**Context** is a dictionary carries some information like *user ID, user language, Timezone,* etc by default. You can pass any kind of data/information in context as per your need either from xml or from python (methods) files and then based on that extra information you can write your process code.

Here, I am going to explain how to use context for relational fields and context to pass when fetching possible values with examples.

1. Use the context to pass in default values for relational object fields.

* ?Set value in many2one field’s child field as default by using keyword *defualt\_<many2one field’s child field name>.*
  + *For example, set company in journal\_id’s many2one field company\_id*.

<field name="journal\_id" context="{'default\_company\_id': company\_id}"

* Use a context value in python function by fetching value from xml.
  + *For example, suppose you need to access sale order field value directly into sale order line method through context. For that you can pass the sale order field value in order lines context through xml and access it in a python function as per below code.*

Pass value in context

<page string="Order Lines">

<field name="'partner\_id'" invisible="1"/>  <!-- sale order field-->

   <field name="line\_ids" context="{'partner\_id': partner\_id}">

       <form string="Sales Order Lines">

          <!-- order lines fields -->

        </form>

    </field>

</field>

</page>

Access value in python function

def your\_order\_line\_function(self):

  # to fetch sale order partner from context.

  partner = self.env.context.get('partner\_id')

  # rest of the code

  return True

1. Use the context in  *\_defaults* method to set any field value.

\_defaults = {

       'order\_partner\_id' : context.get('partner\_id', False), #return id or False

   }

1. Use the context to pass default values for normal fields.

* *For example, Set default value in partner name.*

<field name="partner\_id" context="{'default\_name': ‘Agrolait’}">

1. Use for Context is on buttons to pass parameters to object functions.

<button string="Submit" type="object" name="my\_func" context="{'param' : 'param\_val'}" />

1. Use for Context to set default filter and group by records.

* *For example, Set default filter to display only leave allocation request and group by all records with employees.*

<record id="holidays\_allocation\_approve" model="ir.actions.act\_window">

           <field name="name">Leaves Allocation</field>

           <field name="res\_model">hr.holidays</field>

           <field name="context">{'default\_type':'add', 'group\_by':'employee\_id'}</field>

</record>

**Domain** is a condition which is used to filter your data or for searching. A domain expression is a list of conditions. Each condition is a (*'field\_name'*, *'operator'*, *‘value'*) tuple.

Where,

* **field\_name** : The field name in which you want to set domain.
* **value** : The value is evaluated as a Python expression. It can use literal values, such as numbers, booleans, strings, or lists, and can use fields and identifiers available in the evaluation context.
* **operator** :
  + The usual comparison operators are < , > , <= , >= , = , != .
  + '=like' matches against a pattern, where the underscore symbol, \_ , matches any single character, and the percentage symbol, % , matches any sequence of characters.
  + 'like' matches against a '%value%' pattern. The 'ilike' is similar but case insensitive.
  + The 'not like' and 'not ilike' operators are also available.
  + 'child of' finds the children values in a hierarchical relation, for the models configured to support them.
  + 'in' and 'not in' are used to check for inclusion in a given list, so the value should be a list of values. When it is to be used When it is to be used in "to-many" relation field the ‘in’ operator behaves like a CONTAINS operator.

Here, I am going to explain how to use domain for relational fields and filters to apply when displaying existing records for selection with examples.

1. Use the domain to filter relational object fields records with condition.

* Suppose you need to display only customers in partner field, so you can set it in domain as per below code.

In xml file

<field name="partner\_id" domain="[('customer','=',True)]" />

or

In python file

partner\_id = fields.Many2one('sale.order’, string='Customer',domain=[('customer', '=', true)])

* Suppose you need to display projects for selected client/partner only, if partner is not selected then display all projects. In that case you can use the ternary operator in domain condition as shown in below code.

<field name="project\_ids" widget="many2many\_tags" domain="[('partner\_id','=?',partner\_id)]"/>

* Use parent object field value in relational field using domain.

<field name="tax\_id" widget="many2many\_tags" domain="{[('company\_id','=',parent.company\_id)]}"/>

* Use dynamic domain through onchange function by following code.

@api.onchange('project\_ids')

   def \_onchange\_project\_ids(self):

       domain = {}

       partner\_list = []

       if not self.project\_ids:

           partner\_obj = self.env['res.partner'].search([('customer', '=', True)])

           for partner\_ids in partner\_obj:

               partner\_list.append(partner\_ids.id)

          # to assign parter\_list value in domain

           domain = {'partner\_id': [('id', '=', partner\_list)]}

       return {'domain': domain}

1. Use the domain display specific records in listview based on condition.

* *For example to display only those orders which are in ‘sale’ state in sale order list view.*

<record id="action\_orders" model="ir.actions.act\_window">

           <field name="name">Sales Orders</field>

           <field name="type">ir.actions.act\_window</field>

           <field name="res\_model">sale.order</field>

           <field name="view\_type">form</field>

           <field name="view\_mode">tree,kanban,form,calendar,pivot,graph</field>

           <field name="search\_view\_id" ref="view\_sales\_order\_filter"/>

           <field name="domain">[('state', 'in', (‘sale’))]</field>

</record>

1. Use the domain in search view filter.

<search string="Product">

      <!-- searches on multiple fields at once -->

      <field name="name" string="Sales Order" filter\_domain="['|',('product\_variant\_ids.name','ilike',self),('product\_variant\_ids.attribute\_value\_ids.name','ilike',self)]"/>

      <filter string="Services" name="services" domain="[('type','=','service')]"/>

</search>

1. Use the domain in *fields\_view\_get()* method to set dynamic values for domain filters for a particular field.

* *For* *example set department id of the login users.*

def fields\_view\_get(self, cr, uid, view\_id=None, view\_type='form', context=None, toolbar=False, submenu=False):

   login\_user\_dpt\_id = users\_obj.browse(cr, SUPERUSER\_ID, uid, context=context).emp\_department\_id

   for node in eview.xpath("//field[@name='user\_id']"):

       if login\_user\_dpt\_id:

           user\_filter =  "[('emp\_department\_id', '='," + str(login\_user\_dpt\_id.id) + " )]"

           node.set('domain',user\_filter)

1. Use the domain in record rule.

* *For example, set domain force to display own payslip which is in draft and done state to that login user.*

<record model="ir.rule" id="property\_rule\_employee\_payslip">

      <field name="name">Employee Payslip</field>

      <field name="model\_id" ref="model\_hr\_payslip"/>

      <field name="domain\_force">[('employee\_id.user\_id', '=', user.id),('state','in',['done','draft'])]</field>

      <field name="perm\_create" eval="False"/>

      <field name="perm\_write" eval="False"/>

      <field name="perm\_unlink" eval="False"/>

      <field name="groups"  eval="[(4,ref('base.group\_hr\_user')),(4,ref('base.group\_user'))]"/>

</record>

Basically **domain** is a filter you cannot expand beyond - **context** is a filter you can expand to the point that domain allows