odoo

How to use API to connect external tools

SEINLET Nicolas, TECHNICAL CONSULTANT

Introduction

- A carrier needs picking information on a regular basis in a file stored on a local computer
- A customer wants to send orders and receive invoices using EDIFACT
- An Odoo user wants to set pictures on products (50k+) based on an external product database available on Internet
- Import external products catalogue, and update prices on a regular basis

  - How to do it with Odoo ?
  - Is it possible to automate ?
  - Even on SaaS ?

odoo

odoo

XML/JSON RPC

Languages, Libraries

XML RPC

> *XML-RPC is a remote procedure call (RPC) protocol which uses XML to encode its calls and HTTP as a transport mechanism.*

en.wikipedia.org/wiki/XML-RPC

XML RPC

- Libraries exists in many languages
- Well documented on the Odoo side
  - https://www.odoo.com/documentation/master/api_integratio
  - Sample in many langages in the official documentation
- ACL and Record rules are respected
- Access to any object
- Call every method, except browse
- number of high-level APIs in various langages
    - https://github.com/akretion/ooor
    - https://pypi.python.org/pypi/openerp-client-lib/1.1.2
    - https://github.com/tinyerp/erppeek
    - ...

odoo

Basics

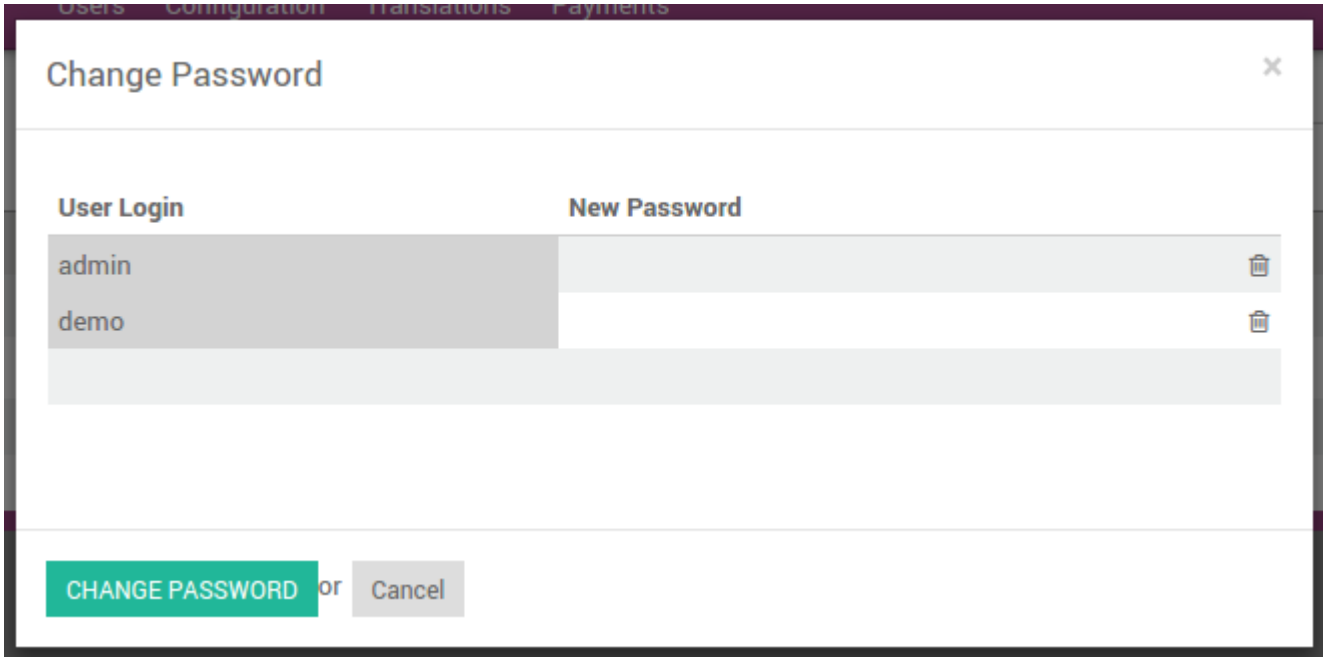
Connect, read, write, call methods

Let's try documentation

```
1  import functools
2  import xmlrpclib
3
4  #Connection
5  url = 'http://127.0.0.1:8069'
6  db = 'xp2015'
7  username = 'admin'
8  password = 'admin'
9
10 common = xmlrpclib.ServerProxy('{}/xmlrpc/2/common'.format(url))
11 common.version()
12
13 #Logging in
14 uid = common.authenticate(db, username, password, {})
15
16 #Calling methods
17 models = xmlrpclib.ServerProxy('{}/xmlrpc/2/object'.format(url))
18 print models.execute_kw(db, uid, password,
19     'res.partner', 'check_access_rights',
20     ['read'], {'raise_exception': False})
21
22 #List records
23 partner_ids = models.execute_kw(db, uid, password,
24     'res.partner', 'search',
25     [[['is_company', '=', True], ['customer', '=', True]]])
26
27 print partner_ids
28
```

On saas ?

- Do not use Oauth
- Set local password for the user

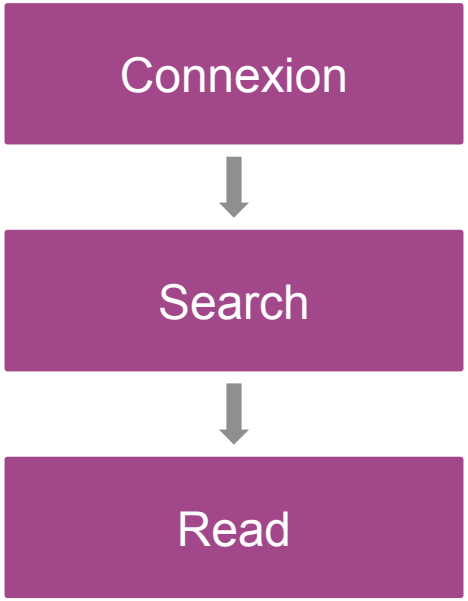


- Create a dedicated user
- Give required access rights

odoo

Read

- Using python
  - Using openerplib
  - Using jsonrpc

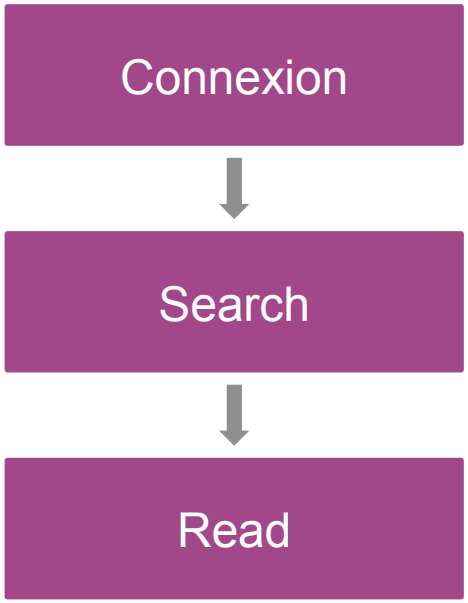


- Connect to instance
- Search for partners called Fletcher
- Display name and company name

Hello Michel

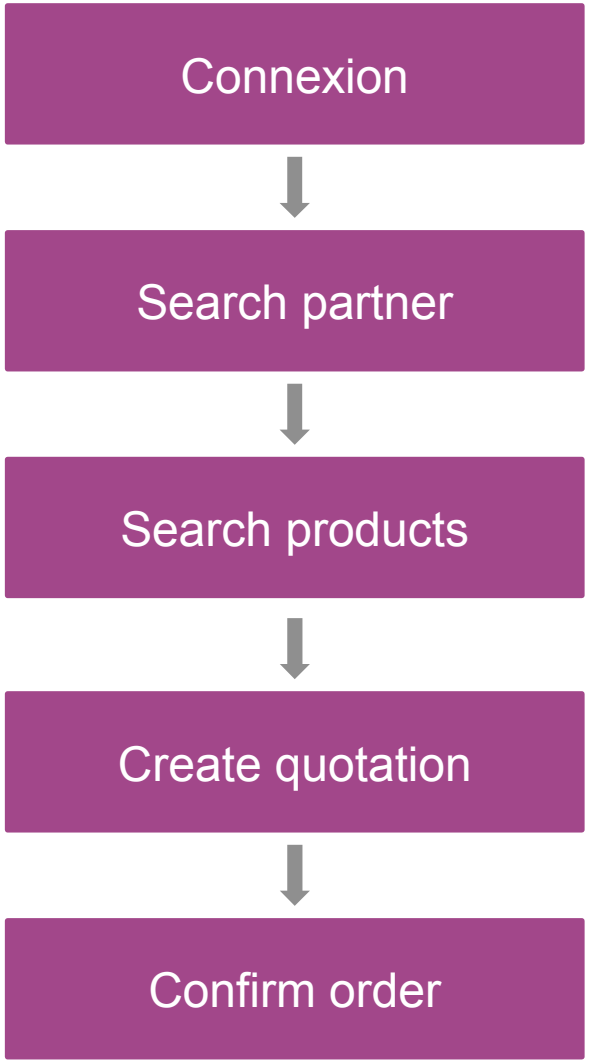
```python
import openerplib

class HelloWorld():
    def say_hello(self, connection):
        partner_model = connection.get_model('res.partner')

        partner_ids = partner_model.search([('name', 'ilike', 'fletcher')])
        partners = partner_model.read(partner_ids, ['name', 'parent_id'])
        for partner in partners:
            print partner
            res = "Hello %s" % partner['name']
            if partner['parent_id']:
                res = "%s from %s" % (res, partner['parent_id'][1])
            print res

if __name__ == '__main__':
    #Connect by xml-rpc
    connection = openerplib.get_connection(hostname="localhost",
                                           port=8069,
                                           database="xp2015",
                                           login="admin",
                                           password="admin",
                                           protocol="jsonrpc",
                                           user_id=1)

    connection.check_login()

    imp = HelloWorld()
    imp.say_hello(connection)

```

Hello World



- Supply user_id
- Using domain
- Return a dict
- M2O as list of id and name
- Supply list of fields

Write



- Check if all informations exists before create order (Products, partners, …)
- Avoid multiple call for the same object creation (Use magic numbers)
- Avoid read when unnecessary
- Call methods on objects

Create an order

```python
import openerplib

class Order():
    def create_order(self, connection):
        partner_model = connection.get_model('res.partner')
        so_model = connection.get_model('sale.order')
        product_model = connection.get_model('product.product')

        partner_ids = partner_model.search([('name', 'ilike', 'fletcher')])
        product_ids = product_model.search([('name', 'ilike', 'ipad')])

        order_id = so_model.create({
            'partner_id': partner_ids[0],
            'order_line': [(0,0,{'product_id': product_ids[0],
                                 'product_uom_qty':1}),
                           (0,0,{'product_id': product_ids[1],
                                 'product_uom_qty':2}),
                          ],

        })
        so_model.action_button_confirm([order_id,])


if __name__ == '__main__':
    #Connect by xml-rpc
    connection = openerplib.get_connection(hostname="localhost",
                                           port=8069,
                                           database="xp2015",
                                           login="admin",
                                           password="admin",
                                           protocol="jsonrpc",
                                           user_id=1)

    connection.check_login()

    imp = Order()
    imp.create_order(connection)

```

- Create order and lines in 1 call
- Return the created id
- Call methods

odoo

RPC vs Method in a module

```
def create_order(self, connection):
    partner_model = connection.get_model('res.partner')
    so_model = connection.get_model('sale.order')
    product_model = connection.get_model('product.product')

    partner_ids = partner_model.search([('name', 'ilike', 'fletcher')])
    product_ids = product_model.search([('name', 'ilike', 'ipad')])

    order_id = so_model.create({
        'partner_id': partner_ids[0],
        'order_line': [(0,0,{'product_id': product_ids[0],
                             'product_uom_qty':1}),
                       (0,0,{'product_id': product_ids[1],
                             'product_uom_qty':2}),
                      ],

    })
    so_model.action_button_confirm([order_id,])
```

- Transaction around the whole method VS each call

```
@api.model
def create_order(self):
    partner_model = self.env['res.partner']
    so_model = self.env['sale.order']
    product_model = self.env['product.product']

    partners = partner_model.search([('name', 'ilike', 'fletcher')])
    products = product_model.search([('name', 'ilike', 'ipad')])

    order = so_model.create({
        'partner_id': partners.id,
        'order_line': [(0,0,{'product_id': products[0].id,
                             'product_uom_qty':1}),
                       (0,0,{'product_id': products[1].id,
                             'product_uom_qty':2}),
                      ],

    })
    order.action_button_confirm()
```

- What about an error between the SO creation and the confirmation ?

Write pictures

- Encode the picture file content in base64
- Write it as a regular field

```
with open(localfile, "rb") as image_file:
    encoded_string = base64.b64encode(image_file.read())
return encoded_string
```

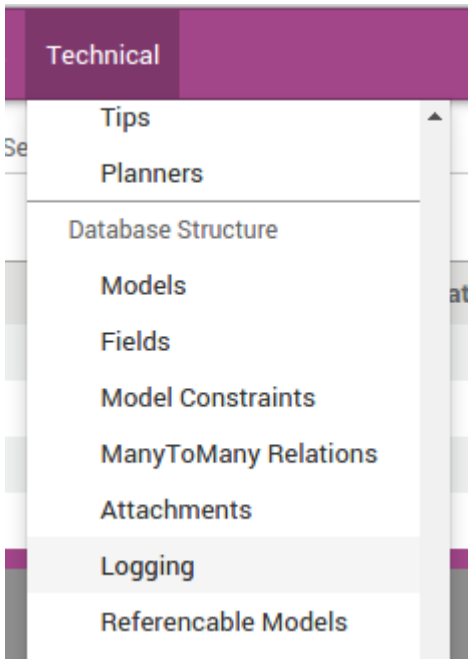
odoo

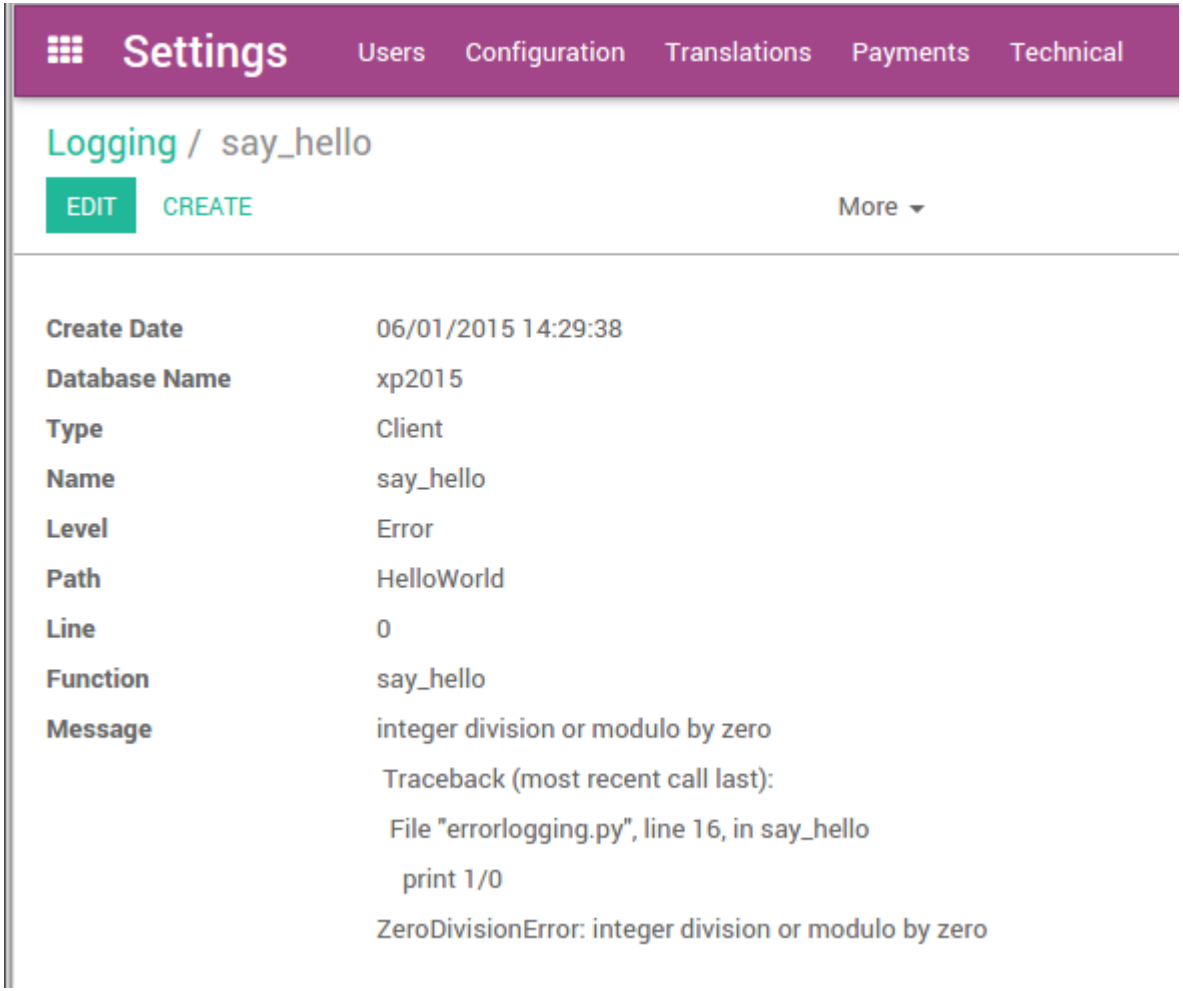
Create pretty much anything, even fields

```
3  class Partner():
4      def add_field(self, connection):
5          model_model = connection.get_model('ir.model')
6          field_model = connection.get_model('ir.model.fields')
7
8          model_id = model_model.search([('model', '=', 'res.partner')])
9          field_id = field_model.create({
10             'name': 'x_demo',
11             'field_description': 'Demo field',
12             'model': 'res.partner',
13             'model_id': model_id[0],
14             'ttype': 'char',
15             'state': 'manual',
16         })
17
18         partner_model = connection.get_model('res.partner')
19         partner_ids = partner_model.search([('name', 'ilike', 'fletcher')])
20         partner_model.write(partner_ids, {'x_demo': 'xperience 2015'})
21         partners = partner_model.read(partner_ids, ['x_demo', 'name'])
22         for partner in partners:
23             print '%s : %s' % (partner['name'], partner['x_demo'])
24
```

Error logging ?

- ir.logging model
- Readable on the instance





odoo

Error logging !

```
1  import openerplib
2  import traceback
3
4  class HelloWorld():
5      def say_hello(self, connection):
6          partner_model = connection.get_model('res.partner')
7          try:
8              partner_ids = partner_model.search([('name', 'ilike', 'fletcher')])
9              partners = partner_model.read(partner_ids, ['name', 'parent_id'])
10             for partner in partners:
11                 print partner
12                 res = "Hello %s" % partner['name']
13                 if partner['parent_id']:
14                     res = "%s from %s" % (res, partner['parent_id'][1])
15                 print res
16                 print 1/0
17
18         except Exception, e:
19             tb = traceback.format_exc()
20             log_model = connection.get_model('ir.logging')
21             datadict = {
22                 'name': "say_hello",
23                 'type': 'client',
24                 'dbname': 'xp2015',
25                 'level': 'Error',
26                 'message': "%s \r\n %s" % (unicode(e), tb),
27                 'path': "HelloWorld",
28                 'func': "say_hello",
29                 'line': "0",
30             }
31             log_model.create(datadict)
```

odoo

Advanced

CSV, multithread

Read using CSV

```python
import openerplib

class Products():
    def read_using_csv(self, connection):
        product_model = connection.get_model('product.product')

        product_ids = product_model.search([('name', 'ilike', 'ipad')])

        products = product_model.export_data(product_ids, ['id', 'name', 'list_price'])
        for product in products['datas']:
            print product
```

```
$ python readcsv.py
[u'product.product_product_6', u'iPad Mini', u'320.0']
[u'product.product_product_4', u'iPad Retina Display', u'750.0']
[u'product.product_product_4b', u'iPad Retina Display', u'750.0']
[u'product.product_product_4c', u'iPad Retina Display', u'750.0']
```

- XML ID instead of ID
- List of lists instead of list of dicts
- Read linked records (O2M, M2M, M20)

odoo

Write using CSV

```python
import openerplib

class Products():
    def update_using_csv(self, connection):
        product_model = connection.get_model('product.product')

        product_ids = product_model.search([('name', 'ilike', 'ipad')])

        columns = ['id', 'list_price']
        lines = []

        products = product_model.export_data(product_ids, ['id', 'name', 'list_price'])
        for product in products['datas']:
            lines.append([product[0], float(product[2])*1.1])
        product_model.load(columns, lines)

```

- Write a list of lists instead of a dict
- Faster than calling write multiple times
- If a line failed, the whole « load » is rolled back
- Write on linked fields

odoo

Sliced reading

```
1  import openerplib
2
3  class Products():
4      def read_using_csv(self, connection):
5          slicing = 1000
6          product_model = connection.get_model('product.product')
7
8          product_ids = product_model.search([])
9
10         product_list = []
11         # To avoid the "CPU time limit exceeded.",
12         # we slice the records to export in blocks
13         begin=0
14         while begin<len(product_ids):
15             products = product_model.export_data(product_ids[begin:(begin+slicing)],
16                 ['id', 'name', 'list_price'])
17             product_list += products['datas']
18             begin+=slicing
19
20         print product_list
```

- Avoid the « CPU time limit exceeded. »
- Read per ~1k records, depending on read fields (O2M, M2O, M2M)

odoo

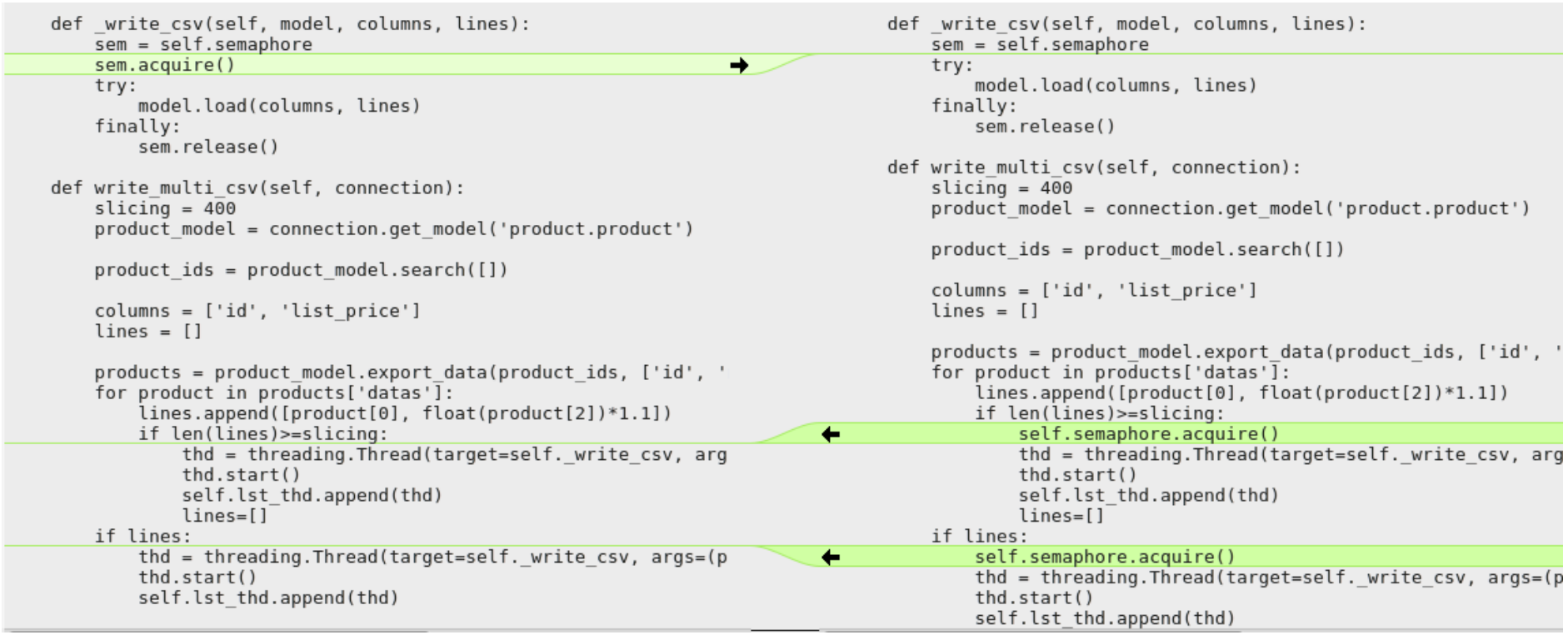
Multi thread write

```
class Products():
    max_connections = 3
    semaphore = threading.BoundedSemaphore(max_connections)
    lst_thd = []

    def _write_csv(self, model, columns, lines):
        sem = self.semaphore
        sem.acquire()
        try:
            model.load(columns, lines)
        finally:
            sem.release()

    def write_multi_csv(self, connection):
        slicing = 400
        product_model = connection.get_model('product.product')

        product_ids = product_model.search([])

        columns = ['id', 'list_price']
        lines = []

        products = product_model.export_data(product_ids, ['id', 'name', 'list_price'])
        for product in products['datas']:
            lines.append([product[0], float(product[2])*1.1])
            if len(lines)>=slicing:
                thd = threading.Thread(target=self._write_csv, args=(product_model, columns, copy.deepcopy(lines)))
                thd.start()
                self.lst_thd.append(thd)
                lines=[]
        if lines:
            thd = threading.Thread(target=self._write_csv, args=(product_model, columns, lines))
            thd.start()
            self.lst_thd.append(thd)

        for thd in self.lst_thd:
            thd.join()
```

Multi thread write

- Write ~400 records per thread
- Better writing rate
- Write on linked fields
- Harder to manage errors
- Avoid using all Odoo workers → 1 thread = 1 Worker
- 1 failing line = the whole thread is rolled back
- If many fields, or linked fields, or binary fields, take care of CPU time → lower the number of records
- Avoid too many threads on the script running machine → change the place of the semaphore

Multi thread write/read



```
def _write_csv(self, model, columns, lines):
    sem = self.semaphore
    sem.acquire()
    try:
        model.load(columns, lines)
    finally:
        sem.release()

def write_multi_csv(self, connection):
    slicing = 400
    product_model = connection.get_model('product.product')

    product_ids = product_model.search([])

    columns = ['id', 'list_price']
    lines = []

    products = product_model.export_data(product_ids, ['id', '
    for product in products['datas']:
        lines.append([product[0], float(product[2])*1.1])
        if len(lines)>=slicing:
            thd = threading.Thread(target=self._write_csv, arg
            thd.start()
            self.lst_thd.append(thd)
            lines=[]
    if lines:
        thd = threading.Thread(target=self._write_csv, args=(p
        thd.start()
        self.lst_thd.append(thd)
```

```
def _write_csv(self, model, columns, lines):
    sem = self.semaphore
    try:
        model.load(columns, lines)
    finally:
        sem.release()

def write_multi_csv(self, connection):
    slicing = 400
    product_model = connection.get_model('product.product')

    product_ids = product_model.search([])

    columns = ['id', 'list_price']
    lines = []

    products = product_model.export_data(product_ids, ['id', '
    for product in products['datas']:
        lines.append([product[0], float(product[2])*1.1])
        if len(lines)>=slicing:
            self.semaphore.acquire()
            thd = threading.Thread(target=self._write_csv, arg
            thd.start()
            self.lst_thd.append(thd)
            lines=[]
    if lines:
        self.semaphore.acquire()
        thd = threading.Thread(target=self._write_csv, args=(p
        thd.start()
        self.lst_thd.append(thd)
```

- Main thread waiting writing threads
- Also usable with read (especially with multiple models)

odoo

No track in mail.thread

Improve performance without tracking modifications in mail.thread

```
class Products():
    def update_using_csv(self, connection):
        product_model = connection.get_model('product.template')

        product_ids = product_model.search([])

        columns = ['id', 'list_price']
        lines = []
        lineswithtrack = []

        products = product_model.export_data(product_ids, ['id', 'name', 'list_price'])
        for product in products['datas']:
            lines.append([product[0], float(product[2])*1.1])
            lineswithtrack.append([product[0], float(product[2])*0.9])

        start = time.time()
        product_model.load(columns, lineswithtrack)
        middle = time.time()
        product_model.load(columns, lines, context={'tracking_disable': True})
        end = time.time()

        print middle - start
        print end - middle
```

odoo

odoo

Interface other softwares

Some real world examples

Ex 1 : Export pickings

- Medium size company
- Carrier with on-site software
- Must generate text files on disk
- Odoo on SaaS

Wizard

- ✔ Fully integrated
- ✖ Manual trigger
- ✔ Easy use
- ✖ Manual save of file
- ✔ Easy maintenance

XML RPC

- ✔ Fully automatic
- ✖ Harder maintenance
- ✖ Delay due to cron
- ✔ Transparent for user

Ex 1 : Export pickings

- Add custom fields for export date
- Customize barcode interface for packs
- Check weight of all products

odoo

Ex 2 : EDIFACT link

- Small size company
- Customers with EDIFACT
- EDI platform require local windows software
- Odoo on SaaS

Wizard	XML RPC
✔ Same interface	✔ No matter the location of the server
✖ No direct access to disk	✖ Harder maintenance
✔ Easy maintenance	✖ Code readability
✔ Easier « on premise »	✔ Do not break the server in case of error
	✔ Transparent for user

odoo

Ex 3 : Pictures from web service

- Free IceCat
- Based on Barcode
- Odoo on SaaS
    - Cool e-commerce in minutes
    - Cool POS in minutes

- Source code :

    https://bitbucket.org/nseinlet/xperience2015

odoo

odoo

Thank You

Questions ?

Odoo
sales@odoo.com
+32 (0) 2 290 34 90
www.odoo.com

R&D and services office
Chaussée de Namur 40
B-1367 Grand Rosière

Sales office
Avenue Van Nieuwenhuyse 5
B-1160 Brussels