use below for passing values inside controller

***use below for passing values inside controller
<a class="btn btn-primary" t-attf-href="/permanent/disconnection/request?gas_account={{gas_status.name}} &gas_status={{'PD'}}"" role="button">Disconnection

to get the key,value in selection field
feedback_rating_values = dict(self.env['ebs.feedback']._fields['rating'].selection)
for key_id, value_name in feedback_rating_values.items():
_logger.info(key_id)
_logger.info(value_name)
rating_values.append({'key_id': key_id,
'value_name': value_name})

Sign up - Reset Password

To Check Uploaded files are public
select create_uid,reference,* from ir_attachment
where
type='binary' and
public = true
and res_model = 'ir.ui.view'
and reference is not null

Update : Uploaded attachment from public to private
update ir_attachment set public = false
where id in (select id from ir_attachment
where
type='binary' and
public = true
and res_model = 'ir.ui.view'
and reference is not null)

copy=False
If you put copy=False to a field, When you duplicate a record, That value is not copied to duplicating (Newly created)record.

ondelete='cascade'

If you put this to a Many2one field, the deletion of record will cause to deletion of one2one records according to that record.

to get the latest record

```
attachment_ids = self.env['ir.attachment'].search([('reference', '=', gsa_name)],
limit=1,order='create_date desc')
```

Onchange Function Get current ID
This can be found at self._origin.id

https://linuxhint.com/check-if-object-empty-javascript/

```
@api.constrains('payment_reference')
def check_payment_reference(self):
for li in self:
gas_account_status = self.env['ebs.customer.gas.account.status'].search([('payment_reference', '=',
li.payment_reference)])
if gas_account_status:
if len(gas_account_status) > 1:
raise ValueError(('Payment Reference %s already exist!' % (li.payment_reference)))
```

```
email_template_obj = self.env['mail.template'].browse(template_id)
# added_attachments = self.env['ir.attachment'].browse(self.doc_attachment_id.ids)
# for i in email_template_obj.attachment_ids:
# email_template_obj.write({"attachment_ids": [(3, i.id)]})
# for attachment in added_attachments:
# for new_attachment in attachment:
# email_template_obj.write({"attachment_ids": [(4, new_attachment.id)]})
email_template_obj.send_mail(self.id, force_send=True)
```

passing many2many fields from many2many

```
"approval_person_ids": [(6, 0, type.approval_person_ids.ids)],
```

filtered

```
partner_ids = self.env['res.partner'].search([])
supplier_ids = partner_ids.filtered(lambda l: l.supplier)
customer_ids = partner_ids.filtered(lambda l: l.customer)
```

In the above example, the variable partner_ids stores all the records from the table ‘res.partner’ as a recordset. We need to get all the suppliers from this same recordset as a separate recordset to manipulate some other kinds of operations. For that, we do not need to search the entire database table (res_partner) again. We will be able to find out the suppliers from this same record set with the help of filtered and lambda functions. The same thing can be done for the customers also by checking if the field ‘customer’ is True.

mapped()

mapped() takes a very important role in code optimization. It avoids user-defined loops. It returns a set of field values. Its argument accepts a string value as the column name and returns all the possible values from the recordset.

```
order_ids = self.env['sale.order'].search([])
order_names = order_ids.mapped('name')
```

In the above snippet of codes, order_ids stores all the values from the model sale.order. We have to get all the values from the field name to do some other stuff. Then we can use the function mapped() to achieve this. It returns a list with all possible names of the table sale.order. The same function returns an object reference if we are using any Many2one fields like partner_id. Overall, the mapped() will check the behavior (type) of the given field and will return the values as an object or list based on the argument(field) type.

4 days less than today

```
('date_completed','<=',(context_today() - relativedelta(days=4)).strftime('%Y-%m-%d'))
```