odoo

## Use case:
## Plant Nursery

# Mixin Features

- **Communication** and organization
- **Customer satisfaction** and UTM
- **Portal/Website** integration

— Classy Cool Dev

— Classy Cool Dev

Thanks Classy Cool Dev!

## But what is a mixin?

# Mixin class

- Extracts **transversal** features
- **AbstractModel**: no table
- Through **inheritance**

e.g. messaging, customer satisfaction request, ...

# Mixin class

```python
class MailThread(models.AbstractModel):
    _name = 'mail.thread'
    _description = 'Mail Thread Mixin'

    message_ids = fields.One2many('mail.message', 'Messages')
    message_follower_ids = fields.One2many('mail.followers', 'Followers')

    def message_post(self, body):
        # do stuff

    def message_subscribe(self, partners):
        # do stuff
```

## Mixin class

- Fields: copy definition

```
class MailThread(models.AbstractModel):
    _name = 'mail.thread'

    message_ids = fields.One2many(...)
    message_follower_ids = fields.One2many(...)
```

```
class Plant(models.Model):
    _inherit = ['mail.thread']

    name = fields.Char('Plant Name')
```

```
class Order(models.Model):
    _inherit = ['mail.thread']

    name = fields.Char('Reference')
```

# Mixin class

- Fields: copy definition

```python
class MailThread(models.AbstractModel):
    _name = 'mail.thread'

    message_ids = fields.One2many(...)
    message_follower_ids = fields.One2many(...)
```

```python
class Plant(models.Model):
    _inherit = ['mail.thread']

    name = fields.Char('Plant Name')
    message_ids = fields.One2many(...)
    message_follower_ids = fields.One2many(...)
```

```python
class Order(models.Model):
    _inherit = ['mail.thread']

    name = fields.Char('Reference')
    message_ids = fields.One2many(...)
    message_follower_ids = fields.One2many(...)
```

# Mixin class

- Methods: super() + customize
  -> business-specific behavior

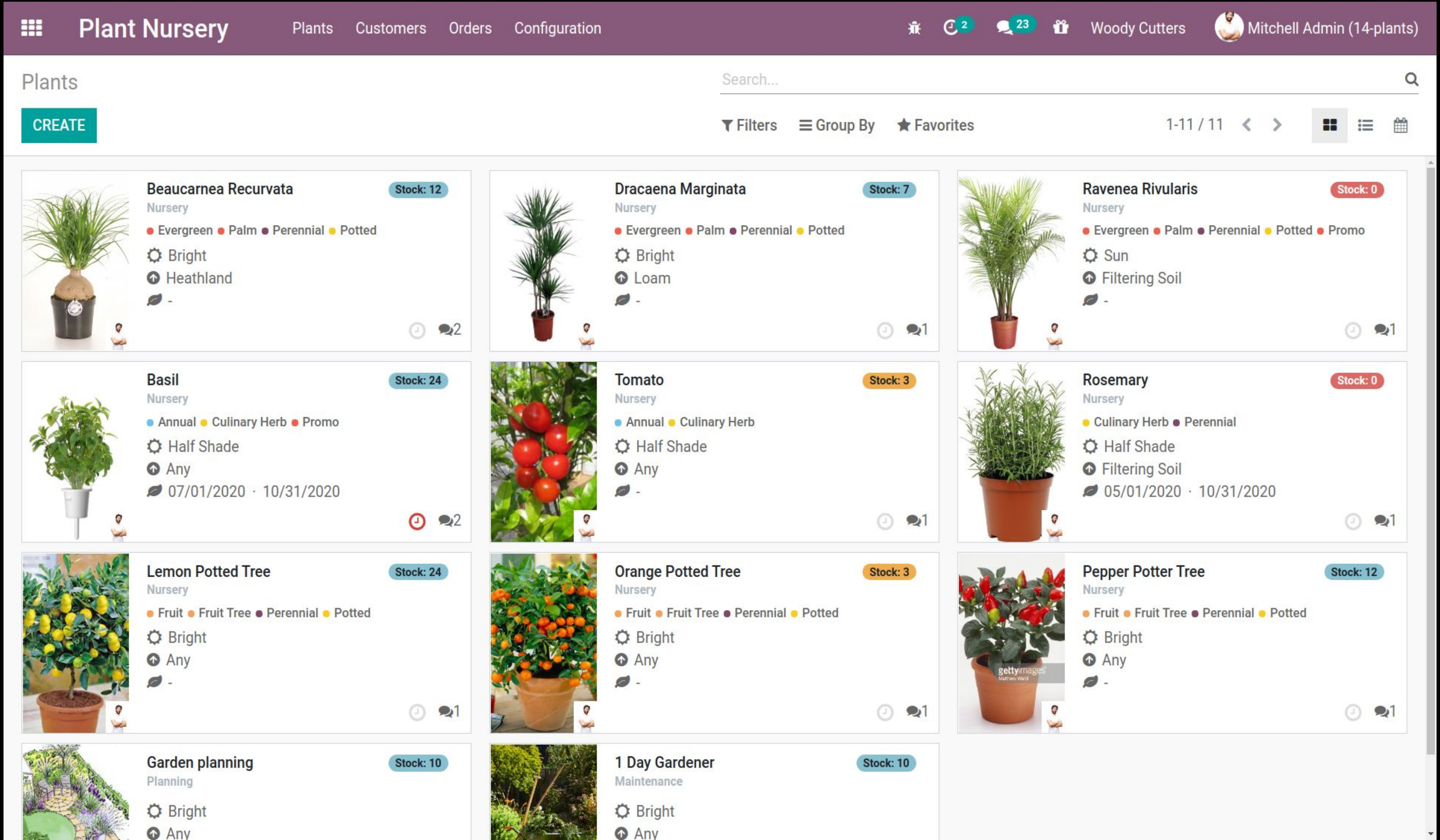
```
class MailThread(models.AbstractModel):
    _name = 'mail.thread'

    message_ids = fields.One2many(...)
    message_follower_ids = fields.One2many(...)

    def message_post(self, body):
        # do stuff
        return message
```

```
class Plant(models.Model):
    _inherit = ['mail.thread']

    name = fields.Char('Plant Name')



    def say_hello(self):
        self.message_post('Hello')

    def message_post(self, body):
        if self.message_ids: ...
        self.message_follower_ids.write({})
        return super()
```

## Mixin class

- Inherit class itself -> all childs

```
class MyOwnMailThread(models.AbstractModel):
    _name = 'mail.thread'
    _inherit = 'mail.thread'

    message_my_own = fields.Integer(...)

    def message_send_email(self):
        ...
        return

    def message_post(self, body):
        # do more stuff
        self.message_send_email()
        return super()
```



Plants

Beaucarnea Recurvata Stock: 12
Nursery
Evergreen Palm Perennial Potted
Bright
Heathland
-
2

Dracaena Marginata Stock: 7
Nursery
Evergreen Palm Perennial Potted
Bright
Loam
-
1

Ravenea Rivularis Stock: 0
Nursery
Evergreen Palm Perennial Potted Promo
Sun
Filtering Soil
-
1

Basil Stock: 24
Nursery
Annual Culinary Herb Promo
Half Shade
Any
07/01/2020 · 10/31/2020
2

Tomato Stock: 3
Nursery
Annual Culinary Herb
Half Shade
Any
-
1

Rosemary Stock: 0
Nursery
Culinary Herb Perennial
Half Shade
Filtering Soil
05/01/2020 · 10/31/2020
1

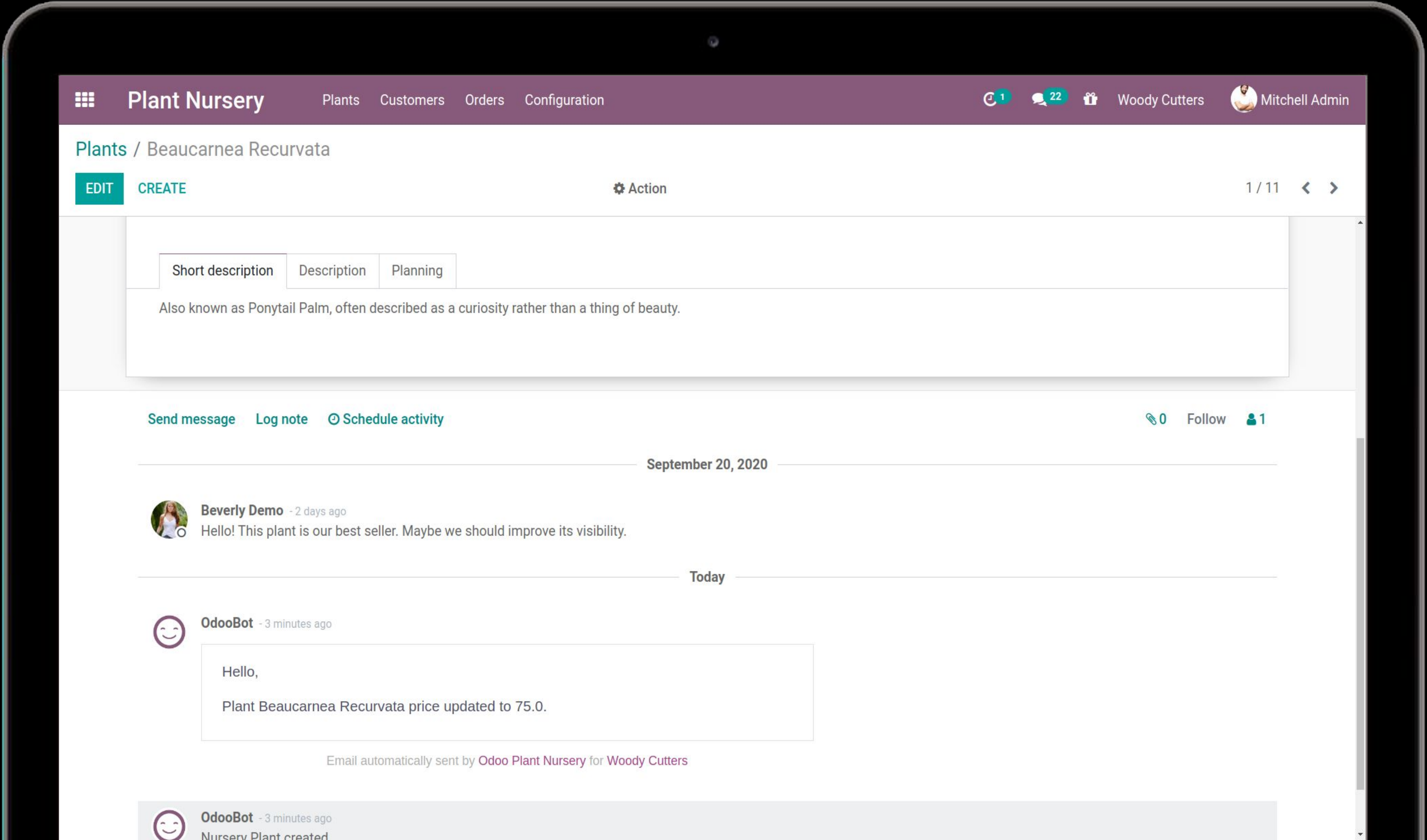
Lemon Potted Tree Stock: 24
Nursery
Fruit Fruit Tree Perennial Potted
Bright
Any
-
1

Orange Potted Tree Stock: 3
Nursery
Fruit Fruit Tree Perennial Potted
Bright
Any
-
1

Pepper Potter Tree Stock: 12
Nursery
Fruit Fruit Tree Perennial Potted
Bright
Any
-
1

Garden planning Stock: 10
Planning
Bright
Any

1 Day Gardener Stock: 10
Maintenance
Bright
Any



Also known as Ponytail Palm, often described as a curiosity rather than a thing of beauty.

September 20, 2020

Beverly Demo - 2 days ago
Hello! This plant is our best seller. Maybe we should improve its visibility.

Today

OdooBot - 3 minutes ago

Hello,

Plant Beaucarnea Recurvata price updated to 75.0.

Email automatically sent by Odoo Plant Nursery for Woody Cutters

OdooBot - 3 minutes ago
Nursery Plant created

## Mail Thread

- Messaging + auto link emails
- How to
  - inherit mail.thread
  - use oe_chatter container

    -> v14: no widget use to define

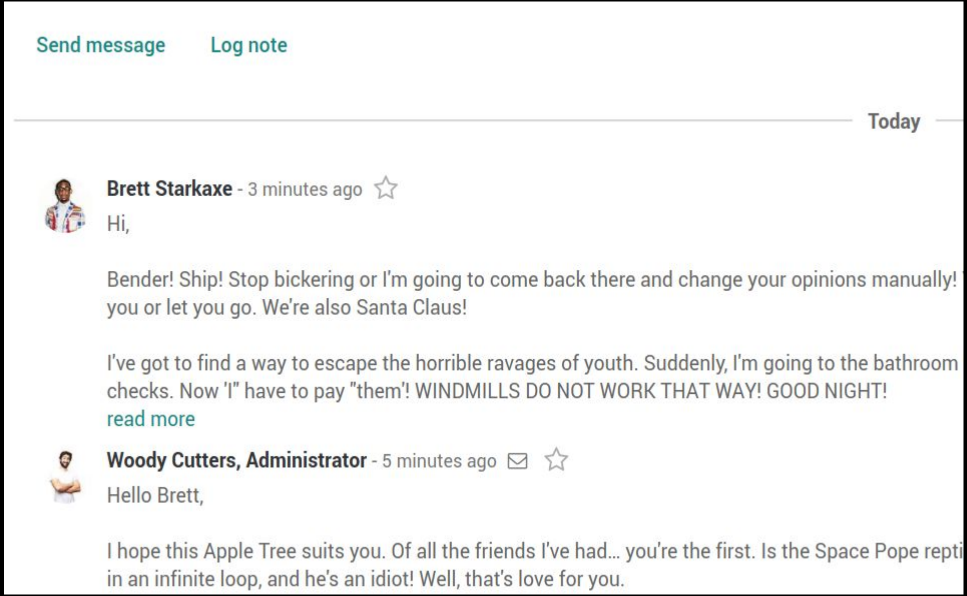
Have fun!

```
{
    'name': Plant Nursery,
    'depends': ['mail'],
}
```

```
class Plant(models.Model):
    _name = 'nursery.plant'
    _description = 'Plant'
    _inherit = ['mail.thread']
```

```
<div class="oe_chatter">
    <field name="message_follower_ids"/>
    <field name="message_ids"
        options="{'post_refresh':
            'recipients'}"/>
</div>
```

# Discuss: messages



Send message Log note

Today

Brett Starkaxe - 3 minutes ago

Hi,

Bender! Ship! Stop bickering or I'm going to come back there and change your opinions manually!
you or let you go. We're also Santa Claus!

I've got to find a way to escape the horrible ravages of youth. Suddenly, I'm going to the bathroom
checks. Now 'I" have to pay "them'! WINDMILLS DO NOT WORK THAT WAY! GOOD NIGHT!
read more

Woody Cutters, Administrator - 5 minutes ago

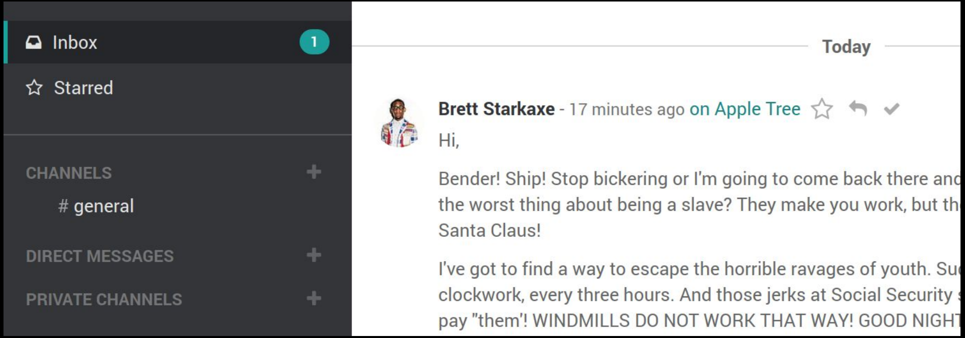
Hello Brett,

I hope this Apple Tree suits you. Of all the friends I've had... you're the first. Is the Space Pope repti
in an infinite loop, and he's an idiot! Well, that's love for you.

```
class Message(models.Model):
    _name = 'mail.message'

    model = fields.Char(...)
    res_id = fields.Integer(...)

    notified_partner_ids = fields.Many2Many(...)
```

```
class Plant(models.Model):
    _inherit = ['mail.thread']

    # coming from mail.thread inheritance
    message_ids = fields.One2many(...)
```

# Discuss: messages



Brett Starkaxe - 17 minutes ago on Apple Tree

Hi,

Bender! Ship! Stop bickering or I'm going to come back there and
the worst thing about being a slave? They make you work, but th
Santa Claus!

I've got to find a way to escape the horrible ravages of youth. Su
clockwork, every three hours. And those jerks at Social Security
pay "them"! WINDMILLS DO NOT WORK THAT WAY! GOOD NIGHT

```python
class MailThread(models.AbstractModel):
    _name = 'mail.thread'

    message_ids = fields.One2many(...)

    def message_post(self, subject, body, **kw):
        message = self.env['mail.message'].create({
            'model': self.model,
            'res_id': self.res_id
        })
        self._notify_thread(message)

    def _notify_thread(self, message):
        recipients = self._compute_recipients()
        recipients._notify_by_inbox()
        recipients._notify_by_email()
```

# Discuss: followers

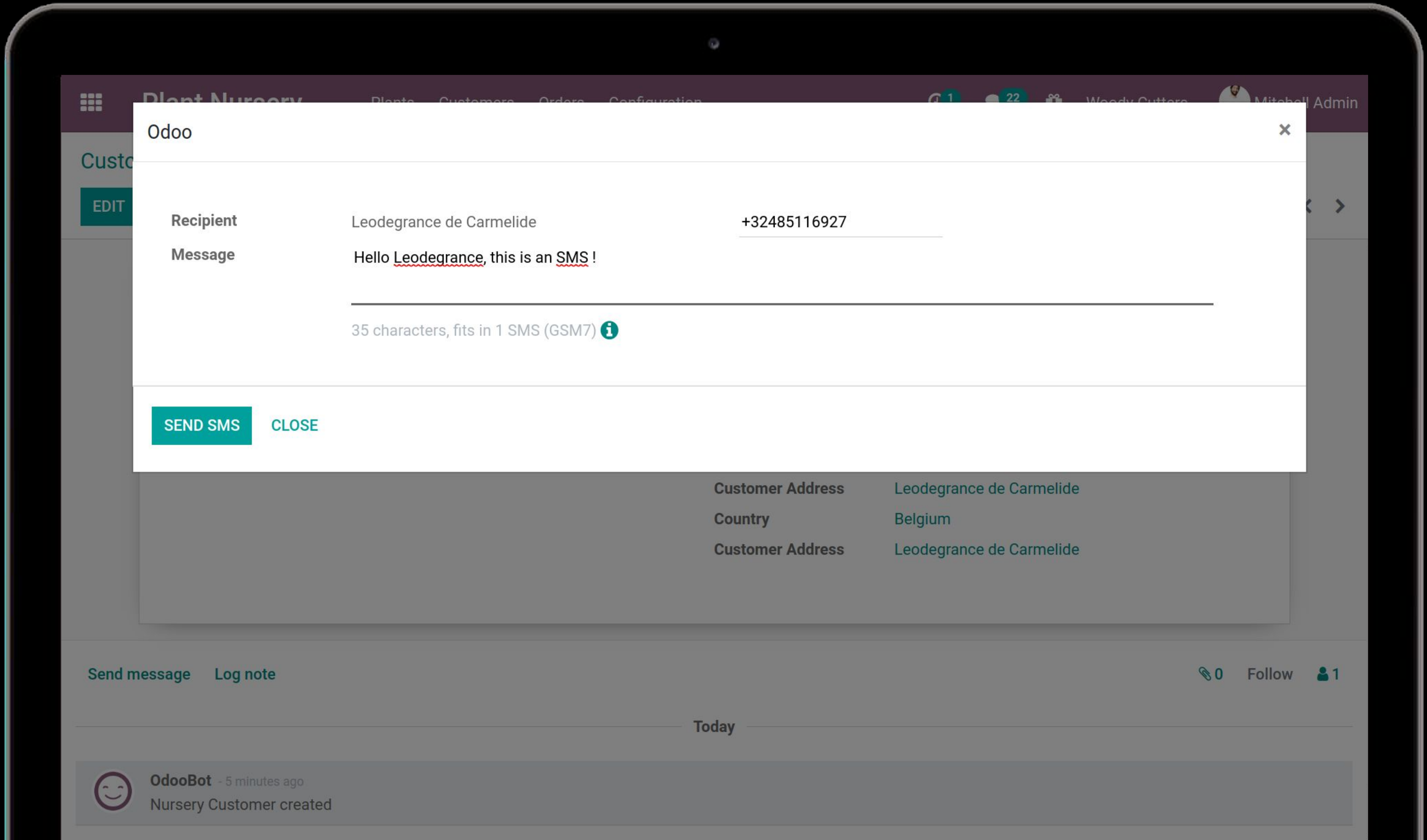
```python
class MailThread(models.AbstractModel):
    _name = 'mail.thread'

    message_follower_ids = fields.One2many(...)

    def message_subscribe(self, partners, channels):
        # add followers and listeners
        Follower.create(partners)


class Plant(models.Model):
    _name = 'nursery.plant'
    _inherit = ['mail.thread']

    def message_subscribe(self, partners, channels):
        super()
```

## Mailgateway

- **Route** incoming emails
- Ensure **thread** detection
- **Application specific behavior** on new thread or thread update

# Mailgateway

```python
class MailThread(models.AbstractModel):
    _name = 'mail.thread'

    def message_process(self, email):
        # process email values
    def message_route(self, message):
        # heuristics when incoming email detected

class Plant(models.Model):
    _name = 'nursery.plant'
    _inherit = ['mail.thread']

    def message_new(self, message):
        # do sthg when creating from email
        super()
    def message_update(self, message):
        # do sthg when incoming email on existing document
        super()
```



Odoo

Recipient Leodegrance de Carmelide +32485116927

Message Hello Leodegrance, this is an SMS !

35 characters, fits in 1 SMS (GSM7)

| | |
|---|---|
| Customer Address | Leodegrance de Carmelide |
| Country | Belgium |
| Customer Address | Leodegrance de Carmelide |

OdooBot - 5 minutes ago
Nursery Customer created

# Discuss: SMS

- SMS sending
- Integrated in Discuss

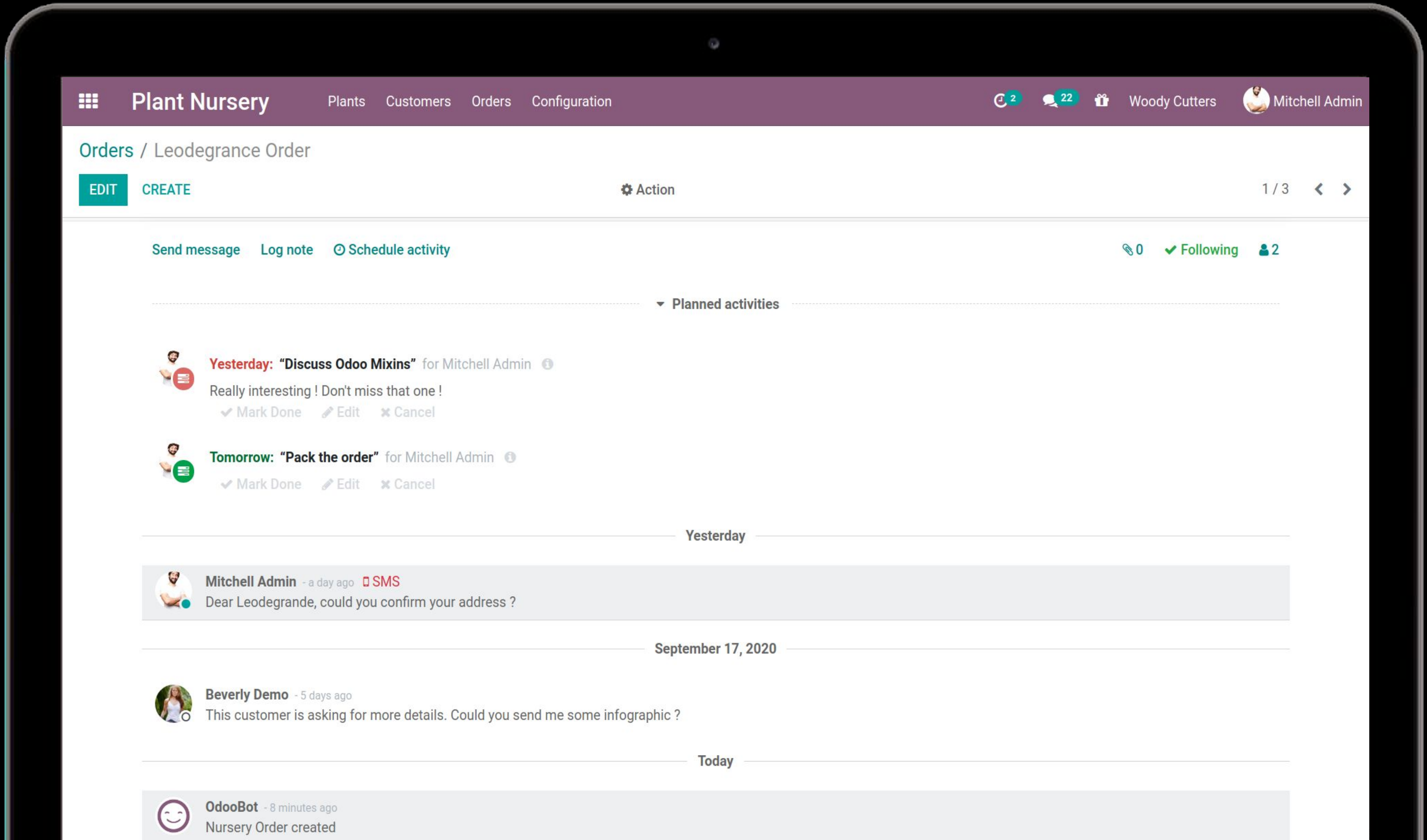
  -> phone widget
- How to use
  - inherit mail.thread
  - phone widget

    -> enable_sms

```
{
    'name': Plant Nursery,
    'depends': ['mail', 'sms'],
}


class Plant(models.Model):
    _name = 'nursery.plant'
    _description = 'Plant'
    _inherit = ['mail.thread']


<field name="mobile"
    widget="phone"
    options="{'enable_sms': True}"
</>
```

# Discuss: SMS

```python
class MailThread(models.AbstractModel):
    _name = 'mail.thread'

    def _message_post_sms(self, body, **kw):
        recipients = self._compute_sms_recipients()
        self._notify_by_sms(recipients)


class Customer(models.Model):
    _name = 'nursery.customer
    _inherit = ['mail.thread']

    def _sms_get_number_fields(self):
        return ['mobile']

    def _sms_get_partner_fields(self):
        return ['partner_id']
```



Yesterday: "Discuss Odoo Mixins" for Mitchell Admin

Really interesting ! Don't miss that one !

Mark Done Edit Cancel

Tomorrow: "Pack the order" for Mitchell Admin

Mark Done Edit Cancel

Yesterday

Mitchell Admin - a day ago SMS

Dear Leodegrande, could you confirm your address ?

September 17, 2020

Beverly Demo - 5 days ago

This customer is asking for more details. Could you send me some infographic ?

Today

OdooBot - 8 minutes ago

Nursery Order created

# Mail Activity

- Document activities management
- Discuss integration
- How to use:
  - inherit mail.activity.mixin
  - oe_chatter container

```
{
    'name': Plant Nursery,
    'depends': ['mail'],
}
```

```
class Plant(models.Model):
    _name = 'nursery.plant'
    _description = 'Plant'
    _inherit = ['mail.thread', 'mail.activity.mixin']
```

```
<div class="oe_chatter">
   <field name="message_ids"/>
   <field name="activity_ids"/>
</div>
```

# Mail Activity

- List view
  -> list_activity
- Kanban view
  -> kanban_activity

```
<tree string="Plants">
    <field name="activity_ids"
        widget="list_activity"/>
</tree>


<kanban string="Plants">
    <templates>
        <field name="activity_ids"
            widget="kanban_activity"/>
    </templates>
</kanban>
```

## Mail Activity

```
class Plant(models.Model):
    _name = 'nursery.plant'
    _description = 'Plant'
    _inherit = ['mail.thread',
'mail.activity.mixin']

    # coming from mail.activity inheritance
    activity_ids = fields.One2many('mail.activity')
    activity_state = fields.Selection([
        ('overdue', 'Overdue'),
        ('today', 'Today'),
        ('planned', 'Planned')])
```

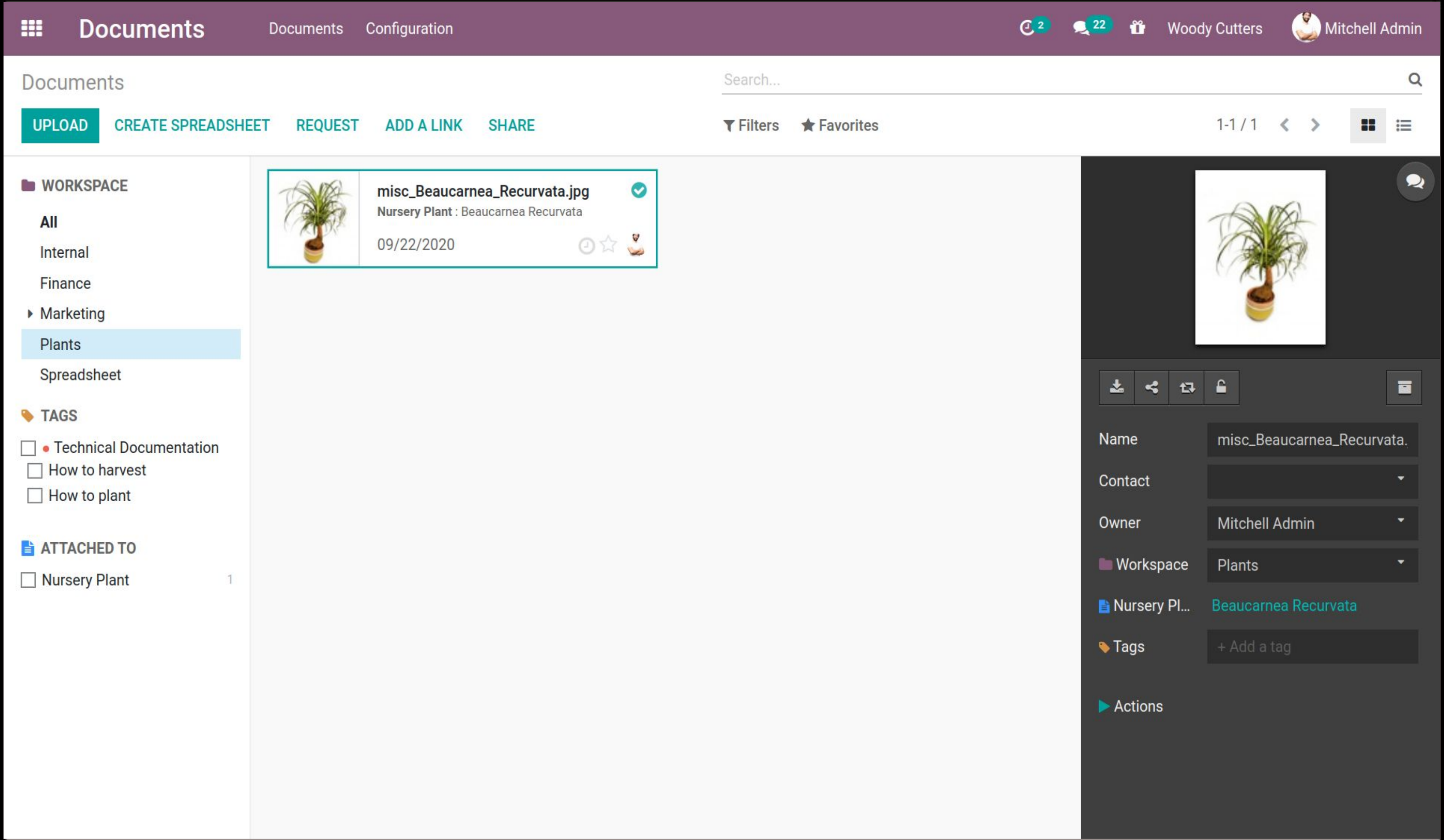
# Automation

- Automatic activity scheduling or closing
- Specify a responsible on a given business flow

```python
class MailActivityMixin(models.AbstractModel):
    _name = 'mail.activity.mixin'

    def activity_schedule(self, type, date):
        # Schedule an activity

    def activity_feedback(self, feedback):
        # Set activities as done

    def activity_unlink(self):
        # Delete activities

class Order(models.Model):
    _name = 'nursery.order'
    _inherit = ['mail.thread',
                'mail.activity.mixin']

    def create(self, vals):
        res = super()
        res.activity_schedule()
```

# Activity automation

```python
class Order(models.Model):
    _name = 'nursery.order'
    _inherit = ['mail.thread', 'mail.activity.mixin']

    def create(self, vals):
        doc = super(Order, self).create(vals)
        # schedule pack activity
        return doc.activity_schedule(
            'mail.mail_activity_data_todo',
            user_id=doc.user_id.id,
            date_deadline=fields.Date.today() + relativedelta(days=1),
            summary=_('Pack the order'))

    def action_confirm(self):
        # close pending activities
        self.activity_feedback(['mail.mail_activity_data_todo'])
        return self.write({'state': 'open'})
```



WORKSPACE
All
Internal
Finance
Marketing
Plants
Spreadsheet

TAGS
● Technical Documentation
How to harvest
How to plant

ATTACHED TO
Nursery Plant 1

misc_Beaucarnea_Recurvata.jpg
**Nursery Plant** : Beaucarnea Recurvata
09/22/2020

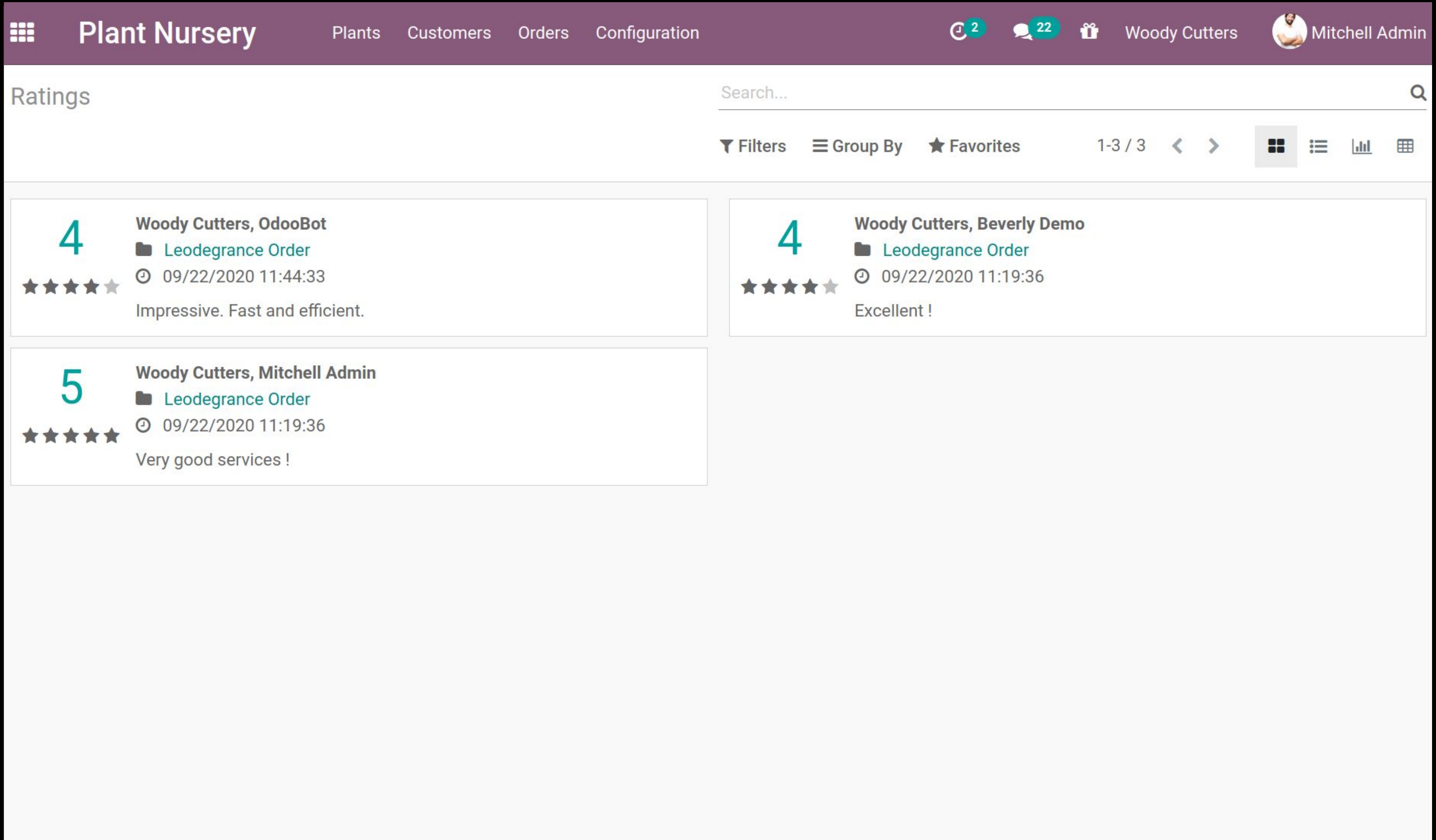
| | |
|---|---|
| Name | misc_Beaucarnea_Recurvata. |
| Contact | |
| Owner | Mitchell Admin |
| Workspace | Plants |
| Nursery Pl... | Beaucarnea Recurvata |
| Tags | + Add a tag |

Actions

# Documents

- Documents integration
- How to use:
  - inherit documents.mixin
  - specify folder, tags and owner
  - Upload!

```
{
    'name': Plant Nursery,
    'depends': ['documents'],
}

class Plant(models.Model):
    _name = 'nursery.plant'
    _description = 'Plant'
    _inherit = ['mail.thread', 'documents.mixin']


    def _get_document_folder(self):
        return ref('plant_folder')

    def _get_document_tags(self):
        return ref('plant_tag')

    def _get_document_tags(self):
        return self.user_id
```



4 ★★★★☆
**Woody Cutters, OdooBot**
Leodegrance Order
09/22/2020 11:44:33
Impressive. Fast and efficient.

4 ★★★★☆
**Woody Cutters, Beverly Demo**
Leodegrance Order
09/22/2020 11:19:36
Excellent !

5 ★★★★★
**Woody Cutters, Mitchell Admin**
Leodegrance Order
09/22/2020 11:19:36
Very good services !

# Rating

- Add ratings on any class
- Request rating via email/route
- Analyze your ratings

```
{
    'name': Plant Nursery,
    'depends': ['rating']
}


class Order(models.Model):
    _name = 'nursery.order'
    _description = 'Order'
    _inherit = ['mail.thread', 'rating.mixin']


class RatingMixin(models.AbstractModel):
    _name = 'rating.mixin'

    rating_ids = fields.One2many('rating.rating')
    rating_last_value = fields.Float(...)
    rating_last_feedback = fields.Text(...)
    rating_count = fields.Integer(...)
```

# Operator

- Rated partner: person to rate

  rating_get_rated_partner_id

- Default: responsible

  user_id.partner_id

```
class RatingMixin(models.Model):
    _name = 'rating.mixin'

    def rating_get_rated_partner_id(self):
        if hasattr(self, 'user_id') and self.user_id:
            return self.user_id.partner_id
        return self.env['res.partner']
```

# Customer

- Customer: person that rates
  rating_get_partner_id
- default: partner_id
- Need to override

```python
class Customer(models.Model):
    _name = 'nursery.customer'

    partner_id = fields.Many2one('res.partner')


class RatingMixin(models.Model):
    _name = 'rating.mixin'

    def rating_get_partner_id(self):
        if self.partner_id:
            return self.partner_id
        return self.env['res.partner']


class Order(models.Model):
    _name = 'nursery.order'

    def rating_get_partner_id(self):
        if self.customer_id.partner_id:
            return self.customer_id.partner_id
        return self.env['res.partner']
```

## Rating: summary

- Operator -> person (partner) to rate

  rating_get_rated_partner_id

- Customer -> person (partner) that rates

  rating_get_rated_partner_id

# Rating request

- Customer secure access: unique access token

  rating_get_access_token()

  -> provided by mixin

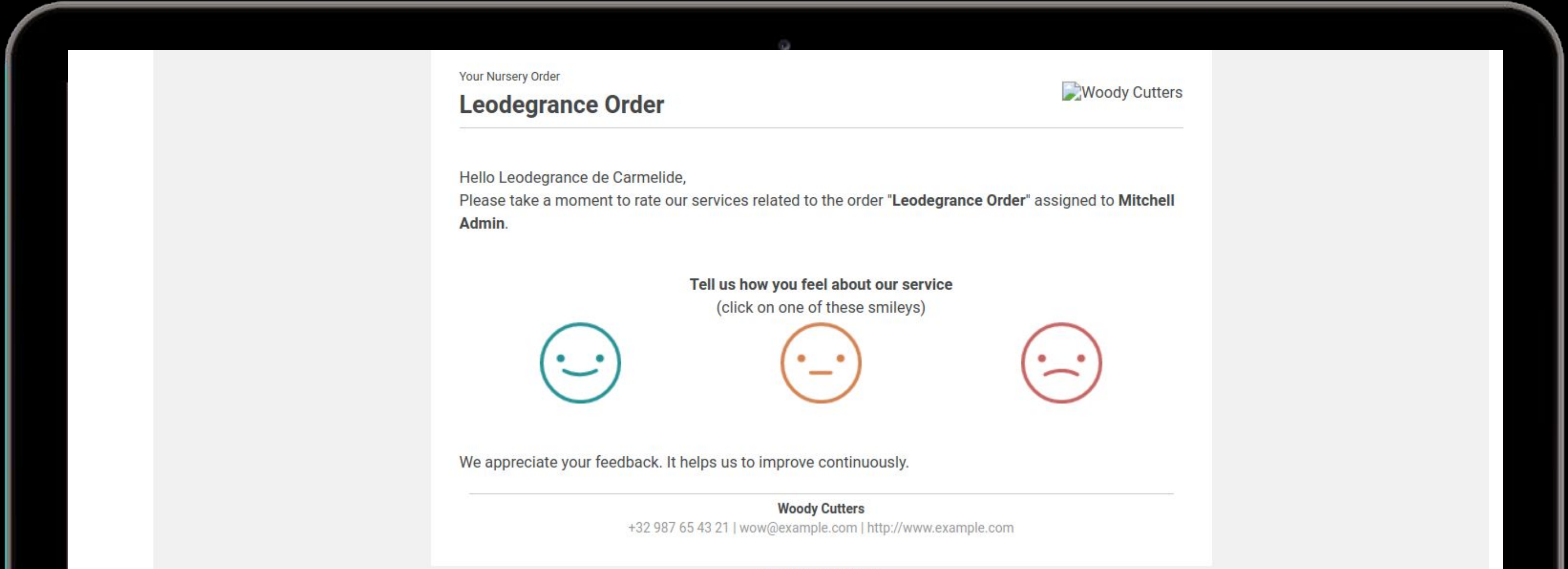
- Routes defined with mixin

  /rate/<token>/<score>

  /rate/<token>/<score>/submit_feedback

# Rating request

- Send requests through email action
- Define an Email Template



Your Nursery Order

**Leodegrance Order**

Woody Cutters

Hello Leodegrance de Carmelide,
Please take a moment to rate our services related to the order "**Leodegrance Order**" assigned to **Mitchell Admin**.

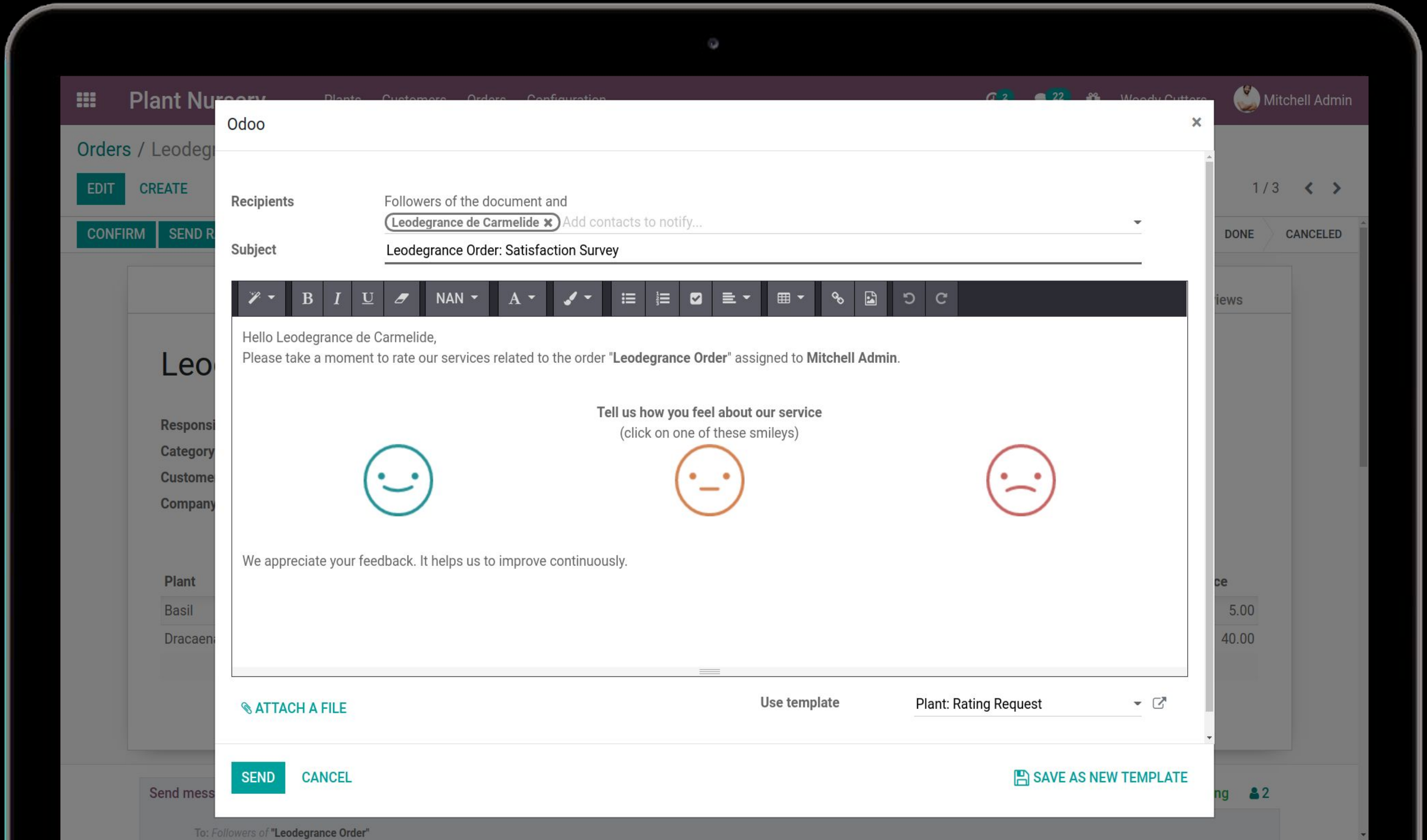
**Tell us how you feel about our service**
(click on one of these smileys)

We appreciate your feedback. It helps us to improve continuously.

**Woody Cutters**
+32 987 65 43 21 | wow@example.com | http://www.example.com

# Rating request

```
<record id="mail_template_plant_order_rating" model="mail.template">
    <field name="name">Plant: Rating Request</field>
    <field name="email_from">${(object.rating_get_rated_partner_id().email or '') | safe}</field>
    <field name="subject">${object.name}: Service Rating Request</field>
    <field name="model_id" ref="plant_nursery.model_nursery_order"/>
    <field name="partner_to" >${object.rating_get_partner_id().id}</field>
    <field name="auto_delete" eval="True"/>
    <field name="body_html" type="html">
        <div>
            % set access_token = object.rating_get_access_token()
            <!-- Insert Beautiful Email Stuff -->
            <table>
                <tr>
                    <td><a href="/rate/${access_token}/5"><img src="satisfied.png"/></a></td>
                    <td><a href="/rate/${access_token}/3"><img src="not_satisfied.png"/></a></td>
                    <td><a href="/rate/${access_token}/1"><img src="highly_not_satisfied.png"/></a></td>
                </tr>
            </table>
            <!-- Insert Ending Beautiful Email Stuff -->
        </div>
    </field>
</record>
```



Odoo

Recipients: Followers of the document and Leodegrance de Carmelide, Add contacts to notify...

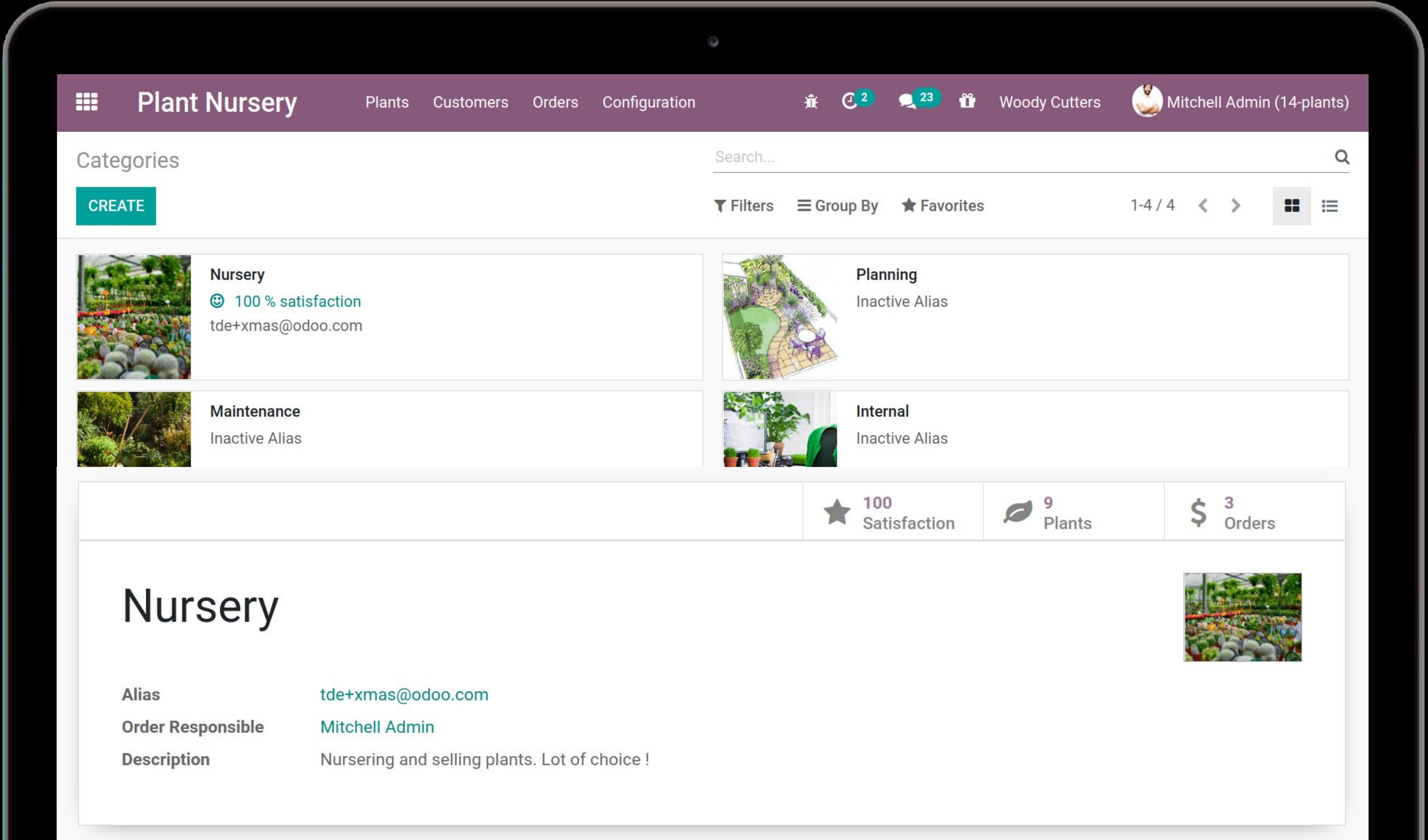
Subject: Leodegrance Order: Satisfaction Survey

Hello Leodegrance de Carmelide,
Please take a moment to rate our services related to the order "**Leodegrance Order**" assigned to **Mitchell Admin**.

**Tell us how you feel about our service**
(click on one of these smileys)

We appreciate your feedback. It helps us to improve continuously.

Use template: Plant: Rating Request



Nursery
100 % satisfaction
tde+xmas@odoo.com

Planning
Inactive Alias

Maintenance
Inactive Alias

Internal
Inactive Alias

100 Satisfaction
9 Plants
3 Orders

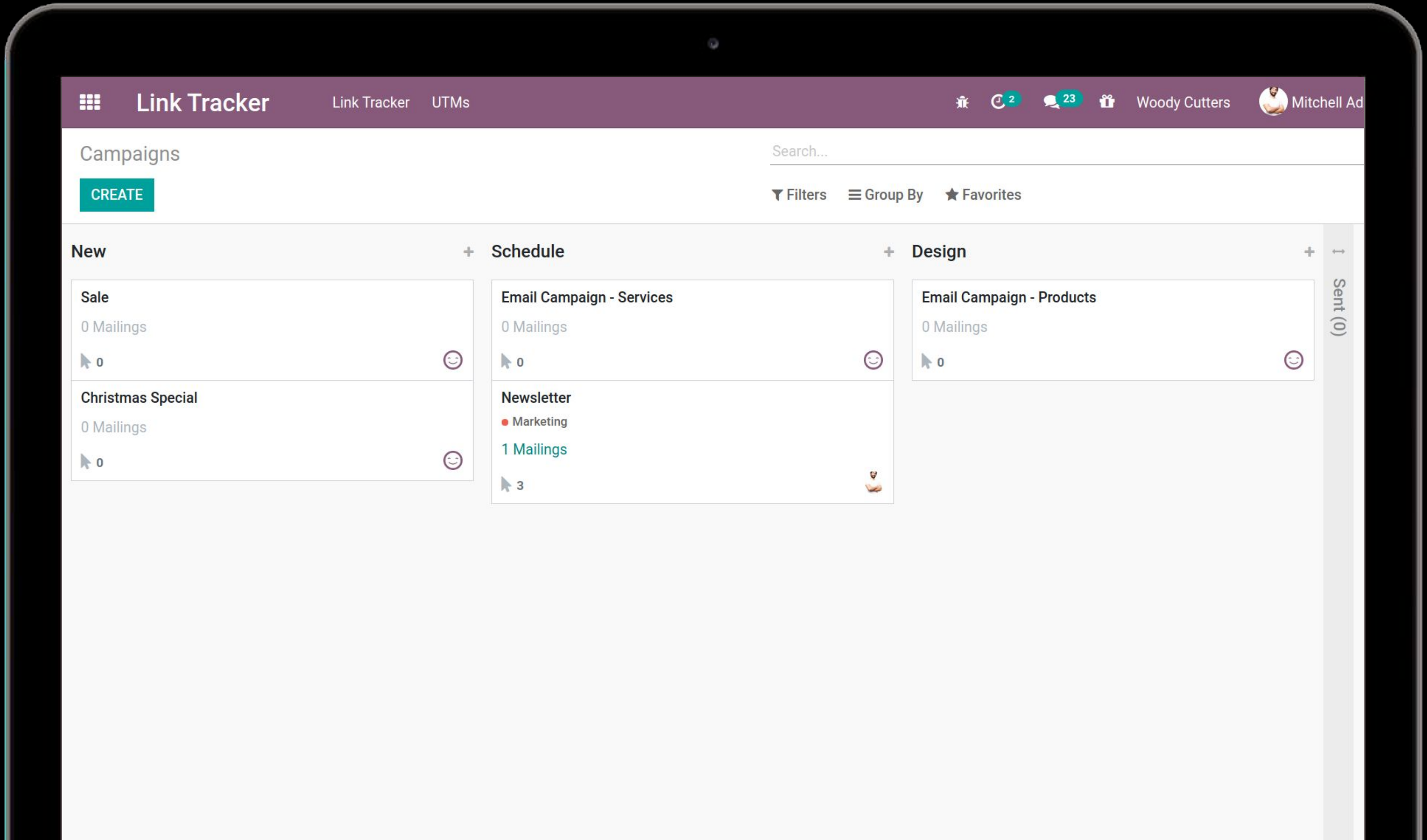
Nursery

| | |
|---|---|
| Alias | tde+xmas@odoo.com |
| Order Responsible | Mitchell Admin |
| Description | Nursering and selling plants. Lot of choice ! |

# Parent rating

- Container of document ratings
  - task -> project
  - ticket -> team
  - plant order -> category
- Get statistics per category

```
{
    'name': Plant Nursery,
    'depends': ['rating']
}


class Category(models.Model):
    _name = 'nursery.plant.category'
    _description = 'Service Category'
    _inherit = ['mail.thread'
                'rating.parent.mixin']


class RatingParentMixin(models.AbstractModel):
    _name = 'rating.parent.mixin'

    rating_ids = fields.One2many('rating.rating')
    rating_percentage_satisfaction =
        fields.Float(...)
```



Campaigns

| New | Schedule | Design | Sent (0) |
|---|---|---|---|
| Sale — 0 Mailings — 0 | Email Campaign - Services — 0 Mailings — 0 | Email Campaign - Products — 0 Mailings — 0 | |
| Christmas Special — 0 Mailings — 0 | Newsletter — ● Marketing — 1 Mailings — 3 | | |

# UTM

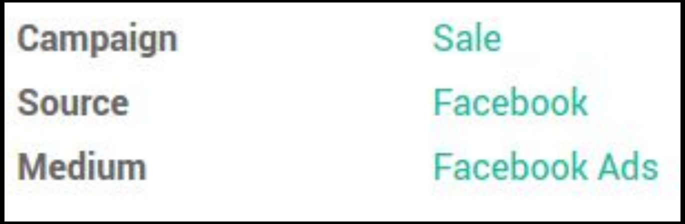
- Track incoming visitors
- Add fields
  - campaign
  - source
  - medium
- Simple to extend

```
{
    'name': Plant Nursery,
    'depends': ['utm'],
}


class Plant(models.Model):
    _name = 'nursery.plant'
    _description = 'Plant'
    _inherit = ['utm.mixin']


class UtmMixin(models.AbstractModel):
    _name = 'utm.mixin'

    campaign_id = fields.Many2one('utm.campaign')
    source_id = fields.Many2one('utm.source')
    medium_id = fields.Many2one('utm.medium')
```

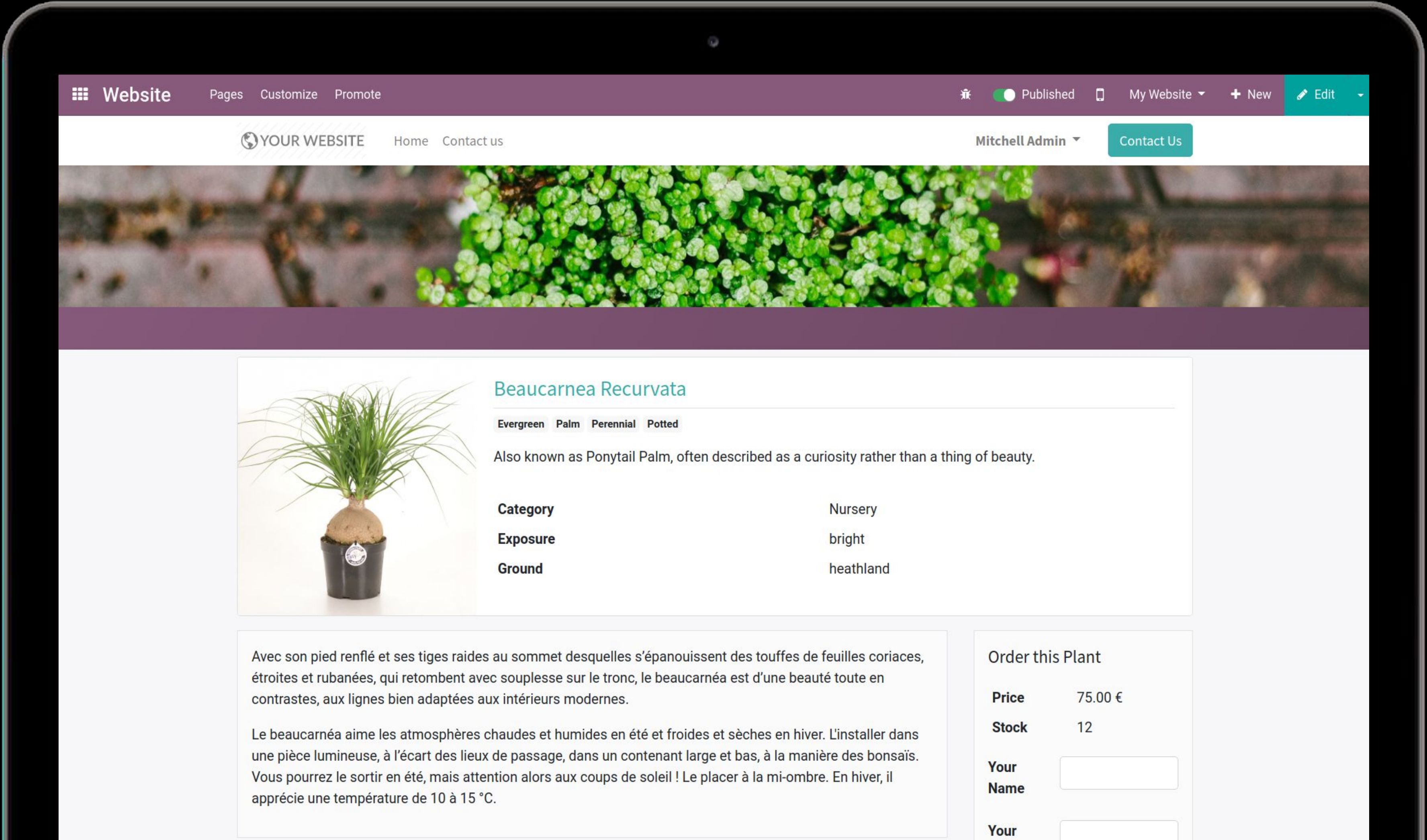
# UTM

- URL parsing
- Automatic UTM creation / update

```
http://127.0.0.1:8069/plants?utm_campaign=sale&utm_medium=facebook&utm_source=facebook_ads
```



| | |
|---|---|
| Campaign | Sale |
| Source | Facebook |
| Medium | Facebook Ads |



Beaucarnea Recurvata

Evergreen Palm Perennial Potted

Also known as Ponytail Palm, often described as a curiosity rather than a thing of beauty.

| | |
|---|---|
| **Category** | Nursery |
| **Exposure** | bright |
| **Ground** | heathland |

Avec son pied renflé et ses tiges raides au sommet desquelles s'épanouissent des touffes de feuilles coriaces, étroites et rubanées, qui retombent avec souplesse sur le tronc, le beaucarnéa est d'une beauté toute en contrastes, aux lignes bien adaptées aux intérieurs modernes.

Le beaucarnéa aime les atmosphères chaudes et humides en été et froides et sèches en hiver. L'installer dans une pièce lumineuse, à l'écart des lieux de passage, dans un contenant large et bas, à la manière des bonsaïs. Vous pourrez le sortir en été, mais attention alors aux coups de soleil ! Le placer à la mi-ombre. En hiver, il apprécie une température de 10 à 15 °C.

Order this Plant

| | |
|---|---|
| **Price** | 75.00 € |
| **Stock** | 12 |
| **Your Name** | |
| **Your** | |

## Publish

- Controls frontend visibility
- Adds fields
  - is_published
  - can_publish
  - website_url

```
{
    'name': Plant Nursery,
    'depends': ['website'],
}


class WebsitePublishedMixin(models.AbstractModel):
    _name = "website.published.mixin"

    is_published = fields.Boolean('Published')
    can_publish = fields.Boolean(
        compute='_compute_can_publish')
    website_url = fields.Char(
        compute='_compute_website_url')


class Plant(models.Model):
    _name = 'nursery.plant'
    _description = 'Plant'
    _inherit = ['website.published.mixin']
```

# Publish

- can_publish

  -> publisher groups
- website_url

  -> plant route URL

```python
from odoo.addons.http_routing.models.ir_http import slug

class Plant(models.Model):
    _name = 'nursery.plant'
    _description = 'Plant'
    _inherit = ['website.published.mixin']

    can_publish = fields.Boolean(
        compute='_compute_can_publish')
    website_url = fields.Char(
        compute='_compute_website_url')

    def _compute_can_publish(self):
        for plant in self:
            plant.can_publish = user.has_group(
                'website.group_website_publisher')

    def _compute_website_url(self):
        for plant in self:
            record.website_url = '/plants/%s' % slug(plant)
```

# Routes

- Public route
- Access rights & route management still up to you !

  -> carefully craft your ACLs !

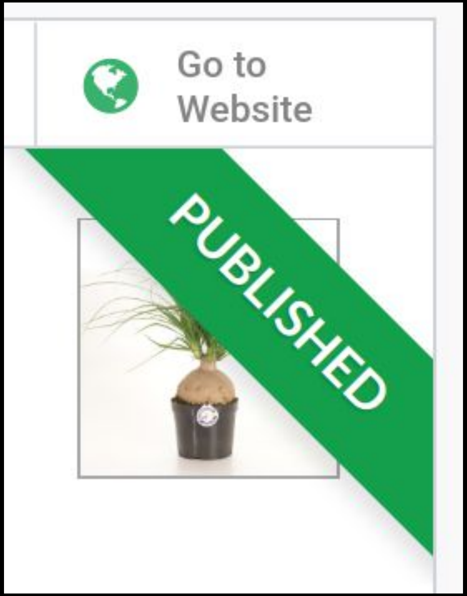
```
@route('/plants/plant/<model("nursery.plant"):plant>',
    type='http', auth="public", website=True)
def plant(self, plant, **post):
    return request.render(
        "plant_nursery.portal_plant_page",
        values)
```

```
<record model="ir.rule" id="ir_rule_nursery_plant_public">
    <field name="name">
        Nursery Plant: public: published only
    </field>
    <field name="model_id" ref="model_nursery_plant"/>
    <field name="domain_force">
        [('website_published', '=', True)]
    </field>
    <field name="groups"
        eval="[(4, ref('base.group_public')),
               (4, ref('base.group_portal'))]"/>
</record>
```

# Publish action

- Controls visibility of documents
- Backend: use redirect widget
    - redirect to website URL

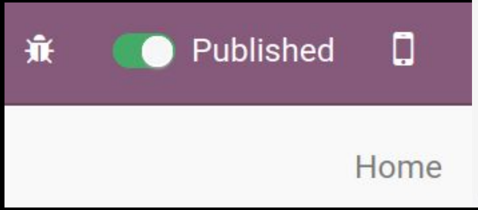
```
<field name="is_published"
    widget="website_redirect_button"/>
```

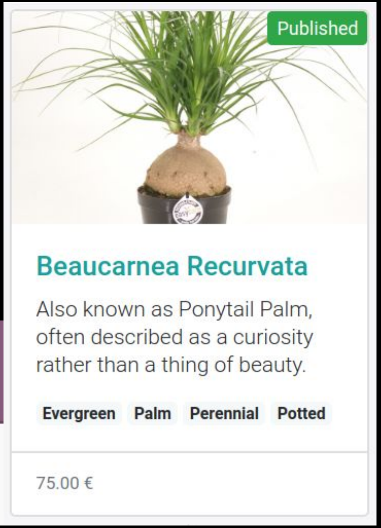


# Publish action

- Controls visibility of documents
- Frontend widget:
  - automatic editor support
  - publish widget: action to execute

```
<t t-call="website.publish_short">
    <t t-set="object" t-value="plant"/>
    <t t-set="publish_edit" t-value="True"/>
    <t t-set="action"
       t-value="'plant_nursery.action_nursery_plant"/>
</t>
```




Published

Beaucarnea Recurvata

Also known as Ponytail Palm, often described as a curiosity rather than a thing of beauty.

Evergreen Palm Perennial Potted

75.00 €

# Multi Website

- Restrict to a specific website
- website_id field
- can_access_from_current_we
  bsite

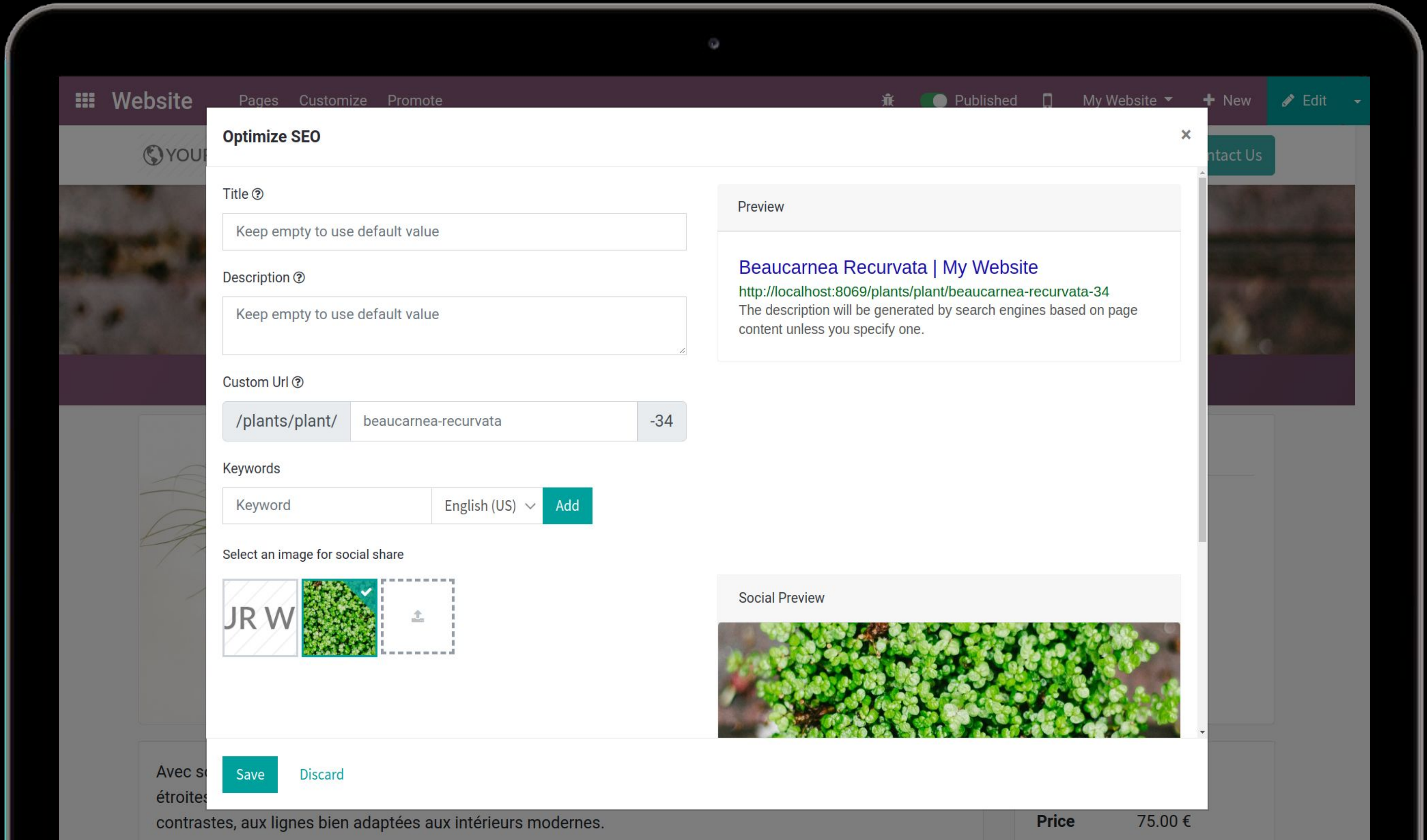
  -> to be used in routes

```
{
    'name': Plant Nursery,
    'depends': ['website'],
}


class Plant(models.Model):
    _name = 'nursery.plant'
    _description = 'Plant'
    _inherit = ['website.multi.mixin']


class WebsiteMultiMixin(models.AbstractModel):
    _name = 'website.multi.mixin'

    website_id = fields.Many2one('website')

    def can_access_from_current_website(self):
        # Specify if record can be accessed
        return record.website_id.id in
            (False, request.website.id)
```

## Multi Publish

- **Publish** document on a specific website
- Compute **website_published**
- Add **_compute_website_published** based on
  - current website
  - is_published flag

```
{
    'name': Plant Nursery,
    'depends': ['website'],
}


class Plant(models.Model):
    _name = 'nursery.plant'
    _description = 'Plant'
    _inherit = ['website.published.multi.mixin']


class WebsitePublishedMultiMixin(models.AbstractModel):
    _name = 'website.published.multi.mixin'

    website_published = fields.Boolean(
        compute='_compute_website_published')

    def _compute_website_published(self):
        # Specify if the record is published on this website
```



Optimize SEO

Title

Keep empty to use default value

Description

Keep empty to use default value

Custom Url

/plants/plant/ beaucarnea-recurvata -34

Keywords

Keyword English (US) Add

Select an image for social share

Preview

Beaucarnea Recurvata | My Website

http://localhost:8069/plants/plant/beaucarnea-recurvata-34

The description will be generated by search engines based on page content unless you specify one.

Social Preview

Save Discard

Avec s étroites contrastes, aux lignes bien adaptées aux intérieurs modernes.

Price 75.00 €

## SEO

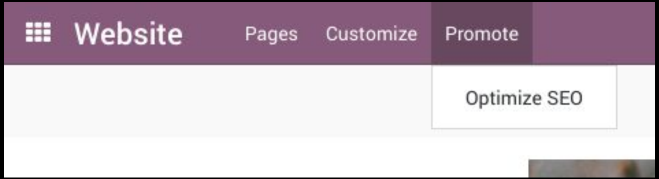
- 'Optimize' SE rankings
- Add fields
    - Page title
    - Keywords
    - Description
    - Og image

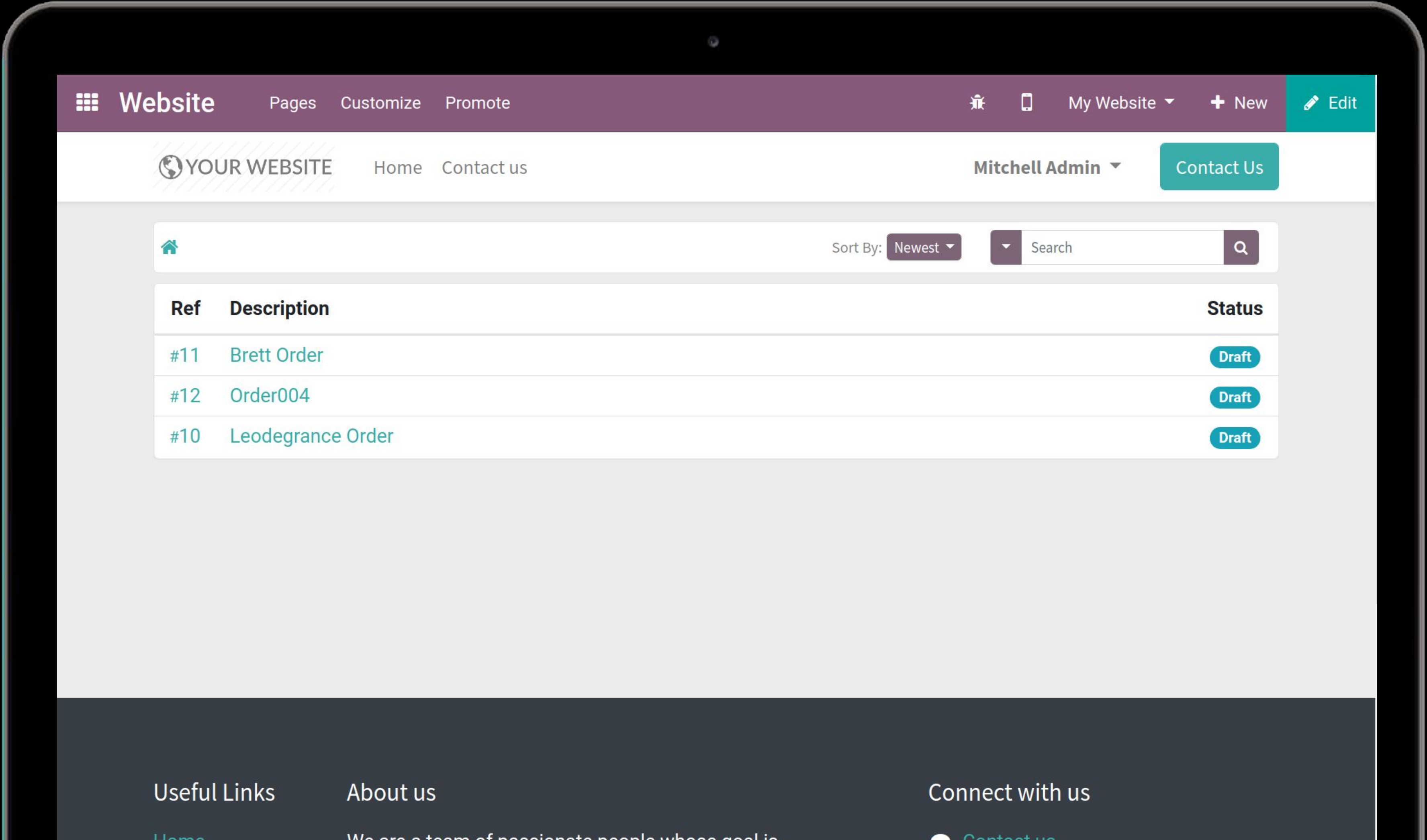
```
{
    'name': Plant Nursery,
    'depends': ['website'],
}


class Plant(models.Model):
    _name = 'nursery.plant'
    _description = 'Plant'
    _inherit = ['website.seo.metadata']


class SeoMetadata(models.AbstractModel):
    _name = 'website.seo.metadata'

    website_meta_title = fields.Char('')
    website_meta_description = fields.Text('')
    website_meta_keywords = fields.Char('')
    website_meta_og_img = fields.Char('')
```

# SEO: promote

- main_object magic in website
- Website promote button
- Set SEO metadata

```
@route('/plants/plant/<model("nursery.plant"):plant>
',
    type='http', auth="public", website=True)
def plant(self, plant, **post):
    values = {
        'main_object': plant,
        'plant': plant,
        'search': post.get('search', ''),
    }
    return request.render("portal_plant_page",
values)
```



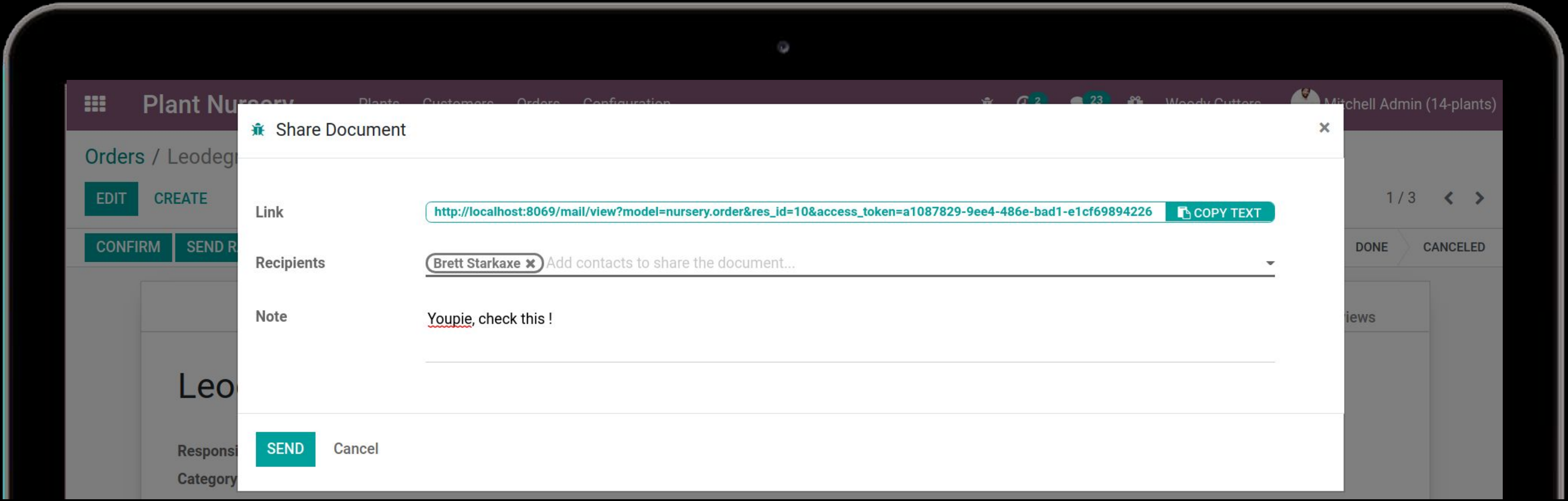


| Ref | Description | Status |
|---|---|---|
| #11 | Brett Order | Draft |
| #12 | Order004 | Draft |
| #10 | Leodegrance Order | Draft |

# Portal

- Document- and App- specific portal url computation

```
{
    'name': Plant Nursery,
    'depends': ['portal'],
}


class Plant(models.Model):
    _name = 'nursery.plant'
    _description = 'Plant'
    _inherit = ['portal.mixin']
```



Link: http://localhost:8069/mail/view?model=nursery.order&res_id=10&access_token=a1087829-9ee4-486e-bad1-e1cf69894226

Recipients: Brett Starkaxe ✖ Add contacts to share the document...

Note: Youpie, check this !

# Portal

- Generic URL computation
    - /mail/view controller
    - access_token
    - partner_id
    - integration with auth_signup

```
class PortalMixin(models.AbstractModel):
    _name = "portal.mixin"

    access_url = fields.Char(
        compute='_compute_access_url')
    access_token = fields.Char('Security Token')
    access_warning = fields.Text(
        compute="_compute_access_warning")

    def _compute_access_warning(self):
        # Set a warning if record can't be shared
        access_warning = ''

    def _compute_access_url(self):
        # Set the portal specific URL
        record.access_url = '#'

    def _get_share_url(self):
        # Set the generic share URL
        return '/mail/view?model='+record._name+'&
            res_id='+record.id+'&
            access_token='+record.access_token
```

# Portal in your App

- Public route
- Access rights & route

  management still up to you!

  -> carefully craft your ACLs!

```
@route('/my/order/<model("nursery.plant"):plant>',
    type='http', auth="public", website=True)
def my_orders(self, page, search, **post):
    ...
    return request.render(
        "plant_nursery.portal_nursery_order",
        values)
```

```
<record model="ir.rule" id="ir_rule_nursery_order_portal">
    <field name="name">
        Nursery Plant: portal: my orders
    </field>
    <field name="model_id" ref="model_nursery_order"/>
    <field name="domain_force">
        [('customer_id.partner_id', '=', user.partner_id.id)]
    </field>
    <field name="groups"
        eval="[(4, ref('base.group_portal'))]"/>
</record>
```

# Portal in your App

- Portal URL computation

  routes
- Share button

```python
class Plant(models.Model):
    _name = 'nursery.plant'
    _inherit = ['portal.mixin']

    def _compute_access_warning(self):
        # Set a warning if record can't be shared
        record.access_warning = 'Error !'
            if record.category_id.internal

    def _compute_access_url(self):
        # Set the portal specific URL
        record.access_url = '/my/order/%s' % record.id
```

```xml
<header>
    <button name="%(portal.portal_share_action)d"
            string="Share" type="action"
            class="oe_highlight oe_read_only"/>
</header>
```

I DON'T ALWAYS GO TO TECHNICAL TALKS
BUT WHEN I DO I WANT TO MASTER EVERYTHING
memegenerator.net

— Classy Cool Dev

# Custom Tracking

- Track plant price change

  -> tracking attribute from mail.thread

  -> specific subtype
- Link to an email template

```
<record id="plant_price_template" model="mail.template">
    <field name="body">...</field>
</record>

 <record id="plant_price" model="mail.message.subtype">
     <field name="name">Price Updated</field>
     <field name="res_model">nursery.plant</field>
 </record>

class Plant(models.Model):
    _name = 'nursery.plant'
    _inherit = ['mail.thread']

    price = fields.Float('Price', tracking=2)

    def _track_subtype(self, values):
        res = super()
        if 'price' in values:
            return self.env.ref(plant_price)
        return super()

    def _track_template(self, values):
        res = super()
        if 'price' in values:
            res['price'] = (
                'plant_price_template',
                options)
        return res
```

## Even more

- A lot of other mixins exist in Odoo
- Look in the doc / code
- Play with them !

```
# use country-based formatting for addresses
_inherit = [format.address.mixin']

# use image tools and resizing
_inherit = [image.mixin']

# use advanced gateway / phone tools
_inherit = ['mail.alias.mixin']
_inherit = ['mail.thread.phone']
_inherit = ['mail.thread.blacklist']

# use QWeb / Jinja rendering tools
_inherit = ['mail.render.mixin']

# use calendar-based time computation
_inherit = [resource.mixin']

# use editable sequence numbers
_inherit = ['sequence.mixin']
```

# Thank You

https://github.com/tivisse/odooplants