': u'sidebyside', 'options': { 'contextall': 1, 'contextlines': 2, 'ignorecase': 0, 'ignoreblanklines': 0, 'ignorewhitespace': 1 } }`

trac.versioncontrol.svn_fs -- Subversion backend for Trac

This module can be considered to be private. However, it can serve as an example implementation of a version control backend.

Speaking of Subversion, we use its `svn.fs` layer mainly, which means we need direct (read) access to the repository content.

Though there's no documentation for the Python API per se, the doxygen documentation for the **C libraries** are usually enough. Another possible source of inspiration are the **examples** and the helper classes in the **bindings** themselves.

Note about Unicode

The Subversion bindings are not unicode-aware and they expect to receive UTF-8 encoded `string` parameters. On the other hand, all paths manipulated by Trac are `unicode` objects.

Therefore:

- before being handed out to SVN, the Trac paths have to be encoded to UTF-8, using `_to_svn()`
- before being handed out to Trac, a SVN path has to be decoded from UTF-8, using `_from_svn()`

Whenever a value has to be stored as utf8, we explicitly mark the variable name with "_utf8", in order to avoid any possible confusion.

Warning:

`SubversionNode.get_content()` returns an object from which one can read a stream of bytes. NO guarantees can be given about what that stream of bytes represents. It might be some text, encoded in some way or another. SVN properties *might* give some hints about the content, but they actually only reflect the beliefs of whomever set those properties...

Components

`class trac.versioncontrol.svn_fs.SubversionConnector`

Concrete classes

`class trac.versioncontrol.svn_fs.SubversionRepository (path, params, log)`

Repository implementation based on the svn.fs API.

`clear (youngest_rev=None)`

Reset notion of `youngest` and `oldest`

`has_node (path, rev=None, pool=None)`

Check if `path` exists at `rev` (or latest if unspecified)

`normalize_path (path)`

Take any path specification and produce a path suitable for the rest of the API

`normalize_rev (rev)`

Take any revision specification and produce a revision suitable for the rest of the API

`close ()`

Dispose of low-level resources associated to this repository.

`get_base ()`

Retrieve the base path corresponding to the Subversion repository itself.

This is the same as the `path` property minus the intra-repository scope, if one was specified.

`get_quickjump_entries (rev)`

Retrieve known branches, as (name, id) pairs.

Purposedly ignores `rev` and always takes the last revision.

`get_path_url (path, rev)`

Retrieve the "native" URL from which this repository is reachable from Subversion clients.

`get_changeset (rev)`

Produce a `SubversionChangeset` from given revision specification

`get_changeset_uid (rev)`

Build a value identifying the `rev` in this repository.

`get_node (path, rev=None)`

Produce a `SubversionNode` from given path and optionally revision specifications. No revision given means use the latest.

`get_oldest_rev ()`

Gives an approximation of the oldest revision.

`get_youngest_rev ()`

Retrieve the latest revision in the repository.

```

    (wraps fs.youngest_rev)

previous_rev (rev, path='')
    Return revision immediately preceeding rev, eventually below given path or globally.

next_rev (rev, path='', find_initial_rev=False)
    Return revision immediately following rev, eventually below given path or globally.

rev_older_than (rev1, rev2)
    Check relative order between two revision specifications.

get_path_history (path, rev=None, limit=None)
    Retrieve creation and deletion events that happened on given path.

get_changes (old_path, old_rev, new_path, new_rev, ignore_ancestry=0)
    Determine differences between two arbitrary pairs of paths and revisions.
    (wraps repos.svn_repos_dir_delta)

class trac.versioncontrol.svn_fs.SubversionNode (path, rev, repos, pool=None,
parent_root=None)

    get_content ()
        Retrieve raw content as a "read()"able object.

    get_entries ()
        Yield SubversionNode corresponding to entries in this directory.
        (wraps fs.dir_entries)

    get_history (limit=None)
        Yield change events that happened on this path

    get_annotations ()
        Return a list the last changed revision for each line. (wraps client.blame2)

    get_properties ()
        Return dict of node properties at current revision.
        (wraps fs.node_proplist)

    get_content_length ()
        Retrieve byte size of a file.
        Return None for a folder. (wraps fs.file_length)

    get_content_type ()
        Retrieve mime-type property of a file.
        Return None for a folder. (wraps fs.revision_prop)

    get_last_modified ()
        Retrieve timestamp of last modification, in micro-seconds.
        (wraps fs.revision_prop)

    get_branch_origin ()
        Return the revision in which the node's path was created.
        (wraps fs.revision_root_revision(fs.closest_copy))

    get_copy_ancestry ()
        Retrieve the list of (path, rev) copy ancestors of this node. Most recent ancestor first. Each ancestor
        (path, rev) corresponds to the path and revision of the source at the time the copy or move operation was
        performed.

class trac.versioncontrol.svn_fs.SubversionChangeset (repos, rev, scope, pool=None)

```

get_properties ()

Retrieve `dict` of Subversion properties for this revision (revprops)

get_changes ()

Retrieve file changes for a given revision.

(wraps `repos.svn_repos_replay`)

trac.web.api -- Trac Web Request Handling

Primary interface for handling web requests.

Interfaces

The following interfaces allow components to interact at various stages of the web requests processing pipeline.

class trac.web.api.IRequestHandler

Decide which `trac.core.Component` handles which `Request`, and how.

See also [trac.web.api.IRequestHandler extension points](#)

match_request (req)

Return whether the handler wants to process the given request.

process_request (req)

Process the request.

Return a `(template_name, data, content_type)` tuple, where `data` is a dictionary of substitutions for the Genshi template.

"text/html" is assumed if `content_type` is `None`.

Note that if template processing should not occur, this method can simply send the response itself and not return anything.

Since 1.0: Clearsilver templates are no longer supported.

class trac.web.api.IRequestFilter

Enable components to interfere with the processing done by the main handler, either before and/or after it enters in action.

See also [trac.web.api.IRequestFilter extension points](#)

pre_process_request (req, handler)

Called after initial handler selection, and can be used to change the selected handler or redirect request.

Always returns the request handler, even if unchanged.

post_process_request (req, template, data, content_type)

Do any post-processing the request might need; typically adding values to the template `data` dictionary, or changing the Genshi template or mime type.

`data` may be updated in place.

Always returns a tuple of `(template, data, content_type)`, even if unchanged.

Note that `template, data, content_type` will be `None` if:

- called when processing an error page
- the default request handler did not return any result

Since 0.11: there's a `data` argument for supporting Genshi templates; this introduced a difference in arity which made it possible to distinguish between the `IRequestFilter` components still targeted at ClearSilver templates and the newer ones targeted at Genshi templates.

Since 1.0: Clearsilver templates are no longer supported.

For how the main content itself can be generated, see `trac.web.chrome`.

class trac.web.api.ITemplateStreamFilter

Transform the generated content by filtering the Genshi event stream generated by the template, prior to its serialization.

See also [trac.web.api.ITemplateStreamFilter extension points](#)

filter_stream (*req, method, filename, stream, data*)

Return a filtered Genshi event stream, or the original unfiltered stream if no match.

req is the current request object, **method** is the Genshi render method (xml, xhtml or text), **filename** is the filename of the template to be rendered, **stream** is the event stream and **data** is the data for the current template.

See the [Genshi](#) documentation for more information.

class trac.web.api.IAuthenticator

Extension point interface for components that can provide the name of the remote user.

See also [trac.web.api.IAuthenticator extension points](#)

authenticate (*req*)

Return the name of the remote user, or **None** if the identity of the user is unknown.

Classes

class trac.web.api.Request (*environ, start_response*)

Represents a HTTP request/response pair.

This class provides a convenience API over WSGI.

Create the request wrapper.

Parameters:

- **environ** -- The WSGI environment dict
- **start_response** -- The WSGI callback for starting the response
- **callbacks** -- A dictionary of functions that are used to lazily evaluate attribute lookups

authname

The name associated with the user after authentication or **'anonymous'** if no authentication took place.

This corresponds to the **remote_user** when the request is targeted to an area requiring authentication, otherwise the authname is retrieved from the **trac_auth** cookie.

href

An **Href** instance for generating *relative* URLs pointing to resources within the current Trac environment.

abs_href

An **Href** instance for generating *absolute* URLs pointing to resources within the current Trac environment.

method

The HTTP method of the request

path_info

Path inside the application

query_string

Query part of the request

remote_addr

IP address of the remote user

remote_user

Name of the remote user.

Will be **None** if the user has not logged in using HTTP authentication.

scheme

The scheme of the request URL

base_path

The root path of the application

server_name

Name of the server

server_port

Port number the server is bound to

add_redirect_listener (listener)

Add a callable to be called prior to executing a redirect.

The callable is passed the arguments to the **redirect()** call.

get_header (name)

Return the value of the specified HTTP header, or **None** if there's no such header in the request.

send_response (code=200)

Set the status code of the response.

send_header (name, value)

Send the response header with the specified name and value.

value must either be an **unicode** string or can be converted to one (e.g. numbers, ...)

end_headers ()

Must be called after all headers have been sent and before the actual content is written.

check_modified (datetime, extra='')

Check the request "If-None-Match" header against an entity tag.

The entity tag is generated from the specified last modified time (**datetime**), optionally appending an **extra** string to indicate variants of the requested resource.

That **extra** parameter can also be a list, in which case the MD5 sum of the list content will be used.

If the generated tag matches the "If-None-Match" header of the request, this method sends a "304 Not Modified" response to the client. Otherwise, it adds the entity tag as an "ETag" header to the response so that consecutive requests can be cached.

redirect (url, permanent=False)

Send a redirect to the client, forwarding to the specified URL.

The **url** may be relative or absolute, relative URLs will be translated appropriately.

send_file (path, mimetype=None)

Send a local file to the browser.

This method includes the "Last-Modified", "Content-Type" and "Content-Length" headers in the response, corresponding to the file attributes. It also checks the last modification time of the local file against the "If-Modified-Since" provided by the user agent, and sends a "304 Not Modified" response if it matches.

read (size=None)

Read the specified number of bytes from the request body.

write (data)

Write the given data to the response body.

data must be a **str** string, encoded with the charset which has been specified in the "Content-Type" header or 'utf-8' otherwise.

Note that the "Content-Length" header must have been specified. Its value either corresponds to the length of **data**, or, if there are multiple calls to **write**, to the cumulated length of the **data** arguments.

class trac.web.api.RequestDone

Marker exception that indicates whether request processing has completed and a response was sent.

Helper Functions

trac.web.api.arg_list_to_args (arg_list)

Convert a list of (**name**, **value**) tuples into a **_RequestArgs**.

`trac.web.api.parse_arg_list (query_string)`

Parse a query string into a list of (**name**, **value**) tuples.

trac.web.auth -- Trac Authentication

This module deals with web request authentication, and provides the default implementation for the **IA Authenticator** interface.

Component

`class trac.web.auth.LoginModule`

User authentication manager.

This component implements user authentication based on HTTP authentication provided by the web-server, combined with cookies for communicating the login information across the whole site.

This mechanism expects that the web-server is setup so that a request to the path '/login' requires authentication (such as Basic or Digest). The login name is then stored in the database and associated with a unique key that gets passed back to the user agent using the 'trac_auth' cookie. This cookie is used to identify the user in subsequent requests to non-protected resources.

check_ip

Whether the IP address of the user should be checked for authentication ("since 0.9").

ignore_case

Whether login names should be converted to lower case ("since 0.9").

auth_cookie_lifetime

Lifetime of the authentication cookie, in seconds.

This value determines how long the browser will cache authentication information, and therefore, after how much inactivity a user will have to log in again. The default value of 0 makes the cookie expire at the end of the browsing session. ("since 0.12")

auth_cookie_path

Path for the authentication cookie. Set this to the common base path of several Trac instances if you want them to share the cookie. ("since 0.12")

Support Classes

A few classes are provided for directly computing the REMOTE_USER information from the HTTP headers for Basic or Digest authentication. This will be used by the **AuthenticationMiddleware**.

`class trac.web.auth.BasicAuthentication (htpasswd, realm)`

`class trac.web.auth.DigestAuthentication (htdigest, realm)`

A simple HTTP digest authentication implementation (**RFC 2617**).

load (filename)

Load account information from apache style htdigest files, only users from the specified realm are used

send_auth_request (environ, start_response, stale='false')

Send a digest challenge to the browser. Record used nonces to avoid replay attacks.

trac.web.chrome -- Trac content generation for the Web

Content presentation for the web layer.

The Chrome module deals with delivering and shaping content to the end user, mostly targeting (X)HTML generation but not exclusively, RSS or other forms of web content are also using facilities provided here.

Interfaces

class `trac.web.chrome.INavigationContributor`

Extension point interface for components that contribute items to the navigation.

See also [trac.web.chrome.INavigationContributor extension points](#)

get_active_navigation_item (*req*)

This method is only called for the `IRequestHandler` processing the request.

It should return the name of the navigation item that should be highlighted as active/current.

get_navigation_items (*req*)

Should return an iterable object over the list of navigation items to add, each being a tuple in the form (category, name, text).

class `trac.web.chrome.ITemplateProvider`

Extension point interface for components that provide their own Genshi templates and accompanying static resources.

See also [trac.web.chrome.ITemplateProvider extension points](#)

get_htdocs_dirs ()

Return a list of directories with static resources (such as style sheets, images, etc.)

Each item in the list must be a (`prefix`, `abspath`) tuple. The `prefix` part defines the path in the URL that requests to these resources are prefixed with.

The `abspath` is the absolute path to the directory containing the resources on the local file system.

get_templates_dirs ()

Return a list of directories containing the provided template files.

Components

class `trac.web.chrome.Chrome`

Web site chrome assembly manager.

Chrome is everything that is not actual page content.

navigation_contributors

List of components that implement `INavigationContributor`

template_providers

List of components that implement `ITemplateProvider`

stream_filters

List of components that implement `ITemplateStreamFilter`

shared_templates_dir

Path to the `//shared` templates directory//.

Templates in that directory are loaded in addition to those in the environments `templates` directory, but the latter take precedence.

("since 0.11")

shared_htdocs_dir

Path to the `//shared` htdocs directory//.

Static resources in that directory are mapped to `/chrome/shared` under the environment URL, in addition to common and site locations.

This can be useful in `site.html` for common interface customization of multiple Trac environments.

("since 1.0")

auto_reload

Automatically reload template files after modification.

genshi_cache_size

The maximum number of templates that the template loader will cache in memory. The default value is 128. You may want to choose a higher value if your site uses a larger number of templates, and you have enough memory to spare, or you can reduce it if you are short on memory.

htdocs_location

Base URL for serving the core static resources below `/chrome/common/`.

It can be left empty, and Trac will simply serve those resources itself.

Advanced users can use this together with `[TracAdmin trac-admin ... deploy <deploydir>]` to allow serving the static resources for Trac directly from the web server. Note however that this only applies to the `<deploydir>/htdocs/common` directory, the other deployed resources (i.e. those from plugins) will not be made available this way and additional rewrite rules will be needed in the web server.

jquery_location

Location of the jQuery JavaScript library (version 1.7.2).

An empty value loads jQuery from the copy bundled with Trac.

Alternatively, jQuery could be loaded from a CDN, for example: <http://code.jquery.com/jquery-1.7.2.min.js>,
<http://ajax.aspnetcdn.com/ajax/jquery/jquery-1.7.2.min.js> or

<https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js>.

("since 1.0")

jquery_ui_location

Location of the jQuery UI JavaScript library (version 1.8.21).

An empty value loads jQuery UI from the copy bundled with Trac.

Alternatively, jQuery UI could be loaded from a CDN, for example:
<https://ajax.googleapis.com/ajax/libs/jqueryui/1.8.21/jquery-ui.min.js> or

<http://ajax.aspnetcdn.com/ajax/jquery.ui/1.8.21/jquery-ui.min.js>.

("since 1.0")

jquery_ui_theme_location

Location of the theme to be used with the jQuery UI JavaScript library (version 1.8.21).

An empty value loads the custom Trac jQuery UI theme from the copy bundled with Trac.

Alternatively, a jQuery UI theme could be loaded from a CDN, for example:
<https://ajax.googleapis.com/ajax/libs/jqueryui/1.8.21/themes/start/jquery-ui.css> or

<http://ajax.aspnetcdn.com/ajax/jquery.ui/1.8.21/themes/start/jquery-ui.css>.

("since 1.0")

metanav_order

Order of the items to display in the **metanav** navigation bar, listed by IDs. See also `TracNavigation`.

mainnav_order

Order of the items to display in the **mainnav** navigation bar, listed by IDs. See also `TracNavigation`.

logo_link

URL to link to, from the header logo.

logo_src

URL of the image to use as header logo. It can be absolute, server relative or relative.

If relative, it is relative to one of the `/chrome` locations: `site/your-logo.png` if `your-logo.png` is located in the `htdocs` folder within your TracEnvironment; `common/your-logo.png` if `your-logo.png` is located in the folder mapped to the `[#trac-section htdocs_location]` URL. Only specifying `your-logo.png` is equivalent to the latter.

logo_alt

Alternative text for the header logo.

logo_width

Width of the header logo image in pixels.

logo_height

Height of the header logo image in pixels.

show_email_addresses

Show email addresses instead of usernames. If false, we obfuscate email addresses. ("since 0.11")

never_obfuscate_mailto

Never obfuscate `mailto:` links explicitly written in the wiki, even if `show_email_addresses` is false or the user has not the EMAIL_VIEW permission ("since 0.11.6").

show_ip_addresses

Show IP addresses for resource edits (e.g. wiki). ("since 0.11.3")

resizable_textareas

Make `<textarea>` fields resizable. Requires !JavaScript. ("since 0.12")

auto_preview_timeout

Inactivity timeout in seconds after which the automatic wiki preview triggers an update. This option can contain floating-point values. The lower the setting, the more requests will be made to the server. Set this to 0 to disable automatic preview. The default is 2.0 seconds. ("since 0.12")

default_dateinfo_format

The date information format. Valid options are 'relative' for displaying relative format and 'absolute' for displaying absolute format. ("since 1.0")

environment_created ()

Create the environment templates directory.

get_all_templates_dirs ()

Return a list of the names of all known templates directories.

prepare_request (req, handler=None)

Prepare the basic chrome data for the request.

Parameters:

- **req** -- the request object
- **handler** -- the `IRequestHandler` instance that is processing the request

load_template (filename, method=None)

Retrieve a Template and optionally preset the template data.

Also, if the optional `method` argument is set to 'text', a `NewTextTemplate` instance will be created instead of a `MarkupTemplate`.

render_template (req, filename, data, content_type=None, fragment=False)

Render the `filename` using the `data` for the context.

The `content_type` argument is used to choose the kind of template used (`NewTextTemplate` if 'text/plain', `MarkupTemplate` otherwise), and tweak the rendering process (use of XHTML Strict doctype if 'text/html' is given).

When `fragment` is specified, the (filtered) Genshi stream is returned.

cc_list (cc_field)

Split a CC: value in a list of addresses.

format_emails (context, value, sep=', ')

Normalize a list of e-mails and obfuscate them if needed.

Parameters:

- **context** -- the context in which the check for obfuscation should be done
- **value** -- a string containing a comma-separated list of e-mails
- **sep** -- the separator to use when rendering the list again

get_email_map ()

Get the email addresses of all known users.

add_textarea_grips (req)

Make `<textarea class="trac-resizable">` fields resizable if enabled by configuration.

add_wiki_toolbars (req)

Add wiki toolbars to `<textarea class="wikitext">` fields.

add_auto_preview (req)

Setup auto-preview for `<textarea>` fields.

add_jquery_ui (req)

Add a reference to the jQuery UI script and link the stylesheet.

Functions

Most of the helper functions are related to content generation, and in particular, (X)HTML content generation, in one way or another.

`trac.web.chrome.web_context (req, resource=None, id=False, version=False, parent=False, absurls=False)`

Create a rendering context from a request.

The **perm** and **href** properties of the context will be initialized from the corresponding properties of the request object.

```
>>> from trac.test import Mock, MockPerm
>>> req = Mock(href=Mock(), perm=MockPerm())
>>> context = web_context(req)
>>> context.href is req.href
True
>>> context.perm is req.perm
True
```

Parameters:

- **req** -- the HTTP request object
- **resource** -- the **Resource** object or realm
- **id** -- the resource identifier
- **version** -- the resource version
- **absurls** -- whether URLs generated by the **href** object should be absolute (including the protocol scheme and host name)

Returns: a new rendering context

Return type: **RenderingContext**

`trac.web.chrome.add_meta (req, content, http_equiv=None, name=None, scheme=None, lang=None)`

Add a `<meta>` tag into the `<head>` of the generated HTML.

Web resources

`trac.web.chrome.add_stylesheet (req, filename, mimetype='text/css', media=None)`

Add a link to a style sheet to the chrome info so that it gets included in the generated HTML page.

If the filename is absolute (i.e. starts with a slash), the generated link will be based off the application root path. If it is relative, the link will be based off the `/chrome/` path.

`trac.web.chrome.add_javascript (req, filename)`

Deprecated: use `add_script` instead.

`trac.web.chrome.add_script (req, filename, mimetype='text/javascript', charset='utf-8', ie_if=None)`

Add a reference to an external javascript file to the template.

If the filename is absolute (i.e. starts with a slash), the generated link will be based off the application root path. If it is relative, the link will be based off the `/chrome/` path.

```
trac.web.chrome.add_script_data (req, data={}, **kwargs)
```

Add data to be made available in javascript scripts as global variables.

The keys in `data` and the keyword argument names provide the names of the global variables. The values are converted to JSON and assigned to the corresponding variables.

Page admonitions

```
trac.web.chrome.add_warning (req, msg, *args)
```

Add a non-fatal warning to the request object. When rendering pages, any warnings will be rendered to the user.

```
trac.web.chrome.add_notice (req, msg, *args)
```

Add an informational notice to the request object. When rendering pages, any notice will be rendered to the user.

Contextual Navigation

```
trac.web.chrome.add_link (req, rel, href, title=None, mimetype=None, classname=None, **attrs)
```

Add a link to the chrome info that will be inserted as `<link>` element in the `<head>` of the generated HTML

```
trac.web.chrome.add_ctxtnav (req, elm_or_label, href=None, title=None)
```

Add an entry to the current page's ctxtnav bar.

```
trac.web.chrome.prevnexnav (req, prev_label, next_label, up_label=None)
```

Add Previous/Up/Next navigation links.

Parameters:

- `req` -- a `Request` object
- `prev_label` -- the label to use for left (previous) link
- `up_label` -- the label to use for the middle (up) link
- `next_label` -- the label to use for right (next) link

Miscellaneous

```
trac.web.chrome.auth_link (req, link)
```

Return an "authenticated" link to `link` for authenticated users.

If the user is anonymous, returns `link` unchanged. For authenticated users, returns a link to `/login` that redirects to `link` after authentication.

`trac.web.href` -- Creation of URLs

This module mainly proposes the following class:

```
class trac.web.href.Href (base, path_safe="/!~*'()", query_safe="/!~*'()")
```

Implements a callable that constructs URLs with the given base. The function can be called with any number of positional and keyword arguments which then are used to assemble the URL.

Positional arguments are appended as individual segments to the path of the URL:

```
>>> href = Href('/trac')
>>> href('ticket', 540)
'/trac/ticket/540'
>>> href('ticket', 540, 'attachment', 'bugfix.patch')
'/trac/ticket/540/attachment/bugfix.patch'
>>> href('ticket', '540/attachment/bugfix.patch')
'/trac/ticket/540/attachment/bugfix.patch'
```

If a positional parameter evaluates to `None`, it will be skipped:

```
>>> href('ticket', 540, 'attachment', None)
'/trac/ticket/540/attachment'
```

The first path segment can also be specified by calling an attribute of the instance, as follows:

```
>>> href.ticket(540)
'/trac/ticket/540'
>>> href.changeset(42, format='diff')
'/trac/changeset/42?format=diff'
```

Simply calling the Href object with no arguments will return the base URL:

```
>>> href()
'/trac'
```

Keyword arguments are added to the query string, unless the value is None:

```
>>> href = Href('/trac')
>>> href('timeline', format='rss')
'/trac/timeline?format=rss'
>>> href('timeline', format=None)
'/trac/timeline'
>>> href('search', q='foo bar')
'/trac/search?q=foo+bar'
```

Multiple values for one parameter are specified using a sequence (a list or tuple) for the parameter:

```
>>> href('timeline', show=['ticket', 'wiki', 'changeset'])
'/trac/timeline?show=ticket&show=wiki&show=changeset'
```

Alternatively, query string parameters can be added by passing a dict or list as last positional argument:

```
>>> href('timeline', {'from': '02/24/05', 'daysback': 30})
'/trac/timeline?daysback=30&from=02%2F24%2F05'
>>> href('timeline', {})
'/trac/timeline'
>>> href('timeline', [('from', '02/24/05')])
'/trac/timeline?from=02%2F24%2F05'
>>> href('timeline', ()) == href('timeline', []) == href('timeline', {})
True
```

The usual way of quoting arguments that would otherwise be interpreted as Python keywords is supported too:

```
>>> href('timeline', from_='02/24/05', daysback=30)
'/trac/timeline?from=02%2F24%2F05&daysback=30'
```

If the order of query string parameters should be preserved, you may also pass a sequence of (name, value) tuples as last positional argument:

```
>>> href('query', (('group', 'component'), ('groupdesc', 1)))
'/trac/query?group=component&groupdesc=1'
```

```
>>> params = []
>>> params.append(('group', 'component'))
>>> params.append(('groupdesc', 1))
>>> href('query', params)
'/trac/query?group=component&groupdesc=1'
```

By specifying an absolute base, the function returned will also generate absolute URLs:

```
>>> href = Href('http://trac.edgewall.org')
>>> href('ticket', 540)
'http://trac.edgewall.org/ticket/540'
```

```
>>> href = Href('https://trac.edgewall.org')
>>> href('ticket', 540)
'https://trac.edgewall.org/ticket/540'
```

In common usage, it may improve readability to use the function-calling ability for the first component of the URL as mentioned earlier:

```
>>> href = Href('/trac')
>>> href.ticket(540)
'/trac/ticket/540'
>>> href.browser('/trunk/README.txt', format='txt')
'/trac/browser/trunk/README.txt?format=txt'
```

The `path_safe` argument specifies the characters that don't need to be quoted in the path arguments. Likewise, the `query_safe` argument specifies the characters that don't need to be quoted in the query string:

```
>>> href = Href('')
>>> href.milestone('<look,here>', param='<here,too>')
'/milestone/%3Clook%2Chere%3E?param=%3Chere%2Ctoo%3E'
```

```
>>> href = Href('', path_safe='/<,', query_safe=',>')
>>> href.milestone('<look,here>', param='<here,too>')
'/milestone/<look,here%3E?param=%3Chere,too>'
```

trac.web.main -- Trac Web Entry Point

Entry point for dispatching web requests.

trac.web.dispatch_request

The WSGI compliant callable. It adapts the `environ` information passed from the WSGI gateway and retrieve the appropriate **Environment** from it, creates a **Request** instance and let the **RequestDispatcher** component forward it to the component implementing a matching **IRequestHandler**.

`trac.web.main.dispatch_request (environ, start_response)`

Main entry point for the Trac web interface.

Parameters:

- **environ** -- the WSGI environment dict
- **start_response** -- the WSGI callback for starting the response

Components

`class trac.web.main.RequestDispatcher`

Web request dispatcher.

This component dispatches incoming requests to registered handlers. Besides, it also takes care of user authentication and request pre- and post-processing.

authenticators

List of components that implement **IAuthenticator**

handlers

List of components that implement **IRequestHandler**

filters

Ordered list of filters to apply to all requests ("since 0.10").

default_handler

Name of the component that handles requests to the base URL.

Options include `TimelineModule`, `RoadmapModule`, `BrowserModule`, `QueryModule`, `ReportModule`, `TicketModule` and `WikiModule`. The default is `WikiModule`. ("since 0.9")

default_timezone

The default timezone to use

default_language

The preferred language to use if no user preference has been set. ("since 0.12.1")

default_date_format

The date format. Valid options are 'iso8601' for selecting ISO 8601 format, or leave it empty which means the default date format will be inferred from the browser's default language. ("since 1.0")

use_xsendfile

When true, send a `x-sendfile` header and no content when sending files from the filesystem, so that the web server handles the content. This requires a web server that knows how to handle such a header, like Apache with `mod_xsendfile` or `lighttpd`. ("since 1.0")

dispatch (req)

Find a registered handler that matches the request and let it process it.

In addition, this method initializes the data dictionary passed to the the template and adds the web site chrome.

Classes

`class trac.web.main.RequestWithSession (environ, start_response)`

A request that saves its associated session when sending the reply.

Create the request wrapper.

Parameters:

- **environ** -- The WSGI environment dict
- **start_response** -- The WSGI callback for starting the response
- **callbacks** -- A dictionary of functions that are used to lazily evaluate attribute lookups

Helper Functions

`trac.web.main.get_environments (environ, warn=False)`

Retrieve canonical environment name to path mapping.

The environments may not be all valid environments, but they are good candidates.

`trac.web.main.get_tracignore_patterns (env_parent_dir)`

Return the list of patterns from `env_parent_dir/.tracignore` or a default pattern of `".*"` if the file doesn't exist.

Miscellaneous

`trac.web.main.default_tracker = 'http://trac.edgewall.org'`

This URL is used for semi-automatic bug reports (see `send_internal_error`). Please modify it to point to your own Trac instance if you distribute a patched version of Trac.

trac.wiki.api -- The Wiki API**Interfaces**

The wiki module presents several possibilities of extension, for interacting with the Wiki application and also for extending the Wiki syntax.

First, components can be notified of the changes happening in the wiki.

`class trac.wiki.api.IWikiChangeListener`

Components that want to get notified about the creation, deletion and modification of wiki pages should implement that interface.

See also [trac.wiki.api.IWikiChangeListener extension points](#).

wiki_page_added (page)

Called whenever a new Wiki page is added.

wiki_page_changed (page, version, t, comment, author, ipnr)

Called when a page has been modified.

wiki_page_deleted (page)

Called when a page has been deleted.

wiki_page_version_deleted (page)

Called when a version of a page has been deleted.

wiki_page_renamed (page, old_name)

Called when a page has been renamed.

Components can also interfere with the changes, before or after they're made.

`class trac.wiki.api.IWikiPageManipulator`

Components that need to do specific pre- and post- processing of wiki page changes have to implement this interface.

Unlike change listeners, a manipulator can reject changes being committed to the database.

See also [trac.wiki.api.IWikiPageManipulator extension points](#).

prepare_wiki_page (req, page, fields)

Validate a wiki page before rendering it.

Parameters:

- **page** -- is the `wikiPage` being viewed.
- **fields** -- is a dictionary which contains the wiki `text` of the page, initially identical to `page.text` but it can eventually be transformed in place before being used as input to the formatter.

validate_wiki_page (req, page)

Validate a wiki page after it's been populated from user input.

Parameters: **page** -- is the `wikiPage` being edited.

Returns: a list of (`field`, `message`) tuples, one for each problem detected. `field` can be `None` to indicate an overall problem with the page. Therefore, a return value of `[]` means everything is OK.

Then, the Wiki syntax itself can be extended. The first and less intrusive way is to provide new Wiki macros or Wiki processors. Those are basically the same thing, as they're implemented using the following interface. The difference comes from the invocation syntax used in the Wiki markup, which manifests itself in the `args` parameter of `IWikiMacroProvider.expand_macro()`.

`class trac.wiki.api.IWikiMacroProvider`

Augment the Wiki markup with new Wiki macros.

Changed in version 0.12: new Wiki processors can also be added that way.

See also `WikiMacroBase` and [wiki/WikiMacros#DevelopingCustomMacros](#) and [trac.wiki.api.IWikiMacroProvider extension points](#).

get_macros ()

Return an iterable that provides the names of the provided macros.

`get_macro_description (name)`

Return a tuple of a domain name to translate and plain text description of the macro or only the description with the specified name.

Changed in version 1.0: `get_macro_description` can return a domain to translate the description.

`render_macro (req, name, content)`

Return the HTML output of the macro :deprecated:

`is_inline (content)`

Return `True` if the content generated is an inline XHTML element.

New in version 1.0.

`expand_macro (formatter, name, content, args=None)`

Called by the formatter when rendering the parsed wiki text.

New in version 0.11: This form is preferred over `render_macro`, as you get the `formatter`, which knows the current `context` (and the `req`, but ideally you shouldn't use it in your macros).

Changed in version 0.12: added the `args` parameter

Parameters:

- **formatter** -- the wiki `Formatter` currently processing the wiki markup
- **name** -- is the name by which the macro has been called; remember that via `get_macros`, multiple names could be associated to this macros. Note that the macro names are case sensitive.
- **content** -- is the content of the macro call. When called using macro syntax (`[[Macro(content)]]`), this is the string contained between parentheses, usually containing macro arguments. When called using wiki processor syntax (`{{#!Macro ...}}`), it is the content of the processor block, that is, the text starting on the line following the macro name.
- **args** -- will be a dictionary containing the named parameters passed when using the Wiki processor syntax. The named parameters can be specified when calling the macro using the wiki processor syntax: `{{#!Macro arg1=value1 arg2="value 2" ... some content ... }}` In this example, `args` will be `{'arg1': 'value1', 'arg2': 'value 2'}` and `content` will be `"... some content ..."`. If no named parameters are given like in: `{{#!Macro ... }}` then `args` will be `{}`. That makes it possible to differentiate the above situation from a call made using the macro syntax: `[[Macro(arg1=value1, arg2="value 2", ... some content...)]]` in which case `args` will always be `None`. Here `content` will be the `"arg1=value1, arg2="value 2", ... some content..."` string. If like in this example, `content` is expected to contain some arguments and named parameters, one can use the `parse_args` function to conveniently extract them.

The Wiki syntax can also be extended by introducing new markup.

`class trac.wiki.api.IWikiSyntaxProvider`

Enrich the Wiki syntax with new markup.

See also [wiki:TracDev/IWikiSyntaxProviderExample](#) and [trac.wiki.api.IWikiSyntaxProvider extension points](#).

`get_wiki_syntax ()`

Return an iterable that provides additional wiki syntax.

Additional wiki syntax correspond to a pair of (`regexp`, `cb`), the `regexp` for the additional syntax and the callback `cb` which will be called if there's a match. That function is of the form `cb(formatter, ns, match)`.

`get_link_resolvers ()`

Return an iterable over (`namespace`, `formatter`) tuples.

Each formatter should be a function of the form:


```
def format(formatter, ns, target, label, fullmatch=None):
    pass
```

and should return some HTML fragment. The `label` is already HTML escaped, whereas the `target` is not. The `fullmatch` argument is optional, and is bound to the regexp match object for the link.

The Wiki System

The wiki system provide an access to all the pages.

```
class trac.wiki.api.WikiSystem
```

Wiki system manager.

change_listeners

List of components that implement `IWikiChangeListener`

macro_providers

List of components that implement `IWikiMacroProvider`

syntax_providers

List of components that implement `IWikiSyntaxProvider`

ignore_missing_pages

Enable/disable highlighting CamelCase links to missing pages ("since 0.9").

split_page_names

Enable/disable splitting the WikiPageNames with space characters ("since 0.10").

render_unsafe_content

Enable/disable the use of unsafe HTML tags such as `<script>` or `<embed>` with the HTML [wiki:WikiProcessors WikiProcessor] ("since 0.10.4").

For public sites where anonymous users can edit the wiki it is recommended to leave this option disabled (which is the default).

safe_schemes

List of URI schemes considered "safe", that will be rendered as external links even if [wiki] **render_unsafe_content** is false. ("since 0.11.8")

pages

Return the names of all existing wiki pages.

get_pages (prefix=None)

Iterate over the names of existing Wiki pages.

Parameters: `prefix` -- if given, only names that start with that prefix are included.

has_page (pagename)

Whether a page with the specified name exists.

PAGE_SPLIT_RE = `<_sre.SRE_Pattern object at 0x4486dc8>`

make_label_from_target (target)

Create a label from a wiki target.

A trailing fragment and query string is stripped. Then, leading `./`, `../` and `/` elements are stripped, except when this would lead to an empty label. Finally, if **split_page_names** is true, the label is split accordingly.

Other Functions

```
trac.wiki.api.parse_args (args, strict=True)
```

Utility for parsing macro "content" and splitting them into arguments.

The content is split along commas, unless they are escaped with a backquote (see example below).

Parameters:

- **args** -- a string containing macros arguments
- **strict** -- if `True`, only Python-like identifiers will be recognized as keyword arguments

Example usage:

```
>>> parse_args('')
([], {})
>>> parse_args('Some text')
(['Some text'], {})
>>> parse_args('Some text, mode= 3, some other arg\, with a comma.')
(['Some text', ' some other arg, with a comma.'], {'mode': ' 3'})
>>> parse_args('milestone=milestone1,status!=closed', strict=False)
([], {'status!': 'closed', 'milestone': 'milestone1'})
```

`trac.wiki.api.validate_page_name (pagename)`

Utility for validating wiki page name.

Parameters: `pagename` -- wiki page name to validate

trac.wiki.macros -- The standard set of Wiki macros

The standard set of components corresponding to Wiki macros are not meant to be used directly from the API. You may study their implementation though, for getting inspiration. In particular, you'll see they all subclass the `WikiMacroBase` class, which provides a convenient way to implement a new `IWikiMacroProvider` interface.

`class trac.wiki.macros.WikiMacroBase`

Abstract base class for wiki macros.

See also [wiki/WikiMacros#DevelopingCustomMacros](#).

tracopt.mimeview -- Optional content generation modules

Syntax Highlighters

`class tracopt.mimeview.enscript.EnscriptRenderer`

Syntax highlighter using GNU Enscript.

`class tracopt.mimeview.enscript.EnscriptDeuglifier`

```
def rules(cls):
    return [
        r'(?P<comment><FONT COLOR="#B22222">)',
        r'(?P<keyword><FONT COLOR="#5F9EA0">)',
        r'(?P<type><FONT COLOR="#228B22">)',
        r'(?P<string><FONT COLOR="#BC8F8F">)',
        r'(?P<func><FONT COLOR="#0000FF">)',
        r'(?P<prep><FONT COLOR="#B8860B">)',
        r'(?P<lang><FONT COLOR="#A020F0">)',
        r'(?P<var><FONT COLOR="#DA70D6">)',
        r'(?P<font><FONT.*?>)',
        r'(?P<endfont></FONT>)'
    ]
```

See also `trac.util.html.Deuglifier`.

`class tracopt.mimeview.php.PHPRenderer`

Syntax highlighter using the PHP executable.

```
class tracopt.mimeview.php.PhpDeuglifier
```

```
def rules(cls):
    colors = dict(comment='FF8000', lang='0000BB', keyword='007700',
                  string='DD0000')
    # rules check for <font> for PHP 4 or <span> for PHP 5
    return [r'(?P<%s><(?:font color="|span style="color: )#%s">)' % c
            for c in colors.items()] + [r'(?P<font><font.*?>)', r'(?P<endfont></font>)']
```

See also `trac.util.html.Deuglifier`.

```
class tracopt.mimeview.silvercity.SilverCityRenderer
```

Syntax highlighting based on SilverCity.

Testing in Trac

So, you'd like to make sure Trac works in your configuration. Excellent. Here's what you need to know.

Running the tests

Prerequisites

Beyond the standard installation prereqs, you also need:

- [Pygments](#) (0.8+)
- [Twill](#) (0.9+)
- [Coverage](#) (for coverage)
- [Figleaf](#) (0.6.1+, alternative for coverage)

Additionally, if you're on Windows, you need to get `fcrypt`. See [Prerequisites on Windows](#) below for more information.

Invoking the tests

Just run **make test** in the Trac tree once you have everything installed. This will run the unit tests first, then the functional tests (if you have the dependencies) against SQLite. On a reasonably fast machine, the former takes 10 seconds and the latter a couple of minutes.

A few environment variables will influence the way tests are executed:

TRAC_TEST_DB_URI

Use another database backend than the default in-memory SQLite database. See [Using an alternate database backend](#) for more.

TRAC_TEST_TRACD_OPTIONS

Provide additional options to the standalone **tracd** server used for the functional tests.

The `Makefile` is actually written in a way that allow you to get more control, if you want.

Other possible usages:

```
make test=trac/tests/allwiki.py # run all the Wiki formatter tests
make unit-test db=postgres # run only the unit tests with PostgreSQL
make functional-test db=mysql # run only the functional tests with MySQL
make test python=2.4 # run all the tests using Python 2.4
```

If you're running the tests on Windows and don't have cygwin, you'll need to manually run the tests using **python tracetest.py**, but this will run all the tests interleaved.

Understanding failures

Functional test failures can happen a few different ways.

Running trac-admin fails every time:	Make sure the prereqs are met. In particular, that new enough Genshi is available and has python setup.py egg_info run.
Repo creation fails:	Subversion is required for the tests; they are not designed to run without it.
Repo creation works, other repo access fails:	Probably a mismatch in svn bindings versus the svn binary.
Twill errors which save to HTML:	Check the html and see if there's a traceback contained in it. Chances are it has an obvious traceback with an error -- these are triggered on the server, not the tester, so they're difficult for us to show in the failure itself. If you can't decipher what the problem is from viewing the HTML, run the server manually and see what state that particular page is in.
Random weird platform issues:	Please report them.
Can't remove files on Windows:	Ugh. Please report them.
Reload tests fail:	Chances are, you're on a Windows VM that has an unstable clock and FAT32 filesystem (which has a granularity of several seconds). If that's not the case, report it.
Coverage doesn't work with functional tests:	Known issue, patches welcome...

Prerequisites on Windows

- You have to install **fcrypt**
- You may install **pywin32** (optional, improve **subprocess** performance)

Writing Tests for Core

Where tests belong

If it's a regression, it belongs in `trac/tests/functional/testcases.py` for now. Module-specific tests generally already have a `trac/$MOD/tests/functional.py` which you can add to. The environment is brought up as few times as possible, for speed, and your test order is guaranteed to be run in the order it's added to the suite, at the end of the file.

Using Twill

The definitive guide for Twill commands is the **Command Reference**, but 90% of what you need is contained by convenience methods in `FunctionalTester.go_to_*` and the following few commands:

tc.find: Looks for a regex on the current page. If it isn't found, raise an exception.

Example:

```
tc.find("\bPreferences\b")
```

tc.notfind: Like find, but raises an exception if it *is* there.

Example:

```
tc.find(r"\bPreferences\b")
```

tc.follow: Find a link matching the regex, and simulate clicking it.

Example:

```
tc.follow("Login")
```

tc.fv: Short for **formvalue**, fill in a field.

Example:

```
tc.fv("searchform", "q", "ponies")
```

tc.submit: Submit the active form.

Example:

```
tc.submit()
```

Example

This is how you might construct a test that verifies that admin users can see detailed version information.

Start with the navigation. You shouldn't rely on the browser being in any specific state, so begin with `FunctionalTester.go_to_*`.

```
def test_about_page(self):
    self._tester.logout()          # Begin by logging out.
    self._tester.go_to_front()     # The homepage has a link we want
    tc.follow("About")             # Follow the link with "About" in it

    tc.find("Trac is a web-based software")
    tc.notfind("Version Info")

    self._tester.login("admin")
    self._tester.go_to_front()
    tc.follow("About")

    tc.find("Trac is a web-based software")
    tc.find("Version Info")
```

Test Environment Helpers

Functional Test Environment

Object for creating and destroying a Trac environment for testing purposes. Provides some Trac environment-wide utility functions, and a way to call **trac-admin** without it being on the path.

`class trac.tests.functional.testenv.FunctionalTestEnvironment (dirname, port, url)`

Common location for convenience functions that work with the test environment on Trac. Subclass this and override some methods if you are using a different **VCS**.

FunctionalTestEnvironment requires a **dirname** in which the test repository and Trac environment will be created, **port** for the **tracd** webserver to run on, and the **url** which can access this (usually `localhost`).

Create a **FunctionalTestEnvironment**, see the class itself for parameter information.

destroy ()

Remove all of the test environment data.

init ()

Hook for modifying settings or class attributes before any methods are called.

create_repo ()

Hook for creating the repository.

destroy_repo ()

Hook for removing the repository.

post_create (env)

Hook for modifying the environment after creation. For example, to set configuration like:

```
def post_create(self, env):
```

```
    env.config.set('git', 'path', '/usr/bin/git') env.config.save()
```

get_enabled_components ()

Return a list of components that should be enabled after environment creation. For anything more complicated, use the **post_create()** method.

create ()

Create a new test environment. This sets up Trac, calls **create_repo()** and sets up authentication.

adduser (user)

Add a user to the environment. The password will be set to the same as username.

start ()

Starts the webserver, and waits for it to come up.

stop ()

Stops the webserver, if running

FIXME: probably needs a nicer way to exit for coverage to work

restart ()

Restarts the webserver

get_trac_environment ()

Returns a Trac environment object

repo_path_for_initenv ()

Default to no repository

Functional Tester

The **FunctionalTester** object provides a higher-level interface to working with a Trac environment to make test cases more succinct.

```
class trac.tests.functional.testers.FunctionalTester (url)
```

Provides a library of higher-level operations for interacting with a test environment.

It makes assumptions such as knowing what ticket number is next, so avoid doing things manually in a **FunctionalTestCase** when you can.

Create a **FunctionalTester** for the given Trac URL and Subversion URL

login (username)

Login as the given user

logout ()

Logout

create_ticket (summary=None, info=None)

Create a new (random) ticket in the test environment. Returns the new ticket number.

Parameters:

- **summary** -- may optionally be set to the desired summary
- **info** -- may optionally be set to a dictionary of field value pairs for populating the ticket. `info['summary']` overrides summary.

summary and **description** default to randomly-generated values.

quickjump (search)

Do a quick search to jump to a page.

go_to_front ()

Go to the Trac front page

go_to_ticket (ticketid)

Surf to the page for the given ticket ID. Assumes ticket exists.

go_to_wiki (name)

Surf to the page for the given wiki page.

go_to_timeline ()

Surf to the timeline page.

go_to_query ()

Surf to the custom query page.

go_to_admin ()

Surf to the webadmin page.

go_to_roadmap ()

Surf to the roadmap page.

add_comment (ticketid, comment=None)

Adds a comment to the given ticket ID, assumes ticket exists.

attach_file_to_ticket (ticketid, data=None, tempfilename=None, description=None, replace=False)

Attaches a file to the given ticket id, with random data if none is provided. Assumes the ticket exists.

clone_ticket (ticketid)

Create a clone of the given ticket id using the clone button.

create_wiki_page (page, content=None)

Creates the specified wiki page, with random content if none is provided.

attach_file_to_wiki (name, data=None)

Attaches a file to the given wiki page, with random content if none is provided. Assumes the wiki page exists.

create_milestone (name=None, due=None)

Creates the specified milestone, with a random name if none is provided. Returns the name of the milestone.

create_component (name=None, user=None)

Creates the specified component, with a random camel-cased name if none is provided. Returns the name.

create_enum (kind, name=None)

Helper to create the specified enum (used for priority, severity, etc). If no name is given, a unique random word is used. The name is returned.

create_priority (name=None)

Create a new priority enum

create_resolution (name=None)

Create a new resolution enum

create_severity (name=None)

Create a new severity enum

create_type (name=None)

Create a new ticket type enum

create_version (name=None, releasetime=None)

Create a new version. The name defaults to a random camel-cased word if not provided.

create_report (title, query, description)

Create a new report with the given title, query, and description

ticket_set_milestone (ticketid, milestone)

Set the milestone on a given ticket.

Using an alternate database backend

The unit tests don't really touch the db. The functional tests will, however, but if you're not using sqlite you need to setup the database yourself. Once it's set up, just set **TRAC_TEST_DB_URI** to the connection string you would use for an **trac-admin initenv** and run the tests.

Postgres

Testing against Postgres requires you to setup a postgres database and user for testing, then setting an environment variable. The test scripts will create a schema within the database, and on consecutive runs remove the schema.

Warning

Do not run this against a live Trac db schema, the schema *will* be removed if it exists.

On OS X and Linux, you can run the following to create the test database:

```
$ sudo -u postgres createuser -S -D -r -P -e tracuser
$ sudo -u postgres createdb -O tracuser trac
```

Windows:

```
> createuser -U postgres -S -D -r -P -e tracuser
> createdb -U postgres -O tracuser trac
```

Prior to running the tests, set the **TRAC_TEST_DB_URI** variable. If you do not include a schema in the URI, the schema `tractest` will be used.

OS X and Linux:

```
$ export TRAC_TEST_DB_URI=postgres://tracuser:password@localhost:5432/trac?schema=tractest
$ make test
```

Windows:

```
set TRAC_TEST_DB_URI=postgres://tracuser:password@localhost:5432/trac?schema=tractest
```


Finally, run the tests as usual. Note that if you have already a test environment set up from a previous run, the settings in `testenv/trac/conf/trac.ini` will be used. In particular, they will take precedence over the `TRAC_TEST_DB_URI` variable. Simply edit that `trac.ini` file or even remove the whole `testenv` folder if this gets in the way.

If in some cases the tests go wrong and you can't run the tests again because the schema is already there, you can drop the schema manually like this:

OS X and Linux:

```
> echo 'drop schema "tractest" cascade' | psql trac tracuser
```

Windows:

```
> echo drop schema "tractest" cascade | psql trac tracuser
```

If you later want to remove the test user and database, use the following:

On OS X and Linux, you can run the following to create the test database:

```
$ sudo -u postgres dropdb tractest
$ sudo -u postgres dropuser tractest
```

Windows:

```
> dropdb -U postgres trac
> dropuser -U postgres tracuser
```

MySQL

Create the database and user as you normally would. See the [MySQLDb](#) page for more information.

Example:

```
$ mysql -u root
CREATE DATABASE trac DEFAULT CHARACTER SET utf8 COLLATE utf8_bin;
CREATE USER tracuser IDENTIFIED BY 'password';
GRANT ALL ON trac.* TO tracuser;
FLUSH PRIVILEGES;
^D
$ export TRAC_TEST_DB_URI=mysql://tracuser:password@localhost/trac
$ make test
...
$ mysql -u root
DROP DATABASE trac
DROP USER tracuser
^D
```

If you have better ideas on automating this, please contact us.

Troubleshooting

If you hit the following error message:

```
trac.core.TracError: The Trac Environment needs to be upgraded.
```

This is because the test environment clean-up stopped half-way: the `testenv/trac` environment is still there, but the `testenv/trac/conf/trac.ini` file has already been removed. The default ticket workflow then requests an environment upgrade. Simply remove manually the whole `testenv` folder and, when using Postgres, remove the `tractest` schema manually as explained above.

Writing Tests for Plugins

Testing a VCS backend

You'll need to make several subclasses to get this working in the current test infrastructure. But first, we start with some imports. These are pretty much required for all plugin tests:

```
from trac.tests.functional import (FunctionalTestSuite,
                                  FunctionalTestCaseSetup,
                                  FunctionalTwillTestCaseSetup, tc)
from trac.tests.functional import testenv
```

Now subclass `FunctionalTestEnvironment`. This allows you to override methods that you need to set up your repo instead of the default Subversion one:

```
class GitFunctionalTestEnvironment(testenv.FunctionalTestEnvironment):
    repotype = 'git'

    def create_repo(self):
        os.mkdir(self.repodir)
        self.call_in_repo(["git", "init"])
        self.call_in_repo(["git", "config", "user.name", "Test User"])
        self.call_in_repo(["git", "config", "user.email", "test@example.com"])

    def get_enabled_components(self):
        return ['tracext.git.*']

    def get_repourl(self):
        return self.repodir + '/.git'
    repourl = property(get_repourl)
```

Now you need a bit of glue that sets up a test suite specifically for your plugin's repo type. Any testcases within this test suite will use the same environment. No other changes are generally necessary on the test suite:

```
class GitFunctionalTestSuite(FunctionalTestSuite):
    env_class = GitFunctionalTestEnvironment
```

Your test cases can call functions on either the *tester* or *twill commands* to do their job. Here's one that just verifies we were able to sync the repo without issue:

```
class EmptyRepoTestCase(FunctionalTwillTestCaseSetup):
    def runTest(self):
        self._tester.go_to_timeline()
        tc.notfind('Unsupported version control system')
```

Lastly, there's some boilerplate needed for the end of your test file, so it can be run from the command line:

```
def suite():
    # Here you need to create an instance of your subclass
    suite = GitFunctionalTestSuite()
    suite.addTest(EmptyRepoTestCase())
    # ...
    suite.addTest(AnotherRepoTestCase())
    return suite

if __name__ == '__main__':
    unittest.main(defaultTest='suite')
```

Todo

write more, with testing-specific steps.

Glossary

VCS

Version Control System, what you use for versioning your source code

Documentation TODO

Todo

write more, with testing-specific steps.

(The *original entry* is located in /usr/local/virtualenv/src/trac-0.13doc/svn-1.0-stable/doc/dev/testing.rst, line 16.)

Indices and tables

- *General Index*
- *modindex*
- *search*
- *Glossary*
- *Documentation TODO*

Index

A

[abs_href](#) ([trac.env.Environment](#) attribute)
([trac.web.api.Request](#) attribute)

[add_alias\(\)](#)
([trac.versioncontrol.api.DbRepositoryProvider](#) method)

[add_auto_preview\(\)](#) ([trac.web.chrome.Chrome](#) method)

[add_comment\(\)](#)
([trac.tests.functional.testenv.FunctionalTester](#) method)

[add_ctxtnav\(\)](#) (in module [trac.web.chrome](#))

[add_interval\(\)](#) ([trac.ticket.roadmap.TicketGroupStats](#) method)

[add_javascript\(\)](#) (in module [trac.web.chrome](#))

[add_jquery_ui\(\)](#) ([trac.web.chrome.Chrome](#) method)

[add_link\(\)](#) (in module [trac.web.chrome](#))

[add_meta\(\)](#) (in module [trac.web.chrome](#))

[add_notice\(\)](#) (in module [trac.web.chrome](#))

[add_redirect_listener\(\)](#) ([trac.web.api.Request](#) method)

[add_repository\(\)](#)
([trac.versioncontrol.api.DbRepositoryProvider](#) method)

[add_script\(\)](#) (in module [trac.web.chrome](#))

[add_script_data\(\)](#) (in module [trac.web.chrome](#))

[add_stylesheet\(\)](#) (in module [trac.web.chrome](#))

[add_textarea_grips\(\)](#) ([trac.web.chrome.Chrome](#) method)

[add_warning\(\)](#) (in module [trac.web.chrome](#))

[add_wiki_toolbars\(\)](#) ([trac.web.chrome.Chrome](#) method)

[adduser\(\)](#)
([trac.tests.functional.testenv.FunctionalTestEnvironment](#) method)

[annotate_row\(\)](#)
([trac.mimeview.api.IHTMLPreviewAnnotator](#) method)

[annotators](#) ([trac.mimeview.api.Mimeview](#) attribute)

[appendrange\(\)](#) ([trac.util.Ranges](#) method)

[apply_ticket_permissions\(\)](#) (in module [trac.ticket.roadmap](#))

[arg_list_to_args\(\)](#) (in module [trac.web.api](#))

[arity\(\)](#) (in module [trac.util](#))

[as_bool\(\)](#) (in module [trac.util](#))

[as_int\(\)](#) (in module [trac.util](#))

[AtomicFile](#) (class in [trac.util](#))

[attach_file_to_ticket\(\)](#)
([trac.tests.functional.testenv.FunctionalTester](#) method)

[attach_file_to_wiki\(\)](#)
([trac.tests.functional.testenv.FunctionalTester](#) method)

[Attachment](#) (class in [trac.attachment](#))

[attachment_added\(\)](#)
([trac.attachment.IAttachmentChangeListener](#) method)

[attachment_data\(\)](#) ([trac.attachment.AttachmentModule](#) method)

[attachment_deleted\(\)](#)
([trac.attachment.IAttachmentChangeListener](#) method)

[attachment_reparented\(\)](#)
([trac.attachment.IAttachmentChangeListener](#) method)

[AttachmentAdmin](#) (class in [trac.attachment](#))

[AttachmentModule](#) (class in [trac.attachment](#))

[auth_cookie_lifetime](#) ([trac.web.auth.LoginModule](#) attribute)

[auth_cookie_path](#) ([trac.web.auth.LoginModule](#) attribute)

[auth_link\(\)](#) (in module [trac.web.chrome](#))

[authenticate\(\)](#) ([trac.web.api.IAuthenticator](#) method)

[authenticators](#) ([trac.web.main.RequestDispatcher](#) attribute)

[authname](#) ([trac.web.api.Request](#) attribute)

[auto_preview_timeout](#) ([trac.web.chrome.Chrome](#) attribute)

[auto_reload](#) ([trac.web.chrome.Chrome](#) attribute)

B

[backup\(\)](#) ([trac.env.Environment](#) method)

[base_path](#) ([trac.web.api.Request](#) attribute)

[base_url](#) ([trac.env.Environment](#) attribute)

[base_url_for_redirect](#) ([trac.env.Environment](#) attribute)

[BasicAuthentication](#) (class in [trac.web.auth](#))

[breakable_path\(\)](#) (in module [trac.util.text](#))

C

[cached\(\)](#) (in module [trac.cache](#))

[CachedProperty](#) (class in [trac.cache](#))

[CachedPropertyBase](#) (class in [trac.cache](#))

[CachedSingletonProperty](#) (class in [trac.cache](#))

[CacheManager](#) (class in [trac.cache](#))

[can_view\(\)](#) ([trac.versioncontrol.api.Changeset](#) method)
([trac.versioncontrol.api.Node](#) method)
([trac.versioncontrol.api.Repository](#) method)

[captioned_button\(\)](#) (in module [trac.util.presentation](#))

[cc_list\(\)](#) ([trac.web.chrome.Chrome](#) method)

default_tracker (in module trac.web.main)

DefaultTicketGroupStatsProvider (class in trac.ticket.roadmap)

delete() (trac.attachment.Attachment method)

delete_all() (trac.attachment.Attachment class method)

destroy()
(trac.tests.functional.testenv.FunctionalTestEnvironment method)

destroy_repo()
(trac.tests.functional.testenv.FunctionalTestEnvironment method)

detect_unicode() (in module trac.mimeview.api)

Deuglifier (class in trac.util.html)

diff_blocks() (in module trac.versioncontrol.diff)

DigestAuthentication (class in trac.web.auth)

disable_component() (trac.core.ComponentManager method)

dispatch() (trac.web.main.RequestDispatcher method)

dispatch_request() (in module trac.web.main)

display_rev() (trac.versioncontrol.api.Repository method)

E

embedded_numbers() (in module trac.util)

empty (in module trac.util.text)

enable_component() (trac.env.Environment method)

end_headers() (trac.web.api.Request method)

EnscriptDeuglifier (class in tracopt.mimeview.enscript)

EnscriptRenderer (class in tracopt.mimeview.enscript)

Environment (class in trac.env)

environment variable

TRAC_TEST_DB_URI [1] [2]

TRAC_TEST_TRACD_OPTIONS

environment_created()
(trac.env.IEnvironmentSetupParticipant method)

(trac.web.chrome.Chrome method)

environment_needs_upgrade()
(trac.env.IEnvironmentSetupParticipant method)

escape() (in module trac.util.html)

exception_to_unicode() (in module trac.util.text)

expand_macro() (trac.wiki.api.IWikiMacroProvider method)

expand_markup() (in module trac.util.html)

expand_tabs
(trac.mimeview.api.IHTMLPreviewRenderer attribute)

expandtabs() (in module trac.util.text)

ExtensionPoint (class in trac.core)

extensions() (trac.core.ExtensionPoint method)

F

file_or_std (class in trac.util)

filter_stream() (trac.web.api.ITemplateStreamFilter method)

filters (trac.web.main.RequestDispatcher attribute)

find_element() (in module trac.util.html)

first_last() (in module trac.util.presentation)

fix_eol() (in module trac.util.text)

FixedOffset (class in trac.util.datefmt)

format_date() (in module trac.util.datefmt)

format_datetime() (in module trac.util.datefmt)

format_emails() (trac.web.chrome.Chrome method)

format_time() (in module trac.util.datefmt)

FormTokenInjector (class in trac.util.html)

fq_class_name() (in module trac.util)

from_request() (trac.mimeview.api.RenderingContext static method)

from_utimestamp() (in module trac.util.datefmt)

FunctionalTestEnvironment (class in trac.tests.functional.testenv)

FunctionalTester (class in trac.tests.functional.testenv)

G

genshi_cache_size (trac.web.chrome.Chrome attribute)

get() (trac.cache.CacheManager method)

get_active_navigation_item()
(trac.web.chrome.INavigationContributor method)

get_all_repositories()
(trac.versioncontrol.api.RepositoryManager method)

get_all_templates_dirs() (trac.web.chrome.Chrome method)

get_annotation_data()
(trac.mimeview.api.IHTMLPreviewAnnotator method)

get_annotation_type()
(trac.mimeview.api.IHTMLPreviewAnnotator method)

get_annotation_types() (trac.mimeview.api.Mimeview method)

get_annotations() (trac.versioncontrol.api.Node method)

(trac.versioncontrol.svn_fs.SubversionNode method)

get_base() (trac.versioncontrol.api.Repository method)

[\(trac.versioncontrol.svn_fs.SubversionRepository method\)](#)
[get_branch_origin\(\)](#)
 (trac.versioncontrol.svn_fs.SubversionNode method)
[get_branches\(\)](#) (trac.versioncontrol.api.Changeset method)
[get_change_extent\(\)](#) (in module trac.versioncontrol.diff)
[get_changes\(\)](#) (trac.versioncontrol.api.Changeset method)
 (trac.versioncontrol.api.Repository method)
 (trac.versioncontrol.svn_fs.SubversionChangeset method)
 (trac.versioncontrol.svn_fs.SubversionRepository method)
[get_changeset\(\)](#) (trac.versioncontrol.api.Repository method)
 (trac.versioncontrol.svn_fs.SubversionRepository method)
[get_changeset_uid\(\)](#) (trac.versioncontrol.api.Repository method)
 (trac.versioncontrol.svn_fs.SubversionRepository method)
[get_changesets\(\)](#) (trac.versioncontrol.api.Repository method)
[get_charset\(\)](#) (trac.mimeview.api.Mimeview method)
[get_content\(\)](#) (trac.versioncontrol.api.Node method)
 (trac.versioncontrol.svn_fs.SubversionNode method)
[get_content_length\(\)](#) (trac.versioncontrol.api.Node method)
 (trac.versioncontrol.svn_fs.SubversionNode method)
[get_content_type\(\)](#) (trac.versioncontrol.api.Node method)
 (trac.versioncontrol.svn_fs.SubversionNode method)
[get_copy_ancestry\(\)](#)
 (trac.versioncontrol.svn_fs.SubversionNode method)
[get_date_format_hint\(\)](#) (in module trac.util.datefmt)
[get_date_format_jquery_ui\(\)](#) (in module trac.util.datefmt)
[get_datetime_format_hint\(\)](#) (in module trac.util.datefmt)
[get_day_names_jquery_ui\(\)](#) (in module trac.util.datefmt)
[get_db_cnx\(\)](#) (trac.env.Environment method)
[get_default_repository\(\)](#)
 (trac.versioncontrol.api.RepositoryManager method)
[get_diff_options\(\)](#) (in module trac.versioncontrol.diff)
[get_doc\(\)](#) (in module trac.util)

[get_email_map\(\)](#) (trac.web.chrome.Chrome method)
[get_enabled_components\(\)](#)
 (trac.tests.functional.testenv.FunctionalTestEnvironment method)
[get_entries\(\)](#) (trac.versioncontrol.api.Node method)
 (trac.versioncontrol.svn_fs.SubversionNode method)
[get_environments\(\)](#) (in module trac.web.main)
[get_extra_mimetypes\(\)](#)
 (trac.mimeview.api.IHTMLPreviewRenderer method)
[get_filtered_hunks\(\)](#) (in module trac.versioncontrol.diff)
[get_first_week_day_jquery_ui\(\)](#) (in module trac.util.datefmt)
[get_frame_info\(\)](#) (in module trac.util)
[get_header\(\)](#) (trac.web.api.Request method)
[get_hint\(\)](#) (trac.mimeview.api.RenderingContext method)
[get_history\(\)](#) (trac.attachment.AttachmentModule method)
 (trac.versioncontrol.api.Node method)
 (trac.versioncontrol.svn_fs.SubversionNode method)
[get_htdocs_dir\(\)](#) (trac.env.Environment method)
[get_htdocs_dirs\(\)](#) (trac.web.chrome.ITemplateProvider method)
[get_hunks\(\)](#) (in module trac.versioncontrol.diff)
[get_known_users\(\)](#) (trac.env.Environment method)
[get_last_modified\(\)](#)
 (trac.versioncontrol.svn_fs.SubversionNode method)
[get_last_traceback\(\)](#) (in module trac.util)
[get_lines_from_file\(\)](#) (in module trac.util)
[get_link_resolvers\(\)](#) (trac.wiki.api.IWikiSyntaxProvider method)
[get_log_dir\(\)](#) (trac.env.Environment method)
[get_macro_description\(\)](#)
 (trac.wiki.api.IWikiMacroProvider method)
[get_macros\(\)](#) (trac.wiki.api.IWikiMacroProvider method)
[get_max_preview_size\(\)](#) (trac.mimeview.api.Mimeview method)
[get_mimetype\(\)](#) (in module trac.mimeview.api)
 (trac.mimeview.api.Mimeview method)
[get_module_path\(\)](#) (in module trac.util)
[get_month_names_jquery_ui\(\)](#) (in module trac.util.datefmt)
[get_navigation_items\(\)](#)
 (trac.web.chrome.INavigationContributor method)
[get_node\(\)](#) (trac.versioncontrol.api.Repository method)

get_repository_id() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_repository_id() (trac.versioncontrol.api.RepositoryManager method)
get_oldest_rev() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_resource_url() (trac.attachment.AttachmentModule method)
get_pages() (trac.wiki.api.WikiSystem method)	get_search_results() (trac.attachment.AttachmentModule method)
get_path_history() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_sources() (in module trac.util)
get_path_url() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_supported_conversions() (trac.mimeview.api.IContentConverter method)
get_path_url() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_supported_types() (trac.versioncontrol.svn_fs.SubversionRepository method)
get_pkginfo() (in module trac.util)	get_supported_types() (trac.versioncontrol.api.IRepositoryConnector method)
get_previous() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_system_info() (trac.versioncontrol.svn_fs.SubversionRepository method)
get_properties() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_systeminfo() (trac.env.ISystemInfoProvider method)
get_properties() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_systeminfo() (trac.env.Environment method)
get_properties() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_tags() (trac.versioncontrol.svn_fs.SubversionRepository method)
get_quality_ratio() (trac.mimeview.api.IHTMLPreviewRenderer method)	get_templates_dir() (trac.versioncontrol.svn_fs.SubversionRepository method)
get_quickjump_entries() (trac.mimeview.api.IHTMLPreviewRenderer method)	get_templates_dirs() (trac.versioncontrol.svn_fs.SubversionRepository method)
get_read_db() (trac.env.Environment method)	get_ticket_group_stats() (trac.web.chrome.ITemplateProvider method)
get_real_repositories() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_tickets_for_milestone() (trac.ticket.roadmap.ITicketGroupStatsProvider method)
get_reporter_id() (in module trac.util)	get_time_format_jquery_ui() (in module trac.ticket.roadmap)
get_repositories() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_timeline_events() (in module trac.util.datefmt)
get_repositories() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_timezone() (trac.attachment.AttachmentModule method)
get_repositories() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_timezone_list_jquery_ui() (in module trac.util.datefmt)
get_repositories_by_dir() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_trac_environment() (trac.tests.functional.testenv.FunctionalTestEnvironment method)
get_repository() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_tracignore_patterns() (in module trac.web.main)
get_repository() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_version() (trac.wiki.api.IWikiSyntaxProvider method)
get_repository() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_wiki_syntax() (trac.wiki.api.IWikiSyntaxProvider method)
get_repository_by_path() (trac.versioncontrol.svn_fs.SubversionRepository method)	get_youngest_rev() (trac.versioncontrol.svn_fs.SubversionRepository method)
	getuser() (trac.versioncontrol.svn_fs.SubversionRepository method)
	go_to_admin() (in module trac.util)
	go_to_front() (trac.tests.functional.testenv.FunctionalTestEnvironment method)
	go_to_front() (trac.tests.functional.testenv.FunctionalTestEnvironment method)

[go_to_query\(\)](#)
 (trac.tests.functional.testers.FunctionalTester method)
[go_to_roadmap\(\)](#)
 (trac.tests.functional.testers.FunctionalTester method)
[go_to_ticket\(\)](#)
 (trac.tests.functional.testers.FunctionalTester method)
[go_to_timeline\(\)](#)
 (trac.tests.functional.testers.FunctionalTester method)
[go_to_wiki\(\)](#)
 (trac.tests.functional.testers.FunctionalTester method)
[group\(\)](#) (in module trac.util.presentation)
[grouped_stats_data\(\)](#) (in module trac.ticket.roadmap)

H

[handlers](#) (trac.web.main.RequestDispatcher attribute)
[has_hint\(\)](#) (trac.mimeview.api.RenderingContext method)
[has_node\(\)](#) (trac.versioncontrol.api.Repository method)
 (trac.versioncontrol.svn_fs.SubversionRepository method)
[has_page\(\)](#) (trac.wiki.api.WikiSystem method)
[hdf_diff\(\)](#) (in module trac.versioncontrol.diff)
[hex_entropy\(\)](#) (in module trac.util)
[Href](#) (class in trac.web.href)
[href](#) (trac.env.Environment attribute)
 (trac.web.api.Request attribute)
[htdocs_location](#) (trac.web.chrome.Chrome attribute)
[html](#) (in module trac.util.html)
[http_date\(\)](#) (in module trac.util.datefmt)

I

[IAttachmentChangeListener](#) (class in trac.attachment)
[IAttachmentManipulator](#) (class in trac.attachment)
[IAuthenticator](#) (class in trac.web.api)
[IContentConverter](#) (class in trac.mimeview.api)
[IEnvironmentSetupParticipant](#) (class in trac.env)
[ignore_case](#) (trac.web.auth.LoginModule attribute)
[ignore_missing_pages](#) (trac.wiki.api.WikiSystem attribute)
[IHTMLPreviewAnnotator](#) (class in trac.mimeview.api)
[IHTMLPreviewRenderer](#) (class in trac.mimeview.api)
[ILegacyAttachmentPolicyDelegate](#) (class in trac.attachment)
[implements\(\)](#) (in module trac.core)
 (trac.core.Component static method)

[import_namespace\(\)](#) (in module trac.util)
[INavigationContributor](#) (class in trac.web.chrome)
[init\(\)](#)
 (trac.tests.functional.testenv.FunctionalTestEnvironment method)
[insert\(\)](#) (trac.attachment.Attachment method)
[Interface](#) (class in trac.core)
[invalidate\(\)](#) (trac.cache.CacheManager method)
[InvalidAttachment](#) (class in trac.attachment)
[IRepositoryChangeListener](#) (class in trac.versioncontrol.api)
[IRepositoryConnector](#) (class in trac.versioncontrol.api)
[IRepositoryProvider](#) (class in trac.versioncontrol.api)
[IRequestFilter](#) (class in trac.web.api)
[IRequestHandler](#) (class in trac.web.api)
[is_24_hours\(\)](#) (in module trac.util.datefmt)
[is_binary\(\)](#) (in module trac.mimeview.api)
 (trac.mimeview.api.Mimeview method)
[is_component_enabled\(\)](#)
 (trac.core.ComponentManager method)
 (trac.env.Environment method)
[is_default\(\)](#) (in module trac.versioncontrol.api)
[is_enabled\(\)](#) (trac.core.ComponentManager method)
[is_inline\(\)](#) (trac.wiki.api.IWikiMacroProvider method)
[is_path_below\(\)](#) (in module trac.util)
[is_viewable\(\)](#) (trac.versioncontrol.api.Changeset method)
 (trac.versioncontrol.api.Node method)
 (trac.versioncontrol.api.Repository method)
[istext\(\)](#) (in module trac.util.presentation)
[ISystemInfoProvider](#) (class in trac.env)
[ITemplateProvider](#) (class in trac.web.chrome)
[ITemplateStreamFilter](#) (class in trac.web.api)
[ITicketGroupStatsProvider](#) (class in trac.ticket.roadmap)
[IWikiChangeListener](#) (class in trac.wiki.api)
[IWikiMacroProvider](#) (class in trac.wiki.api)
[IWikiPageManipulator](#) (class in trac.wiki.api)
[IWikiSyntaxProvider](#) (class in trac.wiki.api)

J

[javascript_quote\(\)](#) (in module trac.util.text)
[jquery_location](#) (trac.web.chrome.Chrome attribute)
[jquery_ui_location](#) (trac.web.chrome.Chrome attribute)

jquery_ui_theme_location (trac.web.chrome.Chrome attribute)

L

lazy (class in trac.util)

levenshtein_distance() (in module trac.util.text)

load() (trac.web.auth.DigestAuthentication method)

load_template() (trac.web.chrome.Chrome method)

LocalTimezone (class in trac.util.datefmt)

log_file (trac.env.Environment attribute)

log_format (trac.env.Environment attribute)

log_level (trac.env.Environment attribute)

log_type (trac.env.Environment attribute)

login() (trac.tests.functional.testers.FunctionalTester method)

LoginModule (class in trac.web.auth)

logo_alt (trac.web.chrome.Chrome attribute)

logo_height (trac.web.chrome.Chrome attribute)

logo_link (trac.web.chrome.Chrome attribute)

logo_src (trac.web.chrome.Chrome attribute)

logo_width (trac.web.chrome.Chrome attribute)

logout() (trac.tests.functional.testers.FunctionalTester method)

M

macro_providers (trac.wiki.api.WikiSystem attribute)

mainnav_order (trac.web.chrome.Chrome attribute)

make_label_from_target() (trac.wiki.api.WikiSystem method)

makedirs() (in module trac.util)

manipulators (trac.attachment.AttachmentModule attribute)

match_request() (trac.web.api.IRequestHandler method)

max_preview_size (trac.mimeview.api.Mimeview attribute)

max_size (trac.attachment.AttachmentModule attribute)

max_zip_size (trac.attachment.AttachmentModule attribute)

md5crypt() (in module trac.util)

metanav_order (trac.web.chrome.Chrome attribute)

method (trac.web.api.Request attribute)

MilestoneModule (class in trac.ticket.roadmap)

Mimeview (class in trac.mimeview.api)

modify_repository() (trac.versioncontrol.api.DbRepositoryProvider method)

MySQL

testing on

N

NaivePopen (class in trac.util)

navigation_contributors (trac.web.chrome.Chrome attribute)

needs_upgrade() (trac.env.Environment method)

never_obfuscate_mailto (trac.web.chrome.Chrome attribute)

next_rev() (trac.versioncontrol.api.Repository method)
(trac.versioncontrol.svn_fs.SubversionRepository method)

Node (class in trac.versioncontrol.api)

normalize_path() (trac.versioncontrol.api.Repository method)

(trac.versioncontrol.svn_fs.SubversionRepository method)

normalize_rev() (trac.versioncontrol.api.Repository method)

(trac.versioncontrol.svn_fs.SubversionRepository method)

normalize_whitespace() (in module trac.util.text)

NoSuchChangeset (class in trac.versioncontrol.api)

NoSuchNode (class in trac.versioncontrol.api)

notify() (trac.versioncontrol.api.RepositoryManager method)

O

obfuscate_email_address() (in module trac.util.text)

open_environment() (in module trac.env)

P

PAGE_SPLIT_RE (trac.wiki.api.WikiSystem attribute)

pages (trac.wiki.api.WikiSystem attribute)

paginate() (in module trac.util.presentation)

Paginator (class in trac.util.presentation)

pairwise() (in module trac.util)

parent (trac.mimeview.api.RenderingContext attribute)

parent_revs() (trac.versioncontrol.api.Repository method)

parse_arg_list() (in module trac.web.api)

parse_args() (in module trac.wiki.api)

parse_date() (in module trac.util.datefmt)

`partition()` (in module `trac.util`)
`path_info` (`trac.web.api.Request` attribute)
`path_to_unicode()` (in module `trac.util.text`)
`pathjoin()` (in module `trac.util`)
`PhpDeuglifier` (class in `tracopt.mimeview.php`)
`PHPRenderer` (class in `tracopt.mimeview.php`)
`plaintext()` (in module `trac.util.html`)
`post_create()`
(`trac.tests.functional.testenv.FunctionalTestEnvironment` method)

`post_process_request()` (`trac.web.api.IRequestFilter` method)

Postgres

testing on

PostgreSQL

testing on

`pre_process_request()` (`trac.web.api.IRequestFilter` method)

`prepare_attachment()`
(`trac.attachment.IAttachmentManipulator` method)

`prepare_request()` (`trac.web.chrome.Chrome` method)

`prepare_wiki_page()`
(`trac.wiki.api.IWikiPageManipulator` method)

prerequisites

tests

`pretty_size()` (in module `trac.util.text`)

`pretty_timedelta()` (in module `trac.util.datefmt`)

`preview_data()` (`trac.mimeview.api.Mimeview` method)

`previous_rev()` (`trac.versioncontrol.api.Repository` method)

(`trac.versioncontrol.svn_fs.SubversionRepository` method)

`prevnext_nav()` (in module `trac.web.chrome`)

`print_table()` (in module `trac.util.text`)

`printerr()` (in module `trac.util.text`)

`printout()` (in module `trac.util.text`)

`process_request()` (`trac.web.api.IRequestHandler` method)

`project_admin` (`trac.env.Environment` attribute)

`project_admin_trac_url` (`trac.env.Environment` attribute)

`project_description` (`trac.env.Environment` attribute)

`project_footer` (`trac.env.Environment` attribute)

`project_icon` (`trac.env.Environment` attribute)

`project_name` (`trac.env.Environment` attribute)

`project_url` (`trac.env.Environment` attribute)

`providers` (`trac.versioncontrol.api.RepositoryManager` attribute)

Q

`query_string` (`trac.web.api.Request` attribute)

`quickjump()`
(`trac.tests.functional.testenv.FunctionalTester` method)

`quote_query_string()` (in module `trac.util.text`)

R

`Ranges` (class in `trac.util`)

`raw_input()` (in module `trac.util.text`)

`read()` (`trac.web.api.Request` method)

`read_file()` (in module `trac.util`)

`redirect()` (`trac.web.api.Request` method)

`reload_repositories()`
(`trac.versioncontrol.api.RepositoryManager` method)

`remote_addr` (`trac.web.api.Request` attribute)

`remote_user` (`trac.web.api.Request` attribute)

`remove_repository()`
(`trac.versioncontrol.api.DbRepositoryProvider` method)

`rename()` (in module `trac.util`)

`render()` (`trac.mimeview.api.IHTMLPreviewRenderer` method)

(`trac.mimeview.api.Mimeview` method)

`render_macro()` (`trac.wiki.api.IWikiMacroProvider` method)

`render_template()` (`trac.web.chrome.Chrome` method)

`render_unsafe_content`
(`trac.attachment.AttachmentModule` attribute)

(`trac.wiki.api.WikiSystem` attribute)

`renderers` (`trac.mimeview.api.Mimeview` attribute)

`RenderingContext` (class in `trac.mimeview.api`)

`reparent_all()` (`trac.attachment.Attachment` class method)

`repo_path_for_initenv()`
(`trac.tests.functional.testenv.FunctionalTestEnvironment` method)

`repositories_section`
(`trac.versioncontrol.api.RepositoryManager` attribute)

`Repository` (class in `trac.versioncontrol.api`)

`repository_dir`
(`trac.versioncontrol.api.RepositoryManager` attribute)

`repository_sync_per_request`
(`trac.versioncontrol.api.RepositoryManager` attribute)

`repository_type`
(`trac.versioncontrol.api.RepositoryManager` attribute)

RepositoryManager (class in trac.versioncontrol.api)
Request (class in trac.web.api)
RequestDispatcher (class in trac.web.main)
RequestDone (class in trac.web.api)
RequestWithSession (class in trac.web.main)
reset_metadata() (trac.cache.CacheManager method)
resizable_textareas (trac.web.chrome.Chrome attribute)
resource (trac.versioncontrol.api.Changeset attribute)
(trac.versioncontrol.api.Node attribute)
restart()
(trac.tests.functional.testenv.FunctionalTestEnvironment method)
returns_source
(trac.mimeview.api.IHTMLPreviewRenderer attribute)
rev_older_than() (trac.versioncontrol.api.Repository method)
(trac.versioncontrol.svn_fs.SubversionRepository method)

RFC

RFC 2617

RoadmapModule (class in trac.ticket.roadmap)

running

tests

S

safe__import__() (in module trac.util)
safe_repr() (in module trac.util)
safe_schemes (trac.wiki.api.WikiSystem attribute)
scheme (trac.web.api.Request attribute)
secure_cookies (trac.env.Environment attribute)
select() (trac.attachment.Attachment class method)
send_auth_request()
(trac.web.auth.DigestAuthentication method)
send_converted() (trac.mimeview.api.Mimeview method)
send_file() (trac.web.api.Request method)
send_header() (trac.web.api.Request method)
send_response() (trac.web.api.Request method)
separated() (in module trac.util.presentation)
server_name (trac.web.api.Request attribute)
server_port (trac.web.api.Request attribute)
set_hints() (trac.mimeview.api.RenderingContext method)
setup_config() (trac.env.Environment method)
setup_log() (trac.env.Environment method)

setup_participants (trac.env.Environment attribute)
shared_htdocs_dir (trac.web.chrome.Chrome attribute)
shared_plugins_dir (trac.env.Environment attribute)
shared_templates_dir (trac.web.chrome.Chrome attribute)
short_rev() (trac.versioncontrol.api.Repository method)
shorten_line() (in module trac.util.text)
show_email_addresses (trac.web.chrome.Chrome attribute)
show_ip_addresses (trac.web.chrome.Chrome attribute)
shutdown() (trac.env.Environment method)
(trac.versioncontrol.api.RepositoryManager method)
SilverCityRenderer (class in tracopt.mimeview.silvercity)
split_page_names (trac.wiki.api.WikiSystem attribute)
start()
(trac.tests.functional.testenv.FunctionalTestEnvironment method)
stats_provider (trac.ticket.roadmap.MilestoneModule attribute)
(trac.ticket.roadmap.RoadmapModule attribute)

stop()
(trac.tests.functional.testenv.FunctionalTestEnvironment method)
stream_encoding() (in module trac.util.text)
stream_filters (trac.web.chrome.Chrome attribute)
stripws() (in module trac.util.text)
SubversionChangeset (class in trac.versioncontrol.svn_fs)
SubversionConnector (class in trac.versioncontrol.svn_fs)
SubversionNode (class in trac.versioncontrol.svn_fs)
SubversionRepository (class in trac.versioncontrol.svn_fs)
sync() (trac.versioncontrol.api.Repository method)
sync_changeset() (trac.versioncontrol.api.Repository method)
syntax_providers (trac.wiki.api.WikiSystem attribute)
system_info_providers (trac.env.Environment attribute)

T

tab_width (trac.mimeview.api.Mimeview attribute)
template_providers (trac.web.chrome.Chrome attribute)
testing on
MySQL

PostgreSQL

Postgres

tests

prerequisites

running

text_width() (in module trac.util.text)

ticket_set_milestone()
(trac.tests.functional.testenv.FunctionalTester method)

TicketGroupStats (class in trac.ticket.roadmap)

timezone() (in module trac.util.datefmt)

to_datetime() (in module trac.util.datefmt)

to_js_string() (in module trac.util.text)

to_json() (in module trac.util.presentation)

to_ranges() (in module trac.util)

to_timestamp() (in module trac.util.datefmt)

to_unicode() (in module trac.util.text)
(trac.mimeview.api.Mimeview method)

to_utf8() (in module trac.util.text)
(trac.mimeview.api.Mimeview method)

to_utimestamp() (in module trac.util.datefmt)

trac.attachment (module)

trac.cache (module)

trac.core (module)

trac.env (module)

trac.mimeview.api (module)

trac.tests.functional.testenv (module)

trac.tests.functional.testenv (module)

trac.ticket.roadmap (module)

trac.util (module)

trac.util.datefmt (module)

trac.util.datefmt.all_timezones (in module trac.util.datefmt)

trac.util.datefmt.localtime (in module trac.util.datefmt)

trac.util.html (module)

trac.util.presentation (module)

trac.util.text (module)

trac.versioncontrol.api (module)

trac.versioncontrol.diff (module)

trac.versioncontrol.svn_fs (module)

trac.web.api (module)

trac.web.auth (module)

trac.web.chrome (module)

trac.web.href (module)

trac.web.main (module)

trac.wiki.api (module)

trac.wiki.macros (module)

TRAC_TEST_DB_URI [1]

TracError (class in trac.core)

TracHTMLSanitizer (class in trac.util.html)

tracopt.mimeview (module)

tracopt.mimeview.enscript (module)

tracopt.mimeview.php (module)

tracopt.mimeview.silvercity (module)

TransposingElementFactory (class in trac.util.html)

treat_as_binary (trac.mimeview.api.Mimeview attribute)

truncate() (trac.util.Ranges method)

U

unescape() (in module trac.util.html)

unicode_from_base64() (in module trac.util.text)

unicode_passwd (class in trac.util.text)

unicode_quote() (in module trac.util.text)

unicode_quote_plus() (in module trac.util.text)

unicode_to_base64() (in module trac.util.text)

unicode_unquote() (in module trac.util.text)

unicode_urlencode() (in module trac.util.text)

unified_diff() (in module trac.versioncontrol.diff)

unquote_label() (in module trac.util.text)

upgrade() (trac.env.Environment method)

upgrade_environment()
(trac.env.IEnvironmentSetupParticipant method)

urandom (in module trac.util)

use_xsendfile (trac.web.main.RequestDispatcher attribute)

user_time() (in module trac.util.datefmt)

V

validate_attachment()
(trac.attachment.IAttachmentManipulator method)

validate_page_name() (in module trac.wiki.api)

validate_wiki_page()
(trac.wiki.api.IWikiPageManipulator method)

VCS

verify() (trac.env.Environment method)

viewable_attachments()
(trac.attachment.AttachmentModule method)

W

`web_context()` (in module `trac.web.chrome`)

`wiki_page_added()` (`trac.wiki.api.IWikiChangeListener` method)

`wiki_page_changed()`
(`trac.wiki.api.IWikiChangeListener` method)

`wiki_page_deleted()` (`trac.wiki.api.IWikiChangeListener` method)

`wiki_page_renamed()`
(`trac.wiki.api.IWikiChangeListener` method)

`wiki_page_version_deleted()`
(`trac.wiki.api.IWikiChangeListener` method)

`WikiMacroBase` (class in `trac.wiki.macros`)

`WikiSystem` (class in `trac.wiki.api`)

`WindowsError` (class in `trac.util`)

`with_transaction()` (`trac.env.Environment` method)

`wrap()` (in module `trac.util.text`)

`write()` (`trac.web.api.Request` method)

Python Module Index

a

[trac.attachment](#)

c

[trac.cache](#)

[trac.core](#)

e

[trac.env](#)

m

[trac.mimeview](#)

[trac.mimeview.api](#)

[tracopt.mimeview](#)

[tracopt.mimeview.enscript](#)

[tracopt.mimeview.php](#)

[tracopt.mimeview.silvercity](#)

t

[trac.tests](#)

[trac.tests.functional.testenv](#)

[trac.tests.functional.testter](#)

[trac.ticket](#)

[trac.ticket.roadmap](#)

u

[trac.util](#)

[trac.util.datefmt](#)

[trac.util.html](#)

[trac.util.presentation](#)

[trac.util.text](#)

v

[trac.versioncontrol](#)

[trac.versioncontrol.api](#)

[trac.versioncontrol.diff](#)

[trac.versioncontrol.svn_fs](#)

w

[trac.web](#)

[trac.web.api](#)

[trac.web.auth](#)

[trac.web.chrome](#)

[trac.web.href](#)

[trac.web.main](#)

[trac.wiki](#)

[trac.wiki.api](#)

[trac.wiki.macros](#)