django-ulogin Documentation

Выпуск 0.3.0

Mikhail Porokhovnichenko

авг. 13, 2020

Оглавление

https://github.com/marazmiki/django-ulogin

Глава 1

Описание

django-ulogin — подключаемое универсальное django-приложение для социальной аутентификации пользователей с помощью внешнего интернет-сервиса uLogin

Внимание: Создатели приложения никак не связаны с интернет-сервисом uLogin, поэтому все вопросы, касающиеся непосредственно работы сервиса, а не этого приложения, просьба отправлять на team@ulogin.ru.

Глава 2

Содержание

2.1 Системные требования

- Разработка и тестирование проводилось под управлением OS Ubuntu, Debian, Mac OS. Production-использование — Ubuntu, Debian, RHEL, FreeBSD. Приложение не использует каких-либо специфичных для той или иной операционной системы особенностей языка, поэтому, скорее всего, всё будет работать и на остальных системах, для которых существуют требуемые версии Python
- Интерпретатор Python `2.7+`, `3.4`, `3.5`, `3.6`. На других интерпретаторах работоспособность не проверялась.
- `Django 1.7+`, включая поддержку `Django 2.x` (доступно только для `Python 3.4+`)
- Библиотека python-requests

2.2 Установка

Установка текущей стабильной версии django-ulogin производится из PyPI с помощью утилиты `pip`:

```
$ pip install django-ulogin
```

Для установки dev-верии из репозитория нужно добавить к командной строке ключ `-e`:

```
$ pip install -e git+github.com/marazmiki/django-ulogin.git#egg=django-ulogin
```

Если нужно установить какую-то конкретную ревизию, просто укажите её в URL репозитория:

```
$ pip install -e git+github.com/marazmiki/django-ulogin.git@{rev}#egg=django-ulogin
```

Все внешние зависимости будут утановлены автоматически

2.3 Конфигурация проекта

Для подключения к проекту откройте `settings.py` и добавьте в кортеж `INSTALLED_APPS` приложение `django_ulogin`

```
INSTALLED_APPS += [
    'django_ulogin'
]
```

Внимание: Атрибут `INSTALLED_APPS` по умолчанию стал списком (`list`) начиная с Django версии 1.9. В более ранних версиях он является кортежем (`tuple`), поэтому, если Вы не изменяли тип, добавлять надо кортеж:

```
INSTALLED_APPS += (
    'django_ulogin'
)
```

Для работы `django-ulogin` необходим подключенный контекст-процессор `django.template.context_processors.request`:

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                # ...
                'django.template.context_processors.request',
            ],
        },
    },
]
```

Внимание: В старых версих Django до версии 1.9 включительно, не было поддержки нескольких систем рендеринга шаблонов, а требуемый контекст-процессор располагался в `django.core.context_processors.request`. Таким образом, подключение выглядит примерно следующим образом:

```
TEMPLATE_CONTEXT_PROCESSORS = (
    # ...
    'django.core.context_processors.request',
)
```

Добавьте схему URL-адресов к списку `urlpatterns` Вашего проекта (`urls.py`):

```
urlpatterns += [
    url(r'^ulogin/', include('django_ulogin.urls')),
]
```

Затем следует синхронизировать базу данных

```
$ ./manage.py migrate
```

2.4 Использование

Внимание: На текущий момент поддерживается работа с единственым шаблонным движком, встроенным в Django. Речь идёт именно о нём.

2.4.1 Быстрый старт

Для использования приложения достаточно в любом месте шаблона вставить подключение шаблонной библиотеки `ulogin_tags` и вызов тега `ulogin_widget`.

```
{% load ulogin_tags %}
{% ulogin_widget %}
```

На месте тега `ulogin_widget` при рендеринге появится код интеграции вашего сайта с uLogin.

Тег `{% ulogin_widget %}` принимает один необязательный аргумент — `scheme_name`, который указывает на имя используемой схемы настроек.

```
{% ulogin_widget "scheme_name" %}
```

Использование различных схем особенно удобно, если нужно на одной странице разместить несколько виджетов, обладающих различными настройками.

2.5 Для опытных пользователей

2.5.1 Тонкая настройка

По умолчанию `django_ulogin` требует от сервиса только одно обязательное поле - `email`. Вы можете указать для проекта список как необходимых полей (определив в `settings` список `ULOGIN_FIELDS`), так и опциональных (`ULOGIN_OPTIONAL`):

```
# Поля first_name и last_name обязательны
ULOGIN_FIELDS = ['first_name', 'last_name']

#  Необязательные поля: пол, URL аватара, дата рождения
ULOGIN_OPTIONAL = ['sex', 'photo', 'bdate']
```

Список всех полей, которые сообщает ULOGIN:

- `first_name`
- `last_name`
- `email`
- `nickname`
- `bdate` *(дата рождения, передаётся в формате dd.mm.yyyy)*
- `sex` *(пол: 1 означает женский, 2 - мужской)*

- `photo` *(аватар, размер 100x100 пикселей)*
- `photo_big`
- `city`
- `country`
- `phone`

Внешний вид виджета определяется параметром `ULOGIN_DISPLAY`. Доступно три варианта:

- `small` *(по умолчанию)*
- `panel`
- `button`

Также можно задать стиль кнопок, установив значение переменной `ULOGIN_THEME`. Принимаются два варианта:

- `classic` *(по умолчанию)*
- `flat`

Список используемых провайдеров определяется директивой `ULOGIN_PROVIDERS`. По умолчанию включены:

- `vkontakte`
- `facebook`
- `twitter`
- `google`
- `livejournal`

Дополнительные провайдеры, которые будут показаны внутри выпадающего меню, определяются в директиве `ULOGIN_HIDDEN`. По умолчанию:

- `yandex`
- `odnoklassniki`
- `mailru`
- `openid`

Директивой `ULOGIN_REDIRECT_URL` можно задать URL, на который пользователь будет переадресован после успешной аутентификации. Лучше всего указать полностью, вместе с доменом и схемой.

Если при входе нужно выполнить какую-то JavaScript-функцию, укажите её в виде строки в переменной `ULOGIN_CALLBACK`.

Если необходимо создать функцию, создающую пользователя Django (это полезно при использовании нестандартной модели), можно указать полный путь до неё в переменной `ULOGIN_CREATE_USER_CALLBACK` (см. ниже)

2.5.2 Схемы

Как упоминалось выше, в некоторых случаях нужно разместить на одной странице несколько виджетов ulogin с различными настройками. В этом случае целесообразно создать нужное количество схем и настроить их.

Схемы определяются как словарь ULOGIN_SCHEMES, ключи которого — названия схем, используемые в шаблонном теге {% ulogin_widget "scheme_name" %}, а значения - словари с настройками.

Ключи этого словаря совпадают с названиями соответствующих «глобальных» настроек, но без префикса ULOGIN_. Это означает, что в пределах настройки схемы ключ DISPLAY будет отвечать за вид панели виджета, как и его глобальный «коллега» ULOGIN_DISPLAY

Кроме того, настройки схем наследуют глобальные настройки. Например, такая настройка:

```
ULOGIN_PROVIDERS = ['google', 'twitter']
ULOGIN_HIDDEN = ['odnoklassniki', 'mailru']
ULOGIN_DISPLAY = 'panel'

ULOGIN_SCHEMES = {
    'default': {'HIDDEN': ['yandex']},
    'comments': {'DISPLAY': 'small'}
}
```

означает, что по умолчанию включены провайдеры google и twitter, odnoklassniki и mailru скрыты, а виджет выводится в раскладке panel.

Однако при использовании схемы default скрытым провайдером окажется yandex, а схема comments будет выведена в раскладке small. Настройки, которые не переопределены, будут браться из глобальной области.

Если в проекте используются бэкенды аутентификации, отличные от стандартных, можно указать настройку ULOGIN_AUTHENTICATION_BACKEND, которая будет использована для хранения в сессии информации о том, через какой бэкенд аутентифицировался пользователь.

2.5.3 Сигналы

При аутентификации пользователя создаётся новый Django-пользователь, username которого заполняется uuid4-хешем. Однако при создании новой аутентификации срабатывает сигнал django_ulogin.signals.assign, в котором передаётся объект request, пользователь Django, аутентификация и флаг registered , показывающий, была ли создана запись.

Чтобы сделать имя поля дружественным пользователю, достаточно создать объект, подписанный на сигнал django_ulogin.signals.assign:

```
from django_ulogin.models import ULoginUser
from django_ulogin.signals import assign

def catch_ulogin_signal(*args, **kwargs):
    """
    Обновляет модель пользователя: исправляет username, имя и фамилию на
    полученные от провайдера.

    В реальной жизни следует иметь в виду, что username должен быть уникальным,
    а в социальной сети может быть много "тёзок" и, как следствие,
    возможно нарушение уникальности.

    """
    user=kwargs['user']
    json=kwargs['ulogin_data']

    if kwargs['registered']:
        user.username = json['username']
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        user.first_name = json['first_name']
        user.last_name = json['last_name']
        user.email = json['email']
        user.save()


assign.connect(receiver=catch_ulogin_signal,
               sender=ULoginUser,
               dispatch_uid='customize.models')
```

Можно изучить тестовый проект, в котором реализована функция сохранения данных, полученных от uLogin:

- https://github.com/marazmiki/django-ulogin/tree/master/test_project
- https://github.com/marazmiki/django-ulogin/blob/master/test_project/customize/models.py#L58

2.5.4 Создание нестандартной модели пользователя

По умолчанию при аутентификации пользователя через социальные сети будет создаваться стандартный пользователь Django; в качестве имени будет использоваться обрезанный до 30 символов (ограничение длины поля `username` в стандартной модели пользователя) UUID4-хеш.

Однако если Вы используете собственную модель, отличную от `django.contrib.auth.models.User`, в которой содержатся другие поля, то можете написать собственную функцию, которая создавала бы пользователя по Вашему сценарию.

Требования к этой функции:

- она должна принимать два аргумента: `request` и `ulogin_response` для передачи объекта `HttpRequest` и `JSON`, полученного от uLogin соответственно;
- возвращать сохранённую модель пользователя

Пример:

```
from my_projects.models import MyUser

def my_user_create(request, ulogin_response):
    return MyUser.objects.create_user(
        username='Vasya_{0}'.format(uuid.uuid4()),
        birthday=datetime.date.today()
    )
```

После этого в настройках проекта в переменной `ULOGIN_CREATE_USER_CALLBACK` указать полный путь этой функции:

```
ULOGIN_CREATE_USER_CALLBACK = "my_projects.utils.my_user_create"
```

2.6 Хотите помочь с django-ulogin?

Если у вас есть желание преобщиться к чудесному миру opensource, или же приложение не устраивает вас в текущем его виде, то вы можете внести свою лепту в его улучшение.

2.6.1 Сообщайте об ошибках

Если вы нашли ошибку, или считаете, что всё должно быть совсем не так, или просто хотите что-то предложить, то к вашим услугам тикет-система на Гитхабе: https://github.com/marazmiki/django-ulogin/issues.

2.6.2 Пишите код

Есть желание поучаствовать в разработке приложения? Форкайте исходные коды https://github.com/marazmiki/django-ulogin и пишите свой. Может, он исправит какую-то ошибку, или описку в документации. А может, реализует какую-то незамеченную, но очень важную новую возможность в программе.

После того, как вы внесёте необходимые изменения в код, смело делайте пулл-реквест, чтобы ваши изменения попали в основной репозиторий и принимайте благодарности.

2.6.3 Рекомендации для написания кода

- Пишите код таким образом, чтобы он соответствовал PEP-0008. Обращайте внимание на советы по оформлению кода, которые даёт IDE (если пользуетесь). Также можно пользоваться линчевателями `pylint` и `flake8`. Если у вас установлен GNU Make (обычно легко ставится или изначально присутствует на большинстве unix- и linux-систем), можете воспользоваться командой `make flake8`, которая найдёт все ошибки:

```
$ make flake8
```

- Не пренебрегайте тестами и не забывайте их запускать. В идеале, тест должен быть написан до написания собственно кода. Быстрый запуск тестов на текущем окружении можно выполить таким образом:

```
$ python setup.py test
```

или, если установлен GNU Make, таким:

```
$ make test
```

Однако, поскольку `django-ulogin` является библиотекой и должна работать в различных окружениях (несколько версий Django, различные версии языка Python), необходимо также проверить, что код будет выполняться там. Это делается с помощью утилиты tox.

```
$ tox
```

Создание всех окружений и прогон на них тестов занимает некоторое время (около 10-15 минут, в зависимости от мощности компьютера и качества подключения к сети). Список поддерживаемых конфигураций, на которых должно отрабатывать приложение, задаётся в секции `[tox:tox]` файла `setup.cfg` из корня репозитория.

- Когда Вы сделаете *PULL REQUEST*, сервис для неприрывных интеграций Travis CI автоматически проверит, работает ли программа на всех заявленных окружениях. И если увидите красную полосу, исправьте, пожалуйста, ошибки.
- Ну и, конечно, не забудьте добавить себя в список авторов, который хранится в `docs/source/authors.rst`. Желательно с указанием настоящего имени и e-mail

2.7 История изменений

2.7.1 1.x

1.1.0

- Избавление от legacy: удалена поддержка `Django<2.1`;
- Минимально поддерживаемая версия `Python`: `3.5`;
- Автоматическая публикация пакета на PyPI с помощью Github Actions;
- Использование Poetry вместо pipenv

1.0.7

- Поддержка `Django==3.1`

1.0.6

- Security-обновление: внешняя зависимость `bleach` обновлена до безопасной версии `3.1.4`

1.0.5

- Security-обновление: внешняя зависимость `bleach` обновлена до безопасной версии `3.1.2`

1.0.4

- Исправлена ошибка #43: `ULoginUser.__str__()` в некоторых случаях возвращал не строку, что приводило к ошибкам при удалении моделей в админке. Спасибо @shot131 за репорт.
- Security-обновление внешней зависимости `bleach`

1.0.3

- Добавлена поддержка `Python 3.8` и `Django 3.0`

1.0.2

Технический релиз, ничего заметного для пользователей тут нет

- Добавлены новыве бейджи
- Исправлена интеграция с Travis CI и Read The Docs

1.0.1

- Добавлена поддержка `Django 2.2`

1.0.0

- Добавлена поддержка стиля кнопок (классические или плоские)
- Добавлена поддержка новых провайдеров: `uID`, `Instagram` и `Wargaming.get`
- Удалена поддержка провайдера `dudu`
- Добавлена поддержка `Python 3.6` и `Python 3.7` и `Django 2.0` и `2.1`. Спасибо Максиму Полежаеву, PR #39
- Удалена поддержка версий `Python` ниже `3.4` и `Django` ниже `1.7`
- В приложение добавлены миграции (ранее не хотелось этого делать из-за необходимости поддерживать как встроенные миграции `Django 1.7+`, так и `South` для более ранних версий `Django`
- Обновлена документация
- PyPI-классификатор `Development Status` переведён в значение `5 - Production/Stable`. Не, ну правда, сколько можно уже? :)
- Максимально облегчен `setup.py`, а вся метаинформация о пакете, а также настройки `bumpversion`, `tox` и `coverage` вынесены в `setup.cfg`
- Исправлены бейджи

2.7.2 0.3x

0.3.0

- Удалена поддержка `Python 2.6`
- Добавлена поддержка `Python 3.5`
- Удалена поддержка `Django` версий `1.5` и `1.6`
- Расширенная документация на Read the Docs
- Добавлена поддержка `Django 1.9`. Спасибо за помощь Максиму Оранскому, PR #27
- Исправлена ошибка `urllib.unquote` в Py3 (спасибо mcd-php)
- Дефолтное имя (UUID) обрезано до 30 символов для совместимости с длиной поля username по умолчанию. Спасибо Haos616)
- Исправлена ошибка при обращении к urllib.unquote в Python 3.x. Исправил mcd-php, PR #32
- Исправлена ошибка, приводящая к UnicodeDecodeError при использовании Python 3.x. Исправил mcd-php, PR #33
- Исправлены бейджи

2.7.3 0.2x

0.2.3

- Обновлена HEAD-версия Django

0.2.2

- Обновлена HEAD-версия Django

0.2.1

- Удалена поддержка `python 3.2`, добавлена поддержка `3.4`
- Улучшен `PEP-0008`
- Добавлен `coverage`

2.7.4 0.1x

0.1.9

- Добавлена обработка подтверждения почты (параметр `verify=1`, см. http://ulogin.ru/help.php#faq)

0.1.8

- Добавлены провайдеры `Google+`, `Foursquare`, `Tumblr`, `Dudu`;
- Отказ от поддержки `Django 1.4x` и ниже;
- Улучшение кода по `PEP-0008`
- Изменение кода виджета
- Поддержка собственной модели пользователя `Django`
- Обновлён демонстрационный проект

0.1.7

- Исправена ошибка, связанная с декодированием uri, в котором встречаются не ascii-символы (исправил mike-grayhat)

0.1.6

- Добавлен параметр `ULOGIN_AUTHENTICATION_BACKEND`
- Грамотная работа с timezones при `USE_TZ=True` (в Django 1.4)

0.1.5

- Отказ от поддержки `Django 1.2x`
- использованы class based views.
- Можно входить в свой аккаунт через различных провайдеров.
- Можно отвязывать аккаунты в соц. сетях от своего пользователя
- Новое поле: `phone`

- Новые провайдеры: * `flickr` * `vimeo` * `webmoney` * `youtube` * `steam` * `soundcloud` * `lastfm` * `linkedin` * `liveid`
- `JS` виджета подключается по `HTTP` или `HTTPS` в зависимости от того, какой схемой пользуется посетитель
- Исправлены неточности в документации (добавлено про обязательные контекст-процессоры)
- Соответствие `pep-0008` :)

0.1.4

- Добавлена поддержка схем виджетов (позволяет разместить на одной странице несколько разных виджетов)
- Обновлена документация
- Изменён HTML-код в соотстветствии с требованиями `ulogin`

2.8 Авторы

Список замечательных людей, принявших участие в развитии проекта:

- Михаил «marazmiki» Пороховниченко <marazmiki@gmail.com>
- Игорь «itoldya» Исаев <i-told@ya.ru>
- Михаил «mike-grayhat» Оськин
- Виталий «vetal4444» Савченко <vetal4444@gmail.com>
- Антон «Forever-Young» Новосёлов <anton.novosyolov@gmail.com>
- Денис «dkopitsa» Копица <denis.kopitsa@gmail.com>
- Максим «sdfsdhgjkbmnmxc» Оранский <maxim.oransky@gmail.com>
- Максим «maxpolezhaev » Полежаев <m.polejaev@gmail.com>
- Haos616 <Haos616@Gmail.com>
- mcd-php

2.9 Лицензия

Приложение распространяется по лицензии MIT

Глава 3

Оглавления и индексы

- genindex
- search