LAMPIRAN

Code Modul Business Scale

Python code pada models.py business scale

```
# -*- coding: utf-8 -*-

from odoo import models, fields, api

class business_scale_partner(models.Model):
    _inherit = 'res.partner'

    # Defined attribute business_scale on python
    person_scale = fields.Selection(
        selection=[('personal', 'Personal'), ('small
retail', 'Small Retail')], string="Business Scale",
        default=False, compute='_compute_person_scale',
inverse='_write_person_scale', store=False)

    company_scale = fields.Selection(
        selection=[('large retail', 'Large Retail')],
string="Business Scale",
        default=False, compute='_compute_company_scale',
inverse='_write_company_scale', store=False)

    business_scale = fields.Char()

    @api.depends('business_scale')
    def _compute_person_scale(self):
        for partner in self:
            if partner.business_scale == 'personal':
                partner.person_scale = 'personal'
            elif partner.business_scale == 'small retail':
                partner.person_scale = 'small retail'
            elif partner.business_scale is False:
                partner.company_scale = False
                partner.person_scale = False

    def _write_person_scale(self):
        for partner in self:
            if partner.person_scale == 'personal':
                partner.write({'business_scale':
'personal'})
            elif partner.person_scale == 'small retail':
                partner.write({'business_scale': 'small
retail'})
            elif partner.person_scale is False and
partner.company_scale is False:
                partner.business_scale = False

    @api.depends('business_scale')
    def _compute_company_scale(self):
        for partner in self:
```

```
            if partner.business_scale == 'large retail':
                partner.company_scale = 'large retail'
            elif partner.business_scale is False:
                partner.person_scale = False
                partner.company_scale = False

    def _write_company_scale(self):
        for partner in self:
            if partner.company_scale == 'large retail':
                partner.write({'business_scale': 'large
retail'})
                print(partner['business_scale'])
            elif partner.company_scale is False and
partner.person_scale is False:
                partner.business_scale = False
```

Code Modul Delivery Method

1. Python code backend.py

```
# -*- coding: utf-8 -*-

import json

import http.client
from odoo.exceptions import UserError


class ApiRequest(object):

    def __init__(self, api_key):
        self.api_key = api_key

    def get_list_city(self):
        conn =
http.client.HTTPSConnection("api.rajaongkir.com")
        headers = {'key': self.api_key}
        conn.request("GET", "/starter/city?id=&province=",
headers=headers)

        res = conn.getresponse()
        raw = res.read()
        data = json.loads(raw.decode('utf-8'))
        conn.close()
        if None in data:
            raise UserError('Please, fill form delivery
method correctly')
        else:
            # Slicing data rajaongkir/results/ to get
results data city keep in my_dict
            list_city = data['rajaongkir']['results']

        return list_city
```

```
    def get_city_id_from_list_city(self, origin,
destination):
        list_city = self.get_list_city()
        for data in list_city:
            if origin in data['city_name']:
                origin_id = data['city_id']
                break
        else:
            raise UserError('Origin City not found')
        for data in list_city:
            if destination in data['city_name']:
                destination_id = data['city_id']
                break
        else:
            raise UserError('Delivery Address not found')
        temp = [origin_id, destination_id]
        return temp

    def _get_cost_rajaongkir(self, origin, destination,
weight, courier, service):
        if weight <= 30000:
            price = 0.0
            conn =
http.client.HTTPSConnection("api.rajaongkir.com")

            payload = "origin=" + str(origin) +
"&destination=" + str(destination) + "&weight=" + str(
                weight) + "&courier=" + str(courier)

            headers = {
                'key': self.api_key,
                'content-type': "application/x-www-form-
urlencoded"
            }

            conn.request("POST", "/starter/cost", payload,
headers)

            res = conn.getresponse()
            raw = res.read()
            data = json.loads(raw.decode('utf-8'))
            conn.close()

            if None in data:
                raise UserError('Please, fill form delivery
method correctly')
            else:
                # There is an array in results, the object
name is 'costs', Slicing data with [0] ex:
rajaongkir/results/[0]
                # to get access data costs keep in my_dict
                my_dict = data['rajaongkir']['results'][0]
                for data in my_dict['costs']:
                    if service in data['service']:
                        cost = data['cost'][0]
                        price = cost['value']
```

```
                        break
                    else:
                        price = 0.0
                return price
        else:
            raise UserError('The minimum weight of the
shipment is 30,000 g')
```

2. Python code pada models.py

```
# -*- coding: utf-8 -*-

from odoo import models, fields, api
from odoo.addons.delivery_carrier_rajaongkir.models.backend
import ApiRequest
from odoo.exceptions import UserError


class RajaongkirProvider(models.Model):

    _inherit = 'delivery.carrier'

    api_key = fields.Char(string="API Key")
    partner_id = fields.Many2one('res.partner',
string='Partner', store=True)
    service_jne = fields.Selection(selection=[('OKE',
'Ongkos Kirim Ekonomis'), ('REG', 'Layanan Reguler'),
                                              ('SPS', 'Super
Speed'), ('YES', 'Yakin Esok Sampai'),
                                              ('CTC', 'JNE
City Courier'), ('CTCYES', 'JNE City Courier YES')],
                                    default=False,
string="Service Type")
    service_pos = fields.Selection(selection=[('Paket Kilat
Khusus', 'Paket Kilat Khusus'), ('Express Next Day Barang',
'Express Next Day Barang'),
                                              ('Paketpos
Dangerous Goods', 'Paketpos Dangerous Goods'), ('Paketpos
Valuable Goods', 'Paketpos Valuable Goods'),
                                              ('Express
Sameday Barang', 'Express Sameday Barang')],
                                    default=False,
string="Service Type")
    delivery_type = fields.Selection(selection_add=[('jne',
'JNE'), ('pos', 'POS')])

    @api.onchange('delivery_type')
    def onchange_delivery_type(self):
        if self.delivery_type is 'jne':
            self.service_pos = False
        elif self.delivery_type is 'pos':
            self.service_jne = False
        else:
            self.service_jne = False
            self.service_pos = False
```

```
    def _get_total_weight_from_order_line(self, order):
        self.ensure_one()
        weight = 0
        for line in order.order_line:
            if line.state == 'cancel':
                continue
            if not line.product_id or line.is_delivery:
                continue
            qty =
line.product_uom._compute_quantity(line.product_uom_qty,
line.product_id.uom_id)
            weight += (line.product_id.weight or 0.0) * qty

        # Conversion kg to gram
        weight = weight * 1000

        return int(weight)


    # ------------------------------------------------ #
    #                   JNE shipping                   #
    # ------------------------------------------------ #

    @api.multi
    def jne_rate_shipment(self, order):
        carrier =
self._match_address(order.partner_shipping_id)
        delivery_address = order.partner_shipping_id['city']
        my_warehouse_location =
order.partner_shipping_id['company_id'][0]['city']
        if not carrier:
            return {'success': False,
                    'price': 0.0,
                    'error_message': ('Error: no matching
grid.'),
                    'warning_message': False}
        try:
            # initialization api_key & delivery_type from
ApiRequestClass
            api_backend = ApiRequest(self.api_key)

            # This method for get city id from JNE
            temp =
api_backend.get_city_id_from_list_city(my_warehouse_location
, delivery_address)

            # This method to get product total weight from
order line
            weight =
self._get_total_weight_from_order_line(order)

            # This method to get cost from jne/pos with
params: origin, destination, weight, type of courier
            price_unit =
api_backend._get_cost_rajaongkir(temp[0], temp[1], weight,
self.delivery_type, self.service_jne)
```

```
        except UserError as e:
            return {'success': False,
                    'price': 0.0,
                    'error_message': e.name,
                    'warning_message': False}

        if price_unit == 0:
            return {'success': False,
                    'price': 0.0,
                    'error_message': 'Shipping service is
not available.',
                    'warning_message': False}
        else:
            if order.company_id.currency_id.id !=
order.pricelist_id.currency_id.id:
                price_unit =
order.company_id.currency_id.with_context(date=order.date_or
der).compute(price_unit,

order.pricelist_id.currency_id)
            return {'success': True,
                    'price': price_unit,
                    'error_message': False,
                    'warning_message': False}

    @api.multi
    def jne_send_shipping(self, pickings):
        res = []
        for p in pickings:
            res = res + [{'exact_price':
p.carrier_id.fixed_price,
                          'tracking_number': False}]
        return res

    @api.multi
    def jne_get_tracking_link(self, picking):
        return False

    @api.multi
    def jne_cancel_shipment(self, pickings):
        raise NotImplementedError()

    # ------------------------------------------------ #
    #                   POS shipping                   #
    # ------------------------------------------------ #

    @api.multi
    def pos_rate_shipment(self, order):
        carrier =
self._match_address(order.partner_shipping_id)
        delivery_address = order.partner_shipping_id['city']
        my_warehouse_location =
order.partner_shipping_id['company_id'][0]['city']
        if not carrier:
            return {'success': False,
```

```
                    'price': 0.0,
                    'error_message': ('Error: no matching
grid.'),
                    'warning_message': False}
        try:
            # initialization api_key & delivery_type from
ApiRequestClass
            api_backend = ApiRequest(self.api_key)

            # This method for get city id from JNE
            temp =
api_backend.get_city_id_from_list_city(my_warehouse_location
, delivery_address)

            # This method to get product total weight from
order line
            weight =
self._get_total_weight_from_order_line(order)

            # This method to get cost from jne/pos with
params: origin, destination, weight, type of courier
            price_unit =
api_backend._get_cost_rajaongkir(temp[0], temp[1], weight,
self.delivery_type, self.service_pos)
        except UserError as e:
            return {'success': False,
                    'price': 0.0,
                    'error_message': e.name,
                    'warning_message': False}

        if price_unit == 0:
            return {'success': False,
                    'price': 0.0,
                    'error_message': 'Shipping service is
not available.',
                    'warning_message': False}
        else:
            if order.company_id.currency_id.id !=
order.pricelist_id.currency_id.id:
                price_unit =
order.company_id.currency_id.with_context(date=order.date_or
der).compute(price_unit,

order.pricelist_id.currency_id)
            return {'success': True,
                    'price': price_unit,
                    'error_message': False,
                    'warning_message': False}

    @api.multi
    def pos_send_shipping(self, pickings):
        res = []
        for p in pickings:
            res = res + [{'exact_price':
p.carrier_id.fixed_price,
                          'tracking_number': False}]
```

```
        return res

    @api.multi
    def pos_get_tracking_link(self, picking):
        return False

    @api.multi
    def pos_cancel_shipment(self, pickings):
        raise NotImplementedError()
```

Code Modul Depresiasi

1. Method dibawah digunakan untuk menghitung seluruh baris depresiasi.

```
@api.multi
def compute_depreciation_board(self):
    self.ensure_one()

    posted_depreciation_line_ids =
self.depreciation_line_ids.filtered(lambda x:
x.move_check).sorted(key=lambda l: l.depreciation_date)
    unposted_depreciation_line_ids =
self.depreciation_line_ids.filtered(lambda x: not
x.move_check)

    # Remove old unposted depreciation lines. We cannot
use unlink() with One2many field
    commands = [(2, line_id.id, False) for line_id in
unposted_depreciation_line_ids]

    if self.value_residual != 0.0:
        amount_to_depr = residual_amount =
self.value_residual
        price_date = datetime.strptime(self.date[:10],
DF).date()

        # if we already have some previous validated
entries, starting date isn't now but last entry + method
period
        if posted_depreciation_line_ids and
posted_depreciation_line_ids[-1].depreciation_date:
            last_depreciation_date =
datetime.strptime(posted_depreciation_line_ids[-
1].depreciation_date,

DF).date()
            depreciation_date = last_depreciation_date +
relativedelta(days=+self.method_period)
        else:
            depreciation_date = price_date

        day = depreciation_date.day
        month = depreciation_date.month
        year = depreciation_date.year
```

```
            total_days = (year % 4) and 365 or 366

            undone_dotation_number = self.method_number

            for x in
    range(len(posted_depreciation_line_ids),
    undone_dotation_number):
                sequence = x + 1
                amount = self._compute_depr_amount(sequence,
    residual_amount, amount_to_depr, undone_dotation_number,
    posted_depreciation_line_ids)
                amount = self.currency_id.round(amount)
                if float_is_zero(amount,
    precision_rounding=self.currency_id.rounding):
                    continue
                residual_amount -= amount
                vals = {
                    'amount': amount,
                    'price_id': self.id,
                    'sequence': sequence,
                    'name': (self.code or '') + '/' +
    str(sequence),
                    'remaining_value': residual_amount,
                    'depreciated_value': self.value -
    (self.salvage_value + residual_amount),
                    'depreciation_date':
    depreciation_date.strftime(DF),
                }
                commands.append((0, False, vals))
                # Considering Depr. Period as days
                depreciation_date = date(year, month, day) +
    relativedelta(days=+self.method_period)
                day = depreciation_date.day
                month = depreciation_date.month
                year = depreciation_date.year

        self.write({'depreciation_line_ids': commands})

        return True
```

2. Method dibawah untuk pengambilan harga dari public price.

```
pricelist_item = fields.Many2one(
    'product.pricelist.item', 'Product',
    domain="[('product_id.type', 'in', ['product',
'consu'])]", required=True, readonly=True, states={'draft':
[('readonly', False)]})

@api.onchange('pricelist_item')
def onchange_account_asset(self):
    for record in self:
            val = record.pricelist_item.fixed_price
            self.value = val
```

3. Ketika journal telah diposting maka method ini akan menurunkan harga dari public price

```
@api.multi
def compute_price_when_posted(self, depreciation):
    for price in self:
        val = price.pricelist_item.fixed_price
        val = val - depreciation
        price.pricelist_item.write({'fixed_price': val})
```

4. Method ini untuk pengiriman notifikasi pada Module Discuss.

```
@api.model
def _prepare_msg_move(self):
    msg_values = {
        'res_id': self.env.ref(

'product_depreciation_price.channel_depreciation_product').i
d,
        'model': 'mail.channel',
        'subject': _('Product Deprecation Price
Notification'),
        'message_type': 'comment',
        'subtype_id': self.env.ref('mail.mt_comment').id}
    return msg_values

@api.model
def send_move_mail(self, depreciation):
    msg_values = self._prepare_msg_move()
    self[0].message_post_with_view(

'product_depreciation_price.message_product_depreciation',
        values={'self': self,
                'origin': depreciation},
        **msg_values)
```