OeMedical Documentation

Release 0.1b

OeMedical Team

May 11, 2016

Contents

CHAPTER 1

Base Documentation

1.1 Introduction.

CHAPTER 2

Module oemedical

2.1 OeMedical Base Module Documentation.

2.2 Gynecologist

Within the module there are 4 different tabs:

2.2.1 Gyneco

This first menu provides a general overview of the basic gynecological conditions of the patient. We can discriminate whether the patient is in a fertil age, is currently pregnant or is menopausal.

2.2.2 Screening

Screening is a strategy used in a population to detect a disease in individuals without signs or symptoms of that disease. These tests are performed on people without any clinical sign of disease. The intention of screening is to identify early disease in a community, thus enabling earlier intervention and management in the hope to reduce mortality and suffering from a disease.

2.2.3 Pap / Colpo

The PAP test is a screening test used to detect potencially pre-cancerous and cancerous processes in the endocervical canal of the female reproductive system. Many premalignant lesions and malignant lesions in these areas have characteristics that can be detected through a colposcopy examination.

2.2.4 Obstetrics

This is a very complete menu that deals with the care of all women's reproductive tracts and their children through specific tabs of perinatal and puerperium info. This section considers important factors such as the number of pregnancies and prenatal evaulations, perinatal info, puerperium monitors or characteristics of the fetus.

CHAPTER 3

Module web_doc_oemedical

3.1 Web tools and Docs on OeMedical.

CHAPTER 4

OeMedical API.

4.1 Coding Guidelines

All the coding guidelines of OeMEdical is based on OpenERP guidelines for coding.

Some exceptions in Styles are in this code that you must to follow to approve merge proposals and ensure the quality of the system.

4.1.1 1 Coding Styles

What will be considered good or bad code to merge in core.

1.1 Your code should be pep8 compliant.

In order to be sure our code is of high wuality, is important use some conventions in the python world the most accepted is the pep8 convention, we will use it in this project.

If you develop in a linux enviroment you can install `pep8` program:

```
sudo apt-get install pep8
```

With this program you will be able to test your code before the merge proposals or even test the code of other person to help us to answer with Q&A reasons before commit or merge changes.

i.e:

```
pep8 yourpythonfile.py
```

It will return a set of results about the quality of your code see `pep8 --help` for more information.

1.2 About new objects

A new object is this one that for functional or design reasons is not included on the core of OpenERP.

1.2.1 New classes signature.

All new classes must start with `OeMedicalName` see it in *CamelCase*.

i.e.:

```
class OeMedicalAppointment(orm.Model):
```

1.2.2 New Object name.

All new objects will complay with OpenERP standard it means separated by one dot between words.

i.e.

```
    _name = 'oemedical.appointment'
```

1.2.3 New method signature.

All new methods will be named with *snake_case*.

1.2.4 File management for new objects

All new objects will be in a new folder, with both xml and py files, see `oemedical` modules for some example.

1.3 About Openerp Version.

In version 1.0 of OeMedical we decide work with version 7.0 of OpenERP, for this reason some specific conventions must be verified.

1.3.1 Views 7.0 compatibles.

All views that use OeMedical will comply with version 7.0 of OpenERP.

1.4 About Data for Standards.

Data loaded should be in an extra module called oemedical_yourstandard_data, to separate data errors from model errors, frequently data dont change too much, because it is based on Standards.

Warning: If the data is necesary for the correct work of your module this rule do not apply

1.5 About Complementary Modules.

Modules that improve functionality working with the core of openerp, should be in an extra module, it means.

i.e:

- Creation of Automated invoices and commercial management: oemedical_account
- Relation with sale orders: oemedical_sale
- Relation with purchase: oemedical_purchase

Avoid use names not self descriptive, as *oemedical_purchase_ext*, the correct sufix should say what it is improving.

1.6 Useability Guidelines.

If a model is inherited, we should create them own view with them own action, to be sure the user is able to use this model in its functional context without unnecesary information in this function.

1.7 Technical requirements.

If a model impact some kind of a more than 3 steps flow, the object `MUST` have Workflow.

4.1.2 2 Documentation Guidelines

In order to be sure the useabilty and to avoid not re-use the job of others members of community will not be merged any module or improvement that is not correct documented.

2.1 How Documentation will be included.

All documentation will be in a web_doc_yourmodule module to be able to embed documentation compiled with sphinx in the same server.

Note: TODO: Put some example or link to a branch with a basic template for this kind of module.

2.2 Guidelines to document your improvement.

We must try to document following pep-0257 convention.

As the standard is a little extense we share some useful snnipets, with some examples.

```
def test_method(self, cr, uid, ids, context=None):
    '''Something in doc

    :param context['mail']: 'new' to send a new mail or 'reply'
    :type context['mail']: str

    If this values are Ok.

    .. note::
        You must be careful with XX YY

    .. code-block:: python

        obj = self.pool.get('your.object')
        somethingelse

    :returns: list -- list with ids of elements
    '''
    return True
```

It is important understand that the documentation with pep standards just propose some good practices from the technical point of view, to be sure all is correct and all community involved has the correct approach about what document and how, some premises are listed below, and the community can propose another ones to follow, in the middle this methodology can be reviewed and improved.

2.3 About comments in code:

At least the model is still on development, we must avoid code commented, all commentaries must be usuables by sphinx to use autodoc embebed.

Try to commit with a really explicit message to avoid unecessary comments

4.1.3 OeMedical API Documentation.