Open Object Community Book

Release 1.0

Tiny SPRL

2009-04-09

CONTENTS

Part I

Introduction

Here you will found informations about the organisation of the community in OpenERP project. It include a description of the different tools used, the role of the differents actors, and the different process for improvement management.

Part II

Our Open Source Vision

As to build up the best enterprise management software ever developed, we need a perfect organisation between all Open ERP's actors. So that we can benefit and leverage the contributions and feedbacks from the community, the market knowledge and creation from partners and the quality control and vision of an editor.

We focus on creating a fully Open Source development methodology. This page summarize the different community efforts on the Open ERP and Open Object projects. For more information, check "How to contribute".

Part III

Summary of resources

Part IV

Working in teams

CHAPTER

ONE

PEOPLE

Who are the different actors in the community of OpenERP.

1.1 Contributors

Contributors are people who wants to help the project getting better, add functionnality and improve stability. Everyone can contribute on the project with his own knowledge by reporting bugs, purposing smart improvment and posting patch.

The community team is available on launchpad: https://launchpad.net/~openerp-community

Member of the quality and commiter team are automatically members of the community.

1.2 Official Commiters

Official commiters are people allowed to commit in the community repository. Those people are approved as commiters by the community once they've done enought good patch and/or contributions in the project.

Their allow to :

- Purpose and post their own patch on bugs report.
- Review patches from contributors, comments and / or improve them.
- Commit well done patch or contribution in the community repository.
- Write a news on the Planet OpenERP RSS

The commiter team is available on launchpad: https://launchpad.net/~openerp-commiter

1.3 Quality Team

Quality team are people from Tiny sprl responsible for the quality of the Official repository. They assume the stability and coherence of the Official version by review the community patch and commit.

They're in charge to merge the commit from Community repository to Official repository. "Extra_addons" and "Others" repository are not part of their responsability.

If a commit isn't good enough, they must remove it from Community repository and report it in the *bug tracker*. In other case, they will apply the purposed commit in the Official version. Like this, everyone will stay aware about what is or isn't accepted.

The quality team is available on launchpad: https://launchpad.net/~openerp

CHAPTER

TWO

TEAMS

2.1 Introduction

As to help developers and contributors to take the right decision when improving OpenERP, we set up experts teams for different management domains. Only people that have a strong experience in OpenERP and the related domain can apply as an expert. We have teams of accountants, manufacturing experts, technical experts, services management experts, ...

Developers can contact our experts mailing list when they need feedback on particular features to be developed. Please contact our experts only for new developments related questions. They don't provide help on current features of OpenERP. Most of our experts have very high positions in the company they work for, so they don't have time to spent providing help or support.

2.2 Expert Teams

- Accounting: https://launchpad.net/~openerp-expert-accounting/+members
- Services Management: https://launchpad.net/~openerp-expert-service/+members
- Manufacturing Industries: https://launchpad.net/~openerp-expert-production/+members

2.2.1 Requesting Advices to A Team

When you create a specification of a new feature on launchpad (called a blueprint), you can assign an expert team as a drafter of the specifications. Then, you can click on request feedback on your blueprint and assign this to an expert team.

They will receive a notification email and will discuss about the requested feature. The team will improve your specifications directly in your blueprint.

2.3 Translators team

2.4 Community Team

Part V

The Planets and announcements

Contents

CHAPTER

THREE

WHAT ARE THE PLANETS ?

The planets are the place where partners, developers and contributors express themselves about the Open Object project. It's an aggregate of the personnal blogs of partners and contributors. We use them to announce new developments, features and events on Open ERP and Open Object.

We have two planets:

- Open Object, the community planet : http://openobject.com/planet
- Open ERP, the editor planet : http://openerp.com/planet

Every Open ERP contributor can post blogs on the Open Object planet, if you follow the writing guidelines. See the section bellow to subscribe.

Our marketing and editorial team will review the Open Object planet frequently. When they notice good announcement, they will write a good announce and publish on the Open ERP planet. If the new feature is very good, we may also write a press announce based on our article.

CHAPTER

FOUR

WRITING GUIDELINES

To post on the Open Object planet, you must:

- Create a personnal blog
- Subscribe your blog on the planet by sending an email to fp AT openerp.com
- Write blogs and attach the label “openobject” to your posts

Some rules to follow:

- All blog entries to be published in the planet must be in english
- Announce only new events, new features or new developments
- Promote the work you did primarily, not the company you represent
- Only write positive messages, not negatives ones
- Avoid commercial messages in the planet. The only accepted commercial messages accepted are those from partners that announce trainings, exhibitions or conferences.

If you don’t follow these rules, we may remove you from the planet.

Part VI

Bazaar, the version control system

The new development process uses Bazaar via launchpad.net instead of Subversion. Bazaar offers a flexibility with this distributed model. You can see our branches on https://code.launchpad.net/~openerp.

Explanation of directories:

Two teams have been created on launchpad:

- OpenERP quality teams –> they can commit on:
  - lp:~openerp/openobject-addons/4.2
  - lp:~openerp/openobject-addons/trunk
  - lp:~openerp/openobject-addons/4.2-extra-addons
  - lp:~openerp/openobject-addons/trunk-extra-addons
  - lp:~openerp/openobject-bi/trunk-addons
  - lp:~openerp/openobject-bi/trunk-cli
  - lp:~openerp/openobject-bi/trunk-client-web
  - lp:~openerp/openobject-client/4.2
  - lp:~openerp/openobject-client/trunk
  - lp:~openerp/openobject-client-web/4.2
  - lp:~openerp/openobject-client-web/trunk
  - lp:~openerp/openobject-server/4.2
  - lp:~openerp/openobject-server/trunk
- 0penERP-commiter –> they can commit on:
  - lp:~openerp/openobject-addons/4.2-extra-addons
  - lp:~openerp/openobject-addons/trunk-extra-addons

In this group, we include some of our partners who will be selected on a meritocracy basis by the quality team.

- Contributors –> they can commit on:
  - lp:~openerp-community

How can I be included in OpenERP-commiter team ?

Any contributor who is interested to become a commiter must show his interest on working for openerp project and his ability to do it in a proper way as the selection for this group is based on meritocracy. It can be by proposing bug fixes, features requested on our *bug tracker* system. You can even suggest additional modules and/or functionalities on our *bug tracker* system.

How can I suggest some additionals modules or functionalities ?

To create some additionals modules and/or functionnalities and include them in the project, this is the way to do:

1. open a branch in launchpad
2. report and suggest your work via your new branch to our *bug tracker* system (there are two way : bugs report for bug and blueprint for idea / functionnality)
3. wait for approval by our quality team

Or the quality team approved your work and merge it into the official branch (like explained in the *bug tracker* section), or they refused it and ask you to improve your work before merging it in our official branch.

CHAPTER

FIVE

INSTALLING BAZAAR

Get Bazaar version control to pull the source from Launchpad.

To install bazaar on any ubuntu distribution, you can edit /etc/apt/sources.list by

```
sudo gedit /etc/apt/sources.list
```

and put these lines in it:

```
deb http://ppa.launchpad.net/bzr/ubuntu intrepid main
deb-src http://ppa.launchpad.net/bzr/ubuntu intrepid main
```

Then, do the following

```
sudo apt-get install bzr
```

To work correctly, bzr version must be at least 1.3. Check it with the command:

```
bzr --version
```

If you don't have at least 1.3 version, you can check this url: http://bazaar-vcs.org/Download On debian, in any distribution, the 1.5 version is working, you can get it on this url: http://backports.org/debian/pool/main/b/bzr/bzr_1.5-1~bpo40+1_i386.deb

If you experience problems with Bazaar, please read the *F.A.Q on Bazaar version control system* (in *Open Object Community Book*) before asking any questions.

CHAPTER

SIX

QUICK SUMMARY

This is the official and proposed way to contribute on OpenERP and OpenObject.

To download the latest sources and create your own local branches of OpenERP, do this:

```
bzr branch lp:openerp
cd openerp
./bzr_set.py
```

This will download all the component of openerp (server, client, addons) and create links of modules in addons in your server so that you can use it directly. You can change the bzr_set.py file to select what you want to download exactly. Now, you can edit the code and commit in your local branch.:

```
EDIT addons/account/account.py
cd addons
bzr ci -m "Testing Modifications"
```

Once your code is good enough and follow the *Coding Guidelines*, you can push your branch in launchpad. You may have to create an account on launchpad first, register your public key, and subscribe to the openerp-community team. Then, you can push your branch. Suppose you want to push your addons:

```
cd addons
bzr push lp:~openerp-community/openobject-addons/YOURLOGIN_YOURBRANCHNAME
bzr bind lp:~openerp-community/openobject-addons/YOURLOGIN_YOURBRANCHNAME
```

After having done that, your branch is public on Launchpad, in the OpenObject project, and commiters can work on it, review it and propose for integration in the official branch. The last line allows you to rebind your branch to the one which is on launchpad, after having done this, your commit will be applied on launchpad directly (unless you use `-local`):

```
bzr pull    # Get modifications on your branch from others
EDIT STUFF
bzr ci    # commit your changes on your public branch
```

If your changes fixe a public bug on launchpad, you can use this to mark the bug as fixed by your branch:

```
bzr ci --fixes=lp:453123   # Where 453123 is a bug ID
```

Once your branch is mature, mark it as mature in the web interface of launchpad and request for merging in the official release. Your branch will be reviewed by a commiter and then the quality team to be merged in the official release.

CHAPTER

SEVEN

HOW TO GET THE LATEST TRUNK SOURCE CODE

Get a clone of each repository:

```
bzr clone lp:~openerp/openobject-server/trunk server
bzr clone lp:~openerp/openobject-client/trunk client
bzr clone lp:~openerp/openobject-client-web/trunk client-web
bzr clone lp:~openerp/openobject-addons/trunk addons
```

If you want to get a clone of the extra-addons repository, you can execute this command:

```
bzr clone lp:~openerp-commiter/openobject-addons/trunk-extra-addons extra-addons
```

run the setup scripts in the respective directories:

```
python2.4 setup.py build
python2.4 setup.py install
```

Currently the initialisation procedure of the server parameter –init=all to populate the database seems to be broken in trunk.

It is recommended to create a new database via the gtk-client. Before that the web-client will not work.

Start OpenERP server like this:

```
./openerp-server.py --addons-path=/path/to/my/addons
```

The `bin/addons` will be considered as default addons directory which can be overriden by the `/path/to/my/addons/`. That is if an addon exists in `bin/addons` as well as `/path/to/my/addons` (custom path) the later one will be given preference over the `bin/addons` (default path).

CHAPTER

EIGHT

HOW TO COMMIT YOUR WORK

If you want to contribute on OpenERP or OpenObject, here is the proposed method:

- You create a branch on launchpad on the project that interest you. It's important that you create your branch on launchpad and not on your local system so that we can easily merge, share code between projects and centralize futur developments.
- You develop your own features or bugfixes in your own branch on launchpad. Don't forget to set the status of your branch (new, experimental, development, mature, ...) so that contributors knows what they can use or not.
- Once your code is good enough, you propose your branch for merging
- Your work will be evaluated by one responsible of the commiters team.
  - If they accept your branch for integration in the official version, they will submit to the quality team that will review and merge in the official branch.
  - If the commiter team refuses your branch, they will explain why so that you can review your code to better fits guidelines (problem for futur migrations, ...)

The extra-addons branch, that stores all extra modules, is directly accessible to all commiters. If you are a commiter, you can work directly on this branch and commit your own work. This branch do not require a validation of the quality team. You should put there your special modules for your own customers.

If you want to propose or develop new modules, we suggest you to create your own branch in the openobject-addons project and develop within your branch. You can fill in a bug to request that your modules are integrated in one of the two branches:

- extra-addons branch : if your module touches a few companies
- addons : if your module will be usefull for most of the companies

We invite all our partners and contributors to work in that way so that we can easily integrate and share the work done between the different projects.

CHAPTER

NINE

USE CASE DEVELOPPERS

This page present the approach your should follow on how to contribute in OpenObject. Suppose you want to develop new features in the addons or simply correct some bugfixes.

If you have the right to modify directly the branch you plan to change, you can do it directly. For example, a quality team member doing a bugfix can do it directly on the main branch. Or commiters can work directly on the extra-addons. If you don't have the right to modify the branch you plan to change or if you want to branch because you are starting big developments that may break the code, the first thing to do is to branch the repository you plan to modify:

```
bzr branch lp:openobject-addons lp:~openerp-commiter/openobject-addons/trunk-new-reporting
```

In that case, the branch created will be for the openerp-commiter team. If you are not a commiter, you can create the branch for the community team openerp-community or just for youself, depending if you accept people to directly commit on your branch or not. For all Tiny employees, we propose to create all branches for the team openerp-commiter. An OpenERP service company may create a team for their company and create branches at the name of their team. This will allow them to avoid others people that will change their customer branch.

Once the branch is created, you must checkout a local copy to work on:

```
bzr co lp:~openerp-commiter/openobject-addons/trunk-new-reporting
```

This will download the branch on your local computer. You can then start developing on it. From time to time, you should commit the work done:

```
bzr ci
```

This will send your modification to the branch: lp:~openerp-commiter/openobject-addons/trunk-new-reporting. Don't forget to change the status of the branch to show others contributors the status of your current work on https://code.launchpad.net/~openerp-commiter/openobject-addons/trunk-new-reporting

For instance, you can switch the status to "In Development" to show you are working on it and put the status to "Mature" when you'd like to have your code integrated in the official release.

During your development, if you want to receive the latests modifications from the parent branches, you can merge it:

```
bzr merge
```

Once your development on this branch are ok, you can ask a commiter to review and merge it or fill in a bug in the bugtracker. A commiter will then review your work and merge it to the official branch if it's good enough.

Part VII

Developing modules

CHAPTER

TEN

INTRODUCTION

Here you will found informations about the organisation of the community in OpenERP project. It include a description of the different tools used, the role of the differents actors, and the different process for improvement management.

The whole organisation is managed through our launchpad projects: http://launchpad.net Our projects on launchpad are currently organised like this:

Project name	URL	Description
openobject	https://launchpad.net/openobject	the main super-project (group) where all bugs, features and faq are managed
openobject-bi	https://launchpad.net/openobject-bin	business intelligence project
openobject-server	https://launchpad.net/openobject-server	the openobject server
openobject-client	https://launchpad.net/openobject-client	the openobject application client (gtk)
openobject-client-web	https://launchpad.net/openobject-client-web	the openobject web client (previously called eTiny)
openobject-addons	https://launchpad.net/openobject-addons	the project for all modules about openobject
openerp	https://launchpad.net/openerp	packaging around openobject (a selection of modules to build applications)

CHAPTER

ELEVEN

GETTING SOURCES

Please refer to *How to get the latest trunk source code* in the Bazaar section

If you don't know the Bazaar version control system, please read the *documentation on Bazaar*

CHAPTER

TWELVE

CODING GUIDELINES

12.1 Development Guidelines

12.1.1 Modules

Organisation of files in modules

The structure of a module should be like this:

```
/module_name/
/module_name/__init__.py
/module_name/__terp__.py
/module_name/module.py
/module_name/module_view.xml
/module_name/module_wizard.xml
/module_name/module_report.xml
/module_name/module_data.xml
/module_name/module_demo.xml
/module_name/module_security.xml
/module_name/wizard/
/module_name/wizard/__init__.py
/module_name/wizard/wizard_name.py
/module_name/wizard/wizard_name_view.xml
/module_name/wizard/wizard_name_workflow.xml
/module_name/report/
/module_name/report/__init__.py
/module_name/report/report_name.sxw
/module_name/report/report_name.rml
/module_name/report/report_name.py
```

Objects and fields namings

Security

Each object defined in your module must have at least one security rule defined on it to make it accessible.

Preventing SQL Injection:

Care must be taken not to introduce SQL injection vulnerabilities to SQL queries. SQL Injection is a kind of vulnerability in which the user is able to introduce undesirable clauses to a SQL query (such as circumventing filters or executing **UPDATE** or **DELETE** commands) due to incorrect quoting in the application.

In order to prevent SQL injection you need to be cautious when constructing SQL query strings. Good advices are to use %d and %f when only numbers are to be substituted and always use psycopg formatting parameters. For example the following expression is incorrect:

```
cr.execute( "SELECT * FROM table_name WHERE name='%s'" % client_supplied_string )
```

and

```
cr.execute( "SELECT * FROM table_name WHERE name=%s", client_supplied_string )
```

should be used instead.

12.1.2 Development

Coding Guidelines

Follow Python PEP 8: http://www.python.org/dev/peps/pep-0008/

12.1.3 Reporting

General Style

- use the Helvetica font everywhere
- margins (in millimeters):
  - top: 14
  - bottom: 16
  - left: between 12 and 13 to allow punching holes without punching in the text area
  - right: between 12 and 13

Note: *the line separator between the header and the body can overlap slightly in the left and right margins: up to 9 millimeters on the left and up to 12 millimeters on the right*

- for Titles use the font *Heveltica-Bold* with the size *14.5*
- put the context on each report: example, for the report account_balance: the context is the fiscal year and periods
- for the name of cells: use Capital Letter if the name contains more than one word (ex: Date Ordered)
- content and name of cells should have the same indentation
- for report, we can define two kinds of arrays:
  - array with general information, like reference, date..., use:
    - *Bold-Helvetica* and size=8 for cells name
    - *Helvetica* size="8" for content
  - array with detailed information, use:
    - *Helvetica-Bold* size *9* for cells names
    - *Helvetica* size *8* for content

Headers and footers for internal reports:

- •Internal report = all accounting reports and other that have only internal use (not sent to customers)
- •height of headers should be shorter
- •take off corporate header and footer for internal report (Use a simplified header for internal reports: Company's name, report title, printing date and page number)
- •header:
  - –company's name: in the middle of each page
  - –report's name: is printed centered after the header
  - –printing date: not in the middle of the report, but on the left in the header
  - –page number: on each page, is printed on the right. This page number should contain the current page number and the total of pages. Ex: page 3/15
- •footer:
  - –to avoid wasting paper, we have taken off the footer

table line separator:

- it's prettier if each line in the table have a light grey line as separator
- use a grey column separator only for array containing general information

table breaking

- •a table header should at least have two data rows (no table header alone at the bottom of the page)
- •when a big table is broken, the table header is repeated on every page

how to differentiate parents and children ?

- •When you have more than one level, use these styles:
- •for the levels 1 and 2:fontSize="8.0" fontName="Helvetica-Bold"
- •from the third level, use:fontName="Helvetica" fontSize="8.0" and increase the indentation with 13 (pixels) for each level
- •underline sums when the element is a parent

12.2 Modules

12.2.1 Naming Convention

The name of the module are all lowercase, each word separated by underscrores. Start always by the most relevant words, which are preferably names of others modules on which it depends.

Exemple:

- account_invoice_layout

12.2.2 Information Required

Each module must contains at least:

- name
- description

12.2.3 Modules Description

12.2.4 Dependencies

Each module must contains:

- A list of dependencies amongst others modules: ['account','sale']
  - Provide only highest requirement level, not need to set ['account','base','product','sale']
- A version requirement string where base is the Open ERP version as a Python expression
  - account>=1.0 && base=4.4

12.2.5 Module Content

Each module must contains dema data for every object defined in the module.

If you implemented workflows in the module, create demo data that passes most branches of your workflow. You can use the module recorder to help you build such demo data.

12.3 User Interface Guidelines

12.3.1 Menus

Organising menus

Here is a good example:

- Invoices (list)
  - Customer Invoices (list)
    - Draft Customer Invoices (list)
    - Open Customer Invoices (list)
    - New Customer Invoice (form)
  - Supplier Invoices
    - ...

Add a *New ...* menu only if the user requires it, otherwise, open all menus as lists. The *New ...* menu open as a form instead of a list. For example, don't put *New ...* in a menu in the configuration part.

If you use folders that are clickable, their child must be of the same object type. (we suppose that inheritancies are the same objects)

List are plurals:

- *Customer Invoice*, should be *Customer Invoices*

If you want to create menu that filters on the user (*All* and *My*) put them at the same level:

- Tasks
- My Tasks

And not:

- Tasks
  - My Tasks

Avoid Abbreviations in menus if possible. Example:

- BoM Lines -> Bill of Materials Lines

Reporting Menu

The dashboard menu must be under the report section of each main menu

- Good: Sales Management / Reporting / Dashboards / Sales Manager
- Bad: Dashboard / Sales / Sales Manager

If you want to manage the *This Month/ALL months* menu, put them at the latest level:

- Reporting/Timesheet by User/All Month
- Reporting/Timesheet by User/This Month

Icons in the menu

- The icon of the menu, must be set according to the end action of the wizard, example:
  - wizard that prints a report, should use a report icon and not a wizard
  - wizard that opens a list at the end, should use a list icon and not a wizard

Order of the menus

The configuration menu must be at the top of the list, use a sequence=0

The *Reporting* menu is at the bottom of the list, use a sequence=50.

Common Mistakes

- Edit Categories -> Categories
- List of Categories -> Categories

12.3.2 Views

Objects with States

- The state field, if any, must be at the bottom left corner of the view
- Buttons to make the state change at the right of this state field

Search Criterions

Search criterions: search available on all columns of the list view

12.3.3 Action Names

12.3.4 Wizards

12.4 Terminology

12.4.1 Default Language

The default language for every development must be U.S. english.

For menus and fields, use uppercase for all first letters, excluding conjections:

- Chart of Accounts

12.4.2 Field Naming Conventions

- Avoid generic terms in fields and use if possible explicit terms, some example:
  - Name -> Sale Order Name
  - Parent -> Bill of Material Parent
  - Rate -> Currency Rate Conversion
  - Amount -> Quantity Sold

Here are some rules to respect:

- many2one fields should respect this regex: '.*_id'
- one2many fields should respect this regex: '.*_ids'
- one2many relation table should respect this regex: '.*_rel'
- many2many fields should respect this regex: '.*_ids'
- use underscore to separate words
- avoid using uppercases
- if a field should be composed of several words, start by the most important words
  - This is good: sale_price, partner_address_id
  - This is bad: is_sellable

12.4.3 Object Naming Conventions

- All objects must start by the name of the module they are defined in.
- If an object is composed of several words, use points to separate words

12.4.4 Some terms

- All Tree of resources are called *XXX's Structure*, unless a dedicated term exist for the concept
  - Good: Location' Structure, Chart of Accounts, Categories' Structure
  - Bad: Tree of Category, Tree of Bill of Materials

CHAPTER

THIRTEEN

MODULE RECORDER

CHAPTER

FOURTEEN

REVIEW QUALITY

Part VIII

Distribute Modules

CHAPTER

FIFTEEN

PUBLISHING MODULES

CHAPTER

SIXTEEN

MANAGE TRANSLATIONS

Part IX

Improving Translations

CHAPTER

SEVENTEEN

TRANSLATING IN LAUNCHPAD

Translations are managed by the Launchpad Web interface. Here, you'll find the list of translatable projects.

Please read the FAQ before asking questions.

CHAPTER

EIGHTEEN

TRANSLATING YOUR OWN MODULE

Changed in version 5.0. Contrary to the 4.2.x version, the translations are now done by module. So, instead of an unique `i18n` folder for the whole application, each module has its own `i18n` folder. In addition, OpenERP can now deal with `.po` [1] files as import/export format. The translation files of the installed languages are automatically loaded when installing or updating a module. OpenERP can also generate a .tgz archive containing well organised `.po` files for each selected module.

[1] http://www.gnu.org/software/autoconf/manual/gettext/PO-Files.html#PO-Files

Part X

Documentation Process

CHAPTER

NINETEEN

BOOKS

The main documentation of OpenERP is composed of a set of books according to the business need. These books are reviewed once a year. We are working with authors/contributors/employees/translators to build chapters on the different aspects of the ERP. As to motivate people to write quality documentation,

This section describe how we collaborate with authors and translators to provide a very good documentation on OpenERP. As to motivate people to write quality documentation/chapters we set up author rights to pay every contributor and translator according to their effort.

19.1 Building a Book

We have contract with several editors to publish books in different countries.

Once we have enough chapters written, we can compose a book and publish it.

Books are first published in a paper version. Three months after, we release it online.

19.2 Author Rights

The typical author rights are between 8% and 10% on the public price, according to the authors, the country and the editor in which the book will be published. This commission is on the public price, no matter of the final selling price per item.

These author rights have to be divided according to people working on the book:

- Reviewers: 10% to be divided by number of reviewers
- Translators: 30% to be divided by the number of translators
- Authors: the rest (60%-90%) to be divided by number of authors

As an example, Geoff and Fabien worked on the french and english book on OpenERP. This book is sold at a public price of 35 EUR with 10% author rights. We had one reviewer for this book from Eyrolles. So author rights are splitted in that way:

- Geoff: 1.575 EUR/book (= 35 * 0.1 * (0.9 / 2))
- Fabien: 1.575 EUR/book
- Reviewer: 0.35 EUR/book (= 35 * 0.1 * 0.1)

Once this book will be translated to Hungarian, with a public price of 30 EUR and author rights of 10% (0.1) we will have:

- Geoff: 1.05 EUR/book (=30 * 0.1 * 0.7 / 2)
- Fabien: 1.05 EUR/book
- Hungarian translator: 0.90 EUR/book (=30 * 0.1 * 0.30)

Author rights are paid every 3 months, after one month. (to be verified according to what we can do with editors)

CHAPTER

TWENTY

PEOPLE

20.1 Authors

Everyone can be an author and write a complete book or just one or several chapters on particular aspects of OpenERP. Chapters are then review

20.2 Authors from Tiny

At Tiny (the editor of OpenERP), each employee can write a few chapters based on new module he wrote for specific customers, at the end of the project. As employees get a salary to write these chapters during their working hours, author rights are computed slightly differently:

- Computed rights are divided by two for the employee: 50%
- Valid until the employee work for Tiny

CHAPTER

TWENTYONE

MODULES

Each module should have a small and minimal documentation.

CHAPTER

TWENTYTWO

BUILDING THE DOCUMENTATION

We use Sphinx, a documentation generator, to build the documentation. So, Sphinx should be installed on your system and you should know how to use it.

You can install it with easy_install. For example, on Ubuntu:

```
sudo easy_install sphinx
```

building the documentation in html:

```
make clean
make html
```

building the documentation in pdf:

```
make clean
make latex
cd build/latex
make all
```

building a book:

For example, if you want to build the *Open ERP for Retail and Industrial Management* book:

```
cd books/book_mrp
make clean
make latex
cd build/latex
make all
```

CHAPTER

TWENTYTHREE

FAQ

How much items can we expect to sell for a book ?

The first french book we wrote is sold at 500 items per month. It's good as it was our the first book on OpenERP but we can expect better results with an english version. So probably between 250 and 1500 items per month for an english book.

Part XI

How to translate the Open Object Documentation in your language

CHAPTER

TWENTYFOUR

PREREQUISITE

You should be able to build the untranslated documentation. So Sphinx should be installed on your system and you should know how to use it.

If this is not the case, please read the Community Guide.

CHAPTER

TWENTYFIVE

UNDERSTANDING THE DIRECTORY STRUCTURE

We are supposing that **<openobject-doc>** is the root of the Open Object documentation bazaar branch.

The *untranslated sources* are located in **<openobject-doc>/source**.

The translated documentation will be located in **<openobject-doc>>/i18n/<lang>/source**.

For example, the documentation in french will be located in **<openobject-doc>/i18n/fr/source** and it will be built in **<openobject-doc>/i18n/fr/build/html** for example.

25.1 Summary

Directory	Description
<openobject-doc>/source	untranslated sources
<openobject-doc>/i18n/<lang>/source	translated sources
<openobject-doc>/i18n/<lang>/build/html	translated documentation in html

CHAPTER

TWENTYSIX

CREATING THE TRANSLATION DIRECTORY STRUCTURE

Use the **make** command (with target **i18n**) to create the translation templates. You'll need to pass the language as an additional parameter to the *make* command.

For example, supposing you want to translate the documentation in french:

```
make i18n LANG=fr
```

This command will do several things:

- create these directories, if they does not exist yet:
  - i18n
  - i18n/fr
  - i18n/fr/source
  - i18n/fr/build
- copy files in *i18n/fr/* required for the html build:
  - MakeFile
  - conf.py
- create the *translation templates* based on the untranslated restructured text files. They will be created in *i18n/fr/source*
- copy all the other necessary files (images for example)

CHAPTER

TWENTYSEVEN

TRANSLATION TEMPLATES

The template structure for a given file is very simple. Each text section is prepended by the original context. Here is a title, for example:

```
.. i18n: %%%%%%%%%%%%%%%%%%%%%%%%%
.. i18n: Open Object Documentation
.. i18n: %%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%
Open Object Documentation
%%%%%%%%%%%%%%%%%%%%%%%%%
```

The context is a commented section starting with **.. i18n:**. It helps you understand the section in its context. It also helps you remember the original section.

And here is the translated section:

```
.. i18n: %%%%%%%%%%%%%%%%%%%%%%%%%
.. i18n: Open Object Documentation
.. i18n: %%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Documentation sur Open Object
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

CHAPTER

TWENTYEIGHT

MANAGING SOURCE CHANGES

If someone adds or changes something in the documentation, that section will have to be retranslated but all the other sections will hopefully keep their translation.

When you will get the documentation changes with bzr pull (for example), the new sections and some changed sections will be reset to the untranslated text when you will rebuild the translation with *make i18n LANG=fr*.

CHAPTER

TWENTYNINE

BUILDING THE DOCUMENTATION IN YOUR LANGUAGE

That is very simple because the directory and file structure is exactly the same as the original structure:

```
i18n
`-- fr
    |-- build
    `-- source
```

For example, in *i18n/fr*, you just have to do a simple *make*:

```
make html
```

And the html documentation will be built in *i18n/fr/build/html*.

CHAPTER

THIRTY

STATUS

At the moment, this script is in alpha status and has not been thoroughly tested. It should work but expect some bugs to pop up at unexpected times.

Part XII

Monthly IRC Meeting

Contents

CHAPTER

THIRTYONE

INTRODUCTION

Once a month, we will organize a community IRC meeting. All topics related to the community, the Open ERP product, the Open Object project can be discussed during this meeting. The meeting is open to partners, the management at Tiny and every member of the community.

It allows all actors on Open ERP to discuss about all aspects of the software or the organisation of the project.

CHAPTER

THIRTYTWO

PREPARATION OF THE MEETING

The community meeting should be planned in the calendar of the Open ERP website, the first tuesday of each month, at 1pm, UTC. You can check the exact date conversion for your country here:

http://www.timeanddate.com/worldclock/fixedtime.html?hour=14&min=0&sec=0&p1=48

A few days before the meeting, a proposition of points to discuss is published in the Open ERP wiki. People are free to add (don't remove) or categorize topics. You can pre-discuss about some topics in the forum or our official IRC channel.

A post on the planet will be made a few days before the meeting. This post must include, at least: * a summary of all topics to cover, in the right order. * a link to the summary of previous meeting * a link to this page that explain the meeting organisation

CHAPTER

THIRTYTHREE

ORGANISATION OF THE MEETING

The meeting is organized at the official Open ERP IRC channel:

- http://openobject.com/irc/

If you missed the beginning of the meeting, you can check our IRC logs that are updated every hour.

timeline is selected at

The phases of a meeting: * Introduction: 10 minuts * Presentations: 5 minuts * Discussion on topics: average of 2 hours * Conclusion: 10 minuts * Summary Post: /

CHAPTER

THIRTYFOUR

THE PHASES OF THE MEETING

34.1 Introduction

At the beginning of each meeting, we will select: * A responsible to organize and respect the list of topics * Someone to write a summary during the meeting These two people are publicly announced.

Then, the responsible of the meeting provide 2 links to be read: * This page: that explains the meeting organization, * The page containing the proposition of topics.

Then, we ask if someone wants to add some topics in the proposed list. If yes, the different elements are added in the list of topics, in the right section.

34.2 Presentations

People can write their names and the company they work for.

34.3 Discussion on topics

The responsible of the meeting announces the topics one by one. Each new topics should be emphasized so that it's clear when we read the logs:

- —- New TOPIC —-

It's important to discuss ONLY about the current topic to keep a clean structure in the meeting organisation. The responsible is the one that should organisze the discussion, by asking people to wait the right topic to discuss on some points.

Topics can not be longer than 15 minuts. If a topics is too long, the responsible of the meeting can request to continue on the next topic and, if needed, ask a new date to discuss this topic specifically.

34.4 Conclusion

The responsible to write the summary copy/paste his summary in the IRC channel. The summary must contain in minimum: * List of assigned tasks (who do what) * List of points to discuss in next meeting

People can criticise this summary, if the responsible missed some points.

The summary should not contain the thoughts or the full discussion of the people, the IRC log is accessible for this.

34.5 Summary Post

The responsible of the summary writes a post in the planet with the summary of the IRC meeting.

Part XIII

Bug Tracker

We use launchpad on the openobject project to track all bugs and features request related to openerp and openobject. the bug tracker is available here:

- Bug Tracker : https://bugs.launchpad.net/openobject
- Ideas Tracker : https://blueprints.launchpad.net/openobject
- FAQ Manager : https://answers.launchpad.net/openobject

Every contributor can report bug and propose bugfixes for the bugs. The status of the bug is set according to the correction.

When a particular branch fixes the bug, a commiter (member of the Commiter Team) can set the status to "Fix Commited". Only commiters have the right to change the status to "Fix Committed.", after they validated the proposed patch or branch that fixes the bug.

The Quality Team have a look every day to bugs in the status "Fix Commited". They check the quality of the code and merge in the official branch if it's ok. To limit the work of the quality team, it's important that only commiters can set the bug in the status "Fix Commited". Once quality team finish merging, they change the status to "Fix Released".

Part XIV

Feature Requests

If you want to suggest or request some additionals modules or functionalities, you use the Launchpad Blueprints.

Part XV

Communication

CHAPTER

THIRTYFIVE

FORUMS

35.1 Users - Forum

URL: http://openerp.com/forum/general-discussion-f11.html

This forum is used to request help on general topics and is more useful for users. It could be functional question, configuration problem and so on. Every request posted on this forum will appear in the according mailing-list. It functions both ways, so every post on the mailing list will also appear in the forum.

35.2 Developers - Forum

URL: http://openerp.com/forum/general-questions-f37.html

This forum is used for comitters to ask for more help about a bug resolution. An automatic message is sent to the according mailing list when you click on a button featured in the *bug tracker*. It functions both ways, so every post on the mailing list will also appear in the forum.

CHAPTER

THIRTYSIX

IRC

CHAPTER

THIRTYSEVEN

MAILING LISTS

37.1 Users Mailing List

This mailing list serves the same purpose as the *Users Forum*

http://tiny.be/mailman/listinfo/tinyerp-users: the web interface to manage your subscription.

37.2 Technical Mailing List

This mailing list serves the same purpose as the *Developers Forum*

http://tiny.be/mailman/listinfo/tinyerp-devel: the web interface to manage your subscription.

37.3 Partners Mailing List

This is the partner communication canal. It is used for request about skills or contributions on project, share development, announcing IRC meeting, ... It is also the communication canal between Tiny and their partners.

http://tiny.be/mailman/listinfo/partners: the web interface to manage your subscription.

CHAPTER

THIRTYEIGHT

WIKI

All documentations about OpenERP and OpenObject are maintained in the official wiki:

- http://openerp.com/wiki

To keep you aware about what is improved on this wiki, use this RSS feed:

- http://openerp.com/wiki/index.php?title=Special:Recentchanges&feed=rss

CHAPTER

THIRTYNINE

BLOG

Where is Planet OpenERP located ?

http://www.openerp.com/planet

How to contribute to Planet OpenERP RSS ?

Only commiters can write in the planet, others can announce on the forum.

If you are a member of the commiter team:

- Create your personnal blog
- Send an email to nva AT openerp.com with your name, photo and address of your blog.

Part XVI

Release Cycle

CHAPTER

FORTY

EXPLANATION OF THE VERSIONS

The development of OpenERP follows two distinct branches:

- Stable: a new version is published every year. This version is intended to be put in production, only bugfixes are applied in this version.
- Development (sometimes called the trunk): It will contain all the latest functionalities and modules. It is intended for testing purpose only, until it is stabilized as the next stable version.

CHAPTER

FORTYONE

INTRODUCTION

41.1 Timeline

A new stable version is published every year. Example:

- May: Community Meeting
  - Define missing and urgent bugfixes and features
  - Tiny and Partners Choose modules from contrib and extra addons that will integrate official addons
  - Define precise timeline and tasks assignation
- July: Branch the trunk to the new stable version
  - You can not add new features in the futur stable
  - Only bugfixes are accepted in this new branch
- August: Publish first releases candidates
- September: Publish the new stable version

CHAPTER

FORTYTWO

PROCESSES

42.1 Community Meeting

Participants: Everybody

Goals:

- Define exact release timeline
- Define required tasks for new version
- Select official addons

42.2 Branching to new stable

Goals:

- After branching, no new features accepted
- Bugfix period

42.3 Release Candidates

Goals:

- Publish a version to be tested by the community
- Different release candidates, every 2 weeks

42.4 Stable Version

Goals:

- Publish when everything is perfect

42.5 Creating Screen cast for OpenERP

Point to be noted to start with Screen cast for OpenERP.

- Make demo in eTiny *Webclient*.(Its not compulsory)
- **We have a specific software to create screen cast. The name is "Screen Flash". It works in windows.** You can purchas license copy of Screen Flash. So, do not use with fake / crack ID.
- For linux, one can use "gtk-Record-my-desktop", but preferable is to use "Screen Flash" on windows to get good result.
- The resolution of your screen at the time of recording should be 800 X 600 for screen cast.
- The speed of recording (cursor / pointer) should be very slow (wait for 1-2 second between to clicks), as you are experts so you know the process, but we are preparing these screen cast for the people, who are not used to with OpenERP. The purpose of screen cast is to make them understand and can learn the process by themselves.
- At the time of typing, be careful for not committing any mistake, avoid erase and retype.
- Close all your other program / application / software running before you start processing for screen cast.
- For windows, screen flash software, kindly configure your short cut for better clarity.
- Auto hide your task bar before starting screen flash on windows/ gtk-Record-my-desktop on linux.
- Always maximize your window at the time of screen cast, so it can capture all the visible portion with perfection and to avoid other part of unused screen object.
- First list out all steps to show, and remember all steps. At the time of recording you must be clear that what will be the next step to show. so you do not hesitate in between choosing inappropriate steps.
- If required, kindly prepare demo data that you want to show for better understanding of users, and also it will not take too much time to type it during recording.
- For required and useful fields highlight the tool-tip and stay for sometimes (At least once we can read it slowly) so the other people can read it and know the exact use of that field.
- One can edit screen cast, if prepared in Screen Flash software on windows. But its better to avoid it by making it proper from the start.

INDEX

V