Unité d'Enseignement en Informatique
Année 2012-2013

Master M1 EFREI – ASI – BI
Devoir Écrit de ERP – Second Session June 3, 2013
(2h00 duration, no document allowed)

Rule: *All answers must be written in English.*

Exercise 1: (5 points)

1) The chronology of software history for Information Systems in companies can be divided in 4 main periods:
   a) 1960 – 1975: … b) 1975 – 1990: … c) 1990 – 2000: … d) Since 2000: …
   Give a title to each of these periods?
2) What is the role of the workflow engine of an ERP?
3) Give two advantages of having only one program to manage all the back-office of a company?
4) What is a BOM (a Bill Of Material)?

Exercise 2: (4 points)

1) We often say that using a proprietary ERP makes a company dependent to the editor. What are the concrete consequences of this "dependency" for the company who is using the ERP software?
2) Is it better for a company to be dependent on a consultant and an editor, or to be dependent on its internal skills?

Exercise 3: (2 points)

1) The situation to model is the following: a brand sells several kinds of t-shirts, and a t-shirt is sold by one (and only one) brand. What relational type would you use to declare a new field `brand_id` in the t-shirt model?
2) When building the information system of a company, what the webservices functionalities of OpenERP can be used for?
3) What are the 5 methods of the ORM (Object-Relational Mapping) which can be called remotely through the webservices of OpenERP?

Exercise 4: (9 points)

The module named "Idea" in OpenERP allows collecting, across time, new ideas of employees, thought an informal way. Thus, these ideas are not forgotten and can help the progression of the company one day. In the following, you will find a part of the source code of this module: the full `idea.py` file and an extract of the `idea_view.xml` file.

In the idea module, the evolution of an idea starts from being a draft document after creation, then being opened, and finally ends by being closed or cancelled. In order to improve this small module for the management of ideas, we would like to add a rating mechanism once a proposed idea has been opened. To do so, one field has to be added to the class idea. This field will not be set by the user creating the idea, but by the user opening the idea (i.e. the field will not be readonly in open state). To define the possible values and the default value for ratings, we will simply use the two global variables declared on lines 6-7-8 of `idea.py` (which are currently defined but not used in the idea module).

To describe your work, for the next four questions, you are going to:

- Give the name of the file and the number of the line you want to edit:

  "In the file *xxx.xx*, I replace the line *n* by ….."
- Give the name of the file and the number of the lines where you want to add new code:

  "In the file *xxx.xx*, between lines *n* and *n+1*, I add the lines ….."

Don't forget to read carefully the documentation provided in this document.

1) Add the new field `rate` to the idea model.

2) Modify the method `idea_draft()`, which is called when entering in draft state, in order to set the field `rate` to the not voted value. Modify the method `idea_open()`, which is called when entering in open state, in order to set the field `rate` to the default value.

3) Now add the possibility of rating an idea from the form view.

4) Define three new filters in order to:

- Filter not voted ideas
- Filter very good ideas
- Filter good or very good ideas (Attention, '75' and '100' are strings, not numbers!)

Documentation:

a) Type of field: selection

Description: A field which allows the user to make a selection between various predefined values.

Syntax:

```
fields.selection(values, 'Field Name' [, Optional Parameters])
```

where the parameter `values` is a list, or a tuple, of 2-tuples formed as `(key, value)`, by example:

```
values = [('n','Unconfirmed'),('c','Confirmed')]
```

or:

```
values = (('n','Unconfirmed'),('c','Confirmed'))
```

Remark: The predefined values can also be given directly:

```
fields.selection((('1','Choice 1'),('2','Choice 2'),('3','Choice 3')),
                 'Field Name', readonly=False)
```

b) The write method of OpenERP objects, which allows to affect values of fields:

(for example, it is called on lines 48, 51, 54 and 57 of `idea.py` in order to set the value of the field `state`)

`write``(cr, user, ids, vals, context=None)`

Update records with given ids with the given field values

Parameters:

- **cr** – database cursor
- **user** (integer) – current user id
- **ids** – object id or list of object ids to update according to vals
- **vals** (dictionary) – field values to update, e.g {'field_name': new_field_value, ...}
- **context** (dictionary) – (optional) context arguments, e.g. {'lang': 'en_us', 'tz': 'UTC', ...}

Returns: `True`

c) The domain of a filter can also evaluate several conditions using Boolean operations with prefix notation (also known as Polish notation). For example, to filter whether the field `country_code` is the one of Belgium OR France would be done like this:

```
['|',('country_code','=','be'),('country_ code','=','fr')]
```

idea.py

```
1  from openerp.osv import osv
2  from openerp.osv import fields
3  from openerp.tools.translate import _
4  import time
5
6  VoteValues = [('-1', 'Not Voted'), ('0', 'Very Bad'), ('25', 'Bad'), \
7                ('50', 'Normal'), ('75', 'Good'), ('100', 'Very Good') ]
8  DefaultVoteValue = '50'
9
10 class idea_category(osv.osv):
11     """ Category of Idea """
12     _name = "idea.category"
13     _description = "Idea Category"
14     _columns = {
15         'name': fields.char('Category Name', size=64, required=True),
16     }
17     _sql_constraints = [
18         ('name', 'unique(name)', 'The name of the category must be unique')
19     ]
20     _order = 'name asc'
21
22
23 class idea_idea(osv.osv):
24     """ Idea """
25     _name = 'idea.idea'
26     _inherit = ['mail.thread']
27     _columns = {
28         'create_uid': fields.many2one('res.users', 'Creator', required=True, readonly=True),
29         'name': fields.char('Idea Summary', size=64, required=True, readonly=True,
   oldname='title', states={'draft': [('readonly', False)]}),
30         'description': fields.text('Description', help='Content of the idea', readonly=True,
   states={'draft': [('readonly', False)]}),
31         'category_ids': fields.many2many('idea.category', string='Tags', readonly=True,
   states={'draft': [('readonly', False)]}),
32         'state': fields.selection([('draft', 'New'),
33             ('open', 'Accepted'),
34             ('cancel', 'Refused'),
35             ('close', 'Done')],
36             'Status', readonly=True, track_visibility='onchange',
37         )
38     }
39     _sql_constraints = [
40         ('name', 'unique(name)', 'The name of the idea must be unique')
41     ]
42     _defaults = {
43         'state': lambda *a: 'draft',
44     }
45     _order = 'name asc'
46
47     def idea_cancel(self, cr, uid, ids, context=None):
48         return self.write(cr, uid, ids, {'state': 'cancel'}, context=context)
49
50     def idea_open(self, cr, uid, ids, context={}):
51         return self.write(cr, uid, ids, {'state': 'open'}, context=context)
52
53     def idea_close(self, cr, uid, ids, context={}):
54         return self.write(cr, uid, ids, {'state': 'close'}, context=context)
55
56     def idea_draft(self, cr, uid, ids, context={}):
57         return self.write(cr, uid, ids, {'state': 'draft'}, context=context)
```

idea_view.xml

```
<!-- New Idea Form View -->

<record model="ir.ui.view" id="view_idea_idea_form">
  <field name="name">idea.idea.form</field>
  <field name="model">idea.idea</field>
  <field name="arch" type="xml">
    <form string="Idea" version="7.0">
      <header>
        <button name="idea_open" string="Open" states="draft" class="oe_highlight"/>
        <button name="idea_close" string="Accept" states="open" class="oe_highlight"/>
        <button name="idea_cancel" string="Refuse" states="open"  class="oe_highlight"/>
        <field name="state" widget="statusbar" statusbar_visible="draft,open,close"/>
      </header>
      <sheet>
        <label for="name" class="oe_edit_only"/>
        <h1><field name="name"/></h1>
        <label for="category_ids" class="oe_edit_only"/>
        <field name="category_ids" widget="many2many_tags"/>
        <label for="description"/><newline/>
        <field name="description"/>
      </sheet>
      <div class="oe_chatter">
        <field name="message_follower_ids" widget="mail_followers"/>
        <field name="message_ids" widget="mail_thread"/>
      </div>
    </form>
  </field>
</record>

<!-- New Idea Tree View -->

<record model="ir.ui.view" id="view_idea_idea_tree">
  <field name="name">idea.idea.tree</field>
  <field name="model">idea.idea</field>
  <field name="arch" type="xml">
  <tree colors="blue:state == 'draft';black:state in ('open','close');gray:state == 'cancel'" string="Ideas">
    <field name="name"/>
    <field name="create_uid"/>
    <field name="state"/>
  </tree>
  </field>
</record>

<!-- Search Idea  -->

<record model="ir.ui.view" id="view_idea_idea_search">
  <field name="name">idea.idea.search</field>
  <field name="model">idea.idea</field>
  <field name="arch" type="xml">
    <search string="Ideas">
      <field name="name" string="Idea"/>
      <filter icon="terp-document-new" string="New" domain="[('state','=', 'draft')]" help="New Ideas"/>
      <filter icon="terp-camera_test" string="In Progress" domain="[('state','=', 'open')]" help="Open Ideas"/>
      <filter icon="terp-check" string="Accepted" domain="[('state','=','close')]" help="Accepted Ideas" />
      <field name="category_ids"/>
      <group expand="0" string="Group By...">
        <filter icon="terp-personal" string="Creator" help="By Creators" context="{'group_by':'create_uid'}"/>
        <filter icon="terp-stock_symbol-selection" string="Category" help="By Idea Category" context="{'group_by':'category_ids'}"/>
        <filter icon="terp-stock_effects-object-colorize" string="Status" help="By States" context="{'group_by':'state'}"/>
      </group>
    </search>
  </field>
</record>
```