django-pagseguro2 Documentation

Versão 1.1.0

Allisson Azevedo

05/03/2016

Sumário

Integração da API v2 do PagSeguro com o Django. Para API v1 do PagSeguro, use o django-pagseguro.

CAPÍTULO 1

Conteúdo

1.1 Tutorial

O django-pagseguro2 necessita do Django versão 1.3+, lembrando que o suporte ao python3 está presente apenas nas versões 1.5+.

1.1.1 Instalação

Instale o django-pagseguro2:

```
pip install django-pagseguro2
```

1.1.2 Configuração

Adicione o app pagseguro no INSTALLED_APPS:

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.staticfiles',
    'django.contrib.admin',
    'pagseguro',
)
```

O django-pagseguro2 suporta as migrações do django 1.7, caso você esteja usando o South, adicione a seguinte configuração no seu settings.py:

```
SOUTH_MIGRATION_MODULES = {
    'pagseguro': 'pagseguro.south_migrations',
}
```

Adicione as configurações no settings.py:

```
PAGSEGURO_EMAIL = 'fulano@cicrano.com'
PAGSEGURO_TOKEN = 'token'
PAGSEGURO_SANDBOX = True # se o valor for True, as requisições a api serão feitas usando o PagSeguro
PAGSEGURO_LOG_IN_MODEL = True # se o valor for True, os checkouts e transações vão ser logadas no da
```

Ao utilizar o django-pagseguro2 em produção, tenha certeza de definir a configuração `PAGSEGURO_SANDBOX` como `False` uma vez que o valor padrão é `True`.

Adicione a view que recebe as notificações no urls.py:

```
urlpatterns = patterns(
    '',
    url(r'^retorno/pagseguro/', include('pagseguro.urls')),
    # a url de retorno será /retorno/pagseguro/
)
```

1.1.3 Trabalhando com a API de checkout

Recomendo que você abra um shell do Django para realizar os testes:

```
python manage.py shell
```

Para começar, precisamos importar duas classes, PagSeguroItem e PagSeguroApi:

```
>>> from pagseguro.api import PagSeguroItem, PagSeguroApi
```

O PagSeguroItem representa um item que será adicionado a transação:

```
>>> item1 = PagSeguroItem(id='0001', description='Meu item 0001', amount='100.00', quantity=1)
>>> item1
<PagSeguroItem: Meu item 0001>
>>> item1.id
u'0001'
>>> item1.amount
Decimal('100.00')
>>> item1.quantity
1
>>> print(item1.shipping_cost)
None
>>> print(item1.weight)
None
```

Você pode adicionar informações como custo de envio e peso:

```
>>> item2 = PagSeguroItem(id='0002', description='Meu item 0002', amount='150.00', quantity=1, shipp
>>> item2
<PagSeguroItem: Meu item 0002>
>>> item2.id
u'0002'
>>> item2.amount
Decimal('150.00')
>>> item2.quantity
1
>>> item2.shipping_cost
Decimal('25.00')
>>> item2.weight
500
```

Agora que temos os itens, podemos fazer o checkout para obter o código da transação:

```
>>> pagseguro_api = PagSeguroApi(reference='id-unico-de-referencia-do-seu-sistema')
>>> # voce poderia passar valores iniciais, ex: pagseguro_api = PagSeguroApi(email='meu@email.com', t
>>> # voce pode passar qualquer valor inicial que a documentacao do PagSeguro informa, exceto os ite
>>> # o email e token são carregados automaticamente pelas variáveis do settings.
>>> # adicionando itens
>>> pagseguro_api.add_item(item1)
>>> pagseguro_api.add_item(item2)
```

```
>>> # verificando os itens
>>> pagseguro_api.get_items()
[<PagSeguroItem: Meu item 0001>, <PagSeguroItem: Meu item 0002>]
>>> # removendo os itens
>>> pagseguro_api.clear_items()
>>> pagseguro_api.get_items()
[]
>>> # fazendo um checkout
>>> pagseguro_api.add_item(item1)
>>> pagseguro_api.add_item(item2)
>>> data = pagseguro_api.checkout()
>>> data
{'date': datetime.datetime(2014, 6, 7, 15, 19, 48, tzinfo=tzoffset(None, -10800)), 'status_code': 200
>>> # agora basta redirecionar o cliente para o data['redirect_url']
>>> data['redirect_url']
'https://sandbox.pagseguro.uol.com.br/v2/checkout/payment.html?code=D0C5A7F8E5E53268849D4F89DA3363E0
```

Você pode consultar os dados de uma transação:

```
>>> pagseguro_api = PagSeguroApi()
>>> data = pagseguro_api.get_transaction('437D1B99-A6E8-46F0-8C00-47B818615AA2')
>>> data['success']
True
>>> data['transaction']
OrderedDict([(u'date', u'2014-06-07T15:25:36.000-03:00'), (u'code', u'437D1B99-A6E8-46F0-8C00-47B8186
>>> data['transaction']['code']
u'437D1B99-A6E8-46F0-8C00-47B818615AA2'
```

Passando parâmetros extras na inicialização do PagSeguroApi:

```
>>> from pagseguro.api import PagSeguroApi
>>> from decimal import Decimal
>>> extra_amount = Decimal('20.00')
>>> sender_email = 'user@email.com'
>>> sender_name = 'Fulano da Silva'
>>> sender_area_code = 83
>>> sender_phone = 11111111
>>> pagseguro_api = PagSeguroApi(reference='id-unico-de-referencia-do-seu-sistema', extraAmount=extra
```

Você pode passar qualquer parâmetro http, exceto os relativos aos itens. Referência.

1.1.4 Trabalhando com a API de checkout transparente

Primeiramente, todas as configurações devem ter sido realizadas como descrito na seção configurações.

Para realizar o checkout transparent você vai precisar de algumas informações adiquiridas utilizando a lib javascript oficial do pagseguro, em conjunto com a nossa api de checkout transparent:

- senderHash (Obrigatório para todas as compras)
- creditCardToken (Obrigatório apenas para cartão de crédito)

Vamos iniciar uma sessão de pagamento para conseguir as informações acima.

Importe a PagSeguroApiTransparent para iniciar uma sessão de pagamento:

```
>>> from pagseguro.api import PagSeguroApiTransparent
>>> # pegando a session id
>>> data = pagseguro_api.get_session_id()
>>> # o método get_session_id retorna um dicionário que contém uma id válida que será utilizada no B
```

```
>>> # uma sessão de chekout transparent.
>>> session_id = data['session_id']
```

No Browser, importe o javascript do pagseguro:

```
<script type="text/javascript" src=
    "https://stc.sandbox.pagseguro.uol.com.br/pagseguro/api/v2/checkout/pagseguro.directpayment.js">
</script>
```

Adicione o id da sessão adquirido ao chamar o método get_session_id:

```
<script type="text/javascript">
    PagSeguroDirectPayment.setSessionId('ID_DA_SESSÃO');
</script>
```

Após iniciar uma sessão de checkout é preciso obter a identificação do comprador **senderHash**, essa informação é obrigatória para realizar o checkout transparent:

```
<script type="text/javascript">
    PagSeguroDirectPayment.getSenderHash();
</script>
```

Apenas para compras no cartão de crédito é obrigatório adquirir o **creditCardToken** que é utilizado para realizar o checkout transparent:

```
<script type="text/javascript">
    PagSeguroDirectPayment.createCardToken({
        cardNumber: {número},
        brand: {bandeira},
        cvv: {código de segurança},
        expirationMonth: {mês de expiração},
        expirationYear: {ano de expiração},
        success: {função de callback para chamadas bem sucedidas},
        error: {função de callback para chamadas que falharam},
        complete: {função de callback para todas chamadas}
    });
</script>
```

Para mais informações consultar a api oficial do pagseguro.

Agora, vamos **realizar o checkout transparent**. Primeiramente, importe a PagSeguroApiTransparent e o PagSeguroItem:

```
>>> from pagseguro.api import PagSeguroApiTransparent, PagseguroItem
>>> # inicializando a api
>>> api = PagseguroApiTransparent()
```

Adicione o item:

```
>>> item1 = PagSeguroItem(id='0001', description='Notebook Prata', amount='24300.00', quantity=1)
>>> api.add_item(item1)
```

Adicione os dados do comprador:

```
>>> sender = {'name': 'Jose Comprador', 'area_code': 11, 'phone': 56273440, 'email': 'comprador@uol.c
>>> api.set_sender(**sender)
```

Adicione o endereço do comprador:

```
>>> shipping = {'street': "Av. Brigadeiro Faria Lima", 'number': 1384, 'complement': '5o andar', 'dis
>>> api.set_shipping(**shipping)
```

Apenas para compras no **boleto**:

```
>>> api.set_payment_method('boleto')
```

Apenas para compras no **débito**:

```
>>> api.set_payment_method('eft')
>>> api.set_bank_name('itau')
```

Apenas para compras no **cartão de crédito**:

```
>>> api.set_payment_method('creditcard')
>>> data = {'quantity': 5, 'value': 125.22, 'name': 'Jose Comprador', 'birth_date': '27/10/1987', 'cp
>>> api.set_creditcard_data(**data)
>>> billing_address = {'street': 'Av. Brig. Faria Lima', 'number': 1384, 'district': 'Jardim Paulista
>>> api.set_creditcard_billing_address(**billing_address)
>>> api.api.set_creditcard_token('token-adquirido-no-browser')
```

Para finalizar, adicione a senderHash adquirida no browser:

```
>>> api.set_sender_hash('hash-adquirida-no-browser')
```

Efetue o checkout transparent:

```
>>> data = api.checkout()
```

1.1.5 Trabalhando com Signals de checkout

Podemos usar o recurso de Signals do Django para capturar informações relacionadas aos checkouts.

Isso é bastante útil para dectectar possíveis problemas na implementação.

Temos os seguintes Signals disponíveis para checkouts:

- **checkout_realizado** (Disparado sempre que um novo checkout for feito).
- **checkout_realizado_com_sucesso**
- **checkout_realizado_com_erro**

Para capturar o Signal **checkout_realizado**:

```
>>> from pagseguro.signals import checkout_realizado
>>> def load_signal(sender, data, **kwargs):
...     print(data['success'])
...
>>> checkout_realizado.connect(load_signal)
```

1.1.6 Trabalhando com Signals de notificação

Após a transação ser concluída pelo cliente, o PagSeguro vai enviar uma requisição do tipo POST para uma url que você escolheu previamente sempre que alguma mudança ocorrer no status.

Para ambiente de testes, eu recomendo que você utilize o PagSeguro Sandbox em conjunto com o serviço Runscope para conseguir visualizar as notificações.

Quando o PagSeguro envia uma nova notificação, Signals são disparados contendo as informações da transação.

Para cada tipo de status, existe um Signal específico, se você quiser ser notificado apenas quando a compra for paga, você deve capturar o Signal **notificacao_status_pago**.

Temos os seguintes Signals disponíveis para notificações:

- **notificacao_recebida** (Disparado sempre que uma notificação for recebida).
- **notificacao_status_aguardando**
- **notificacao_status_em_analise**
- **notificacao_status_pago**
- **notificacao_status_disponivel**
- **notificacao_status_em_disputa**
- **notificacao_status_devolvido**
- **notificacao_status_cancelado**

Para capturar o Signal **notificacao_recebida**:

```
>>> from pagseguro.signals import notificacao_recebida
>>> def load_signal(sender, transaction, **kwargs):
...     print(transaction['status'])
...
>>> notificacao_recebida.connect(load_signal)
```

Exemplo de um objeto **transaction**:

```
>>> transaction
OrderedDict([(u'date', u'2014-06-07T15:25:36.000-03:00'), (u'code', u'437D1B99-A6E8-46F0-8C00-47B8186
>>> transaction.keys()
[u'date', u'code', u'type', u'status', u'lastEventDate', u'paymentMethod', u'grossAmount', u'discount
>>> transaction['status']
u'3'
>>> transaction['code']
u'437D1B99-A6E8-46F0-8C00-47B818615AA2'
```

1.1.7 Logando checkouts e transações no database

Sempre que você configura o PAGSEGURO_LOG_IN_MODEL = True, todos os checkouts e transações são logados no database.

Basta acessar o /admin/ e verificar.

1.1.8 Transações seguras com HTTPS

Caso você esteja usando as parametrizações de segurança do Django, adicione a respectiva linha no settings.py:

```
SECURE_REDIRECT_EXEMPT = 'retorno/pagseguro/'
```

Isso é necessário para que o Pagseguro consiga acessar a url de transação.

1.1.9 CloudFlare

Caso você utilize o serviço CloudFlare em servidor de produção, será necessário fazer algumas parametrizações no serviço para que as notificações enviadas pelo Pagseguro sejam recebidas corretamente, caso contrário, elas serão identificadas como ameaças pelo serviço e o acesso será negado.

Para revolser esse detalhe, basta entrar na página "Threat control" (CloudFlare), clicar em "Add custom role" e adicionar os seguintes IPs disponibilizado pelo PagSeguro abaixo.

- 186.234.16.8
- 186.234.16.9
- 186.234.48.8
- 186.234.48.9
- 186.234.144.17
- 186.234.144.18
- 200.147.112.136
- 200.147.112.137

Após adicioná-los, clique em "Trust +".

CAPÍTULO 2

Autor

- Allisson Azevedo

CAPÍTULO 3

Principais características

- Python: 2.6, 2.7, 3.2, 3.3 e 3.4.
- Django: 1.4 (apenas python2), 1.5, 1.6, 1.7, 1.8.
- Processamento de requisições http/https usando a biblioteca requests.
- Excelente cobertura de testes (> 80%).
- Documentação com exemplos práticos.

Links

- Github
- Travis CI
- Coveralls

CAPÍTULO 5

Referências

- django-pagseguro (Para API v1 do PagSeguro)
- PagSeguro: Api de Pagamento
- PagSeguro: Api de Notificação
- PagSeguro: Sandbox