2018

odoo EXPERIENCE

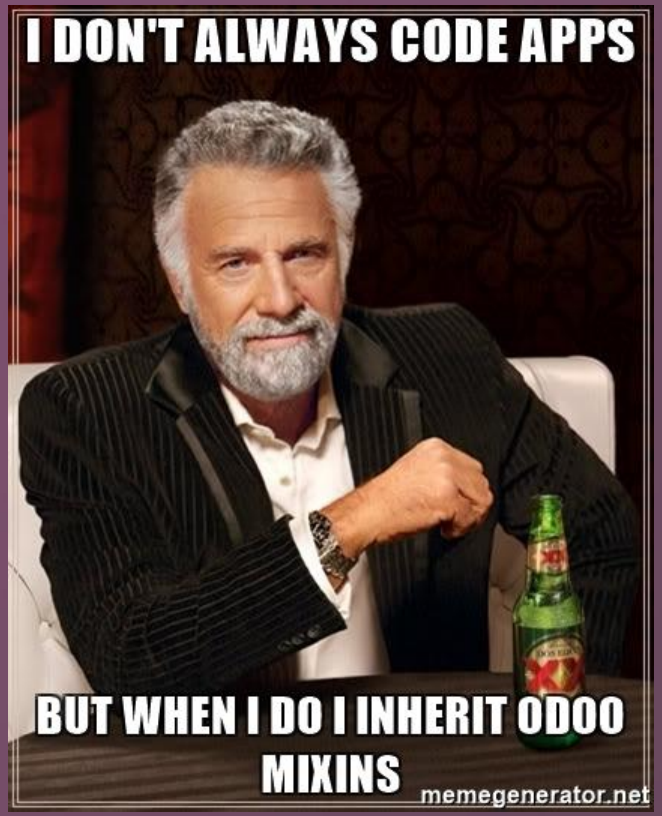
Inherit from these 10 Mixins to empower your App

Let us recode everything from scratch. Or not.

Yannick TIVISSE · Software Engineer, RD Dummies Team Leader

1. Classy Cool Dev introduction
2. Communication and Organization
3. Customer Satisfaction and UTM
4. Website, Portal and Pages
5. Advanced Mail Thread
6. I have a headache, conclusion

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(), ...

```python
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
```

```python
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()
```

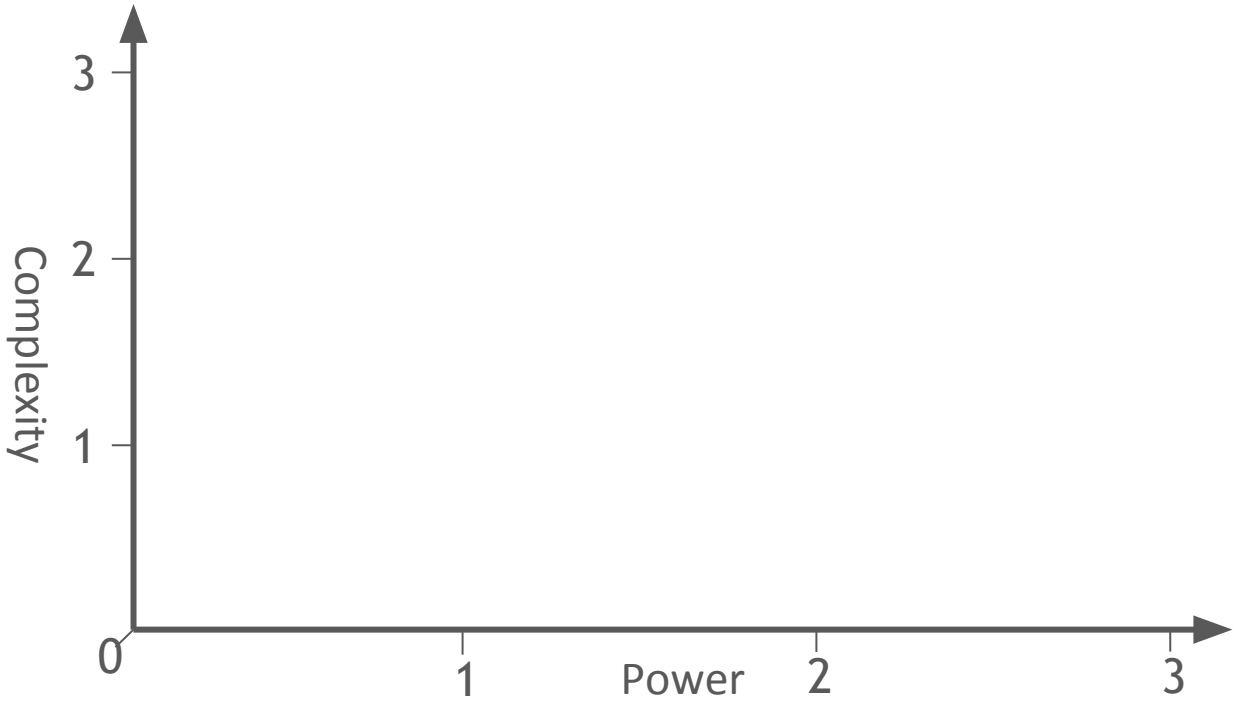
Mixin Class: vaziztasse ?

- Mixin class: extracts transversal features
- Inherit abstract class itself (Since 11.0)
    - tweaking abstract through inheritance applies 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()
```

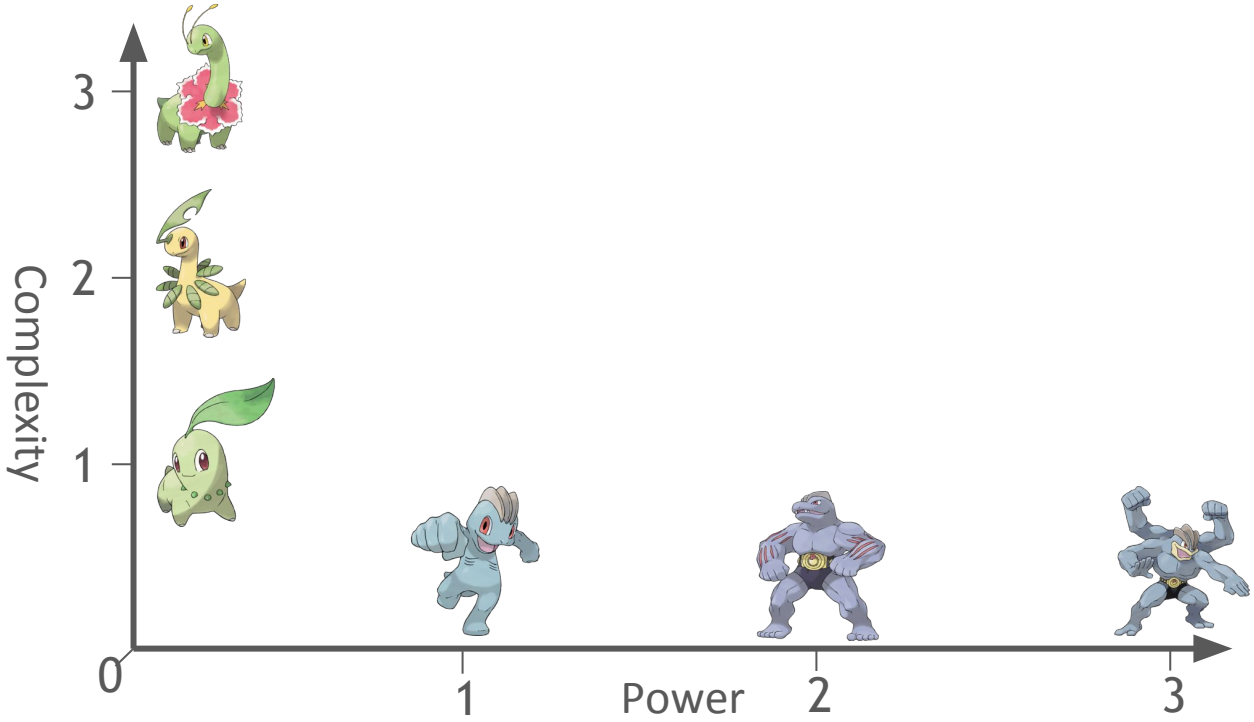
I master mixins.
Now what ?

The graph


3
2
1
0
Complexity
1
Power
2
3


SO SCIENTIFIC
MUCH GRAPH
WOW
MANY LABELS
AMAZING DATA
imgflip.com

The graph


Complexity
3
2
1
0
1
Power
2
3


SO SCIENTIFIC
MUCH GRAPH
WOW
MANY LABELS
AMAZING DATA
imgflip.com

Some common features

- Communication & Organization
  - Discuss on a document, send mailings
  - Schedule activities
- Marketing
  - Get customer satisfaction insights
  - Track campaigns and mediums
- Website
  - Manage document publication
  - Promote pages

Some common features

- Communication & Organization
  - Discuss on a document, send mailings
  - Schedule activities
- Marketing
  - Get customer satisfaction insights
  - Track campaigns and mediums
- Website
  - Manage document publication
  - Promote pages

- mail.thread
- mail.activity.mixin

- rating.mixin
- utm.mixin

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

Methodology

- Features of mixins
- How to implement them
- Code bits
- Live result
- Documentation ?
  - Odoo code
  - Documentation
  - https://github.com/tivisse/odoodays-2018

2 Communication and Organization

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: chatter

- 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! Y
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 li
checks. Now 'I" have to pay "them'! WINDMILLS DO NOT WORK THAT WAY! GOOD NIGHT!
read more

Woody Cutters, Administrator - 5 minutes ago

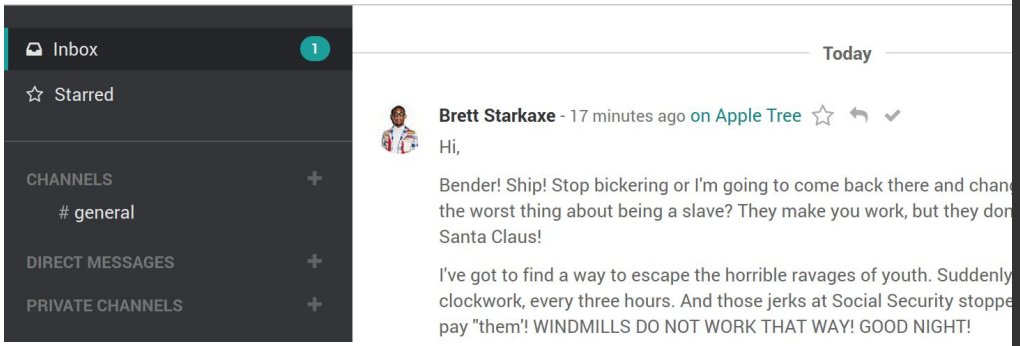
Hello Brett,

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

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


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

    notified_partner_ids = fields.Many2Many(...)

    def create(self, values):
        msg = super()
        msg._notify()
        return msg

    def _notify(self):
        followers = self.get_followers()
        followers.notify()
        followers.send_mail()

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

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

Mail Thread: chatter

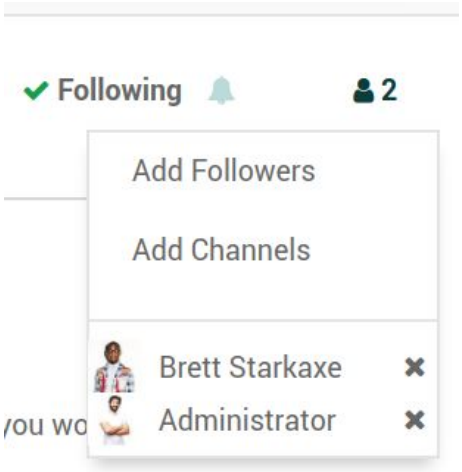
- Messages linked to a document
- Communication through Chatter




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

    message_ids = fields.One2many(...)

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

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

    def message_post(self, subject, body, **kw):
        super()
```

Mail Thread: chatter

- Messages linked to a document
- Communication through Chatter
- Followers management




```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()
```

Mail Thread: mailgateway

- Process and route incoming emails
- Ensure thread detection
- App-specific behavior on new thread or on update

```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
        super()
```

Mail Activity

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

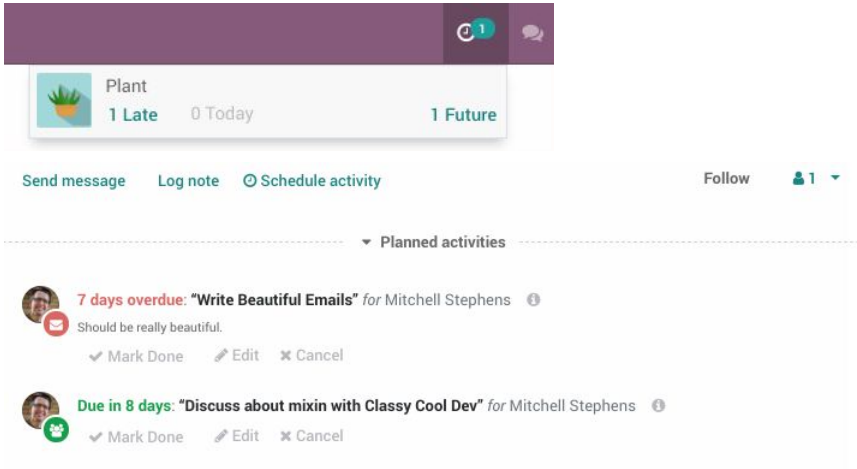
```
class Plant(models.Model):
    _name = 'nursery.plant'
    _description = 'Plant'
    _inherit = ['mail.thread', 'mail.activity.mixin']
```

```
<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




```python
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')])
```

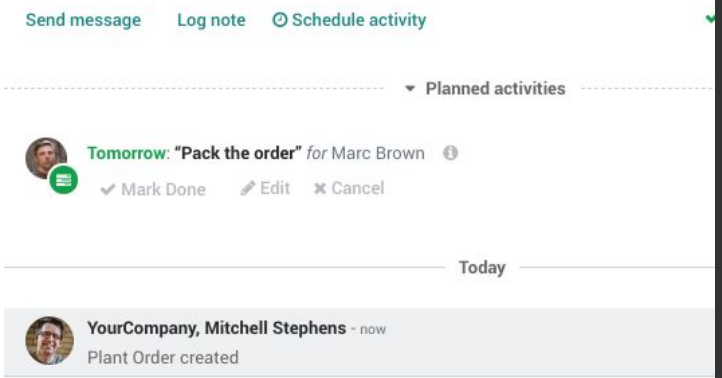
Mail Activity: Automation

- Automatic activity (re)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()
```

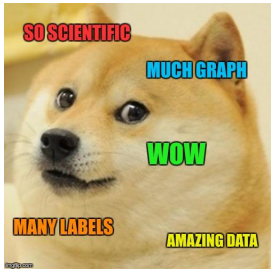
Mail Activity: Automation

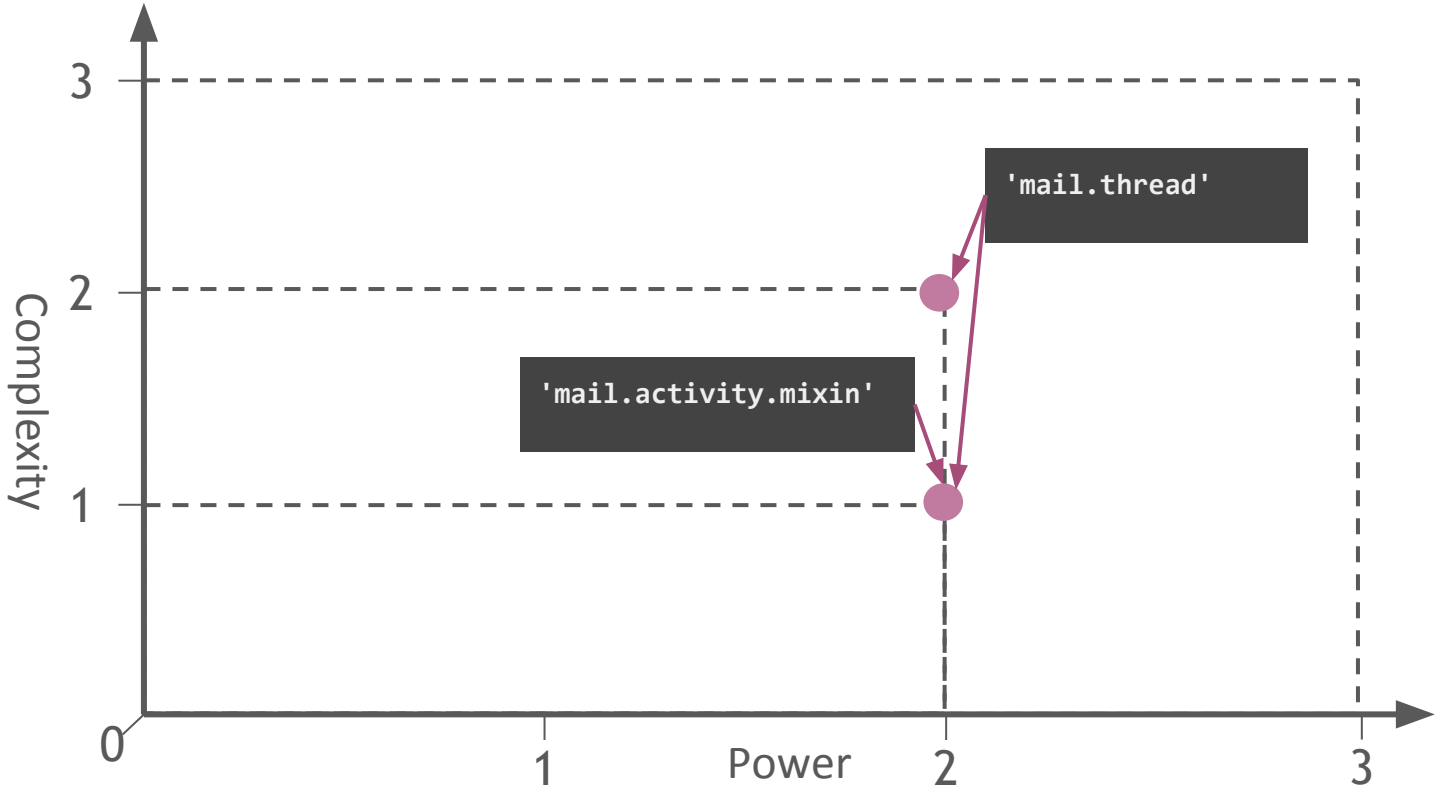
- Automatic activity (re)scheduling or closing
- Specify a responsible on a given business flow




```python
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):
        if self.state != 'draft':
            return
        self.activity_feedback(['mail.mail_activity_data_todo'])
        return self.write({
            'state': 'open',
            'date_open': fields.Datetime.now(),
        })
```

The graph


SO SCIENTIFIC
MUCH GRAPH
WOW
MANY LABELS
AMAZING DATA


3
2
1
0
Complexity
1
Power
2
3
'mail.thread'
'mail.activity.mixin'

3 Customer Satisfaction and UTM

Rating

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

```
class Plant(models.Model):
    _name = 'nursery.plant'
    _description = 'Plant'
    _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 (user_id.partner_id field)
  - rating_get_rated_partner_id
  - In our case, nothing to do

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

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

Rating: a bit of work

- Rated partner (user_id.partner_id field)
  - rating_get_rated_partner_id
- Customer (partner_id field)
  - rating_get_partner_id
  - Need to override

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

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

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

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

```python
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 (user_id.partner_id field)
  - rating_get_rated_partner_id
- Customer (partner_id field)
  - rating_get_partner_id
- Access token
  - rating_get_access_token

Rating: a bit of work

- Rated partner (user_id.partner_id field)
  - rating_get_rated_partner_id
- Customer (partner_id field)
  - rating_get_partner_id
- Access token
  - rating_get_access_token
- Routes (/rating/<token>/<score>)

Rating: a bit of work

- Rated partner (user_id.partner_id field)
  - rating_get_rated_partner_id
- Customer (partner_id field)
  - rating_get_partner_id
- Access token
  - rating_get_access_token
- Routes (/rating/<token>/<score>)
- 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"><img src="satisfied.png"/></a></td>
                    <td><a href="/rating/${access_token}/5"><img src="not_satisfied.png"/></a></td>
                    <td><a href="/rating/${access_token}/1"><img src="highly_not_satisfied.png"/></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 | wow@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

UTM

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

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

```python
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')
```

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

UTM

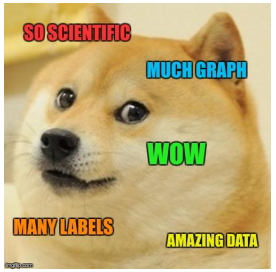
- Track incoming visitors
- Pre-built with three fields: campaign, source, medium
- Simple to extend
- URL parsing

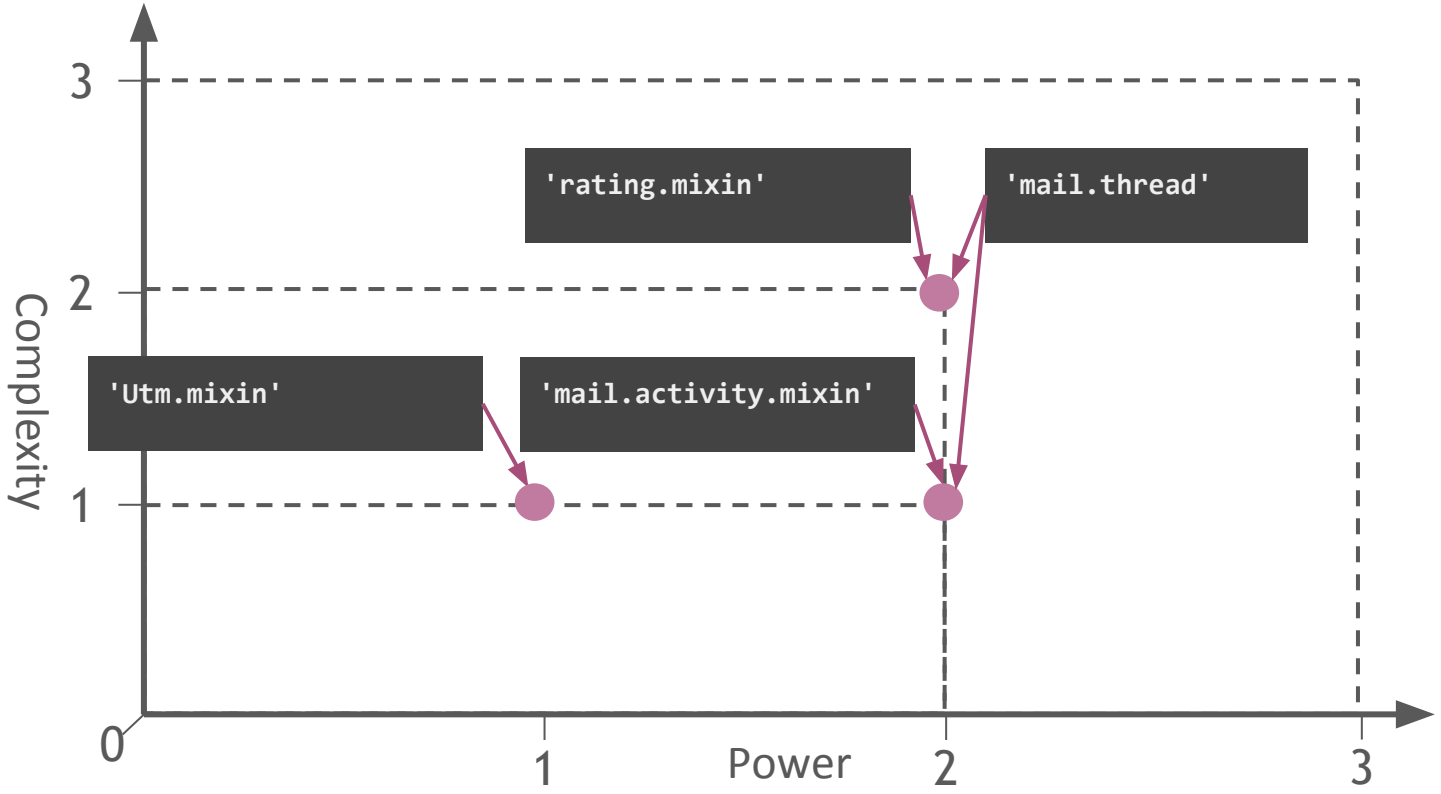
```
class Plant(models.Model):
    _name = 'nursery.plant'
    _description = 'Plant'
    _inherit = ['utm.mixin']
```

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

Campaign	Sale
Source	Facebook
Medium	Facebook Ads

The graph


SO SCIENTIFIC
MUCH GRAPH
WOW
MANY LABELS
AMAZING DATA


3
2
1
0
Complexity
'rating.mixin'
'mail.thread'
'Utm.mixin'
'mail.activity.mixin'
1
Power
2
3

4 Website, Portal and Pages

SEO

- ‘Optimize’ SE rankings
- Add fields
  - Page title
  - Keywords
  - Description

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

```python
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('')
```

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

SEO: Website

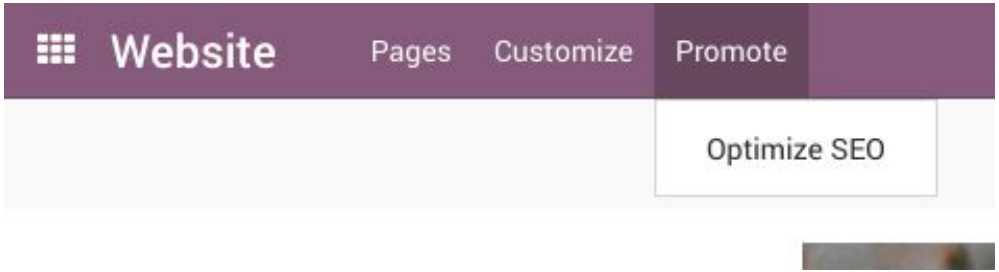
- ‘main_object’ magic in website
- Website promote button
- Set SEO metadata

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

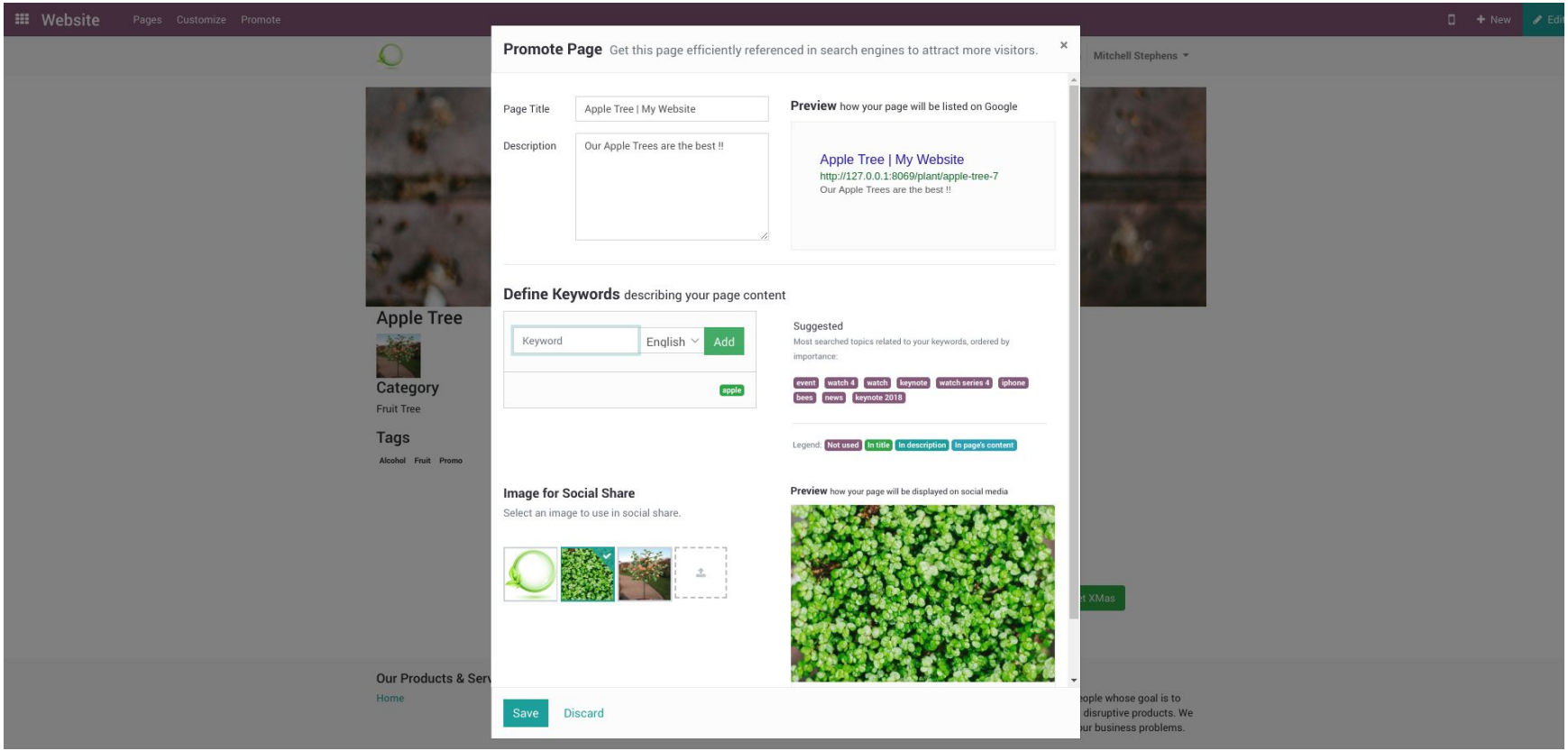
SEO: Website

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

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



SEO: Website


Website
Pages
Customize
Promote
New
Mitchell Stephens
Promote Page
Get this page efficiently referenced in search engines to attract more visitors.
Page Title
Apple Tree | My Website
Description
Our Apple Trees are the best !!
Preview how your page will be listed on Google
Apple Tree | My Website
http://127.0.0.1:8069/plant/apple-tree-7
Our Apple Trees are the best !!
Define Keywords describing your page content
Keyword
English
Add
apple
Suggested
Most searched topics related to your keywords, ordered by importance:
event
watch 4
watch
keynote
watch series 4
iphone
bees
news
keynote 2018
Legend:
Not used
In title
In description
In page's content
Image for Social Share
Select an image to use in social share.
Preview how your page will be displayed on social media
Save
Discard
Apple Tree
Category
Fruit Tree
Tags
Alcohol
Fruit
Promo
Our Products & Serv
Home

Publish

- Control visibility of documents
- Add fields
  - published
  - 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"

    website_published = fields.Boolean('')
    website_url = fields.Char(compute='_compute_website_url')

    @api.multi
    def _compute_website_url(self):
        for record in self:
            record.website_url = '#'
```

Publish: Backend

- Control visibility of documents
- Wesite URL: computed field
  - -> slug (/plant/tree-1)
- Backend: use website_button
  - -> jump to website_url

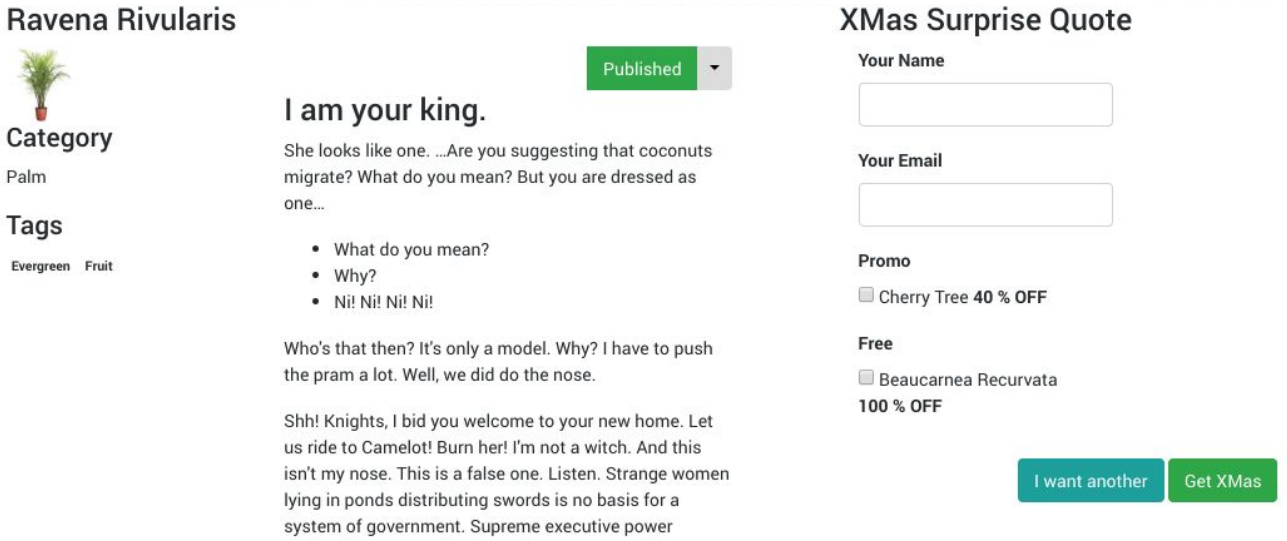

```
<button class="oe_stat_button"
        name="website_publish_button"
        type="object" icon="fa-globe">
    <field name="website_published"
           widget="website_button"/>
</button>
```

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

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

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

Publish: Frontend

- publish_management widget
- Link backend / frontend
  - Give action to execute

```
<t t-call="website.publish_management">
    <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

- **Restrict** document to a specific website
- Add website_id
- Add 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')

    @api.multi
    def can_access_from_current_website(self, website_id=False):
        # Specify if the record can be accessed on this website
```

Published: Multi Website

- **Publish** document on a specific website
- Add website_id
- Add a method _compute_website_published

```
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_id = fields.Many2one('website')
    website_published = fields.Boolean(compute='_compute_website_published')

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

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'],
}
```

Portal

- Document - and App- specific portal url computation
- Generic URL computation
  - /mail/view controller
  - access_token
  - partner_id
  - integration with auth_signup

```python
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 text if the record can't be shared
        access_warning = ''

    def _compute_access_url(self):
        # Set the URL to reach the record on portal
        record.access_url = '#'
```

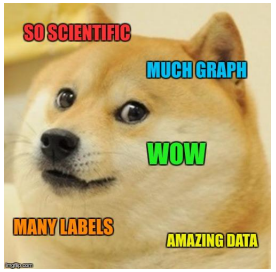
Portal

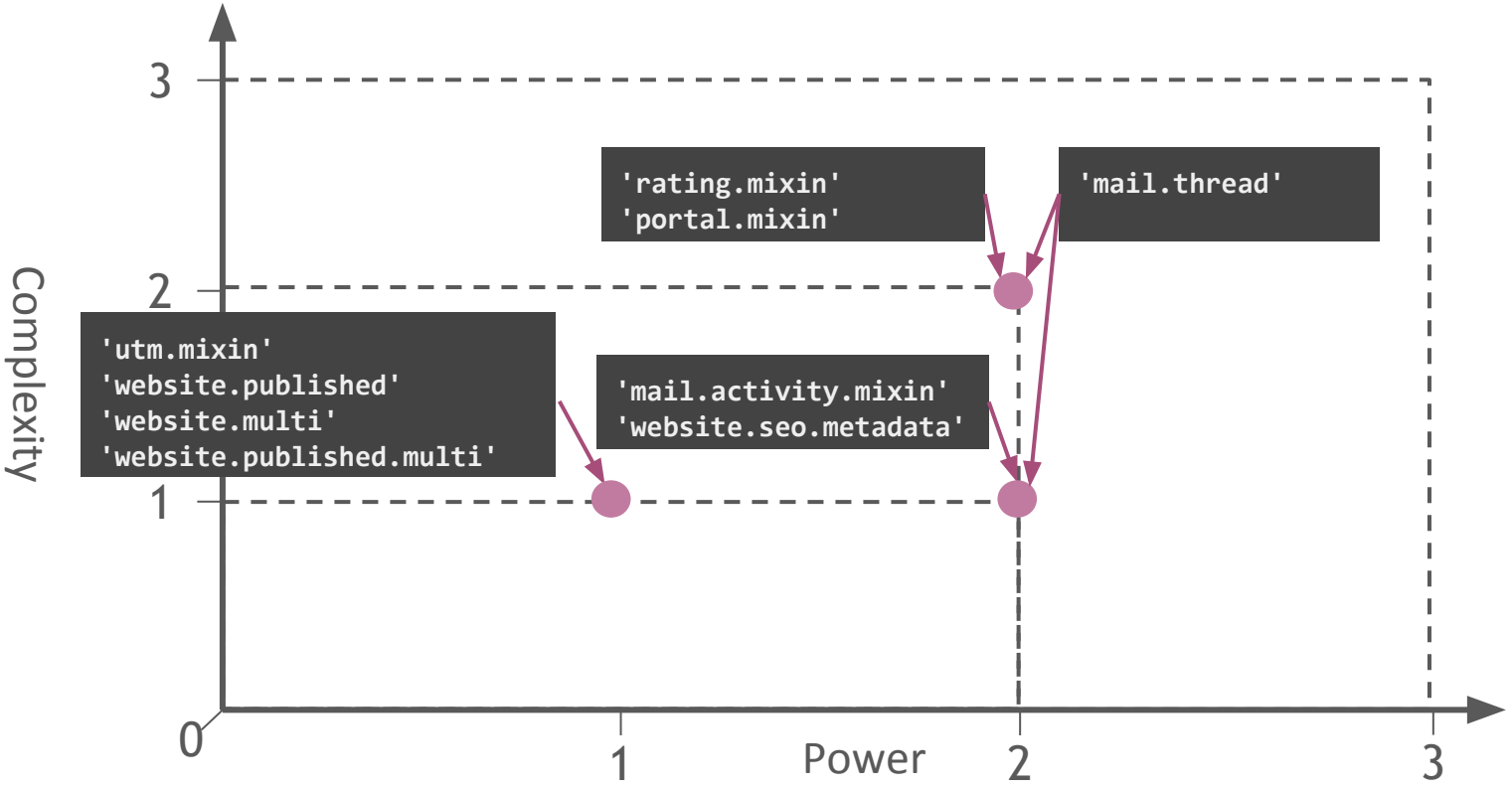
- Document - and App- specific portal url computation
- In your App
  - Portal Implementation...
  - access_token definition
  - Share button
- /mail/view controller check and redirection to portal

```python
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 text if the record can't be shared
        access_warning = ''

    def _compute_access_url(self):
        # Set the URL to reach the record on portal
        record.access_url = '#'
```

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

The graph


SO SCIENTIFIC
MUCH GRAPH
WOW
MANY LABELS
AMAZING DATA

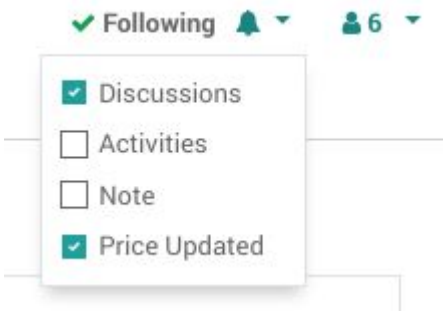

Complexity
3
2
1
0
'rating.mixin'
'portal.mixin'
'mail.thread'
'utm.mixin'
'website.published'
'website.multi'
'website.published.multi'
'mail.activity.mixin'
'website.seo.metadata'
1
Power
2
3

— Still Classy Cool Dev

5 Advanced Mail Thread

Mail Thread: Subtypes

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




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

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

```xml
<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 Thread: Tracking

- Track value change
- track_visibility on field
  - ~~onchange~~
  - ~~Always~~
  - True
- Automatic logging

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

    price = fields.Float('Price', track_visibility=True)
```

YourCompany, Mitchell Stephens - now

Price Updated

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

Tracking & Subtypes

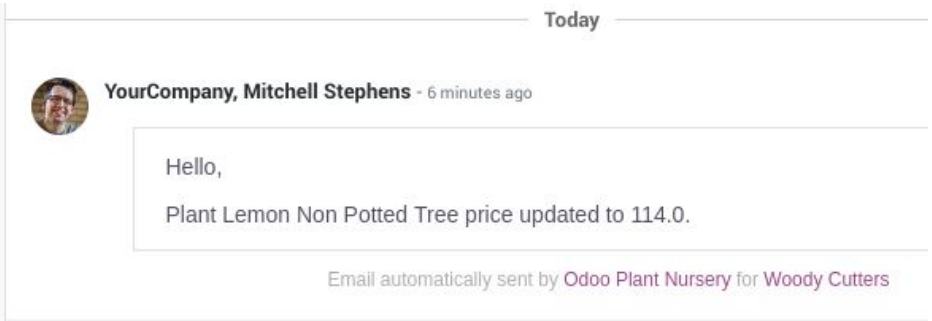
- Warn on some specific value change
- Link a set of tracking and a subtype
  - _track_subtype
  - return xml_id
- Message with tracking generated with the subtype
  - allow people to be notified

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

    price = fields.Float('Price',
        track_visibility='onchange')

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

Tracking & Mailing

- Email on some specific value change
- Link a set of tracking and an automatic mailing
  - _track_template
  - give a mail.template and 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', 'rating.mixin']

    price = fields.Float('Price',
        track_visibility='onchange')

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



Mail Alias

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

```
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

```python
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
```

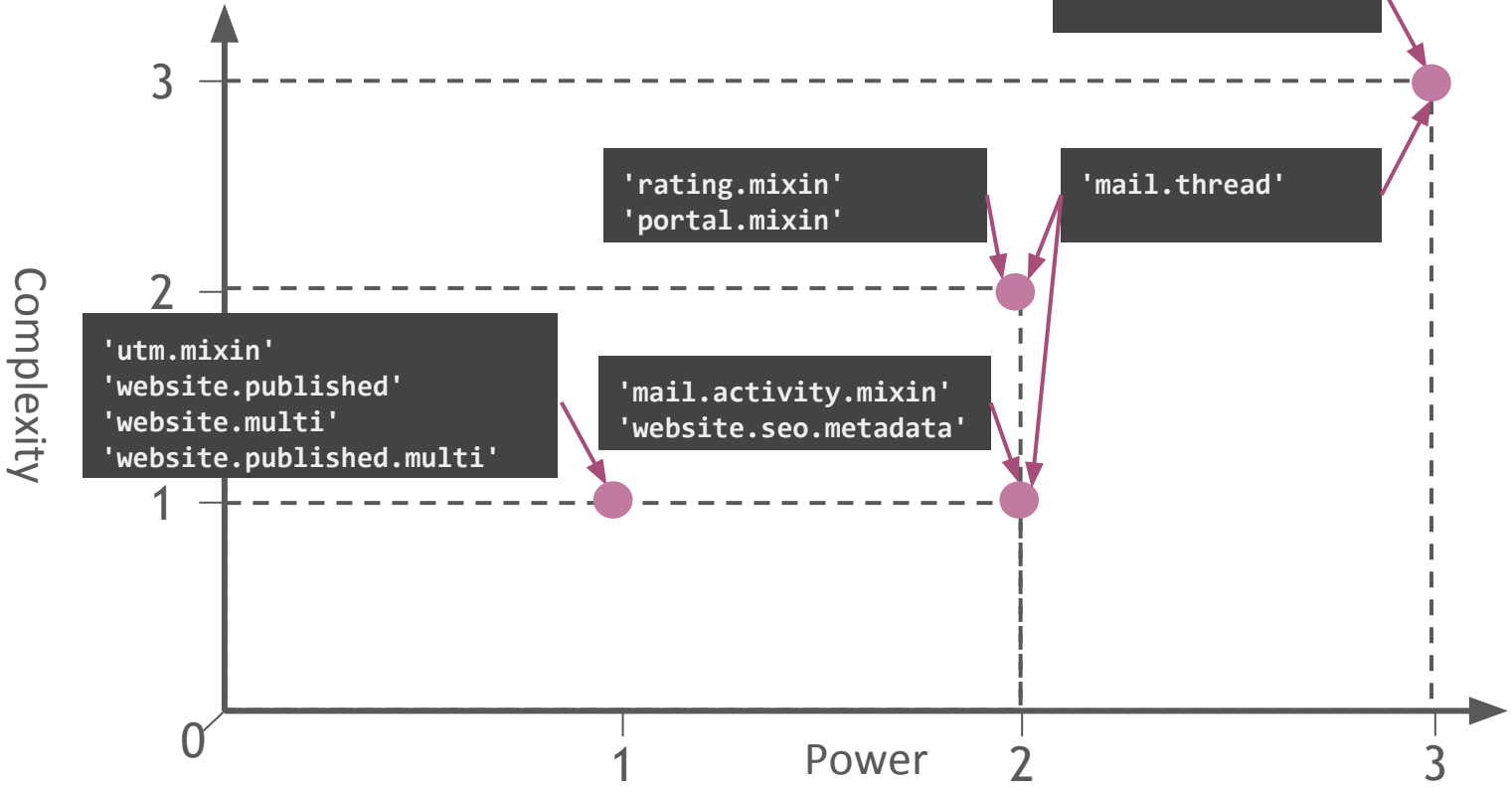
```python
class MailAlias(models.AbstractModel):
    _name = 'mail.alias.mixin'
    _inherit = {'mail.alias' : 'alias_id'}

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

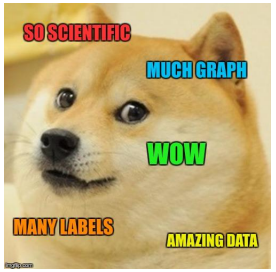
Name	Palm
Alias	yti+xmas@odoo.com

6

I have a headache

The graph


'mail.alias.mixin'
'rating.mixin'
'portal.mixin'
'mail.thread'
'utm.mixin'
'website.published'
'website.multi'
'website.published.multi'
'mail.activity.mixin'
'website.seo.metadata'
Complexity
Power
0
1
2
3


SO SCIENTIFIC
MUCH GRAPH
WOW
MANY LABELS
AMAZING DATA

2018

odoo EXPERIENCE

Thank you.

https://github.com/tivisse/odoodays-2018

#odooexperience

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