Python-блог

воскресенье, 10 марта 2013 г.

MoinMoin2. Документация. Введение в настройку MoinMoin

Типы конфигурационных файлов

Для изменения того, как moinmoin себя ведёт и выглядит, Вы можете настроить его, редактируя конфигурационные файлы:

- Конфигурация движка Wiki
  - файл чаще всего называется wikiconfig.py, но может иметь и другое имя
  - в этом файле находятся классы конфига движка вики
  - он написан на Python
- Конфигурация фреймворка
  - расположена в том же файле, что и конфигурация движка вики
  - содержит некоторые настройки в ВЕРХНЕМ РЕГИСТРЕ в конце файла. Это конфигурация фреймворка (Flask и его расширений)
  - он написан на Python
- Конфигурация логирования
  - опциональна; если Вы не хотите настраивать логирование - будут использованы встроенные настройки
  - это отдельный файл, назвается чаще всего logging.conf
  - имеет .ini подобный формат файла

Меняйте понемногу и храните резервные копии

Начните с простого конфига, поставляемого с moin и меняйте его понемногу и каждый раз проверяйте работоспособность.

Если Вы не знакомы с форматом файла конфигурации, храните копию последнего работающего конфига на случай, если Вы вдруг сделаете трудно отлавливаемую ошибку.

Редактирование файлов Python

Когда Вы редактируете файлы Python, будьте осторожны с отступами, используйте лишь отступы, кратные 4 пробелам и не используйте табуляцию.

Кроме того, будьте внимательны к синтаксису вообще; это должен быть корректный код на Python, иначе Вы получите ошибку при попытке загрузки конфигурации. В таком случае прочитайте внимательно сообщение об ошибке, в нём скорее всего будет содержаться номер строки и описание проблемы. Если у Вас не получается решить проблему - восстановите из резервной копии последний рабочий конфиг.

Зачем использовать Python для конфигурации?

Вы, конечно, можете удивиться, почему мы используем код Python для конфигурации. Одна из причин в том, что это очень мощный язык. Moinmoin сам по себе разработан на Python и использование чего-либо другого потребовало бы больше времени для введения новой функциональности.

wikiconfig.py

wikiconf.py выглядит таким образом:

```
# -*- coding: utf-8 -*-
from MoinMoin.config.default import DefaultConfig

class Config(DefaultConfig):
    # некоторый комментарий
    sometext = u'your value'
    somelist = [1, 2, 3]

MOINCFG = Config # Flask любит только верхний регистр
SOMETHING_FLASKY = 'foobar'
```

Давайте разберём это строка за строкой:

0. Мы объявляем кодировку конфигурационного файла; убедитесь, что ваш редактор использует ту же самую кодировку, особенно, если Вы используете не ASCII символы
1. импортируем класс DefaultConfig из кода moin; в нём содержатся значения по умолчанию для всех настроек, что экономит ваш труд, так как Вы должны указать тут только те параметры, которые Вы хотите переопределить
2. пустая строка для лучшего восприятия

Ярлыки

Постоянные читатели

3. определяем новый класс Config, дочерний классу DefaultConfig. Это и есть настройки движка вики и он переопределяет значения из DefaultConfig.
4. знак # указывает на комментарий в вашем конфиге. Эта строка, как и последующие строки в пределах Config имет отступ в 4 пробела, так как Python определяет блоки по отступам
5. определяем атрибут Config'а с именем sometext и значением u'your value', где "u" указывает на то, что это строка юникода
6. определяем атрибут Config'а с именем somelist и значением [1, 2, 3], это список с эементами 1, 2 и 3
7. пустая строка для лучшего восприятия
8. специальная строка "MOINCFG = Config" должна иметь именно такой вид по техническим причинам
9. код в верхнем регистре находится в конце файла, вне класса Config и является настройкой фреймворка; обычно это что-то для Flask'а или его расширений

Реальный пример wikiconfig.py можно найти в каталоге docs/examples/config/

Настройка движка вики

Настройка интерфейса пользователя

Использование собственного шаблона snippets.html

Пользовательский интерфейс или html элементы, которые чаще всего требуют настройки, определены как макрос в файле шаблона snippets.html.
Если Вы хотите настроить какие-то части, Вы должны сделать копию встроеннго файла MoinMoin/templates/snippets.html и настроить moin, чтобы он использовал вашу копию вместо встроенного файла.
Это делается передачей листа каталога с шаблонами, где moin будет искать шаблоны в первую очередь:

```
template_dirs = ['path/to/my/templates',]
```

Для того, чтобы настроить что-то, обычно Вы должны добавить ваш код между {% macro ... %} и {% endmacro %}. Ниже об этом сказано более подробно.

Лого

Для замены логотипа MoinMoin на ваш логотип используйте следующий код:

```
{%  macro logo() -%}
<img       src=http://wiki.example.org/logos/my_logo.png"       id="moin-img-logo"
alt=Example logo">
{%- endmacro %}
```

Рекомендуется так сделать, чтобы ваши пользователи могли легко распознать вики, на которой они сейчас находятся.

Статические файлы (в том числе и логотипы) хранятся по адресу MoinMoin/static (логотипы хранятся в MoinMoin/static/logos)

Кроме того, совсем не обязательно, чтобы в качестве логотипа у Вас было изображение - это может быть и просто текст.
Будьте уверены, что размер вашего изображения или текста соответствует теме, которую используют ваши пользователи вики.

Отображение информации о лицензии

Если Вам нужно отобразить что-то вроде лицензионной информации вашего контента, используйте макрос:

```
{# Лицензионная информация в нижнем колонтитуле #}
{% macro license_info() -%}
Всё содержимое вики находится под лицензией WTFPL.
{%- endmacro %}
```

Добавление кусков HTML

В некоторых местах Вы можете добавить свои куски html в заголовок или тело темы:

```
{# Дополнительные HTML теги в <head> #}
{% macro head() -%}
{%- endmacro %}

{# Дополнительные HTML теги перед #moin-header #}
{% macro before_header() -%}
{%- endmacro %}

{# Дополнительные HTML теги после #moin-header #}
{% macro after_header() -%}
{%- endmacro %}
{# Дополнительные HTML теги перед #moin-footer #}
{% macro before_footer() -%}
{%- endmacro %}

{# Дополнительные HTML теги после #moin-footer #}
{% macro after_footer() -%}
```

```
{%- endmacro %}
```

Авторы и логотипы

В конце вашей страницы вики обычно расположен какой-то текст и изображения, показывающие, что вики использует MoinMoin, Python, что MoinMoin лицензируется под GPL и т.д.
Если Вы запускаете публичный сайт, используя MoinMoin, мы будем благодарны, если Вы *сохраните* эти ссылки, особенно "MoinMoin powered".
Однако, если по какой-то причине Вы не можете этого сделать, можете спокойно изменить эти макросы для отображения чего-то другого:

```
{# Ссылки на изображения в нижнем колонтитуле #}
{% macro creditlogos(start='<ul id="moin-creditlogos"><li>'|safe,
end='</li></ul>'|safe, sep='</li><li>'|safe) %}
{{ start }}
{{ creditlogo('http://moinmo.in/', url_for('.static',
filename='logos/moinmoin_powered.png'), 'MoinMoin powered', 'This site
uses the MoinMoin Wiki software.') }}
{{ sep }}
{{ creditlogo('http://moinmo.in/Python', url_for('.static',
filename='logos/python_powered.png'), 'Python powered', 'MoinMoin is
written in Python.') }}
{{ end }}
{% endmacro %}

{# Текстовые ссылки в нижнем колонтитуле #}
{% macro credits(start='<p id="moin-credits">'|safe, end='</p>'|safe,
sep='<span>&bull;</span>'|safe) %}
{{ start }}
{{ credit('http://moinmo.in/', 'MoinMoin Powered', 'This site uses the
MoinMoin Wiki software.') }}
{{ sep }}
{{ credit('http://moinmo.in/Python', 'Python Powered', 'MoinMoin is
written in Python.') }}
{{ sep }}
{{ credit('http://moinmo.in/GPL', 'GPL licensed', 'MoinMoin is GPL
licensed.') }}
{{ sep }}
{{ credit('http://validator.w3.org/check?uri=referer', 'Valid HTML 5',
'Click here to validate this page.') }}
{{ end }}
{% endmacro %}
```

Добавляем скрипты

Вы можете добавлять скрипты:

```
{# Дополнительный Javascript #}
{% macro scripts() -%}
<script type="text/javascript" src="http://example.org/cool.js">
</script>
{% endmacro %}
```

Добавляем CSS

Для того, чтобы применить изменения стилей, добавьте собственные css и переопределите стили, которые Вам не нравятся в основной теме:

```
{# Дополнительны Stylesheets (после theme css, до user css #}
{% macro stylesheets() -%}
    <link media="screen"
href="http://wiki.example.org/static/company.css" title="Company CSS"
rel="stylesheet" />
    <link media="screen" href="http://wiki.example.org/static/red.css"
title="Red Style" rel="alternate stylesheet" />
    <link media="screen"
href="http://wiki.example.org/static/green.css" title="Green Style"
rel="alternate stylesheet" />
{%- endmacro %}
```

Вы можете либо добавить обычные css, либо добавить выбор альтернативных css. Смотри:

- CSS media types
- Альтернативные CSS

Хороший способ протестировать стили - сперва использовать их как пользовательские CSS перед использованием их для широкой публики.
Обратите внимание, стили будут включены вне зависимости от того, какую тему выбрал пользователь, так что либо примените стили ко всем доступным темам, либо заставьте всех пользователей использовать одну и ту же тему, чтобы ваш CSS отображался корректно.

Отображение аватаров пользователей

При желании moin может отображать аватары пользователей при помощи gravatar.com. Для того, чтобы это активировать используйте:

```
user_use_gravatar = True
```

Обратите внимание, что использование gravatar'а сопряжено с некоторыми проблемами личных данных:

- для того, чтобы связать аватар с вашим e-mail на gravatar.com, Вы должны предоставить им свой e-mail адрес, который должен быть такой же, как указан в профиле пользователя вики
- когда вики отображает аватар, URL будет использовать как ссылка на провайдер сервиса аватаров, так что они могут приблиительно знать, какие пользователи какой вики пользуются

XStatic пакеты

XStatic - стандарт пакетов для упаковки внешних статических файлов в качестве пакетов Python, чаще всего от сторонних производителей. Таким образом их можно будет легко использовать на разных ОС, не важно, имеют ли они систему управлени пакетами, или нет.
Во многих случаях эти внешние статические файлы обслуживаются кем-то другим (как jQuery библиотекой или другой js библиотекой) и мы определённо не хотим сливать их с нашим проектом.
Для MoinMoin нам требуются следующие XStatic пакеты в setup.py:

- jquery для jquery функциональности, загружаемой в файл шаблона base.html
- jquery_file_upload, загружаемой в файл шаблона представления index. Она позволяет загружать несколько файлов одновременно.
- ckeditor используется в файле шаблона modify_text_html. WYSIWYG редактор, похожий на обычный текстовый процессор
- svgweb используется в base.html для обеспечения SVG поддержки в большинстве браузеров
- svgedit_moin загружается в modify_svg-edit. Это быстрый, основанный на javascript SVG редактор
- twikidraw_moin - Java аплет, загружаемый из файла шаблона или modify_twikidraw. Это простой редактор изображений
- anywikidraw - Java аплет, загружаемый из файла шаблона или modify_anywikidraw. Он может быть использован для редактирования рисунков и диаграм на элементах.
- jquery_multi_download используется в шаблонах или представлении index для множественных параллельных загрузок.

Эти пакеты импортируются в wikiconfig:

```
from xstatic.main import XStatic
mod_names = ['jquery', 'jquery_file_upload',
             'ckeditor', 'svgweb', 'svgedit_moin',
             'twikidraw_moin', 'anywikidraw',
             'jquery_multi_download', ]
pkg = __import__('xstatic.pkg', fromlist=mod_names)
for mod_name in mod_names:
    mod = getattr(pkg, mod_name)
    xs = XStatic(mod, root_url='/static', provider='local',
protocol='http')
    serve_files.update([(xs.name, xs.base_dir)])
```

В файле шаблона Вы можете получить доступ к файлам пакета по имени их модуля:

```
url_for('serve.files', name='имя модуля', filename='имя файла для загрузки')
```

Добавление XStatic пакетов

Следующий пример показывает как Вы можете активировать дополнительный пакет XStatic-MathJax, который используется для mathml или формул latex в содержимом элемента.
Для этого:

- сперва выполните
  `pip install xstatic-mathjax`
- добавьте mathjax в mod_names в файл wikiconfig
- добавьте нужный фрагмент в base.html

```
<script type="text/x-mathjax-config">
MathJax.Hub.Config({
    extensions: ["tex2jax.js"],
    jax: ["input/TeX","output/HTML-CSS"],
    tex2jax: {inlineMath: [["$","$"],["\\(","\\)"]]}
});
</script>
<script src="{{ url_for('serve.files', name='mathjax',
filename='MathJax.js') }}"></script>
```

Пользовательские темы

В случае, если Вы хотите сделать большие изменения в том, как MoinMoin отображает свои страницы, Вы можете также просто написать свою новую тему.
Предупреждение: разработка своей собственный темы подразумевает что Вам придётся ещё и обслуживать её и обновлять её, что обычно подразумевает затраты времени.
Надо добавить подробностей про пользовательские темы

Аутентификация

MoinMoin использует настраиваемый список аутентификаторов auth, так что администратор может настроить кому он хочет дать возможность аутентифицироваться. Moin обрабатывает этот список слева направо.
Каждый аутентификатор является экземпляром определённого класса, настройка которого обычно происходит при помощи передаваемых именованных аргументов. Большинство из них имеют разумные настройки по умолчанию.

MoinAuth

Это аутентификатор используемый moin по умолчанию, если Вы не хотите ничего больше настраивать. Пользователь входит заполняя форму входа, вводя имя пользователя и пароль, moin сравнивает хеш пароля с тем, который хранится в профиле пользователя, и если всё совпадает - пользователь аутентифицирован:

```
from MoinMoin.auth import MoinAuth
auth = [MoinAuth()] # это значение по умолчанию!
```

HTTPAuthMoin

При помощи HTTPAuthMoin moin обеспечивает базовую аутентификацию без помощи веб-сервера:

```
from MoinMoin.auth.http import HTTPAuthMoin
auth = [HTTPAuthMoin(autocreate=True)]
```

При похожей настройке moin будет требовать аутентификации при помощи http заголовков. После этого браузер обычно показвает пользователю диалог входа, где у пользователя запрашивается имя пользователя и пароль. Оба передаются moin и сравниваются с хешем пароля, хранимым в профиле пользователя.
Примечание: при использовании HTTPAuthMoin браузер будет показывать диалог входа, так что пользователю придётся войти в вики перед её использованием.

GivenAuth

При использовании GivenAuth moin полагается на вебсервер, который производит аутентификацию пользователя а результат возвращает moin, обычно при помощи переменной окружения REMOTE_USER:

```
from MoinMoin.auth import GivenAuth
auth = [GivenAuth(autocreate=True, coding='utf-8')]
```

Использование этого метода имеет свои за и против:

- Вы можете использовать только те расширения аутентификации, которые доступны для вашего веб-сервера
- единственная информация, получаемая moin через REMOTE_USER - имя аутентифицированного пользователя, ничего больше. Так, например, для LDAP/AD Вы не получите дополнитульную информацию, хранящуюся в LDAP
- Вы должны будете сами сохранить всё, что нужно, вручую, например, адрес электронной почты и т.п.

Пожалуйста, обратите внимание, что Вы должны передать правильную кодовую страницу, чтобы moin мог декодировать имя пользователя в юникод при необходимости. Для переменной окружения, например, REMOTE_USER, кодировка может зависеть от вашей ОС 5
Если Вы не знаете правильную кодировку, попробуйте 'utf-8', 'iso-8859-1', и так далее
Сделать: добавить стандартные кодировки для некоторых платформ, например, windows)
Чтобы это опробовать, измените конфигурацию, перезапустите moin и затем воспользуйтесь не ASCII именем пользователя (умлауты и т.д.). Если moin не рухнет (с ошибкой Unicode Error в логе) - значит Вы нашли правильную кодировку.
Для пользователей, настраивающих GivenAuth на Apache, пример файла настройки виртуального хоста можно посмотреть в docs/examples/deployment/moin-http-basic-auth.conf

OpenID

Благодаря OpenID moin может использовать аутентификацию, проводимую некоторыми OpenID провайдерами (такими как Google, Yahoo, Microsoft и другими):

```
from MoinMoin.auth.openidrp import OpenIDAuth
auth = [OpenIDAuth()]
```

По умолчанию OpenID аутентификация принимает всех OpenID провайдеров. Если хотите, Вы можете настроить список провайдеров, которым хотите позволить проводить аутентификцию (т.е. тех из них, которым Вы доверяете), передав их URL'ы через именованый аргумент trusted_providers. Если оставить его пустым, moin будет принимать авторизацию от всех провайдеров:

```
# Разрешаем только профили google:
auth = [OpenIDAuth(trusted_providers=
['https://www.google.com/accounts/o8/ud?source=profiles'])]
```

Для того, чтобы пользователь мог воспользоваться OpenID, его OpenID должен быть сохранён в его профиле.

Примечание: Если Вы собираетесь использовать эту аутентификацию, то Вам потребуются пакеты python-openid и python-ldap. Оба они у меня под FreeBSD 9.1 через pip не установились. Но их можно установить из портов: py-openid и py-ldap2. Единственная проблема в том, что создаваемое виртуальное окружение по умолчанию имеет флаг --no-site-packages, из-за чего виртуальное окружение не буедт использовать пакеты из стандартной установки. Для того, чтобы это исправить надо удалить или переименвать файл env/lib/python2.7/no-global-site-packages.txt.

LDAPAuth

При помощи LDAPAuth Вы можете проводить аутентификацию при помощи LDAP или Active Directory

LDAPAuth с одним LDAP сервером

Этот пример показывает как использовать LDAPAuth с одним LDAP/AD сервером:

```
from MoinMoin.auth.ldap_login import LDAPAuth
ldap_common_arguments = dict(
    # значения, приведённые ниже - значения по умолчанию
    # Вы можете удалить их, если они Вам не нравятся
    # примеры, приведённые в комментариях стандартны для
    # Active Directory или OpenLDAP
    bind_dn='', # Мы можем использовать фиксированный пароль для
        # подключения к LDAP. Будьте осторожны если Вам нужно
        # использовать % в этих настройках, так как они
        # используется как форматируемые строки. Вместо
        # этого используйте %%.
        #bind_dn = 'binduser@example.org' # (AD)
        #bind_dn = 'cn=admin,dc=example,dc=org' # (OpenLDAP)
        #bind_pw = 'secret'
        # или мы можем использовать имя пользователя и
        # пароль, полученные от пользователя:
        #bind_dn = '%(username)s@example.org'
        # DN, который мы используем для первого подключения (AD)
        #bind_pw = '%(password)s'
        # пароль, который мы используем для первог подключения
        # или мы можем подключиться анонимно
        # (если это поддерживается вашим каталогом)
        # В любом случае, bind_dn и bind_pw должны быть определены
    bind_pw='',
    base_dn='', # base DN, который мы используем для поиска
    #base_dn = 'ou=SOMEUNIT,dc=example,dc=org'
    scope=2, # диапазон поиска (2 == ldap.SCOPE_SUBTREE)
    referrals=0, # LDAP REFERRALS (0 для AD)
    search_filter='(uid=%(username)s)',
    # ldap фильтр для поиска:
    #search_filter = '(sAMAccountName=%(username)s)' # (AD)
    #search_filter = '(uid=%(username)s)' # (OpenLDAP)
    # можно использовать и более сложные фильтры:
    # "(&(cn=%
(username)s(memberOf=CN=WikiUsers,OU=Groups,DC=example,DC=org))"
    # некоторые имена атрибутов, которые мы будем использовать
    # для извлечения информации из LDAP (если не None,
    # если None, то атрибуты не будут извлечены из LDAP):
    # зачастую 'givenName' - ldap атрибут имени
    givenname_attribute=None,
    # 'sn' - ldap атрибут фамилии
    surname_attribute=None,
    # 'displayName' - ldap атрибут псевдонима (aliasname)
    aliasname_attribute=None,
    # 'mail' - ldap атрибут email
    email_attribute=None,
    # callback функция вызывается для обработки email адреса
    email_callback=None,
    # кодировка, используемая для запросов и ответов ldap
    coding='utf-8',
    timeout=10, # как долго ждём ответ от сервера [сек]
    # использование Transport Layer Security:
    # 0 = Не используем, 1 = Пробуем, 2 = Обязательно
    start_tls=0,
    tls_cacertdir=None,
    tls_cacertfile=None,
    tls_certfile=None,
    tls_keyfile=None,
    # 0 == ldap.OPT_X_TLS_NEVER
    # (нужно для самоподписанных сертификатов)
    tls_require_cert=0,
```

```
    # установите в True для только одного подключения -
    # полезно если настроена для подключения под пользователем
    # при первой попытке
    bind_once=False,
    # задайте в True для автоматического создания / обновления
    # профилей пользователей
    autocreate=True,
    # отправлять сообщение "invalid username or password"
    # ("неверное имя пользователя или пароль") при входе в
    # систему или нет
    report_invalid_credentials=True,
    )

ldap_authenticator1 = LDAPAuth(
    # URI сервера ldap / active directory
    server_uri='ldap://localhost',
    # используйте ldaps://server:636 url для ldaps,
    # ldap://server для ldap без tls (и задайте set start_tls=0),
    # ldap://server для ldap с tls (и задайте start_tls=1 или 2).
    # уникальное имя для ldap сервера, например
    # 'ldap_pdc' и 'ldap_bdc' (или 'ldap1' и 'ldap2')
    name='ldap1',
    **ldap_common_arguments # разложение общих аргументов
    )

# в этом списке Вы можете указать несколько ldap аутентификаторов
# или другие аутентификаторы
auth = [ldap_authenticator1, ]

# используйте настройки пользователей (по желанию, см
# MoinMoin/config/multiconfig для значений по умолчанию)
# Возможно Вы захотите использовать user_checkbox_remove,
# user_checkbox_defaults, user_form_defaults,
# user_form_disable, user_form_remove.
```

LDAPAuth с двумя LDAP серверами

Этот пример показывает как использовать LDAPAuth с двумя LDAP/AD серверами, например, в случае когда у Вас есть основной и резервный сервера:

```
# ... всё как для одного сервера (до строки "auth =") ...
ldap_authenticator2 = LDAPAuth(
    # URI сервера ldap / active для второго сервера
    server_uri='ldap://otherldap',
    name='ldap2',
    **ldap_common_arguments
    )
auth = [ldap_authenticator1, ldap_authenticator2, ]
```

AuthLog

AuthLog не является реальным аутертификатором в том смысле, что люди не могут через него залогиниться или выйти. Он лишь логирует информацию для отладки аутентификации:

```
from MoinMoin.auth import MoinAuth
from MoinMoin.auth.log import AuthLog
auth = [MoinAuth(), AuthLog(),]
```

Вот пример итогового лога:

```
2011-02-05 16:35:00,229 INFO MoinMoin.auth.log:22 login: user_obj=
<MoinMoin.user.User at 0x90a0f0c name:u'ThomasWaldmann' valid:1> kw=
{'username': u'ThomasWaldmann', 'openid': None, 'attended': True,
'multistage': None, 'login_password': u'secret', 'login_username':
u'ThomasWaldmann', 'password': u'secret', 'login_submit': u''}
2011-02-05 16:35:04,716 INFO MoinMoin.auth.log:22 session: user_obj=
<MoinMoin.user.User at 0x90a0f6c name:u'ThomasWaldmann' valid:1> kw={}
2011-02-05 16:35:06,294 INFO MoinMoin.auth.log:22 logout: user_obj=
<MoinMoin.user.User at 0x92b5d4c name:u'ThomasWaldmann' valid:False>
kw={}
2011-02-05 16:35:06,328 INFO MoinMoin.auth.log:22 session:
user_obj=None kw={}
```

Примечание: тут содержится важная информация, такая как имена пользователей и пароли. Убедитесь, что Вы используете это только для тестирования / отладки и удаляете после этого логи.

SMBMount

SMBMount не является реальным аутертификатором в том смысле, что люди не могут через него залогиниться или выйти. Вместо этого он получает имя пользователя и пароль и использует его для подключения SMB шар под этим пользователем.
SMBMount полезен только для очень специфических приложений, например, в комбинации с хранилищем на файловом сервере:

```
from MoinMoin.auth.smb_mount import SMBMount
smbmounter = SMBMount(
    # Вы можете удалить значения по умолчанию, если они Вам
    # не нравятся. Смотрите более детально man mount.cifs
    # (не умолчание) mount.cifs //server/share
    server='smb.example.org',
    share='FILESHARE', # (не умолчание) mount.cifs //server/share
    # (не умолчание) функция от имени пользователя
    # для определения точки монтирования
    mountpoint_fn=lambda username: u'/mnt/wiki/%s' % username,
    # (не умолчание) имя пользователя для получения
    # uid, который используется для mount.cifs -o uid=...
    dir_user='www-data',
    domain='DOMAIN', # (не умолчание) mount.cifs -o domain=...
    dir_mode='0700', # (умолчание) mount.cifs -o dir_mode=...
    file_mode='0600', # (умолчание) mount.cifs -o file_mode=...
    iocharset='utf-8', # (умолчание) mount.cifs -o iocharset=...
    #(попробуйте 'iso8859-1'
    # если значение по умолчанию не работает)
    coding='utf-8', # (умолчание) кодировка, которая
    # используется для имени пользователя/пароля/командной
    # строки (попробуйте 'iso8859-1'
    # если значение по умолчанию не работает)
    log='/dev/null', # (умолчание) файл лога для mount.cifs
    )
auth = [....., smbmounter] # Вам нужен реальный
    # аутентификатор в этом списке перед smbmounter
smb_display_prefix = u"S:" # куда будет монтироваться
    # //server/share для пользователей windows
    # (только для целей отображения)
```

Сделать: проверить, работает ли SMBMount как это описано

Безопасность передачи

Личные данные

Некоторые из описаных методов аутентификации передают аутентификационные данные, такие как имя пользователя и пароль, в незашифрованной форме:

- MoinAuth: кода данные формы входа передаются moin, они содержат имя пользователя и пароль в незашифрованном виде
- HTTPAuthMoin: ваш браузер передаёт имя пользователя и пароль в закодированной (но НЕ зашифрованной) форме при КАЖДОМ запросе
- GivenAuth: проверьте потенциальные проблемы безопасности метода аутентификации, используемого вашим веб-сервером.
- OpenID: проверьте сами

Содержимое

http передаёт всё в открытом виде без какого либо шифрования

Шифрование

Передача незашифрованных данных может создать кучу проблем в разных случаях. Мы рекомендуем Вам использовать зашифрованные подключения, такие как https, VPN или ssh тунель.
Для публичных вики с очень низкими потребностями безопасности может быть не нужно шифровать передаваемые данные, но всё равно ещё остаётся потребность шифрования личных данных.
При использовании незашифрованных подключений пользователи вики должны использовать уникальный пароль, а не использовать пароли, которые он использует для других целей.

Безопасность пароля

Сила пароля

Как Вы можете знать, многие пользователи не умеют выбирать хорошие пароли, а многие ещё и используют очень лёгкие для взлома пароли. Чтобы помочь пользователям выбрать хороший пароль moin имеет встроенную функцию проверки пароля, активированная по умолчанию, которая проверяет качество пароля, чтобы пользователи не могли выбрать лёгкий для взлома пароль.
Если у Вашего сайта низкие требования к безопасности, то эту функцию можно отключить:

```
password_checker = None # не производить проверку пароля
```

Обратите внимание, что встроенная функция делает лишь базовые проверки, то есть не использует словарь плохих паролей.

Хранение паролей

Moin никогда не хранит пароли пользователей вики в открытом виде, вместо этого использует криптографические хеши паролей, получаемы при помощи библиотекеки

passlib. Документация passlib рекомендует 3 схемы хеширования, обеспечивающие хороший уровень безопасности: sha512_crypt, pbkdf2_sha512 и bcrypt (bcrypt требует ещё дополнительные бинарные пакеты, смотрите библиотеку passlib).
По умолчанию мы используем sha512_crypt хеши со значениями по умолчанию (это тот же алгоритм, который используется в moin>=1.9.7)
В случае если вход в систему занимает слишком много времени или Вы по другой причине хотите заменить генератор хешей, обращайтесь к документации passlib. Moin позволяет Вам настроить параметр CryptContext библиотеки passlib в конфиге вики (вот пример по умолчанию)5

```
passlib_crypt_context = dict(
    schemes=["sha512_crypt",],
)
```

Авторизация

Moin использует списки контроля доступа (ACL) для определения того, кто что может делать.
Обратите внимание, что вики обычно часто используют так называемую "мягкую безопасность", что означает, что они с одной стороны дают полную свободу пользователям, но в то же время предоставляют возможность отменить любой причинённый ущерб.
"Жёсткая безопасность" подразумевает, что какие-то элементы можно заблокировать, предотвратив причинение ущерба.
Конфигурация moin по умолчанию старается сочетать как "мягкую", так и "жёсткую" безопасность. Однако Вам могут потребоваться другие настройки, в зависимости от ситуации, с которой столкнулись администратор, хозяин или пользователи вики.
Так что помните:

- если ваша вики довольно открыта, Вы можете разрешить пользователям легко вносить в неё информацию, например, исправлять опечатки. Однако, некоторые пользователи могут использовать вашу вики для размещения спама, хотя Вы и сможете отменить его добавление позже.
- если ваша вики скорее закрыта, то есть Вы требуетет от пользователей сперва создать себе аккаунт, для того, чтобы он мог вносить в неё изменения, то Вы будете получать меньший вклад от пользователей, но и зато меньше спама.

ACL для функций

Этот ACL управляет доступом к некоторым специфическим функциям / представлениям moin:

```
# значение по умолчанию функции acl_rights_functions приведено
# тут для справки, Вам, скорее всего, не потребуется его менять:
#acl_rights_functions = ['superuser', 'notextcha', ]
acl_functions=u'+YourName:superuser TrustedEditorGroup:notextcha'
```

Поддерживаемые возможности (права):

- superuser - пользователь с функциями администратора. Давайте такие права только доверенным пользователям
- notextcha - если у Вас активирован TextChas, пользователи с notextcha не будут получать вопрос. Давайте такие права только доверенным пользователям, которые регулярно вносят правки в вашу вики

Примечание: то есть, для того, чтобы дать кому-то какие-то права (суперпользоателя или notextcha), Вы должнны вручную в файл конфигурации добавить строку acl_functions, где указать нужные имена пользователей.

ACL контента

Эти ACL управляют доступом к контенту, который хранится в вики; они настраиваются для каждого хранилища (см соответствующую документацию) и, по желанию, для метаданных:

```
# значение acl_rights_contents тут для справки
# Вам, скорее всего, не потребуется его менять:
#acl_rights_contents = ['read', 'write', 'create', 'destroy', 'admin',
]
... backend configuration ...
... before=u'YourName:read,write,create,destroy,admin',
... default=u'All:read,write,create',
... after=u'',
... hierarchic=False,
```

Обычно у Вас будут before, on item, default и after, которые обрабатываются именно в таком порядке. ACL default используется только в случае, если в метаданных не определено другого ACL для этого элемента (диаграмма по ссылке).
Как использовать brfore/default/after:

- before обычно используется для принуждения чего-либо, например, если Вы хотите какому-то администратору все права
- default используется в случае, если не задан специальный ACL в метаданных элемента
- after используется реже, обычно в случае когда надо "не забыть что-то, если только что-то другое не определено"

При настройке ACL контента, Вы можете выбрать между "плоской", станадрной обработке ACL и иерархической обработкой ACL. Иерархическая обработка подразумевает, что подэлементы наследуют ACL от родительских элементов, если ACL не определён для них самих.
Обратите внимание, что хотя иерархическая обработка обычно более удобна в некоторых случаях, тем не менее, она делает систему гораздо сложнее. Вы должны быть очень внимательны с изменениями ACL, которые происходят в результате изменения иерархии, когда Вы создаёте, переименовываете или удаляете элементы.
Поддерживаемые возможности (права):

- read - чтение информации
- write - запись (редактирование) информации
- create - создание новых элементов
- destroy - полное уничножение элемента, должно быть разрешено только пользователям, которым Вы на **100% доверяете**
- admin - изменение (создание, удаление) ACL для элементов, должно быть разрешено только пользователям, которым Вы на **100% доверяете**

ACL - специальные группы
В дополнение к группам, поставляемым group backend, есть некоторые имена групп, доступные в ACL:

- All - виртуальная группа, содержащая всех пользователей
- Known - виртуальная группа, содержащая всех залогинившихся пользователей
- Trusted - виртуальная группа, содержащая всех залогинившихся пользователей, которые произвели вход "доверенным" методом аунтентификации

ACL - базовый синтаксис
ACL - это строка юникода с одной или более записью контроля доступа, с разделителями пробелами.
Каждая запись - набор, разделённый двоеточиями, двух значений:

- слева находится разделённый запятыми список пользователей / групп
- справа находится разделённый запятыми список прав для этих пользователей / групп

ACL обрабатываются слева направа, где учитывается первое совпадение слева. Например:

```
u"SuperMan:read,write,create,destroy,admin All:read,write"
```

Если SuperMan на данный момент вошёл в систему и moin обрабатывает даный ACL, то он обнаружит совпадение уже с первым элементом. Если moin хочет знать, может ли этот пользователь уничтожить элемент, то ответ будет "да", так как destroy - одно из прав, которые перечислены в правой части этого элемента.
Если JoeDoe на данный момент вошёл в систему и moin обрабатывает этот ACL, то с первым элементом совпадения не произойдёт, так что moin двинется дальше вправо и посмотрит на вторую запись. Тут у нас указано имя группы All (а JoeDoe очевидно является членом этой группы), так что этот элемент нам подходит.
Если moin хочет знать, может ли этот польователь уничтожать элементы, ответом будет "нет", так как это право не указано среди доступных прав для этой группы.
Если moin хочет знать, имеет ли этот пользователь право на запись - ответ будет "да".
Примечания:

- Как следствие обработки слева направо и поиска первого совпадения, Вы должны расположить ваши элементы ACL начиная с более детальных, заканчивая более общими. Обычно этот будет такой порядок:
  - имя пользователя
  - специальные группы
  - более общие группы
  - Trusted
  - Known
  - All
- Не размещайте пробелы внутри элемента ACL, если только это не является частью имени пользователя или группы
- Право, не заданное явно - не выдано
- Для большинства ACL есть встроенные значения по умолчанию, которые дают ограниченные права

ACL - префиксы элементов
Для того, чтобы сделать систему более гибкой, есть два способа изменить элементы ACL: префиксы + и -.
Если Вы используете один из двух префиксов, MoinMoin будет искать и имя пользователя и пароль, и должны будут совпасть и то и другое, или moin перейдёт к следующему элементу.
+ означает, что MoinMoin должен выдать разрешение(я), заданные справа
- означает, что MoinMoin не выдаёт разрешение(я), заданные справа
Например:

```
u"+SuperMan:create,destroy,admin -Idiot:write All:read,write"
```

Если SuperMan на данный момент вошёл в систему и moin хочет узнать, имеет ли он право на уничтожение элемента, то он найдёт совпадение с перым элементом, так как

совпадут имя пользователя **и** искомое разрешение. Так как префиксом является +, то ответом будет "да". Если moin хочет знать, имеет ли он право на запись, то первый элемент нам не подойдёт, так как нет совпадения со стороны прав, и moin перейдёт к следующему элементу. Он тоже не подойдёт и moin перейдёт к третьему элементу. Конечно, SuperMan является членом группы All, так что тут совпадение будет. Так как право write указано среди выданных прав, ответом будет "да".
Если Idiot на данный момент вошёл в систему и moin хочет знать, имеет ли он право на запись, то совпадение произойдёт со вторым элементом. Так как префиксом является -, то ответом будет "нет" и третий элемент даже не будет рассматриваться.
Примечания:

- Обычно Вы будете использовать эти модификаторы в случае, если большая часть прав для данного пользователя должна быть определена позже, но некая группа или пользователь должны иметь права, несколько отличающиеся от этих прав

ACL - записи по умолчанию
Есть специальная ACL запись, default, которая используется в качестве ACL по умолчанию.
Это полезно в случае если Вы хотите использовать по большей части ACL по умолчанию, но с некоторыми модификациями, однако Вы не хотите указывать эту ACL каждый раз и Вы так же хотите иметь возможность изменить ACL по умолчанию, не редактируя её во многих элементах.
Пример:

```
u"-NotThisGuy:write Default"
```

Это будет работать как обычно, за исключением того, что NotThisGuy никогда не получит прав на зпись.

Анти-спам

TextChas
TextCHA - текстовая альтернатива капче. MoinMoin использует её для предотвращения спама в вики и её эффективность вполне доказана.
Возможности:

- когда регистрируется новый пользователь или сохраняется элемент она может задать случайный вопрос для проверки антибот
- moin сравнивает полученный ответ с регулярным выражением
- вопросы и ответы можно настроить в конфиге вики
- поддерка нескольких языков: пользователь получает вопрос на своём языке или на английском по умолчанию, в зависимоитс от доступности вопросов и ответов на языке пользователя.

Настройка TextCha
Советы для настройки:

- имейте один числовой, один буквенный ответ
- задавайте вопрос, на который сможет ответить типичный пользователь вашей вики
- Не задавайте слишком сложные вопросы
- Не задавайте "вычисляемые" вопросы, типа "1+1" или "2+2"
- Не задавайте слишком очевидные вопросы
- Не делитесь вопросами с другими сайтами и не копируйте их оттуда (или спамеры смогут под это подстроиться)
- Вы должны как минимум задать textcha для английского языка или вашего языка пользователя, если это не английский, так как иначе возникнет ошибка, если MoinMoin не найдёт textcha на языке пользователя.

В вашем конфиге вики используйте что-то вроде этого:

```
textchas = {
    'en': { # ukegst английские примеры textchas
            # (не исользуйте их!)
            u"Enter the first 9 digits of Pi.": ur"3\.14159265",
            u"What is the opposite of 'day'?": ur"(night|nite)",
            # ...
          },
    'de': { # немецкие textchas
            u"Gib die ersten 9 Stellen von Pi ein.": ur"3\.14159265",
            u"Was ist das Gegenteil von 'Tag'?": ur"nacht",
            # ...
          },
    # Вы можете добавить и другие языки
}
```

Обратите внимание, что пользователи с notextcha ACL не будут получать эти вопросы.

Секреты

Moin использует секреты для шифрования или подписывания чего-то вроде:

- textchas
- билеты (tickets)

Секреты - длинные случайные строки, которые **не** используются для ваших паролей. **Не** используйте строки из примера, так как они не являются секретными, ведь они -

часть документации moin. Создайте свои собственные секреты:

```
secrets = {
    'security/textcha': 'kjenrfiefbeiaosx5ianxouanamYrnfeorf',
    'security/ticket': 'asdasdvarebtZertbaoihnownbrrergfqe3r',
}
```

Если Вы не настроите эти секреты, moin это обнаружит и будет использовать SECRET_KEY Flask'а вместо них.

Группы и словари

Moin может получить информацию групп и словарей из поддерживаемого хранилища, такого как конфигурация вики или элементов вики.
Группа - это список имён в юникоде. Она может использоваться для разных целей: какое-нибудь приложение может поределить группы пользователей для использования в ACL.
Словарь отображает юникодовые ключи на юникодовые значения. Он тоже может использоваться для разных приложений. На данный момент он не используется самим moin.

Конфигурация бэкэнда групп

Бэкэнд WikiGroups получает группы из элементов ваики и используется по умолчанию:

```
def groups(self, request):
    from MoinMoin.datastruct import WikiGroups
    return WikiGroups(request)
```

Бэкэнд ConfigGroups использует группы, определённые в конфигурационном файле:

```
def groups(self, request):
    from MoinMoin.datastruct import ConfigGroups
    # Группы определяются тут.
    groups = {u'EditorGroup': [
              u'AdminGroup', u'John', u'JoeDoe', u'Editor1'],
              u'AdminGroup': [u'Admin1', u'Admin2', u'John']
              }
return ConfigGroups(request, groups)
```

CompositeGroups может использовать, в большей части, любую комбинацию бэкэндов. Вот пример использования комбинаци WikiGroups и ConfigGroups:

```
def groups(self, request):
    from MoinMoin.datastruct import ConfigGroups, WikiGroups,
CompositeGroups
    groups = {u'EditorGroup': [
        u'AdminGroup', u'John', u'JoeDoe', u'Editor1'],
              u'AdminGroup': [
        u'Admin1', u'Admin2', u'John']}
# Тут используется конфигурация ConfigGroups и WikiGroups.
# Обратите внимание на важность порядка! Так как ConfigGroups
# упомянут первым, EditorGroup будет получен из него, не из
# WikiGroups.
return CompositeGroups(request,
                       ConfigGroups(request, groups),
                       WikiGroups(request))
```

Конфигурация бэкэнда Dict

WikiDicts бэкэнд получает словари из элементов вики и использует их по умолчанию:

```
def dicts(self, request):
    from MoinMoin.datastruct import WikiDicts
    return WikiDicts(request)
```

ConfigDicts бэкэнд использует словари, определённые в конфигурационном файле

```
def dicts(self, request):
    from MoinMoin.datastruct import ConfigDicts
    dicts = {u'OneDict': {u'first_key': u'first item',
                          u'second_key': u'second item'},
             u'NumbersDict': {u'1': 'One', u'2': 'Two'}
            }
 return ConfigDicts(request, dicts)
```

CompositeDicts бэкэнд может использовать комбинацию бэкэндов. Вот пример:

```
def dicts(self, request):
```

```
    from MoinMoin.datastruct import ConfigDicts, WikiDicts,
CompositeDicts
    dicts = {u'OneDict': {u'first_key': u'first item',
                          u'second_key': u'second item'},
             u'NumbersDict': {u'1': 'One', u'2': 'Two'}}
    return CompositeDicts(request,
                          ConfigDicts(request, dicts),
                          WikiDicts(request))
```

Хранилище

MoinMoin поддерживает бэкэнд хранилищ как разные способы хранения элементов вики. Настройка хранилищ сложна и многослойна и включает:

- router middleware, который отправляет часть пространства имён соответствующему бэкэнду
- middleware проверки ACL, который позволяет нам убедиться, что никто не получит доступ к тому, к чему не надо
- примесь индексирования (index mixin), которая индексирует некоторые данные автоматически при записи, так что эти элементы можно быстрее получить
- бэкэнд хранения, который хранит элементы вики

create_simple_mapping

Это вспомогательная функция, которая облегчает настройку хранения. Она помогает:

- Создать простую настройку, которая использует 3 бэкэнда хранения для следующих частей пространства имён:
  - содержание
  - профили пользователей
- настроить ACL, защищающая эти части пространства имён
- настроить router middleware, который направляет к этим частям пространства имён
- настраивает примесь индексирования

Вызвать её можно так:

```
from MoinMoin.storage import create_simple_mapping
namespace_mapping, acl_mapping = create_simple_mapping(
    uri=...,
    content_acl=dict(before=...,
                     default=...,
                     after=...,
                     hierarchic=..., ),
    user_profile_acl=dict(before=...,)
                          default=...,
                          after=..., ),
)
```

uri зависит от типа бэкэнда хранилища и хранит то, что Вы хотите использовать, см. ниже. Обычно это URL подобная строка такой формы:

```
stores:fs:/srv/mywiki/%(nsname)s/%(kind)s
```

stores - имя бэкэнда, за которым идёт двоеточие, за которым идёт спецификация хранилища. fs - имя хранилища, за которым идёт спецификация, актуальная для fs хранилища (файловой системы), то есть путь с подстановками.
%(nsname)s будет замещено content или userprofiles для соответствующего бэкэнда.
%(kind)s будет замещено meta или data.
В нашем случае созданное отображение будет выглядеть так:

```
+----------------+-----------------------------+
| Namespace part | Filesystem path for storage |
+----------------+-----------------------------+
| /              | /srv/mywiki/content/        |
+----------------+-----------------------------+
| /UserProfiles/ | /srv/mywiki/userprofiles/   |
+----------------+-----------------------------+
```

content_acl и user_profile_acl - это словари, определяющие ACL для этой части пространства имён (содержимое, профиль пользователя). См документацию про ACL.

protecting middleware

Возможности:

- защищает доступ к нижним слоям хранилища ACL'ем
- позволяет убедиться, что не возникнет проблем с безопасностью ACL, даже если есть проблемы с верхними уровнями
- Если Вы используете create_simple_mapping, Вы всего лишь передаёте ACL параметры; middleware будет автоматически настроен moin

routing middleware

Возможности:

- перенаправляет доступ к различным бэкэндам в зависимости от имени элемента

- в терминах POSIX это что-то вроде fstab/mount
- если Вы используете create_simple_mapping, router middleware будет автоматически настроен moin

indexing middleware
Возможности:

- обслуживает индекс для важных значений метаданных
- ускоряет поиск / выборку элементов
- упрощает возможность организации нижних слоёв
- indexing middleware будет настроен автоматически moin

stores backend
Это бэкэнд, который связывает вместе 2 хранилища в бэкэнд: один для метаданных, другой для данных
хранилище fs
Возможности:

- хранение в файловой системе
- хранение метаданных и данных в разных файлах/каталогах

Настройка:

```
from MoinMoin.storage import create_simple_mapping

data_dir = '/srv/mywiki/data'
namespace_mapping, acl_mapping = create_simple_mapping(
    uri='stores:fs:{0}/%(nsname)s/%(kind)s'.format(data_dir),
    content_acl=dict(
        before=u'WikiAdmin:read,write,create,destroy',
        default=u'All:read,write,create',
        after=u'', ),
    user_profile_acl=dict(
        before=u'WikiAdmin:read,write,create,destroy',
        default=u'',
        after=u'', ),
)
```

хранилище sqla
Возможности:

- хранение данных в (SQL) БД/таблице
- может либо использовать 1 БД на хранилище или 1 таблицу на хранилище, в этом случае Вы должны дать таблицам разные имена
- использует sqlalchemy (без ORM) для абстракции БД
- поддерживает множество типов БД, например:
  - sqlite (по умолчанию, встроена в Python)
  - postgresql
  - mysql
  - и другие, смотри документацию sqlalchemy

uri для create_simple_mapping выглядит как-то так:

```
stores:sqla:sqlite:////srv/mywiki/data/mywiki_%(nsname)s_%(kind).db
stores:sqla:sqlite:////srv/mywiki/data/mywiki_%(nsname)s.db::%(kind)s
stores:sqla:mysql://myuser:mypassword@localhost/mywiki_%(nsname)s::%(kind)s
stores:sqla:postgres://myuser:mypassword@localhost/mywiki_%(nsname)s::%(kind)s
```

uri после sqla имеет следующую форму: DBURI::TABLENAME
Пожалуйста, обратитесь к документации sqlalchemy на тему части DBURI.
Передайте пользователю myuser (с паролем mypassword) полный доступ к этим БД.
хранилище sqlite
Возможности:

- напрямую общается с sqlite, без использования sqlalchemy
- хранит данные в sqlite БД, то есть в одном файле
- может либо использовать 1 БД на хранилище или 1 таблицу на хранилище, в этом случае Вы должны дать таблицам разные имена
- может по желанию сжимать данные при помощи zlib. По умолчанию уровень сжатия 0, то есть данные не сжимаются.

uri для create_simple_mapping будет выглядеть следующим образом:

```
stores:sqlite:/srv/mywiki/data/mywiki_%(nsname)s_%(kind)s.db
stores:sqlite:/srv/mywiki/data/mywiki_%(nsname)s.db::%(kind)s
stores:sqlite:/srv/mywiki/data/mywiki_%(nsname)s.db::%(kind)s::1
```

uri часть после "sqlite:" имеет следующую форму: PATH::TABLENAME::COMPRESSION. В качестве разделителя используется ::, чтобы обеспечить поддержку windows, где : может использоваться в пути после имени диска.
хранилище kc
Возможности:

- использует Kyoto Cabinet файл для хранения данных
- очень быстрое
- однопроцессное и локальное

uri для create_simple_mapping выглядит так:

```
stores:kc:/srv/mywiki/data/%(nsname)s_%(kind)s.kch
```

Обратитесь к документации kyoto cabinet на тему того, что должно быть после kc:. Если Вы используете kc для встроенного сервера moin, Вы не можете использовать перезагрузчик (reloader). Отключите эту опцию в командной строке:
`moin moin -r`
хранилище kt
Возможности:

- использует Kyoto Tycoon сервер для хранения
- быстрый
- многопроцессорный, локальный или удалённый

Сделать: добавить пример конфигурации хранилища
хранилище в памяти
Возможности:

- держит всё в ОЗУ
- если ваша система или moin падает, все данные будут потеряны, так что определённо не стоит использовать это для продакшена
- лучше всего предназначено для тестирования
- только один процесс

Сделать: добавить пример конфигурации хранилища
бэкэнд файлового сервера
Возможности:

- предоставляет часть файловой системы как хранилище только для чтения элементов вики:
  - файлы отображаются как элементы вики
    - с одной ревизией
    - с таким количеством метаданных, сколько может быть сохранено в метаданных АС
  - каталоги будут отображаться как элементы индекса, содержащие ссылки на их содержимое
- может быть полезно вместе с псевдо-аутентификатором SMBMount

Настройка почты

отправка почты
Moin может по желанию отправлять e-mail. Возможные использования:

- отправка уведомлений об изменениях элементов
- позволяет восстановить забытый пароль

Вам нужно задать несколько настроек для того, чтобы можно было отправлять почту:

```
# адрес "from:"  [Unicode]
mail_from = u"wiki <wiki@example.org>"
# a) использование SMTP сервера, например, "mail.provider.com"
# (None чтобы отключить отправку почты)
mail_smarthost = "smtp.example.org"
# если Вы используете SMTP AUTH на вашем mail_smarthost:
#mail_username = "smtp_username"
#mail_password = "smtp_password"
# b) альтернативно использованию SMTP,
# Вы можете использовать инструмент командной строки sendmail:
#mail_sendmail = "/usr/sbin/sendmail -t -i"
```

Сделать: описать настройку более подробно
Верификация e-mail адреса
Во время создания аккаунта Moin может потребовать от нового пользователя подтвердить свой адрес электронной почты, нажав на ссылку в отправленном по этому адресу письме.
Убедитесь, что Moin может отправлять электронную почту (см предыдущий раздел) и добавьте следующую строку в ваш конфигурационный ффайл:
`user_email_verification = True`

Настройка фреймоворка

Вот, что Вы можете захотеть настроить для Flask и его расширений (более подробно это описано в их документации):

```
# для Flask
SECRET_KEY = 'вы должны изменить эту строку на что-то секретное'
# используйте True только для разработки, не для публичных сайтов
DEBUG = False
#TESTING = False
#SESSION_COOKIE_NAME = 'session'
#PERMANENT_SESSION_LIFETIME = timedelta(days=31)
#USE_X_SENDFILE = False
#LOGGER_NAME = 'MoinMoin'
# для Flask-Cache:
```

```
#CACHE_TYPE = 'filesystem'
#CACHE_DIR = '/path/to/flask-cache-dir'
```

Настройка логирования

По умолчанию логирование настроено на отправку вывода в stderr. Это будет хорошо работать для встроенного сервера (где вывод будет отображаться в консоли) или для Apache2 и т.п. (где лог будет выведен в error.log)
Логирование настраиваемое и гибкое благодаря использованию модуля logging стандартной библиотеки.
Формат файла настройки описан по ссылке.
Кроме того, примеры настройки можно найти в каталоге docs/examples/config/logging/.
Настройку логирования нужно произвести как можно раньше, обычно это делается ещё в скрипте-адапторе, то есть moin.wsgi:

```
from MoinMoin import log
log.load_config('wiki/config/logging/logfile')
```

Вы должны указать путь к файлу настройки, который отвечает вашим потребностям.
Пожалуйста обратите внимание, что файл конфигурации логирования должен быть отдельным файлом, так что не пытайтесь добавить это в файл настройки вики
Примечание: для изменения конфигурации логирования можно использовать переменную окужения MOINLOGGINGCONF, которая должна содержать путь к файлу с конфигурацией логирования. Для задания переменой используйте:
export MOINLOGGINGCONF=/path/to/config
для удаления переменной:
unset MOINLOGGINGCONF

Автор: Ishayahu Lastov на 18:12

Ярлыки: moinmoin2, moinmoin2 настройка, wiki

Комментариев нет:

Отправить комментарий

Чтобы оставить комментарий, нажмите кнопку ниже и войдите с аккаунтом Google.

ВОЙТИ С АККАУНТОМ GOOGLE

Следующее Главная страница Предыдущее

Подписаться на: Комментарии к сообщению (Atom)