

## Модуль для работы с сервисом Onpay.ru для сайта, построенного на фреймворке Django.

Пользователи других фреймворков могут написать своё приложение, специально для этого основная функциональность собрана в файле `common.py` .

Фреймворко-зависимую часть (запросы, систему хранения, обработку форм, оповещение менеджеров) нужно будет переписать, взяв за основу текущий код.

[Начало](#) > [Опра](#) > Балансы

### Выберите баланс для изменения

[Добавить баланс](#)

Действие:	-----	Выполнить	
	<input type="checkbox"/> <a href="#">2010-06-11 10:44:55.309972</a>	Sandanski_17	450.00
	<input type="checkbox"/> <a href="#">2010-06-11 10:36:23.290767</a>	SNOVKA	550.00
	<input type="checkbox"/> <a href="#">2010-06-10 18:58:51.543544</a>	denger	15.00

3 балансы

[Начало](#) > [Опра](#) > Операции

### Выберите операция для изменения

[Добавить операци](#)

Действие:	-----	Выполнить	Фильтр
	<input type="checkbox"/> <a href="#">2010-06-11 13:03:21.663171</a>	64 Andrei 100.00	
	<input type="checkbox"/> <a href="#">2010-06-11 13:00:47.806951</a>	63 Andrei 100.00	
	<input type="checkbox"/> <a href="#">2010-06-11 13:00:02.753948</a>	62 Andrei 100.00	
	<input type="checkbox"/> <a href="#">2010-06-11 13:00:02.447436</a>	61 Andrei 100.00	
	<input type="checkbox"/> <a href="#">2010-06-11 12:43:48.467187</a>	60 denger 100.00	
	<input type="checkbox"/> <a href="#">2010-06-11 12:41:27.135936</a>	59 denger 100.00	
	<input type="checkbox"/> <a href="#">2010-06-11 12:36:12.455593</a>	58 Elessar 100.00	
	<input type="checkbox"/> <a href="#">2010-06-11 12:36:02.283881</a>	57 Economist 100.00	
	<input type="checkbox"/> <a href="#">2010-06-11 12:11:17.267966</a>	56 Footbolist777 100.00	
	<input type="checkbox"/> <a href="#">2010-06-11 11:57:11.460772</a>	55 bobr 100.00	
	<input type="checkbox"/> <a href="#">2010-06-11 11:49:49.126195</a>	54 DENtt 100.00	
	<input type="checkbox"/> <a href="#">2010-06-11 11:27:16.751208</a>	53 vikont 100.00	
	<input type="checkbox"/> <a href="#">2010-06-11 11:10:51.004491</a>	52 Sandanski_17 100.00	
	<input type="checkbox"/> <a href="#">2010-06-11 11:07:58.969225</a>	51 Sandanski_17 100.00	
	<input type="checkbox"/> <a href="#">2010-06-11 10:57:46.067835</a>	50 Sandanski_17 100.00	

[Добавить операци](#)

#### Фильтр

- По оплачено
- Все
  - Да
  - Нет

## Схема работы Onpay.ru через фреймворк Django

- Сгенерировать тег `iframe` из множества параметров. В этом поможет класс `IframeGenerator` .
- Принять запрос от сервиса, который предоставит данные (кому какую сумму в какой валюте

перечислили).

3. Внести эти данные в базу данных.

## Настройки в личном кабинете Opray.ru для работы с Django

В личном кабинете Opray.ru (Настройки магазина) необходимо настроить параметры API IN:

Уведомлять по API - Да

Проверять MD5 на ссылках - Да (не обязательно)

URL API: <http://вашидомен/onpay/api/> (можно переопределить через URLconf)

Пароль для API IN: ksjgJskLJds - ваш секретный код, который будет нужен при настройках платежного модуля Opray.ru в Django

The screenshot shows the 'Personal Cabinet' interface with the following navigation bar:

- Новости
- Деньги
- Платежи
- Личные настройки**

Under 'Личные настройки', the 'Настройки магазина' tab is selected. The page displays the following configuration fields:

- Название сервиса:** Django onpay demo
- Описание товара или услуги:** Тестирование приложения django-onpay
- Адрес сайта (для посетителей):** http://example.org/
- Настройки API IN** (Section)
  - Уведомлять по API
  - Проверять MD5 на ссылках
- Метод отправки запросов в API:**
  - POST
  - GET
- URL API:** http://example.org/onpay/api/
- Пароль для API IN:** lsdjf3o2uadjfk

At the bottom left is a 'Сохранить' (Save) button.

## Установка Django-Onpay

Приложение требует модуль lxml. Начиная с версии 2.6 python содержит его в стандартной

библиотеке. Если вы пользуетесь более ранней версией (например, той, что идёт в пакетах с debian lenny), необходимо установить пакет python-lxml.

Для собственно установки пакета подойдет один из вариантов:

```
$ hg clone http://bitbucket.org/denger/django-onpay
$ ln -s /path/to/django-onpay/onpay /usr/lib/python2.6/site-packages
или
$ hg clone http://bitbucket.org/denger/django-onpay
$ cd django-onpay
$ sudo python setup.py install
или
$ sudo pip install -e hg+http://bitbucket.org/denger/django-onpay
```

Далее:

1. прописываем `onpay` в `INSTALLED\_APPS` ,
2. `./manage.py syncdb`
3. в `settings.py` добавляем переменную `ONPAY` с минимумом настроек (см. ниже) ,
4. добавляем в `urls.py` : `('^onpay/', include('onpay.urls')),` ,
5. тестируем работу.

Настройка Все параметры хранятся в словаре `ONPAY` в файле `settings.py` .

Обязательные параметры:

```
ONPAY = {
    "onpay_login": "example", # Ваш логин на опрай
    "private_code": "ksjgJskLJds", # Пароль, который вы ввели на сайте опрай
}
```

Необязательные параметры:

```
"url_success": "http://example.org/onpay/api/",
# default: то что задано в настройках на сайте опрай
```

```
"use_balance_table": True,
# записывать в таблицу баланса.
```

```
"pay_mode": "fix",
# "free" - обновление баланса, юзер может изменить цифру
# "fix" - фиксированный платеж, цифра в фрейме только для чтения
```

```
"f": None,
# скин, возможные значения - None, 1, 2, 3
# в зависимости от скина
# подробнее: http://onpay.ru/form/
```

```
"enable_email_notify": None,
```

```
# если включить опцию, при платежах будет отправлен email
# через функцию email_managers

"debug": None,
# на данный момент отправляет через mail_admins запрос от Onpay.ru
```

Чтобы встроить платежную систему в свой дизайн надо переопределить шаблоны из папки onpay.

Можно вместо include в urls.py прописать свои роуты к своим views, если требуется какие-то изменения.

Можно поменять параметры после инициализации IframeGenerator:

```
iframe_generator = IframeGenerator()
iframe_generator.set_f(3)
iframe_generator.width = 100500
iframe_generator.pay_mode = "free"
```

После оплаты отправляется сигнал `onpay.signals.refilled\_balance`, если на него подписать свою функцию, можно добиться любой функциональности. Примеры смотрите в файле `signals.py`.

Ну и, наконец, можно изменить исходные тексты и прислать hg патчи нам - по возможности добавим в репозиторий.

common.py для работы с dom.minidom

```
# coding: UTF-8
import urllib
from hashlib import md5
#from lxml import etree
from xml.dom.minidom import getDOMImplementation

from onpay.conf import get_constant

class IframeGenerator(object):
    def __init__(self):
        self.pay_mode = get_constant("pay_mode", "fix")
        self.currency = get_constant("currency", "RUR")
        self.convert = get_constant("convert", "yes")
        self.url_success = get_constant("url_success")
        self.private_code = get_constant("private_code")
        self.onpay_login = get_constant("onpay_login")
        self.set_f(get_constant("f"))

    def iframe_url_params(self, operation_id, summ, email=None):
        "Функция определения параметров платежной формы."
        query = {
            "pay_mode": self.pay_mode,
```

```
        "currency": self.currency,
        "convert": self.convert,

        "pay_for": operation_id,
        'price_final' : 'true',
        "price": summ,
        "md5": self.md5check(summ, operation_id),
    }
    if self.url_success:
        query["url_success"] = self.url_success
    if email:
        query['user_email'] = email
    if self.f:
        query['f'] = self.f
    return urllib.urlencode(query)

def md5check (self, summ, operation_id):
    return md5(";" .join(
        (self.pay_mode, str(summ), self.currency, str(operation_id),
        self.convert, self.private_code,))).hexdigest().upper()

def set_f(self, f):
    "Определение ширины и высоты в зависимости от текущего скина"
    self.f = f
    self.width, self.height = {
        None: ( 300, 500),
        1:     (1020, 660),
        2:     ( 250, 540),
        3:     ( 960, 800),
    }[f]

def iframe_tag (self, operation_id, summ, email=None):
    url = "http://secure.onpay.ru/pay/%s?%s" % (self.onpay_login,
        self.iframe_url_params(operation_id, summ, email=email))
    options = {
        "src": url,
        "width": self.width,
        "height": self.height,
        "frameborder": "no",
        "scrolling": "no",
        "name": "onpay",
        "id": "onpay",
    }
    options_gen = ((u'%s="%s"' % o_0) for o_0 in options.iteritems())
    return u'<iframe %s></iframe>' % (u" ".join(options_gen))

def answer(type, code, pay_for, order_amount, order_currency, text):
    "функция выдает ответ для сервиса onpay в формате XML на чек запрос"
    array_for_md5 = (type, pay_for, order_amount, order_currency, str(code),
        get_constant('private_code'))
```

```

result_md5 = md5(";" . join(array_for_md5)).hexdigest().upper()

dom = getDOMImplementation()
doc = dom.createDocument(None, 'result', None)

def xml_add(name, value):
    node = doc.createElement(name)
    node.appendChild(doc.createTextNode(value))
    doc.documentElement.appendChild(node)

xml_add('code', str(code))
xml_add('pay_for', pay_for)
xml_add('comment', text)
xml_add('md5', result_md5)

return doc.toxml('UTF-8')

#root = etree.Element("result")
#etree.SubElement(root, "code").text = str(code)
#etree.SubElement(root, "pay_for").text = pay_for
#etree.SubElement(root, "comment").text = text
#etree.SubElement(root, "md5").text = result_md5
#return ''etree.tostring(root, pretty_print=True,
#                      xml_declaration=True, encoding='UTF-8')

```

```

def answer_dict(POST, code, text):
    "Shortcut for call answer with POST or form dict as parameter"
    return answer(
        POST.get("type"),
        code,
        POST.get("pay_for"),
        POST.get("order_amount"),
        POST.get("order_currency"),
        text,
    )

```

```

def answerpay(type, code, pay_for, order_amount, order_currency, text,
onpay_id):
    "функция выдает ответ для сервиса опрэй в формате XML на руу запрос"

```

```

array_for_md5 = (type, pay_for, onpay_id, pay_for, order_amount,
                  order_currency, str(code), get_constant('private_code'))
result_md5 = md5(";" . join(array_for_md5)).hexdigest().upper()

dom = getDOMImplementation()
doc = dom.createDocument(None, 'result', None)

def xml_add(name, value):
    node = doc.createElement(name)
    node.appendChild(doc.createTextNode(value))

```

```
doc.documentElement.appendChild(node)

xml_add('code', str(code))
xml_add('comment', text)
xml_add('onpay_id', onpay_id)
xml_add('pay_for', pay_for)
xml_add('order_id', pay_for)
xml_add('md5', result_md5)

return doc.toxml('UTF-8')
#root = etree.Element("result")
#etree.SubElement(root, "code").text = str(code)
#etree.SubElement(root, "comment").text = text
#etree.SubElement(root, "onpay_id").text = onpay_id
#etree.SubElement(root, "pay_for").text = pay_for
#etree.SubElement(root, "order_id").text = pay_for
#etree.SubElement(root, "md5").text = result_md5
#return etree.tostring(root, pretty_print=True,
#                      xml_declaration=True, encoding='UTF-8')
```

```
def answerpay_dict(request_dict, code, text):
    "Shortcut for call answerpay with POST or form dict as parameter"
    return answerpay(
        request_dict.get('type'),
        code,
        request_dict.get('pay_for'),
        request_dict.get('order_amount'),
        request_dict.get('order_currency'),
        unicode(text),
        request_dict.get('onpay_id'),
    )
```

Автор: Денис Бурый

Лицензия: MIT

From:  
<http://wiki.onpay.ru/> - **Onpay.ru Wiki**

Permanent link:  
<http://wiki.onpay.ru/doku.php?id=django&rev=1312282163>

Last update: **2011/08/02 11:49**

