gnats2w

Yet another Web interface to GNATS
gnats2w version 0.14

Milan Zamazal pdm@freesoft.cz

This manual documents gnats2w, version 0.14. gnats2w is yet another WWW interface to GNATS, See section \Top" in *The GNATS manual*, a bug tracking system.

This is the first draft of the manual. It can be buggy, incomplete, contain obsolete information, and in not much good English. Any improvements are welcome.

1 Introduction

gnats2w is yet another WWW interface to GNATS, See section \Top" in *The GNATS manual*. Why to create yet another WWW interface to a bug tracking system, when several others already exist? I wasn't satis ed with any WWW interface I found | it either required patching GNATS or was a little hacky without reasonable extendibility. So I started an \one evening" project, which became a little more sophisticated program than I had planned originally.

1.1 gnats2w design

The aims of gnats2w are the following ones:

To allow access to all the important GNATS user functions. The system should not disqualify features provided by GNATS. Even more, it could provide some more high level functions for common tasks, which must be composed of several actions in GNATS command line interface.

To allow it with clean interface to GNATS. The Web interface should not relay on current format of GNATS con guration les | it should use GNATS command accessors instead.

To be relatively easily extensible. Some other Web interfaces I have seen were just terrible mess one was already afraid install, even less more to hack.

To allow access regulation. Not all software produced is public, even free software projects can be \hidden" projects at the beginning. GNATS allows access regulation and gnats2w allow access regulations too, with help of authentization classes.

To be well documented. Software without good documentation is useless. Dot. It concerns reference manual as well as on-line help.

To be internationalized. Providing users with access to a bug tracking system in their own language can improve their cooperation signi cantly and make the software much more friendly for them.

gnats2w is written in Python. It allows producing quite modular and readable code and producing it quickly with the help of many libraries (even in case of questionable quality of some modules this allows quick start).

For technical description of gnats2w, see See Chapter 5 [Internals], page 16.

1.2 Copying

You should have received a copy of the GNU General Public License along with this program; see the file COPYING. If not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

1.3 Getting the latest version

gnats2w is distributed with the Debian GNU/Linux distribution. See `http://www.debian.org/debian/dists/unstable/source/web/` for the latest gnats2w upstream sources. The FTP source is `ftp://ftp.debian.org/debian/dists/unstable/source/web/`.

1.4 Known problems

There are two kinds of problems with gnats2w:

GNATS bugs. Some known bugs are caused by GNATS itself. Hopefully they get fixed sometimes. If you can help this, your help will be definitely welcome. Please see GNATS documentation for appropriate contacts.

Real gnats2w bugs. These bugs are supposed to be more frequent. If you find any problems or have any suggestions, write to Milan Zamazal `pdm@freesoft.cz`. Notice that I have not much time to maintain gnats2w, so patches are especially welcome.

Before reporting any bug reports, please take a look at the latest version of gnats2w if it has not been fixed already.

1.5 Future enhancements

The following things are planned to be added to gnats2w sometimes. Your help is always welcome!

Features planned to add before the 1.0 release:

On change of responsible, inform submitter about e-mail address of new responsible.

Logging of actions.

Better support for duplicates.

Features planned to add sometimes after the 1.0 release:

Outputs for a printer.

Allow specifying submitter-id in submit.

Make API for bug submittions.

Write preprocesor for extracting language dependent messages.

Non-editable names and e-mails from known identity.

Make notify addresses available.

Date searching hot buttons.

Make `` `cgiscript.py' `` less ugly.

Frankly, I do not plan to work on gnats2w much soon. I plan to work on other free software projects, so if you think you would be better gnats2w author, please contact me.

1.6 Author

If you have any suggestions, enhancements, etc. of this software, feel free to contact its author, Milan Zamazal pdm@freesoft.cz.

2 Installation

Unfortunately, there is currently no con gure installation process. However, the installation should be straightforward anyway. The best way to install gnats2w is to install them on Debian systems. If your target machine doesn't run Debian, you have to perform the steps described below.

2.1 Required software

The most di cult thing about it is probably getting all the software needed installed. You need to have installed and con gured properly the following software components:

GNATS 3.113

Python 1.5.X

Zope 0.9.5 You need only the ZTemplates module of Zope.

HTMLgen 2.1 This is

Apache May be it works with other HTTP servers too, but I don't know.

All the software is available in Debian.

If you want to use gnats2w with FastCGI, you must have installed the `fcgi' Python module too, See Section 3.5 [FastCGI], page 14.

2.2 Installation steps

The recommended way of installing gnats2w is from its Debian package. The Debian GNU distribution makes the installation one-command process and ensures all the needed software is installed. If you choose this way of installation, you can skip the rest of this section. If you do not choose to install gnats2w from Debian package for whatever reason, follow the steps described below.

After you have all the required software installed and working, See Section 2.1 [Required software], page 5, you can perform the following steps to install gnats2w:

1. Edit con guration variables in `Makefile'. Usually, you should only set the `prefix' variable to the main installation directory and the `CGI_DIR' and `FCGI_DIR' variables to locations, where you want to have placed the CGI and FastCGI links. It can also be a good idea to set `TMP_DIR' to some secure location.
2. Run `make'. This should be totally straightforward.
3. Log in as the `gnats' user and run `make install'. If you have no `gnats' user on your system, create one. No special UID number is required, on Debian systems it is 41.
4. Create the le `config' in the directories with (Fast)CGI scripts. You can change global `config.py' options there. In this way you can have di erent con gurations for di erent GNATS databases or to set di erent access rights for di erent URLs.

After succesfull installation and con guration, you can enter gnats2w at the URL of the `index.cgi' script, See Chapter 4 [Usage], page 15.

3 Configuration

gnats2w can be configured basically by setting the variables in the `config.py' file. This defines basic installation directories, e-mails, URLs, GNATS database, some Web forms element appearance, etc. User specific customization can be achieved through environment variables, which can be set by the Web server with respect to an URL, from which a CGI script was invoked.

Special care is applied to access rights. gnats2w allows you to define fine control about who can access the bug database and who can modify it, allowing any authentization method you can use in Python.

Finally, gnats2w has FCGI support to reduce overhead caused by Python startup times. With a bit of configuration, you can avoid this overhead.

3.1 Basic configuration

The main configuration place is the file `config.py'. It consists mainly of configuration variables, which you should set according to your local installation and your local requirements. Primarily URLs and paths are set here.

The file contains configuration variables and their values separated by the `=' operator. This is standard Python module file but do not worry if you do not know Python. All special you need to know is that strings are enclosed in apostrophes or double quotes and that `None` is something like `nil` in Lisp or `NULL` in C.

3.1.1 Installation specific configuration

You must set all the variables here to values specific to your site and gnats2w installation.

Directory paths:

`GNATS_PRG_DIR`
: The directory, where all the GNATS *internal* programs and scripts sit.

`TEMPLATE_DIR`
: The directory containing DTML templates.

`DICTIONARY_DIR`
: The directory containing language dictionaries.

URLs:

`HOST_URL` Host URL, including protocol and port number (if needed).

`BASE_URL` Base URL for static HTML documents, it must be a full pointer to `index.html'. This should be `HOST_URL` and the `index.html' paths.

Mailing:

`MAIL` Command to send an e-mail given on the standard input.

`ADMIN_MAIL`
: E-mail address of the gnats2w administrator.

3.1.2 Database speci c con guration

The following con guration variables specify your database setup.

`GNATS_DIR`
: The default database directory.

`SITE`
: Site to submit bugs to, as de ned by `GNATS_SITE', See section \con g le" in *The GNATS manual*.

`DEFAULT_CATEGORY`
: The category, which is o ered as the default one in the Web bug submitting form.

`AUTO_CONFIDENTIAL`
: This is boolean ag denoting con dentiality handling. If it is `0`, there is nothing special about con dentiality. If it is `1`, new reports can be set con dential, See section \Problem Report elds" in *The GNATS manual*, automatically for some categories. Such auto-con dential categories are denoted by the * character at the very beginning of their category description in the `categories' le, the second eld, See section \categories le" in *The GNATS manual*.[1]

3.1.3 Language con guration

One important thing in bug hunting is to allow users accessing a bug tracking system in their own language. gnats2w has some support for message translations.[2] gnats2w consideres language codes given by the Web browser to it and selects appropriate available message catalog and HTML templates.

The default language is English. If you want to support another language, you need translated `*.dtml' les and a message catalog. Translated `*.dtml' les have an additional extension, which is the ISO language code, e.g. Czech document templates have the extension `*.dtml.cs'. Message catalogs have names equal to the language code and are placed in the directory de ned by the variable `DICTIONARY_DIR`, See Section 3.1.1 [Installation speci c con guration], page 6. Additionally, you can have your own catalogs in the directory de ned by the variable `DICTIONARY_DIR_LOCAL` (see below) for new messages appearing in your own extension code.

The format of message catalog les is as follows:

```
dictionary = {
  'English message 1': 'translation 1',
  'English message 2': 'translation 2',
  'English message 3': 'translation 3',
  ...
  'English message N': 'translation N'
}
```

I.e. it contains Python variable `dictionary`, which is an associative array with English messages as keys and their translations as values.

The following con guration variables has some relation to the translations:

---

[1] This is gnats2w extension not supported by GNATS.

[2] This is not based on gettext yet.

`LANGUAGE` Default language code. If you do not want to specify some local language, just set it to the empty string (`''`).

`DICTIONARY_DIR_LOCAL`
The directory containing your local message catalogs.

3.1.4 Web interface con guration

The main index page has a part left for local text. For example, you can put links to various category selections here, See Section 3.2 [Environment variables], page 8.

If the variable `LOCAL_INDEX_FUNCTION` is `None`, the local part is left empty. Otherwise `LOCAL_INDEX_FUNCTION` should be set to a Python function generating an HTML 3.2 code, which will be put to the local part of the main index page. The function takes no arguments, but can access global objects, See Chapter 5 [Internals], page 16.

There are also some visibility ags, which allow you to hide some information not interesting to your users. All the following variables can have the value `0` (*not visible*) or `1` (*visible*):

`DISPLAY_CONFIDENTIAL`
Whether con dential button and status should be displayed in forms.

`DISPLAY_ORGANIZATION`
Whether organization name should be displayed in forms.

`DISPLAY_IDREQUEST`
Whether request for registration link should be displayed on the main index page.

3.1.5 Varying con gurations

If the directory with CGI scripts contains a le named `` `config' ``, all settings in this le take precedence over the ones in `` `config.py' ``. This way you can have di erent con gurations for di erent URLs | just make symbolink links to all the CGI scripts in another directory and override the options with `` `config' `` le in the new directory.

One obvious application of this feature is to allow access to di erent GNATS databases through di erent URLs by setting the variable `GNATS_DIR`, See Section 3.1.2 [Database speci c con guration], page 7, in di erent `` `config' `` les.

3.2 Environment variables

Not everything can be set in a static con guration le. For example, user preferences are di cult to store there. To make such problems easier, gnats2w can be customized also by environment variables. Then you can with enough powerful Web servers (like Apache) customize gnats2w e.g. through URL paths to gnats2w scripts. For example, Apache rewrite rules allow you to set the environment variables based on URL, while rewriting the URL into some canonical form.

There are considered two such environment variables:

GNATS_EXPERT_MODE
: By default, gnats2w marks links to help with three question marks and has some verbose explanations in forms. This is because many users can overlook help links consisting of only single question mark and also need more assistance than one would expect. However these extra helps have bad in uence on page arrangement and can make the orientation in them less easy. If you set this variable to the value 'on', the helps are reduced.

GNATS_CATEGORIZE
: If you want to make \thematic" category sets (e.g. on sites with a lot of categories), you achieve this by grouping categories by some common pre xes and setting this variable to one of the pre xes. Then only categories with this pre x are o ered in category lists and only those can be accessed by a user (it can be used thus for simple access regulation, for more sophisticated access regulation see Section 3.3 [Access rights], page 9). If GNATS_CATEGORIZE is an empty string (''), all accessible categories are visible.

3.3 Access rights

THIS SECTION HAS TO BE COMPLETED. Volunteers?

By default, gnats2w allows access everyone to everything. This is often not what you would like to have. So gnats2w is equipped with some support for access regulation and authentization. The authorization and authentization basic gnats2w principles are described below.

The module `access.py' de nes some basic Python classes, of which you can compose your own access customization classes. If you want to change the default behavior, create new access class and assign it to the con guration variable ACCESS, See Section 3.1 [Basic con guration], page 6.

3.3.1 Access classi cation

Access rights are divided into three categories:

Submit The right to submit new problem reports.

View Viewing search results and individual problem reports.

Edit Editing problem reports.

For each of this category, the basic rights can be:

Access denied
: No access allowed.

Restricted access
: Access to non-con dential problem reports only.

Access granted
: Full access to the problem.

The abstract class `Access` de nes appropriate constants for all these categories: `Access.SUBMIT`, `Access.VIEW`, `Access.EDIT`, `Access.DENIED`, `Access.RESTRICTED`, `Access.FULL`.

3.3.2 Authentization

Identi cation of the accessor is composed from the three parts:

Identity Key name as used in the `responsible', See section \responsible le" in *The GNATS manual*, and `submitter', See section \submitters le" in *The GNATS manual*, les.

Name Real human name.

Email E-mail address.

The following authentization classes are prede ned:

`Email_Identity`
: This class gets user's identity from his e-mail address. Note this authentization method is totally unsafe and can be abused easily.

`HTTPS` This class gets user's identity from his SSL certi cate. This authentization method is safe, but only if your Web server is con gured in such a way, that it allows access only with a valid certi cate signed by a trusted certi cation authority.

3.3.3 Access de nition

You can assign whatever subclass of the `Access` class you de ne. One useful prede ned subclass of the `Access` class is `Enumeration`. The following text describes its usage.

Access is de ned by a dictionary with keys as users' identities and values de ning access for a user with a given identity. If the dictionary contains an item with an empty string as a key, it de nes default access. If there is no such an item, access is denied for accessors not presented in the dictionary.

Values are sequences of triples (*ACCESS*, *RESTRICTED_ALLOWED*, *SPECIAL*), where the items mean:

ACCESS One of the `Access` class constants `SUBMIT`, `VIEW`, or `EDIT`.

RESTRICTED_ALLOWED
: I true, access to restricted problems is allowed.

SPECIAL One of the `Enumerated` class constants `RESPONSIBLE` (allow access only for the responsible), `ORIGINATOR` (allow access only for the originator), `RELATED` (allow access only for the responsible or the originator); or `None` (allow access for all).

For a given key, several triples can be de ned in its value sequence, with the same of di erent elements. The most liberal triple for a particular kind of access is considered.

The dictionary described above is simply given as the only argument of the `Enumeration` constructor. The `Enumeration` instance is assigned to the `ACCESS` con guration variable.

3.3.4 Access de nition examples

The simplest thing you can do is to allow access anyone to everything:

```
from gnats2w.access import Access
ACCESS = Access ()
```

Let us look at more complicated example:

```
from gnats2w.access import Enumeration
MY_ACCESS = {'administrator': ((Access.SUBMIT, 1, None),
                               (Access.EDIT, 1, None),
                               (Access.VIEW, 1, None)),
             'certificate': ((Access.SUBMIT, 1, None),
                             (Access.EDIT, 1, Access.RELATED),
                             (Access.VIEW, 1, None)),
             '': ((Access.SUBMIT, 0, None),
                  (Access.EDIT, 0, Access.RELATED),
                  (Access.VIEW, 0, None))}
ACCESS = Enumeration (MY_ACCESS)
```

This con guration has the following features:

For non-con dential categories, See Section 3.1.2 [Database speci c con guration], page 7, access is allowed to anyone, but editting a problem is allowed only to the responsible and the originator of this problem.

Con dential problems are accessible in the same way, but only for users with identi - cation `certificate`.

Finally user identi ed as `administrator` can do everything with the problems.

To let it work reasonably, there has to be de ned mapping on the user identi cations `certificate` and `administrator`. This can be achieved e.g. in the following way:

```
from gnats2w import access

class My_Enumeration (Enumeration, access.HTTPS):

    def identity (self):
        name = self.name ()
        if name == 'John Powerful':
            return 'administrator'
        elif name:
            return 'certificate'
        else:
            return ''

ACCESS = My_Enumeration (MY_ACCESS)
```

3.4 Customizing templates

If you don't like look of gnats2w pages, you can customize them easily. Just change the template `.dtml' les in the `template' subdirectory and try not to throw away any form eld. That's all.

3.4.1 Template variables

You can use variable values generated by the CGI script inside the templates. These values are inserted through the tags of the form <!--#var ... -->, see source codes of the templates for examples. Variable names are case sensitive. For more information about using variable constructs, conditionals, etc. in templates, see Zope's DTML manual.

These are template variables which can be used in any template:

`adminmail`
: GNATS administrator's e-mail as a string.

`base`
: Base URL of the document.

`expert`
: Boolean variable set to true, i expert mode (shorter, for more experienced users) is enabled.

`gen_submit_url`
: Python function to be called for generating URL for some gnats2w CGI script.

`versions`
: Text variable containing names and versions of software pieces used.

In the following table you can nd all the variables provided for inclusion in particular templates:

`` `categories.dtml' ``
: `categories` (HTML table of available categories)

`` `classes.dtml' ``
: `classes` (HTML table of available classes)

`` `edit-error.dtml' ``
: `errors` (HTML description of errors), `problem` (instance of the Python class `Problem`)

`` `edit-ok.dtml' ``
: `dump` (HTML description of the PR), `problem` (instance of the Python class `Problem`)

`` `edit.dtml' ``
: `category` (PR category), `changers` (HTML code of possible submitters), `class` (PR class), `confidentialp` (boolean, true i PR is con dential), `confidential_checked` (boolean, con dential check eld checked), `dump` (HTML description of the PR), `email` (submitter's mail address), `id` (PR number), `mail` (responsible's mail address), `originator` (submitter's name), `priority` (PR priority), `responsible` (responsible's name), `responsibles` (HTML code of possible responsibles), `severity` (PR severity), `state` (PR state)

`` `error.dtml' ``
: `error` (text of the error message)

`` `index.dtml' ``
: `idrequestp` (boolean, true i ID allocation request is allowed), `local` (local HTML code, as speci ed in the con g le)

`problem.dtml'
: *This is only subpart of larger templates, not a complete HTML document.* arrived (PR arrival date), category (PR category), class (PR class), closed (PR closing date), confidential (boolean, PR con dentiality), confidentialp (boolean, true i PR is con dential), description (PR description), environment (environment, in which the problem appeared), fix (x description of the problem), modified (last modi cation time), note (PR notes), organization (submitter's organization), originator (submitter's name), priority (PR priority), release (release stated in the PR), repeat (how to repeat the PR), responsible (responsible's name), severity (PR severity), state (PR state), synopsis (PR synopsis), trail (PR trail), unformatted (unformatted part of the PR)

`query-result.dtml'
: number (number of problems found), problem_list (HTML table of problems found), query (query string)

`query.dtml'
: catcgi (URL of the category listing), categories (<SELECT> of available categories), clacgi (URL of the class listing), classes (<SELECT> of available classes), confidentialp (boolean, true i PR is con dential), responsible (HTML code of possible responsibles), stacgi (URL of the state listing), states (<SELECT> of available states)

`request-error.dtml'
: No special variables.

`request-ok.dtml'
: No special variables.

`request.dtml'
: No special variables.

`states.dtml'
: states (HTML table of available states)

`submit-error.dtml'
: errors (HTML description of errors), problem (instance of the Python class Problem)

`submit-ok.dtml'
: dump (HTML description of the PR), problem (instance of the Python class Problem)

`submit.dtml'
: categories (<SELECT> of available categories), classes (<SELECT> of available classes)

`traceback.dtml'
: basic_info (preformatted text, dump of a Python exception), traceback (preformatted text, Python traceback)

`view.dtml'
: dump (HTML description of the PR), id (PR number)

3.4.2 Contributing templates

If you want to share your templates with other people and to include them into the distribution package, please send them to me, See Section 1.6 [Author], page 4. But please ensure the following conditions are satis ed:

The HTML les are strictly HTML compliant. Run an SGML parser on them to ensure this (one good parser/checker is the `sp' package by James Clark).

Ensure the author of the new/modi ed les is known and that the les can be redistributed under the terms of GPL.

3.5 FastCGI

To avoid slow Python startups of CGI scripts, gnats2w provides an interface to FastCGI. To use gnats2w with FastCGI, you need have installed the module `fcgi.py'. You can nd it at `http://www.fastcgi.com'.

After this, setup your Web server appropriately to get FastCGI support and you can run gnats2w through the `fcgiscripts.py' module and symbolic links made to it.

4 Usage

There is nothing special about usage of the interface. Just enter the top level URL of the interface and walk through links. The interface should be self-explaining and there is an online help under the ? marks on many pages.

If you think this usage information is insu cient and you have an idea, how to improve it, please let me know, See Section 1.4 [Known problems], page 3.

5 Internals

This section provides information about extending gnats2w and using APIs of its modules.

Unfortunately, it is not written yet. So you have to look to documentation strings in code. Maybe you could create some documentation based on them?

THIS SECTION HAS TO BE COMPLETED. Volunteers?

Index

Table of Contents