

Empower your App by Inheriting from Odoo Mixins

Let us recode everything from scratch! Or not.

Thibault DELAVALLEE · Software Engineer, RD Marketing Team Leader

Based on work from **Yannick TIVISSE**, **Martin TRIGAUX** and **Damien BOUVY**

- 1 Classy Cool Dev Enlightening
- 2 Communication and Organization
- 3 Customer Satisfaction and UTM

4 Website, Portal and Pages

5 Advanced Mail Thread

6 I have a headache. Benefits ?

The use case: **A Plant Nursery**



“



— Classy Cool Dev



Thanks Classy Cool Dev!
But what is a Mixin ?

Mixin Class: vaziztasse ?

- Mixin class: extracts transversal features
- AbstractModel: no table, only for definition
- Offer features through inheritance
- e.g. messaging mechanism, customer satisfaction request, ...



Mixin Class: vaziztasse ?

```
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: vaziztasse ?

- Mixin class: extracts transversal features
- Inheritance: “copy” fields on child

```
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: vaziztasse ?

- Mixin class: extracts transversal features
- Inheritance: “copy” fields on child

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



```
class Order(models.Model):  
    _inherit = ['mail.thread']  
  
    name = fields.Char('Reference')  
    message_ids = fields.One2many(...)  
    message_follower_ids = fields.One2many(...)
```

Mixin Class: vaziztasse ?

- Mixin class: extracts transversal features
- Inheritance: class inheritance: methods, super(), ...

```
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):
    _name = 'nursery.plant'
    _inherit = ['mail.thread']

    name = fields.Char('Plant Name')
    price = fields.Integer('Plant Price')

    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: vaziztasse ?

- Mixin class: extracts transversal features
- Inheritance: class inheritance: methods, super(), ...
- Inherit abstract class itself: apply to all childs

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

    message_my_own = fields.Integer(...)

    def message_my_own_stuff(self):
        ...
        return

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



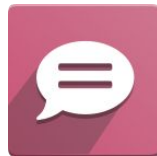
I master mixins.
Now what ?

Methodology

- Features of mixins
- How to implement them
- Code bits
- Live result
- Documentation ?
 - Odoo code
 - [Documentation](#)
 - <https://github.com/tivisse/odoodays-2019>

Covered features

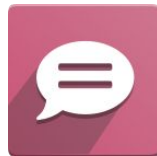
- Communication & Organization
 - Discuss, send mailings, send SMS
 - Activities & Documents
- Marketing
 - Get customer satisfaction insights
 - Track campaigns and mediums
- Website
 - Manage document publication
 - Promote pages



Covered features

- Communication & Organization

- Discuss, send mailings, send SMS
- Activities & Documents



- mail.thread
- mail.activity, documents.mixin

- Marketing

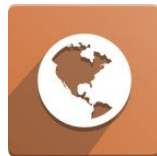
- Get customer satisfaction insights
- Track campaigns and mediums



- rating.(parent.)mixin
- utm.mixin

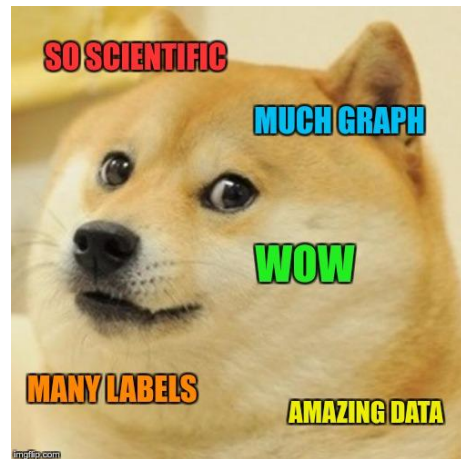
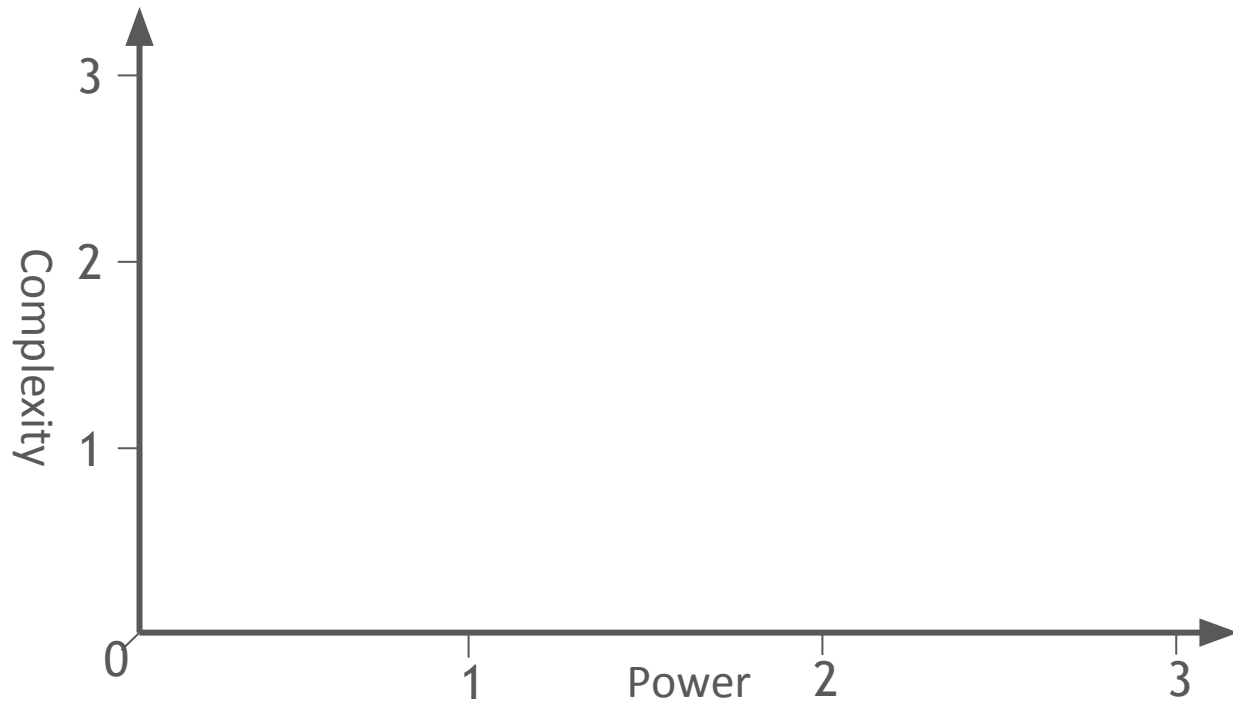
- Website

- Manage document publication
- Promote pages

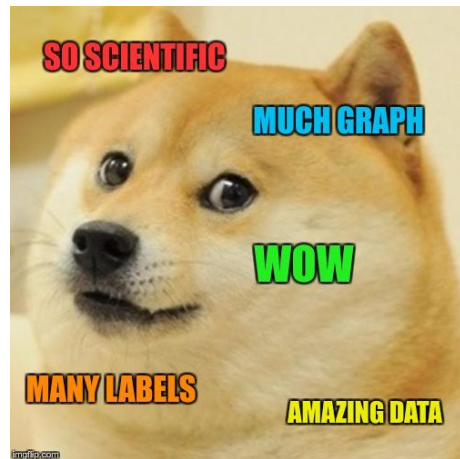
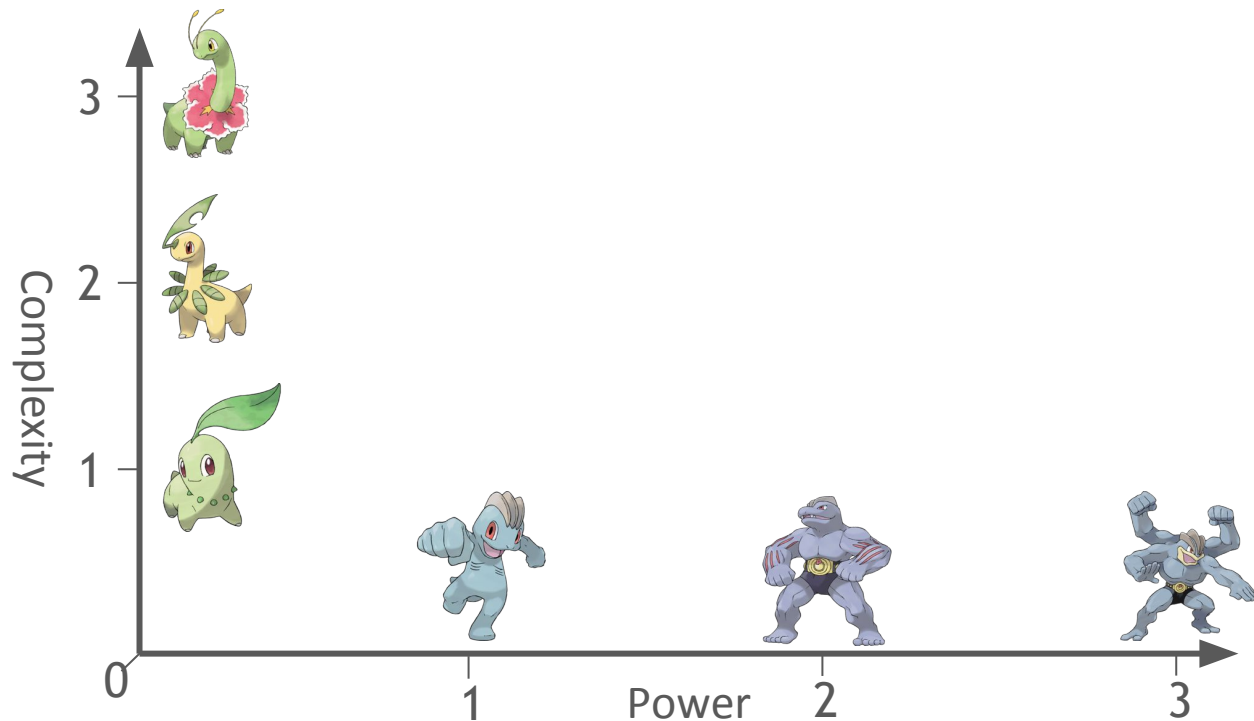


- portal, website.published
- website.seo.metadata

The graph



The graph





Communication and Organization



Communication and Organization

Discuss, send mailings and SMS



Mail Thread: basics



- Add messaging to any class
- Auto-interfacing with e-mail
- How to use:
 - Inherit mail.thread
 - Add widgets
 - Have fun !



Mail Thread: basics



- Add messaging to any class
- Auto-interfacing with e-mail
- How to use:
 - Inherit mail.thread
 - Add widgets
 - Have fun !

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

```
<div class="oe_chatter">  
    <field name="message_follower_ids"  
        widget="mail_followers"/>  
    <field name="message_ids"  
        widget="mail_thread"/>  
</div>
```

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




Mail Thread: Discuss



- Messages linked to a document

[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 know 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 like a man. 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 reptilian!? Leela, Bender, we're going grave robbing. Throw her in the brig. Oh dear! She's stuck 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(...)
```

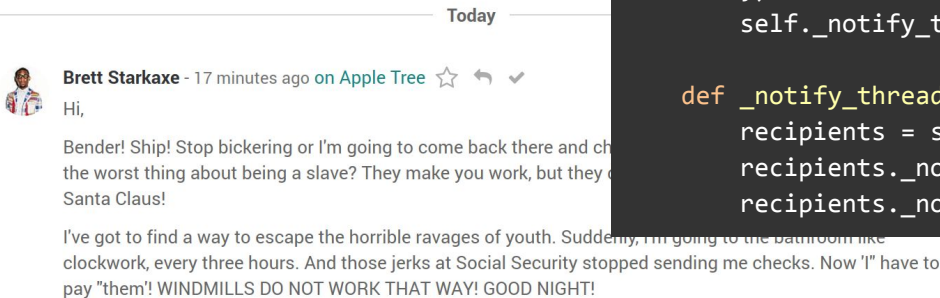
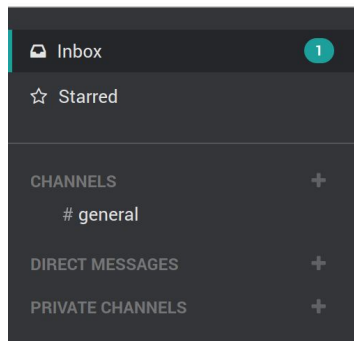
```
class Plant(models.Model):
    _inherit = ['mail.thread']

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



Mail Thread: Discuss

- Messages linked to a document
- Communication using Discuss



```
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)

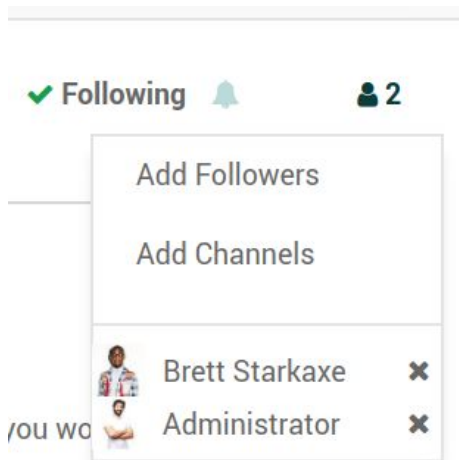
    def _notify_thread(self, message):
        recipients = self._compute_recipients()
        recipients._notify_by_inbox()
        recipients._notify_by_email()
```



Mail Thread: Discuss



- Messages linked to a document
- Communication using Discuss
- Followers management



```
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()
```



Mail Thread: mailgateway



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

```
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
        super()
```



Mail Thread: SMS



- SMS sending
- Auto-addition to mail.thread
- How to use:
 - Inherit mail.thread
 - Add widgets
 - Have fun !

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

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



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



Mail Thread: SMS



- Phone widget, SMS Composer
- Discuss integration

**Woody Cutters, Mitchell Admin** - 9 minutes ago  SMS 

Hello Brett ! I hope everything is fine !

Recipients

Brett Starkaxe (+32456001122)

Message

Hello Brett ! I hope everything is fine !

41 characters, fits in 1 SMS (GSM7) 

SEND SMS

CLOSE

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

    message_ids = fields.One2many(...)

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

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

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



Communication and Organization

Activities and documents



Mail Activity



- Activities management on document
- Discuss integration
- How to use:
 - inherit mail.activity.mixin
 - add widgets
 - have much fun !

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

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

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



Mail Activity




- Activities management on document
- Activity-based state
- Filters

```
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')])
```

[Send message](#) [Log note](#) [🕒 Schedule activity](#)

Follow  1

▼ Planned activities



7 days overdue: "Write Beautiful Emails" for Mitchell Stephens ⓘ

Should be really beautiful.

✓ Mark Done ✎ Edit ✕ Cancel



Due in 8 days: "Discuss about mixin with Classy Cool Dev" for Mitchell Stephens ⓘ

✓ Mark Done ✎ Edit ✕ Cancel



Plant

1 Late

0 Today

1 Future



Mail Activity: Automation



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

```
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()
```



Mail Activity: Automation



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

[Send message](#) [Log note](#) [Schedule activity](#)

✓ Following 2

Planned activities



Tomorrow: "Pack the order" for Marc Brown ⓘ

✓ Mark Done Edit ✕ Cancel

Today



YourCompany, Mitchell Stephens - now

Plant Order created

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

    @api.model
    def create(self, vals):
        res = super(Order, self).create(vals)
        res.activity_schedule(
            'mail.mail_activity_data_todo',
            user_id=self.env.ref('base.user_demo').id,
            date_deadline=fields.Date.today() +
                relativedelta(days=1),
            summary=_('Pack the order'))
        return res

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



Documents



- Attachment management
- Documents Application integration

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

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

The screenshot displays a document management application interface. On the left, a sidebar contains a 'WORKSPACE' section with a tree view including 'All', 'Internal', 'Finance', 'Marketing', and 'Plants' (which is selected). Below this is a 'TAGS' section with a tree view including 'Technical Documentation' (expanded) and its sub-items 'How to harvest' and 'How to plant'. At the bottom of the sidebar is an 'ATTACHED TO' section showing 'Nursery Plant'. The central area shows a document card for 'presentation.pdf' with a PDF icon, the title 'Nursery Plant : Beaucarnea Recurvata', a tag 'How to harvest', and the date '10/01/2019'. The right-hand panel displays the document's metadata, including fields for Name, Contact, Owner, Workspace, and Tags, each with a dropdown menu or input field.



Documents



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

```
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')

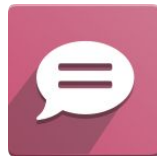
    def _get_document_tags(self):
        return self.user_id
```

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

Covered features

- **Communication & Organization**

- **Discuss, send mailings, send SMS**
- **Activities & Documents**



- **mail.thread**
- **mail.activity, documents.mixin**

- **Marketing**

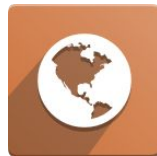
- Get customer satisfaction insights
- Track campaigns and mediums



- rating.(parent.)mixin
- utm.mixin

- **Website**

- Manage document publication
- Promote pages



- portal, website.published
- website.seo.metadata



3

Customer Satisfaction and UTM



Customer Satisfaction and UTM

Customer satisfaction insights



Rating

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



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

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

```
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(...)
```



Rating

- Add rating on any class
- Request rating via email/route
- Analyse your ratings
- How to use: **a bit of work**



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

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

```
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(...)
```



Rating: a bit of work



- Rated partner: **people to rate**
 - **rating_getRatedPartnerId**
 - default: (user_id.partner_id field)
 - in our case: default ok!

```
class RatingMixin(models.Model):  
    _name = 'rating.mixin'  
  
    def rating_getRatedPartnerId(self):  
        if hasattr(self, 'user_id') and self.user_id:  
            return self.user_id.partner_id  
        return self.env['res.partner']
```



Rating: a bit of work



- Rated partner: people to rate: `rating_getRatedPartnerId`
- Customer: people that rates
 - **`rating_getPartnerId`**
 - default: (partner_id field)
 - Need to override

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

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

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

    partner_id = fields.Many2one('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: a bit of work



- Rated partner: people to rate: `rating_getRatedPartnerId`
- Customer: people that rates: `rating_getRatedPartnerId`
- Customer secure access: use an access token
 - **`rating_get_access_token`**
 - provided by mixin, nothing to do



Rating: a bit of work



- Rated partner: people to rate: `rating_getRatedPartnerId`
- Customer: people that rates: `rating_getRatedPartnerId`
- Customer secure access: use an access token
- Routes from mixin:
 - `/rating/<token>/<score>`
 - `/rating/<token>/<score>/submitFeedback`



Rating: a bit of work



- Rated partner: people to rate: `rating_getRatedPartnerId`
- Customer: people that rates: `rating_getRatedPartnerId`
- Customer secure access: use an access token
- Routes from mixin
- Send requests through email
 - -> Email Template using this data



Rating: a bit of work



- Email Template using this data

```
<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" >${object.rating_get_partner_id().id}</field>
  <field name="auto_delete" eval="True"/>
  <field name="body_html" type="html">
    <div>
      % set access_token = object.rating_get_access_token()
      <!-- Insert Beautiful Email Stuff -->
      <table>
        <tr>
          <td><a href="/rating/${access_token}/10"></a></td>
          <td><a href="/rating/${access_token}/5"></a></td>
          <td><a href="/rating/${access_token}/1"></a></td>
        </tr>
      </table>
      <!-- Insert Ending Beautiful Email Stuff -->
    </div>
  </field>
</record>
```



Rating: The Result



Orders / Order003

EDIT CREATE

SEND RATING REQUEST

Your Plant Order
Order006



Your logo

Satisfaction Survey

Hello,

Please take a moment to rate our services related to the order "**Order006**" assigned to **Marc Brown**.

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

Tell us how you feel about our service:

(click on one of these smileys)



--
+Mr Demo

Email automatically sent by Odoo Plant Nursery for Woody Cutters

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

Thanks! We appreciate your feedback.

Your rating has been submitted.



you are **satisfied**
on our services on "**Order006**"
by **Marc Brown**.

Would be great if you can provide more information:

This is awesome !

Send Feedback



We appreciate your feedback!

[Go to our website](#)



Rating: parent mixin



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

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

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

```
class RatingParentMixin(models.AbstractModel):  
    _name = 'rating.parent.mixin'  
  
    rating_ids = fields.One2many('rating.rating')  
    rating_percentage_satisfaction =  
        fields.Float(...)
```



Rating: parent mixin



- Container of document ratings
- Get statistics per category

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



Nursery

😊 100 % satisfaction

5



Woody Cutters, Administrator

📁 Starkaxe Order

🕒 09/30/2019 21:25:26

Very good services !

4.5



Woody Cutters, OdooBot

📁 Starkaxe Order

🕒 09/30/2019 21:25:26

Excellent !



3

Customer Satisfaction and UTM

Track UTMs



UTM



- Track incoming visitors
- Pre-built with three fields:
campaign, source, medium
- Simple to extend

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

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

```
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



- Track incoming visitors
- Pre-built with three fields:
campaign, source, medium
- Simple to extend
- URL parsing

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

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

Campaign

Sale

Source

Facebook

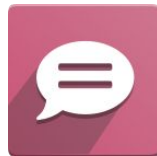
Medium

Facebook Ads

Covered features

- Communication & Organization

- Discuss, send mailings, send SMS
- Activities & Documents



- mail.thread
- mail.activity, documents.mixin

- **Marketing**

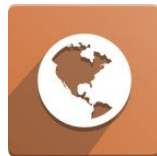
- **Get customer satisfaction insights**
- **Track campaigns and mediums**



- **rating.(parent.)mixin**
- **utm.mixin**

- Website

- Manage document publication
- Promote pages



- portal, website.published
- website.seo.metadata



Website, Portal and Pages



Website, Portal and Pages

Manage document publication



Publish



- Controls visibility of documents
- Adds fields
 - is_published (v13 change)
 - can_publish
 - URL (slug)
- **Access rights & route management still up to you !**

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

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

```
class WebsitePublishedMixin(models.AbstractModel):  
    _name = "website.published.mixin"  
  
    is_published = fields.Boolean('')  
    can_publish = fields.Boolean(  
        compute='_compute_can_publish')  
    website_url = fields.Char(  
        compute='_compute_website_url')
```



Publish: model

- Controls visibility of documents
- Adds fields
 - is_published (v13 change)
 - can_publish
 - URL (slug)
- In our case
 - can_publish -> publisher
 - URL: plants/plant

```
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)
```



Publish: actions



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

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

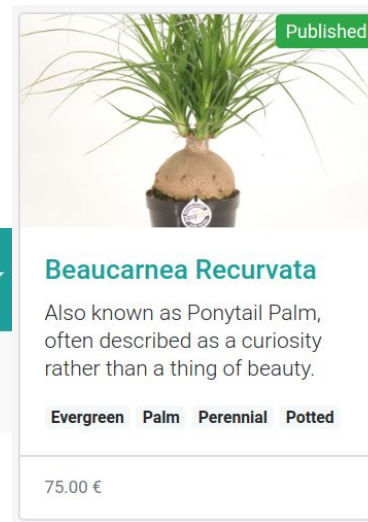
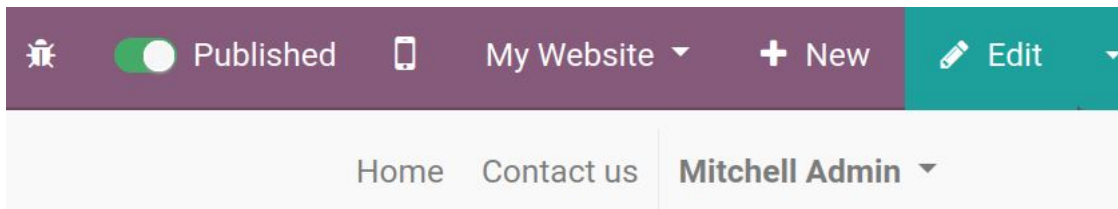




Publish: actions

- Controls visibility of documents
- Backend: use redirect widget
- Frontend:
 - native support in website editor
 - 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>
```





Multi Website



- **Restricts** document to a specific website
- Adds **website_id**
- Adds a **method** to be used in routes

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

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

```
class WebsiteMultiMixin(models.AbstractModel):  
    _name = 'website.multi.mixin'  
  
    website_id = fields.Many2one('website')  
  
    def can_access_from_current_website(self):  
        # Specify if the record can be accessed on  
        this website  
        return record.website_id.id in (False,  
            request.website.id)
```




Published: Multi Website

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

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

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

```
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
```



Website, Portal and Pages

Customer Portal



Portal



- Document- and App- specific portal url computation

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

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

🔗 Share Document

Link

http://localhost:8069/mail/view?model=nursery.order&res_id=1&access_token=10543e23-b7e0-4893-ab17-6a04e304d4ab

📋 COPY TEXT

Recipients

Brett Starkaxe ✕ Add contacts to share the document...

Note

Hello Brett! Here is your access to the order !

SEND

Cancel



Portal



- Document- and App- specific portal url computation
- 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: do it !



- Document- and App- specific portal url computation
- In your App
 - Portal computation
 - Share button

```
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
```

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



Portal: do it !



- Document- and App- specific portal url computation
- In your App
 - Portal computation
 - Share button
 - ... and customer pages
- **Access rights & route management still up to you !**

```
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
```

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



Website, Portal and Pages

SEO



SEO



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

```
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('')
```

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




SEO: Website



- '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)
```

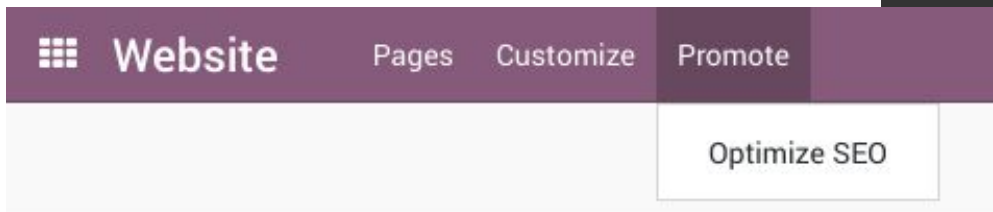


SEO: Website



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

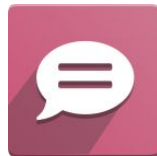
```
@route('/plants/plant/<model("nursery.plant"):plant>/',  
      type='http', auth='public', website=True)  
def plant(self, plant, **post):  
    values = {  
        'main_object': plant,  
        'plant': plant,  
        'search': post.get('search', ''),  
    }  
    return request.render("portal_plant_page",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



Covered features

- Communication & Organization

- Discuss, send mailings, send SMS
- Activities & Documents



- mail.thread
- mail.activity, documents.mixin

- Marketing

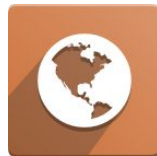
- Get customer satisfaction insights
- Track campaigns and mediums



- rating.mixin
- utm.mixin

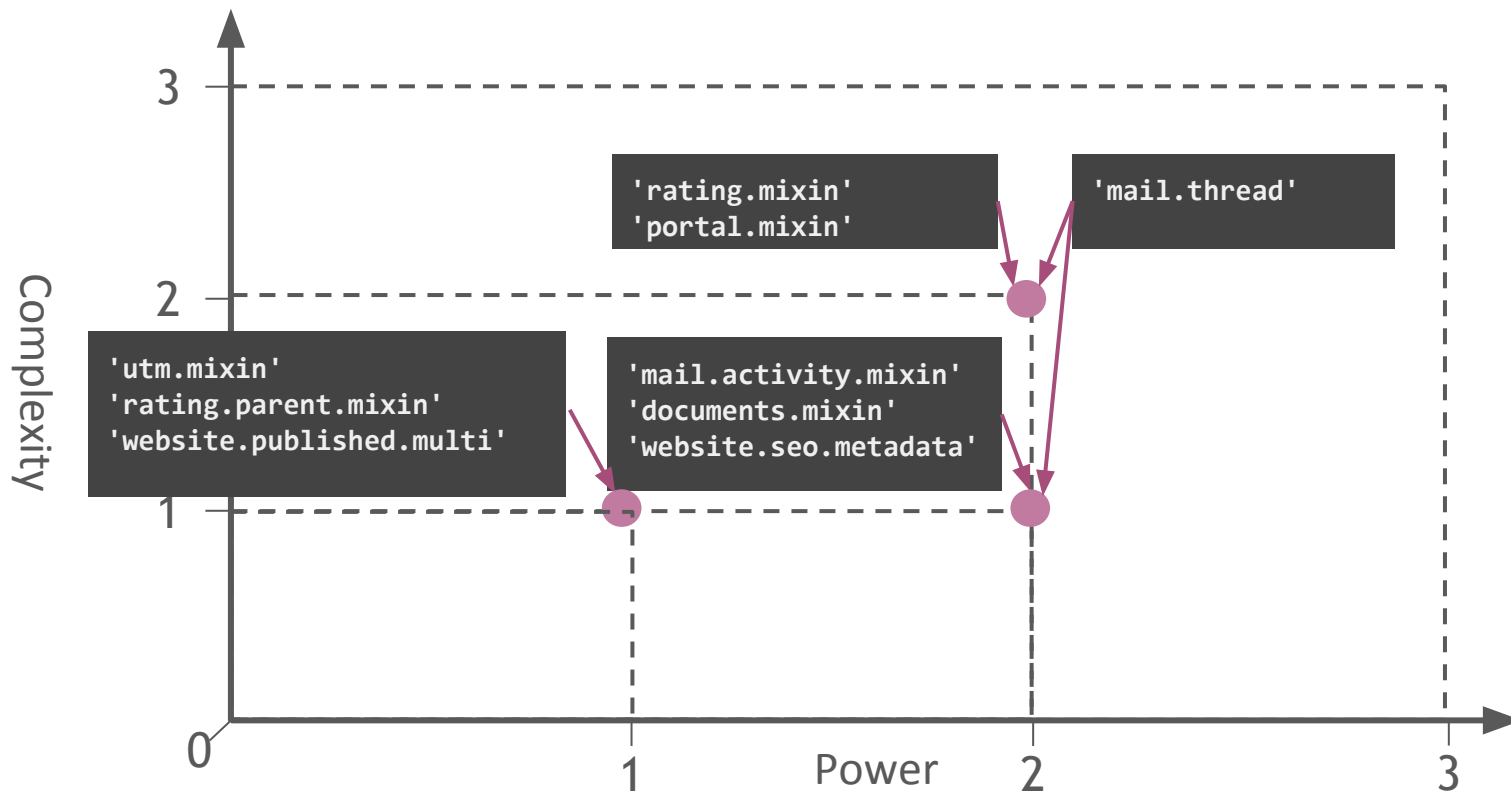
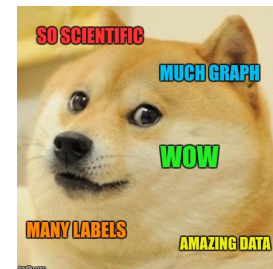
- **Website**

- **Manage document publication**
- **Promote pages**



- **portal, website.published**
- **website.seo.metadata**

The graph



“



— Still Classy Cool Dev



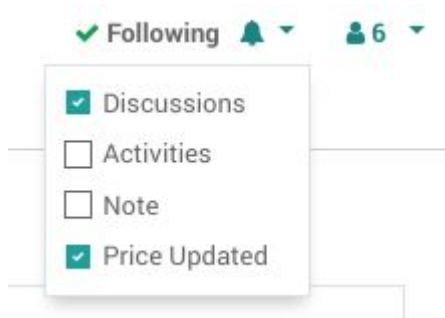
Advanced Mail Thread



Mail: Subtypes



- Characterize / filter messages
- Subtype definition (XML)
 - model specific (e.g. plant)
 - default / hidden
- Use in code: message_post



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

    def write(self, values):
        res = super()
        if 'price' in values:
            self.message_post(
                body=_('Price Updated'),
                subtype='plant_price')
        return res
```

```
<record id="plant_price" model="mail.message.subtype">
  <field name="name">Price Updated</field>
  <field name="res_model">nursery.plant</field>
  <field name="default" eval="True"/>
  <field name="hidden" eval="False"/>
  <field name="description">Price Updated</field>
</record>
```



Mail: Tracking



- Track value change automatically
- **tracking** field attribute
 - sequence (int) or True
- Automatic logging

```
class Plant(models.Model):  
    _name = 'nursery.plant'  
    _inherit = ['mail.thread']  
  
    name = fields.Char('Name', tracking=1)  
    price = fields.Float('Price', tracking=2)
```



YourCompany, Mitchell Stephens - now

Price Updated

- Price: 115 → 114
- Plant Name: Lemon Potted Tree → Lemon Non Potted Tree



Mail: Tracking + Subtypes



- Track value change automatically
- **tracking** field attribute
 - sequence (int) or True
- Link tracking and subtype
 - **_track_subtype**
 - return a subtype xml_id
- Message with tracking + subtype
 - allow people to be notified

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

    name = fields.Char('Name', tracking=1)
    price = fields.Float('Price', tracking=2)

    def _track_subtype(self, values):
        if 'price' in values:
            return 'plant_nursery.plant_price'
        return super()
```



Woody Cutters, Administrator - 2 minutes ago ☆

Price Updated

- Plant Name: Apple Tree
- Price: 50



Mail: Tracking + Mailing



- Email on some specific value change
- Link tracking and mail template
 - **_track_template**
 - give a mail.template xml_id and mailing options
- Template rendered and sent
 - force mailing to people
- Ultra power: add rating in template

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

    name = fields.Char('Name', tracking=1)
    price = fields.Float('Price', tracking=2)

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



YourCompany, Mitchell Stephens - 6 minutes ago

Hello,

Plant Lemon Non Potted Tree price updated to 114.0.

Email automatically sent by Odoo Plant Nursery for Woody Cutters



Mail Alias



- Email integrated with mailgateway
- Purpose: create / update thread in a given model
- Owner: record defining the alias
- Example
 - alias on category
 - creating an order

```
class MailAlias(models.AbstractModel):
    _name = 'mail.alias'

    name = fields.Char('Email')

    # record creation / update
    model = fields.Char('Model')
    thread_id = fields.Integer('Update specific record')

    # record owner
    parent_model = fields.Char('Owner model')
    parent_thread_id = fields.Integer('Owner ID')
```



Mail Alias: How to use



- mail.alias.mixin
- add alias_id field
- Specify what do do with alias by overriding methods

Name

Palm

Alias

yti+xmas@odoo.com

```
class Category(models.Model):
    _name = 'plant.category'
    _inherit = ['mail.alias.mixin']

    def get_alias_model_name(self, values):
        return 'nursery.order'

    def get_alias_values(self):
        values = super()
        values['alias_defaults'] = {}
        return values
```

```
class MailAlias(models.AbstractModel):
    _name = 'mail.alias.mixin'
    _inherit = {'mail.alias' : 'alias_id'}

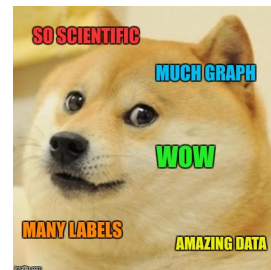
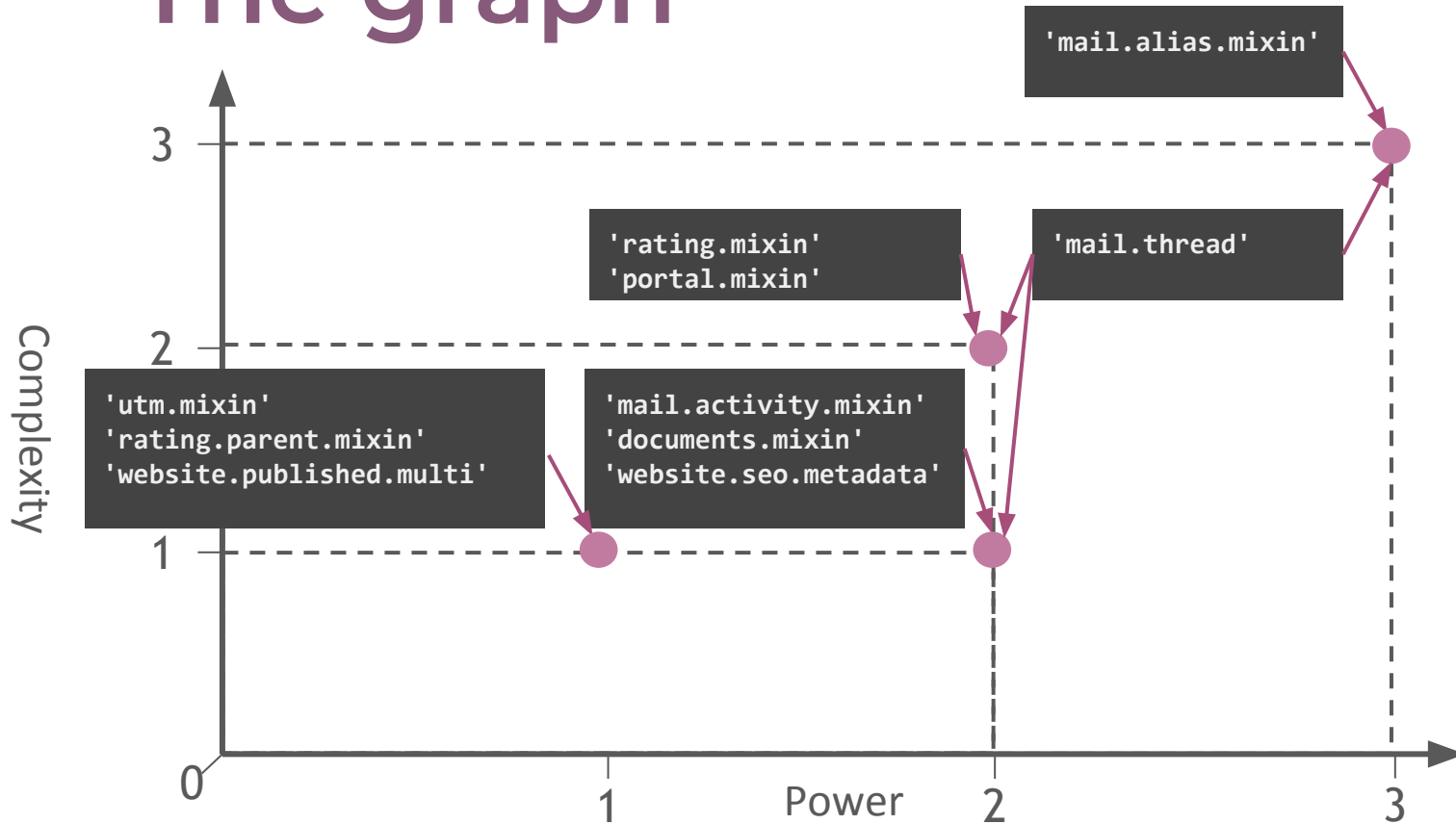
    alias_id = fields.Many2one('Alias')
```



6

I have a headache.
Benefits ?

The graph



Thank you.

<https://github.com/tivisse/odoodays-2019>



#odooexperience

Based on work from Yannick TIVISSE that was based on work from Thibault DELAVALLEE
and Martin TRIGAUX that was based on work from Damien BOUVY

Stop now

<https://github.com/tivisse/odoodays-2019>



#odooexperience

Based on work from Yannick TIVISSE that was based on work from Thibault DELAVALLEE
and Martin TRIGAUX that was based on work from Damien BOUVY

Really

<https://github.com/tivisse/odoodays-2019>



#odooexperience

Based on work from Yannick TIVISSE that was based on work from Thibault DELAVALLEE
and Martin TRIGAUX that was based on work from Damien BOUVY

The graph

