Unité d'Enseignement en Informatique
Année 2012-2013

Master M1 EFREI – ASI – ISI
Devoir Écrit de ERP – Deuxième Session du 3 Juin 2013
(durée 2h00, aucun document autorisé)

Consigne: *Réponses en Français ou en Anglais, terminologie Anglophone autorisée dans les deux cas.*

Exercice 1: (5 points)

1) La chronologie de l'histoire des logiciels pour les Systèmes d'Information des entreprises peut être découpée en 4 grandes périodes :
   a) 1960 – 1975 : … b) 1975 – 1990 : … c) 1990 – 2000 : … d) Depuis 2000 : …
   Donnez un titre à chacune de ces périodes ?
2) Quel est le rôle du moteur de workflow dans un ERP ?
3) Donnez deux avantages d'avoir un seul logiciel pour gérer tout le *back-office* (ou *l'arrière-boutique* en français) d'une entreprise ?
4) Qu'est-ce qu'un BOM (*Bill Of Material*, ou encore *nomenclature* en français) ?

Exercice 2: (4 points)

1) On dit souvent qu'utiliser un logiciel propriétaire rend l'entreprise dépendante de l'éditeur. Par quoi cette « dépendance » se traduit-elle concrètement pour l'entreprise qui utilise le logiciel ?
2) Est-il meilleur pour une entreprise d'être dépendante d'un consultant et d'un éditeur, ou d'être dépendant de ses compétences internes ?

Exercice 3: (2 points)

1) La situation à modéliser est la suivante : une marque peut vendre plusieurs vêtements, et un vêtement est vendu par une (et une seule) marque. Quel type relationnel utiliseriez-vous pour déclarer un nouveau champ `brand_id` dans le modèle du vêtement ?
2) Lors de la construction du système d'information d'une entreprise, en quoi les fonctionnalités de *webservice* (ou service web en français) d'OpenERP peuvent-elles être utiles ?
3) Quelles sont les 5 méthodes de l'ORM (Object-Relational Mapping) qui peuvent être appelées à distance grâce aux services web d'OpenERP ?

Exercice 4: (9 points)

Le module "Idea" d'OpenERP permet de collecter, au fil du temps, les idées nouvelles des employés, de manière informelle. Ainsi, ces idées ne sont pas oubliées et peuvent servir un jour à faire progresser l'entreprise. Dans la suite, vous trouverez des parties du code source de ce module : le fichier `idea.py` en entier et un extrait du fichier `idea_view.xml`.

Dans le module idea, l'évolution d'une idée commence par être un document brouillon (draft), puis un document ouvert (open) et enfin termine en étant close ou annulée (closed ou cancelled). Afin d'améliorer ce petit module de gestion d'idées, nous voudrions ajouter un mécanisme de notation une fois que l'idée proposée a été ouverte. Pour ce faire, un champ doit être ajouté à la classe idea. Ce champ ne sera pas entrés par l'utilisateur lors de la création de l'idée, mais une fois que l'idée a été ouverte (i.e. le champ ne sera pas readonly quand le champ `state` vaut `open`). Pour définir les valeurs possibles et la valeur par défaut pour la notation des idées, nous allons simplement utiliser les deux variables globales déclarées lignes 6-7-8 de `idea.py` (qui ne sont actuellement pas utilisées par le module idea).

Pour décrire votre travail, lors des quatre questions à suivre, vous procéderez ainsi :

- Donner le nom du fichier et le numéro de la ligne que vous souhaitez modifier:
  "Dans le fichier *xxx.xx*, je remplace la ligne *n* par ….."
- Donner le nom du fichier et les numéros des lignes où vous souhaiter insérer du code :
  "Dans le fichier *xxx.xx*, entre les lignes *n* et *n+1*, j'ajoute les lignes ….."

N'oubliez pas de lire attentivement la documentation fournie dans la suite du présent document.

1) Ajouter le nouveau champ `rate` au modèle idea.

2) Modifier la méthode `idea_draft()`, qui est appelée quand une idée entre dans l'état brouillon (draft), afin que le champ `rate` soit affecté avec la valeur de non vote. Modifier la méthode `idea_open()`, qui est appelée quand une idée entre dans l'état ouvert (open), afin que le champ `rate` soit affecté avec la valeur par défaut.

3) Ajouter maintenant la possibilité de noter une idée depuis la vue formulaire.

4) Définissez les trois filtres pour :

- Filtrer les idées non encore votées
- Filtrer les très bonnes idées
- Filtrer les idées bonnes ou très bonnes (attention, '75' et '100' sont des chaînes de caractères, pas des nombres !)

Documentation:

a) Type de champ: 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) La méthode write des objets OpenERP, qui permet d'affecter les valeurs des champs :
(par exemple, elle est appelée lignes 48, 51, 54 et 57 de `idea.py` pour donner sa valeur au champ `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) Le domaine d'un filtrer peut aussi évaluer plusieurs conditions en utilisant les opérateurs booléens en notation préfixée (aussi connue comme notation Polonaise). Par exemple, filtrer si le champ `country_code` vaut celui de la Belgique OU de la France serait fait comme suit :

```
['|',('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>
```