Develop your own ERP with ERP5 Business Templates

ERP5 Tutorial

www.erp5.org

version 0.9
Linux Solutions
February 2, 2005
Paris

Date	01/29/05
Version	0.1
Author	Kevin Deldycke
Contributors	Jean-Paul Smets
Translation Date	05/01/05
Translation Version	0.1
English Translators	Shrenik Bhura, Chris Purves, Yoshinori Okuji
Abstract	Introduction to the installation of Business Templates into ERP5 and to the development of new functionalities
Email	info@nexedi.com
Web	http://www.erp5.org
Rights	Copyright © 2004-2005 Nexedi
Licence	You have the permission to copy, distribute and/or modify this document according to the term of the GNU Free Documentation License, Version 1.1 as published by Free Software Foundation (http://www.gnu.org/copyleft/fdl.html) with the Front-Cover Texts being « ERP5 Tutorial www.erp5.org », with the Back-Cover Texts being « You are free to copy and modify this handbook, like all the free software and documentation products by the company Nexedi », and with the Invariant Section being « ERP5 Commercial Support ».

Contents

1. Introduction.....4

2. ERP5: An ERP for All.....6
2.1. ERP5 without installation : Using ERP5 Live CD.....6
2.2. Traditional ERP5 Installation.....6
2.3. Extending ERP5 with Business Templates 10

3. Creating a Business Template.....16
3.1. The Structural Design of Data.....16
3.2. Variation by categories.....19
3.3. Integrating the data structure with ERP5.....21
3.4. Fast Web Development with portal_classes.....22
3.5. The structure of a Document class in ERP5.....23
3.6. Creating a module in EPR5.....27
3.7. Integrating the classes, data, menus and appearance: the tool portal_types 28
3.8. The user interface with ERP5Form.....29
3.9. Python scripts for calculation.....42
3.10. The reporting.....49
3.11. The creation of the Business Template.....58
3.12. A point on cataloguing.....60

4. Conclusion.....61

5. TODO.....62

1. Introduction

This document is intended for developers who want to adapt ERP5 to their companies in order to meet their specific and unique needs.

With ERP5, it is not the company which conforms to the ERP logic any longer, but the ERP adapts itself to the culture, methods and practices of the company. ERP5 provides a framework and tools necessary for the achievement of this goal. This is most often accomplished by the creation and the modification of flexible ERP packages called «Business Templates».

This document will show you how to quickly install an ERP5 platform, and how to install and use Business Templates, by guiding you through the use of a simple module for payroll management. The second part of the document will describe all the techniques of ERP5 so that you can thereafter create your own Business Templates by yourself.

ERP5 Commercial Support

ERP5 is a free ERP, under the term of GPL, developed and published by the company Nexedi.

Nexedi coordinates a network of partners certified for the deployment of ERP5 solutions for enterprises. Nexedi and its partners assure, through the team of their engineers and their consultants, the assistance, the formation, and the transfer of competences necessary to the control of ERP5 in the field.

http://www.erp5.org/sections/documentation/evangelism/enterprise/

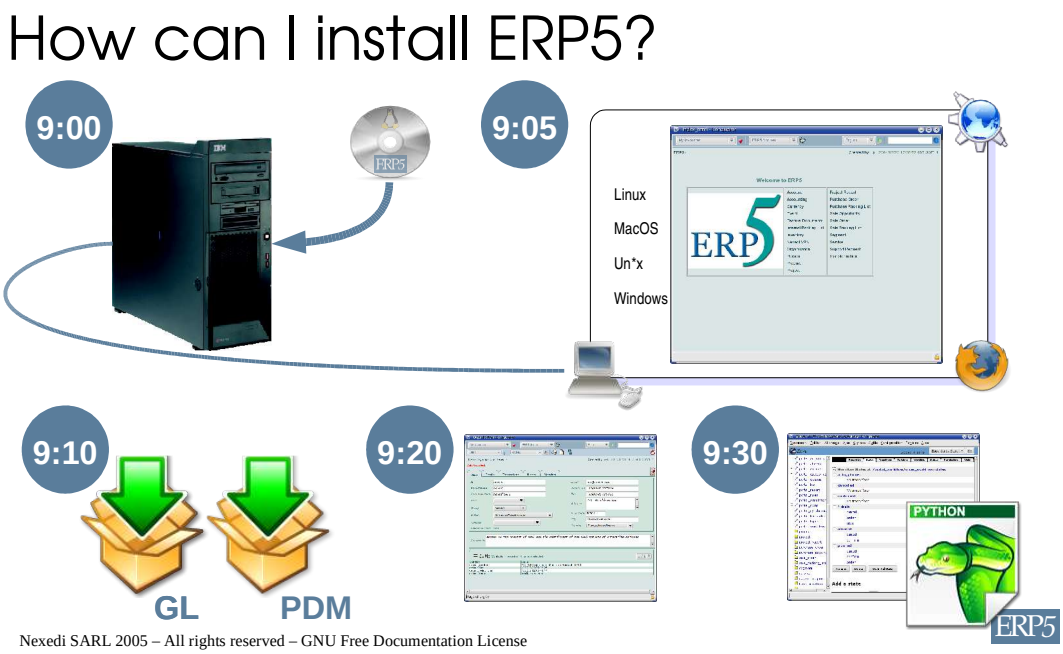
Tel.: +33 662 05 76 14
Mail: info@nexedi.com

2. ERP5: An ERP for All

2.1. ERP5 without installation : Using ERP5 Live CD

The easiest way to get started with ERP5 is to use the Live CD. The ERP5 Live CD saves your valuable time from wasting due to installation problems. It has been conceived as a production and development tool, not only intended for demonstration. It can be downloaded from: https://www.erp5.org/sections/download/erp5_livecd/view .

This illustrates its simplicity:



Insert a CD into a standard PC, and boot up from it. After a few minutes, it starts ERP5. You can then install Business Templates to use, configure, and extend.

2.2. Traditional ERP5 Installation

If the use of the Live CD is not suitable for you, it is possible to install ERP5 in a traditional way. It should take a few tens of minutes (if you use an operating system such as Mandriva Linux for which all packages and their dependences have been created) or a few days (if you use an operating system without ERP5 packages).

ERP5 needs the Zope application server on which it is based as well as a number of other Zope products. You also need to install MySQL-Max (a version of the most popular free database server which supports transactions). MySQL-Max is used as the

search engine of Zope via the ZSQLCatalog product.

For a complete system, it is necessary to have:

- A Zope server (and all its dependences related to Python) with the optimization developed by Nexedi, regarding meta-programming[1].
- The Zope products BTreeFolder, Base18, CMF, CMFActivity, CMFCategory, CMFMailIn, CMFPhoto, CMFReportTool, ERP5, ERP5Catalog, ERP5Form, ERP5SyncML, ERP5Type, ExtFile, Formulator, Localizer, Photo, TranslationService, ZMailIn, ZMySQLDA and ZSQLCatalog.
- MySQL-Max

To conclude this section, it is necessary to discuss packaging for your Linux distribution. You can also take the source and install Zope by a traditional ./configure, make, and make install. The Zope products can be easily and rapidly installed, since the archive files can be extracted in the Zope Products repository (generally `/var/lib/zope/Products` or `/usr/lib/zope/lib/python/Products`), and some extensions (zsqlbrain.py and InventoryBrain.py) can be added into the Extensions repository of Zope (generally `/var/lib/zope/Extensions` or `/usr/lib/zope/lib/python/Extensions`). You will find tutorials and documentation on the Internet for the installation and the total configuration of Zope without difficulties.

The happy users of Mandriva Linux can be satisfied with a simple urpmi ERP5, after having added the media `main`, `contrib` and `nexedi`[2] with their package manager.

Once the package installations finish, use the administration console of Zope, `zopectl`, to change the default password of the administrator (the login by default being `admin`). The addition of a user can be made only if the Zope server is stopped. Note that the installation procedure above automatically may have started the server, you should thus stop it before proceeding:

```
[root@localhost /]# /etc/init.d/zope stop
Stopping zope : . daemon process stopped

[root@localhost /]# zopectl
program: /usr/bin/runzope
daemon manager not running
zopectl> help adduser
adduser <name> <password> -- add a Zope management user
zopectl> adduser admin admin

[root@localhost /]# /etc/init.d/mysql-max start
Starting MySQL Server                                      [  OK  ]

[root@localhost /]# zopectl
```

1 The source code of this optimization is available from the URL http://www.nexedi.org/static/Mandrake/10.1/SRPMS/zope-2.7.2rc1-1nxd.src.rpm

2 http://www.nexedi.org/static/Mandrake/10.1/RPMS

```
program: /usr/bin/runzope
daemon manager not running
zopectl> start
daemon process started, pid=7944
```

Let's check the operation of Zope by accessing the ZMI, the Zope management Web interface, where one will be authenticated with the password of the user `admin` which one has just added:

```
http://localhost:9080/manage
```

Illustration 1: The access to the ZMI requires an authentication

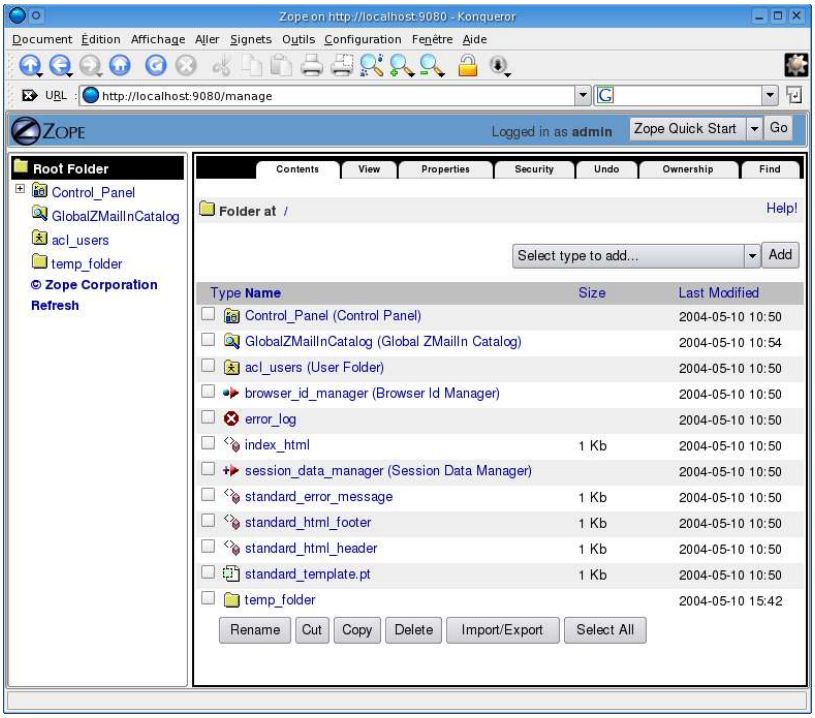
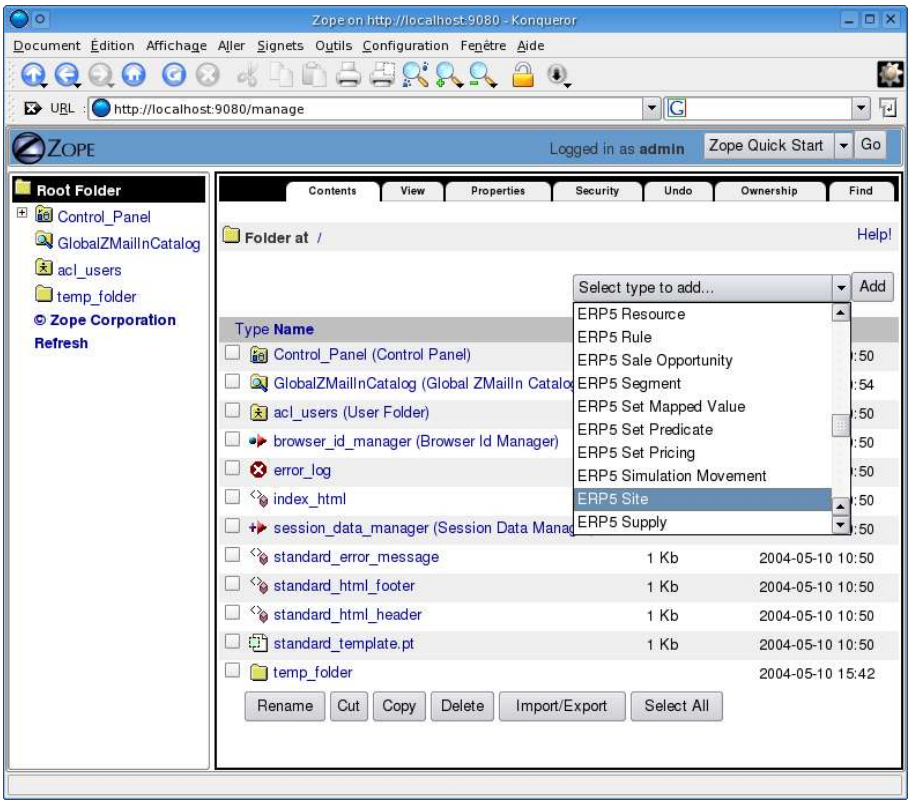
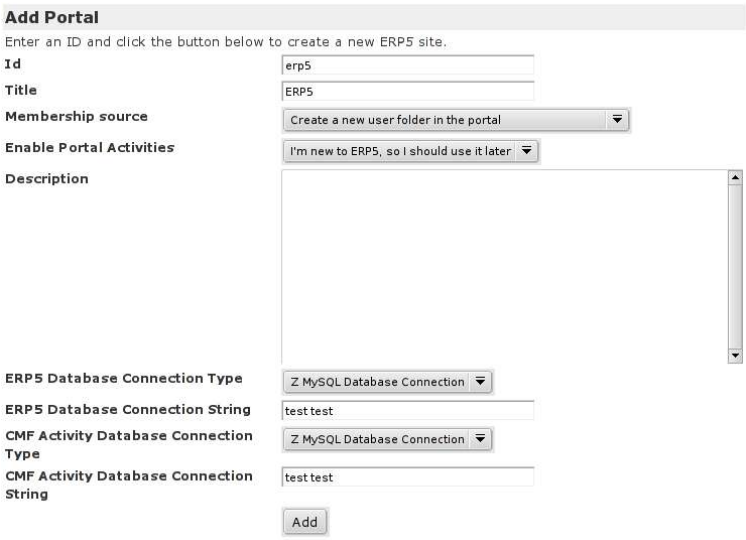


Illustration 2: ZMI, the Web interface for the management of Zope

As the Zope server is operational and accessible now, you will create an instance of ERP5 by selecting type `ERP5 Site` in the drop-down list in the top-right:



A page of properties is displayed in the ZMI, and you will fill it as on the screenshot:



The validation of this form leads to this screen if successful:

But it is frequent to see the following error appearing at this time:

```
Site Error
An error was encountered while publishing this resource.
Error Type: DatabaseError
Error Value: z0_catalog_object is not connected to a database
```

This error means simply that the MySQL-Max server is not started. A restarting of this server is enough so that all returns in the order.

You can now go to the title page of ERP5 via the URL:

```
http://localhost:9080/erp5/
```

Now that ERP5 is operational on your machine, you are ready to begin with it. We would, however, advise to update ERP5 products from the CVS of ERP5[3], to benefit from the latest enchancements and bugfixes.

Warning: Although ERP5 is now installed and usable on your machine or your server, it is not advised to use it « just as it is » in a production. A fine configuration (a cluster, an activity system, etc.) is indeed necessary to obtain high performance, and a thought on the policy of data security is essential to protect the vital data for the company.

2.3. Extending ERP5 with Business Templates

The Business Templates are ERP5 packages, comparable to system packages (such as rpm and deb) in the Linux distributions. The purpose of the RPM is to simplify the

3 http://cvs.erp5.org – also see the script `update_cvs.sh` in `/var/lib/zope/` to automate the updates

installation of software within a system. The Business Templates have the same role as it takes charge of a clean and automated installation of an ERP5 component. From the point of view of a user, installing a Business Template is to add a major functionality to the ERP, by widening its capacities of the management of the company.

There are currently 8 Business Templates, each one dedicated to an aspect of the management of a company:

- Trade : commercial management (purchases and sales, orders, delivery orders, invoices, inventory control)
- PDM (Product Data Management) : management of the data produced (the definition of products, alternatives, categorization, nomenclatures, operational ranges, multi-media catalogues)
- MRP (Manufacturing Resources Planning) : organization and production control (planning work orders for production)
- CRM (Customer Relationship Management) : management of customer relations (databases of the organizations and people, commercial opportunities)
- Accounting : accountancy (accounting transactions, financial reports)
- HR (Human Resources) : management of human resources (payrolls, career management)
- eCommerce : electronic trade (on-line store sales, affiliations)
- CMS (Content Management System) : management of contents via Web (via Nuxeo- CPS[4])

Some of these Business Templates have reached a first stage of maturity (e.g. Accounting makes it possible today to enclose a financial year). The others are still in the course of integration.

Indeed, altough ERP5 is used today in production on important sites with a complete functional cover (e.g. nomenclatures, production management, calculation of needs, stock management, invoicing, etc.), the customers of ERP5 do not wish to finance the cost of integration in the form of generic Business Templates of the specific work which was carried out to adapt the ERP5 framework to their own needs. The work of integration of the Business Templates is thus carried out volunatrily by Nexedi. For this reason, all the functionalities of the ERP5 framework are not covered yet by the Business Templates available for downloading.

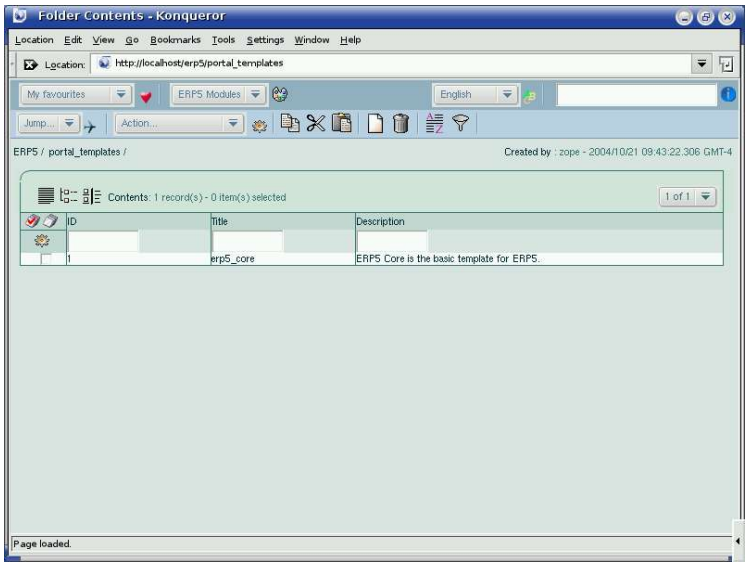
One of the modules designed first which uses the system of Business Templates is

4 http://www.cps-project.org/

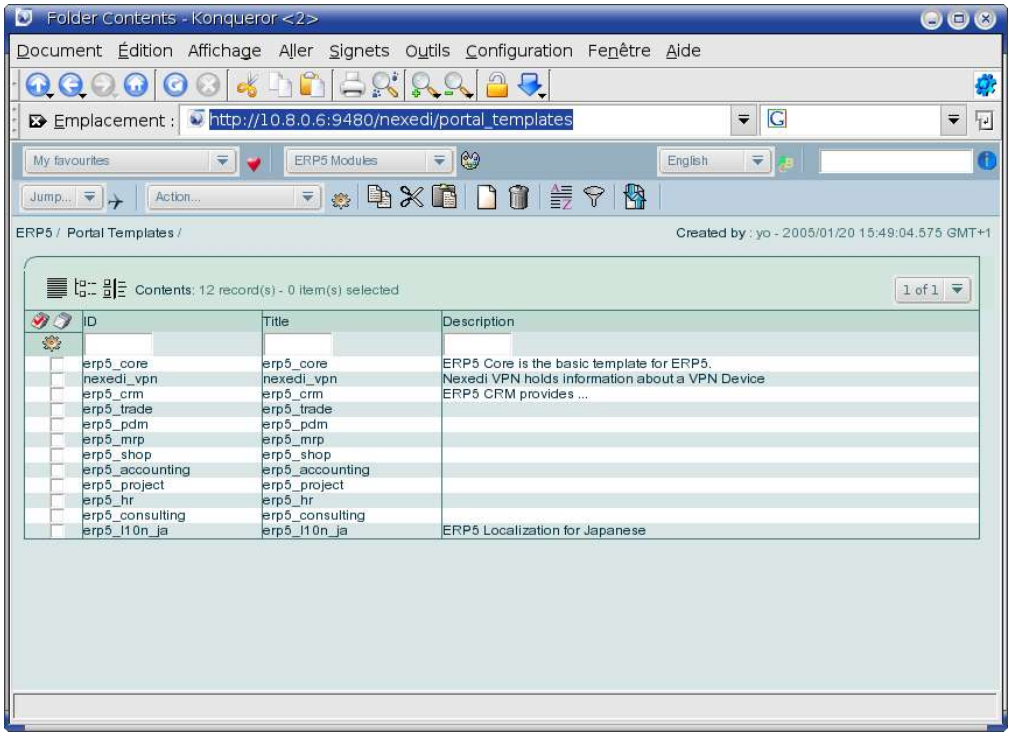
that of payroll management. This module makes it possible to manage a pay-book and to create payrolls.

It is this module which will be used as an example throughout the article. We now will detail the process of installation of this Business Template. This process requires an account with the right `Manager`.

First of all, open the module `portal_templates` with your browser by typing an URL http://localhost/erp5/portal_templates or http://localhost:9480/erp5/portal_templates, according to your configuration. You will be able to then reach the list of the Business Templates:

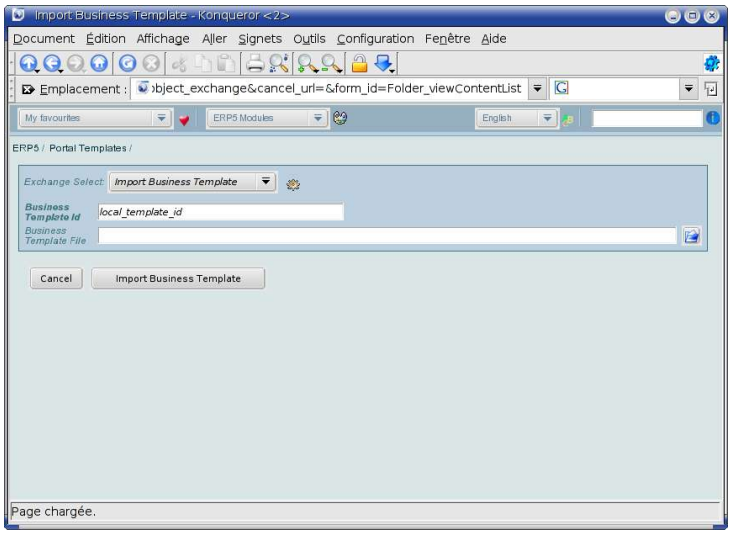


A more complete ERP5 system comprises many Business Templates:

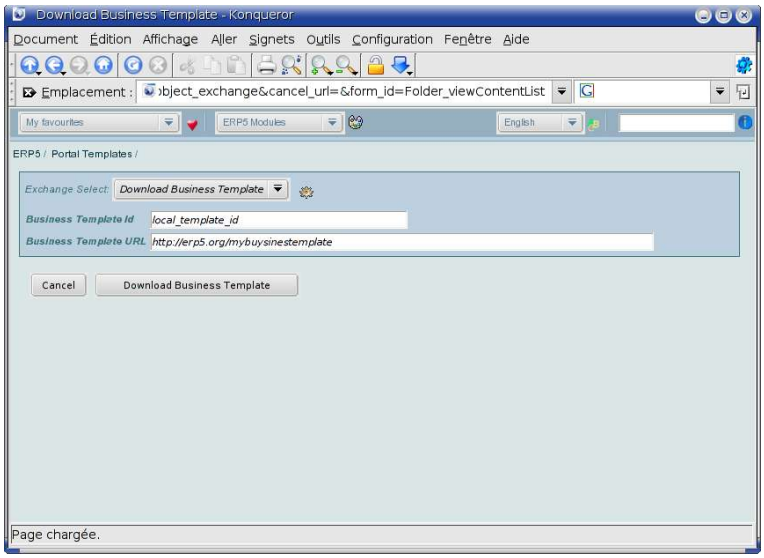


Our objective is thus to pass from a practically empty list of Business Templates to a more complete list. To achieve this goal, it is enough to click on the button « `Import/export` » in the second bar of buttons starting from the top. This button resembles this:

After having clicked above, you arrive at the dialogue for importing a Business Template:



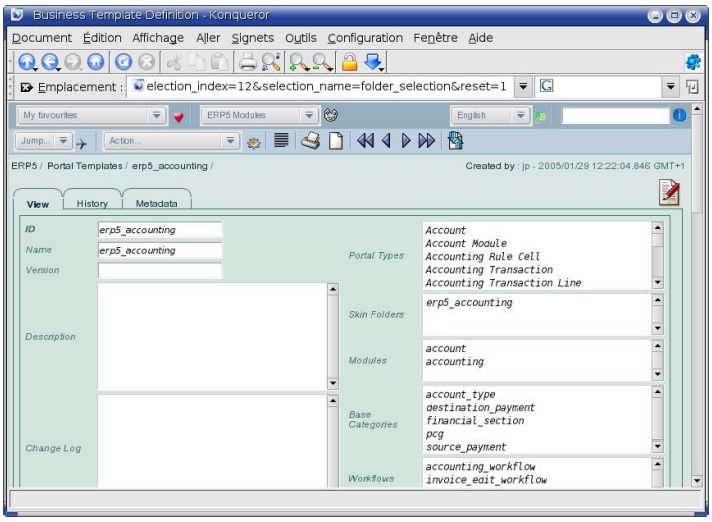
Use the menu « `Exchange Select` » to change the dialogue from « `Import Business Template` » to « `Download Business Template` » :



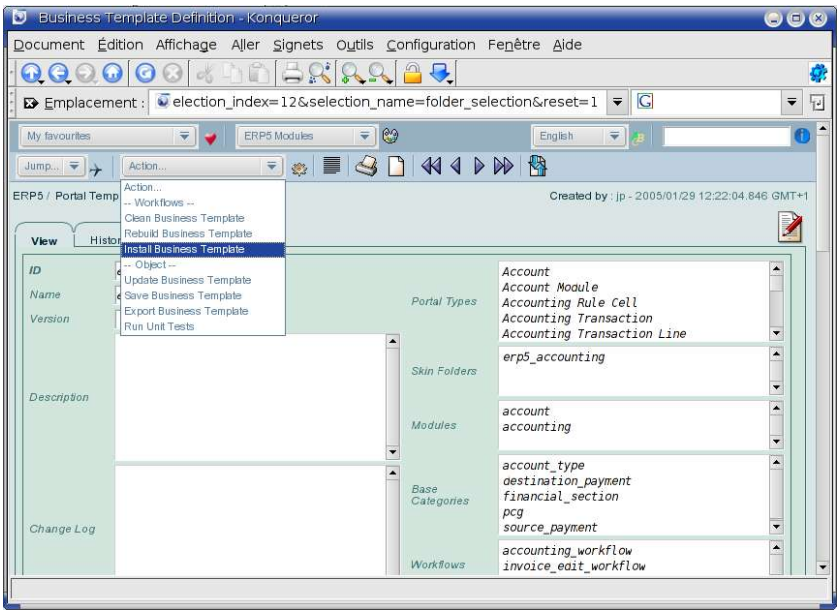
You then have to filll in the two fields, an identifier of the Business Template (e.g. `erp5_accounting`) and the URL of this Business Template (e.g. `http://cvs.erp5.org/cgibin/viewcvs.cgi/*checkout*/erp5_bt5/erp5_trade.bt5?rev=HEAD&content-type=text/plain`), and then click on « `Download Business Template` ». The

downloading will be automatically carried out (provided you have access to the Internet), and, in the case of success, you will see a message « `Business Template Downloaded Successfully` ».

You can then click on the Business Template which you have just downloaded. You will see a form:



Proceeding with the installation, choose « `Install Business Template` » from the drop-down menu « `Action` ». This will prompt you to validate this action:



Once you click on « `Validate Workflow Action` », you just have to wait for a while. This takes from a few seconds to a few minutes, depending on the contents of this Business Template.

You have just installed the ERP5 Business Template for accounting. This module is essential for the Business Template of payroll. Let's continue now with the payroll. The principle is the same. You just have to import a different Business Template by giving it an ID such as `erp5_payroll` and the URL for downloading

http://cvs.erp5.org/cgibin/viewcvs.cgi/*checkout*/erp5_bt5/erp5_payroll.bt5?rev=HEAD&contenttype=text/plain

Here is an example of a payroll generated by this Business Template:

nexedi

FEUILLE DE SALAIRE

Nexedi SARL
943, avenue de la République
59700 Marcq-en-Baroeul
Etablissement : nexedi
Siret : 44004750400012
NAF (ex APE) : 722Z

N°URSSAF : 592 190071289 1013
URSSAF de Roubaix-Tourcoing
28, avenue de la Marne
59208 Tourcoing

Convention collective de branche : Syntec

Paulo Lavabo
4, rue de la Plomberie
99555 St-Amand-les-Eaux
N°Secu.Soc. : 3 70 07 42 555 777 05

Emploi : Directeur Technique
Qualification : ingenieur
Echelon : 1
Coefficient : 1

Periode : du 2003/02/01 au 2003/02/28
Date de paiement : 2003/03/04

Nature	Assiette	Montant	Taux part patronale	Part patronale	Taux part salariale	Part salariale
Salaire brut						**2000.00**
Assurance décès						
Assurance décès	Tranche A	2000.00	1.500 %	-30.00		
Assurance chômage						
Assurance chômage	Tranche A	2000.00	4.000 %	-80.00	2.400 %	-48.00
CRDS						
CRDS	Salaire brut CRDS	1900.00			0.500 %	-9.50
CSG						
CSG non déductible	Salaire brut CSG	1900.00			2.400 %	-45.60
CSG déductible	Salaire brut CSG	1900.00			5.100 %	-96.90
Taxes et participations						
Formation professionnelle	Salaire brut	2000.00	0.150 %	-3.00		
Apprentissage	Salaire brut	2000.00	0.500 %	-10.00		
Mutuelle Convention Syntec						
Mutuelle Syntec	Salaire plafonné SYNTEC	2432.00	0.915 %	-22.25	0.915 %	-22.25
Sécurité sociale						
Assurance maladie	Salaire brut	2000.00	12.800 %	-256.00	0.750 %	-15.00
Aide au logement	Salaire plafonné	2000.00	0.100 %	-2.00		
Accidents du travail	Salaire brut	2000.00	1.000 %	-20.00		
Assurance veuvage	Salaire brut	2000.00			0.100 %	-2.00
Assurance vieillesse	Salaire brut	2000.00	1.600 %	-32.00		
Assurance vieillesse	Salaire plafonné	2000.00	8.200 %	-164.00	6.550 %	-131.00
Allocations familiales	Salaire brut	2000.00	5.400 %	-108.00		
AGS (FHGS)						
AGS	Tranche A	2000.00	0.350 %	-7.00		
Retraite complémentaire						
AGFF	Tranche A	2000.00	1.200 %	-24.00	0.800 %	-16.00
ARRCO	Tranche A	2000.00	4.500 %	-90.00	3.000 %	-60.00
CET	Tranche A	2000.00	0.220 %	-4.40	0.130 %	-2.60
Total des cotisations				**-852.65**		**-448.85**
Salaire Net						**1551.15**
CRDS	Salaire brut CRDS	1900.00			0.500 %	+9.50
CSG non déductible	Salaire brut CSG	1900.00			2.400 %	+45.60
Salaire Net Imposable						**1606.25**

Cumuls annuels (2003)				
Salaire brut	Cotisations salariales	Salaire net	Net imposable	Part patronale
4000.00	897.71	3102.29	3212.49	1705.31

Durée des congés payés : 2 jours
Durée des délais de préavis : 3 mois

Dans votre intérêt et pour vous aider à faire valoir vos droits, conservez ce bulletin de paie sans limitation de durée.

Illustration 3: Une feuille de paie générée par ERP5

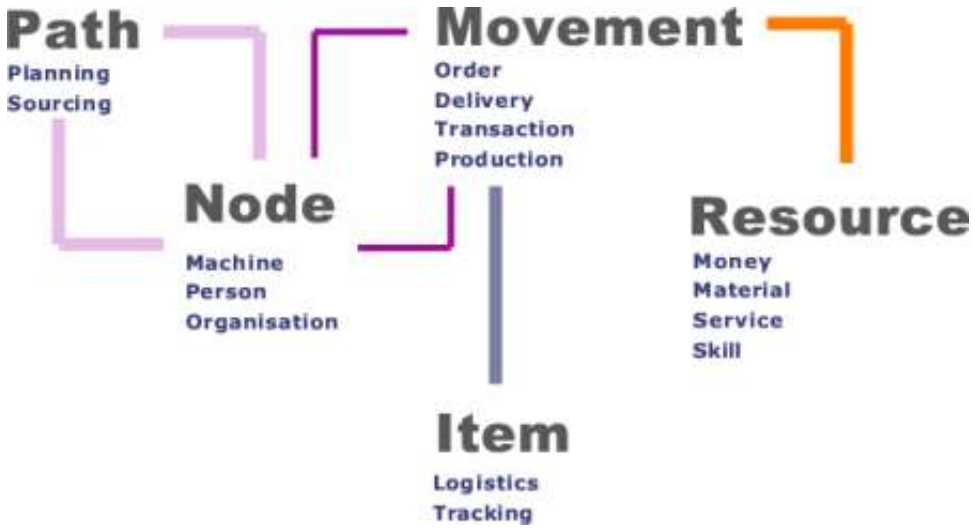
This payroll will be created, based on a document available on the Community site erp5.org (http://www.erp5.org/workspaces/project/erp5_payroll/erp5_pay_sheet_for_n). This English document intended for the beginners explains the procedures to be followed in the end. It will help you in the use of the payroll module. Now that you know the principle of the Business Templates from the point of view of users, you can start at the heart of this article: the creation of a Business Template.

3. Creating a Business Template

The objective of the second part of this article is to offer an overall picture of the ERP5 development platform to developers. The payroll module will be rebuilt in the end. This will give us the opportunity to show the user-friendliness and the power of ERP5 for the rapid creation of functional modules, and also the technological approach and the concepts which are the core of ERP5.

3.1. The Structural Design of Data

All ERP5 objects derive from 5 fundamental classes:



This ideal model of EPR5 has already been the subject of a scientific publication[5] which shall be interesting to consult to understand the motivation which lies behind each key concept of ERP5.

Here is the descriptions of the 5 fundamental classes:

- `Resource`
  A resource describes an abstract resource in a business process, such as an individual's skill, a currency, a raw material, or a product.

- `Node`
  Nodes can receive and send resources. They can relate to physical entities (such as a workshop that receives raw material, processes it, and sends it), or abstract entities (such as a bank account that can receive money). Stocks are a type of node. Metanodes are nodes that contain other nodes. For instance, a company is a metanode, and a project is both a resource and a node.

- `Movement`

5 http://www.computer.org/itpro/cover_stories/smets.htm

This class describes the movement of resources between two nodes at a given time and for a given duration. For example, a movement might send raw material from a stock to a workshop; another movement might send money from one account to another.

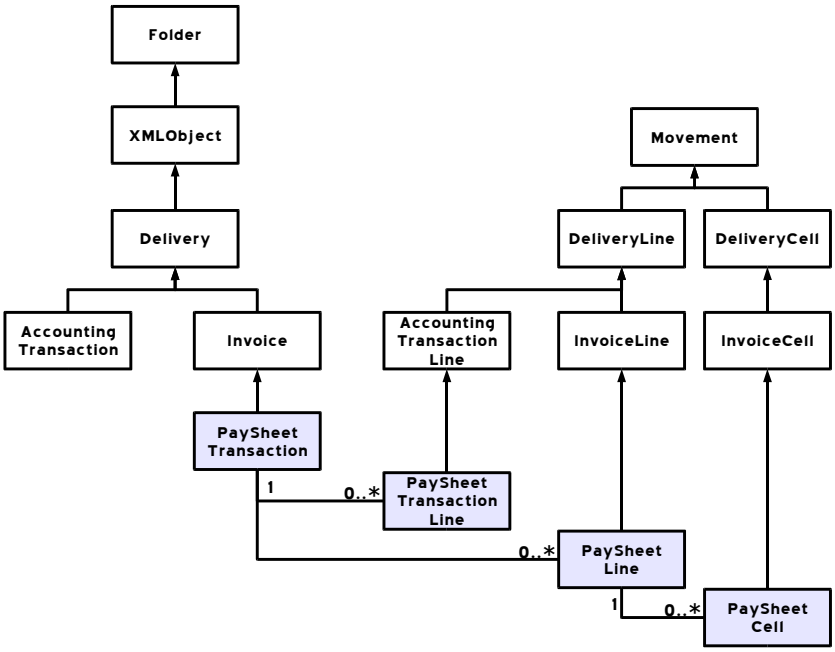
- `Path`
  A path defines a way for a node to access a resource it might need. Prices and commercial profiles can attach to a path to define the default price for a given resource produred at a given maker. A path can also define how a workshop obtains its resource from a stock. A path has a start and end date, and can represent the assignment of an individual to a temporary project.

- `Item`
  A physical instance of a resource is an item. A movement can expand into a series of traceable movements through items. Items also define how resources ship (such as by parcel or by listing the serial numbers of items in each container).

By analyzing a traditional French payroll, we chose to create 4 types of objects:

- `PaySheetTransaction`
- `PaySheetTransactionLine`
- `PaySheetLine`
- `PaySheetCell`

Here is a diagram of classes which gives the hierarchy and the inheritance making it possible to connect the base classes of ERP5 with our 4 new types of objects:



The structure of data of the payrolls is based on the model of invoicing (`Invoice`). This is described by 3 classes: `Invoice`, `InvoiceLine` and `InvoiceCell`. These three classes allow for depicting an invoice (`Invoice`) in lines of invoicing via `InvoiceLine` (a line of

invoice by invoiced product), which can be decomposed according to variations of the same invoiced product. `InvoiceCell` thus makes it possible, for example, to detail the invoicing of a quantity of four products A, of three products A with the color blue, and of one product A with the color green.

As the diagram shows it, the invoices specialize the representation of the delivery (`Delivery`) which implements the origin of the structure `Line/Cell`.

The cells (`Cell`) are the visible part of the concept of variation which ERP5 offers. This concept makes it possible to represent variations of the same resource, like the color, the size, or the speed. This makes it possible to define several configurations of the same product.

The advantage of this system lies in the possibility of very deeply configuring the ERP5 while keeping the benefit of a generic management of data, according to a standard structure. We will detail the concept of variation a little further in this article.

Here is a detailed information on our various classes:

- `PaySheetTransaction`

  The payroll is represented by an object of the type `PaySheetTransaction`.

  This class derives from the basic class `Folder` via `Invoice`. We chose this inheritance, because the purpose of `PaySheetTransaction` is to simply put objects related to a payroll in a single place. This is why `PaySheetTransaction` does not have any ancestor class from the 5 basic classes of ERP5. `Folder` is, to some extent, an "utility class" which allows us to reduce the entropy of the system.

  Although `PaySheetTransaction` is of the type `Folder` and it behaves similarly with an invoice (`Invoice`), some behaviors are specific to a payroll. So we had to create a new class. These particular behaviors take into account the concept of employer and employee as well as other data specific to the payroll (the date of a payment, a gross salary).

- `PaySheetLine`

  As the diagram indicates it, each `PaySheetTransaction` can contain a number of `PaySheetLine` which are the representation of various contributions withdrawn on a gross salary. For each contribution, there is one and only one `PaySheetLine`.

- `PaySheetCell`

  `PaySheetLine` contains `PaySheetCell`, which may represent and detail the amounts of `PaySheetLine`. The variation system will make it possible to distinguish the employer's and employee's share from a given social security contribution, and to preserve a trace of the calculation method of the amounts.

  For a given contribution, the total amount of one part (of a salary or of an

employer's) may be the result of a scientific calculation utilizing one or more rates, associated with a tax base respectively (gross salary, ceiling salary, sliced salary A/B/C, and so on). The calculation of the payrolls is not obvious, especially as the legislation changes every day.

The variation reduces the amount of code in this case, because it is possible to tackle all the complications induced by the legislation of each country with a small piece of code.

- `PaySheetTransactionLine`

  `PaySheetTransactionLine` is the countable representation of the payroll. We have decided to distinguish `PaySheetLine` from `PaySheetTransactionLine`, because the representation of the payroll in an accounting logic gathers certain amounts, and this makes it impossible a fine reporting on the pay.

We thus have within `PaySheetTransaction` two sets:

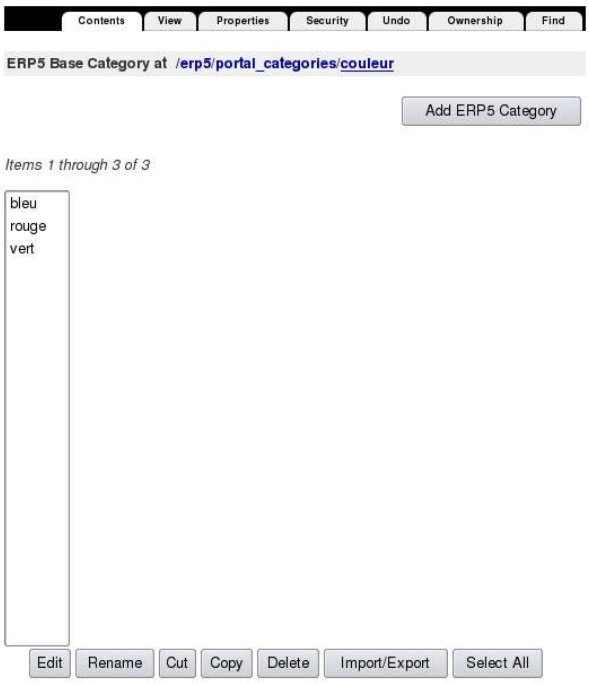
- The data of the couple of `PaySheetLine`/`PaySheetCell` : a set of basic data with which we can display the details of the payroll, and handle the figures in all the directions.

- The data of the type `PaySheetTransactionLine`, which represents the accounting lines of the payroll.

Now that we have a coherent data model, we will implement it in ERP5.

3.2. Variation by categories

The objects of the type `Category` are used in ERP5 for classification. For example, a document can have an attribute `color`. Instead of letting the user enter a free value, one will prefer to allot a category to the object via a text field for the attribute `color`. One will thus create the categories `color/blue`, `color/green`, and `color/red` via the ZMI:



The categories can contain subcategories. For example, to describe the geographical areas:

```
region/europe
region/europe/west/
region/europe/west/france
region/europe/west/germany
region/europe/south/spain
region/americas
region/americas/north
region/americas/north/us
region/americas/south
region/asia
```

In this example, the base category is `region`.

The categories have other advantages, such as automatic cataloguing. This mechanism makes it possible the use of SQL queries to select the ZODB objects, according to the value of the category (refer to the section « The point on cataloguing »).

It is possible to create *virtual* classifications based on existing documents or categories. For example, if a document is accessible with the path `organisation/nexedi`, and there is a base category `client`, the tool `portal_categories` allows the creation of the virtual category `client/organisation/nexedi`. This allows for representing the relation: « the client of such an object is the organisation Nexedi ».

The concept of the virtualisation of categories avoids the duplication of information by offering mechanisms equivalent to a base of relational data.

3.3. Integrating the data structure with ERP5

We will add all the classes defined in the previous section into ERP5. ERP5 has clean mechanisms to define new classes of objects, which permits it to deal with dynamic and automated creation of methods. All we have to do with the new classes is thus describe them. ERP5 will take care of the remainder, like the dynamic creation of the « setter » and « getter ».

The classes are defined in a file under `Document` of the Zope/ERP5 directory in Python. We will create a new class `PaySheetTransaction`, and, to avoid seizing again all the code, make a copy of the file `Invoice.py`:

```
[root@localhost /]# cd /usr/lib/zope/lib/python/Products/ERP5/Document/
[root@localhost Document]# cp Invoice.py \
                             /var/lib/zope/Document/PaySheetTransaction.py
[root@localhost Document]# cd /var/lib/zope/Document/
[root@localhost Document]# vi PaySheetTransaction.py
(…)
[root@localhost Document]# chown zope:zope PaySheetTransaction.py
[root@localhost Document]# chmod 0644 PaySheetTransaction.py
```

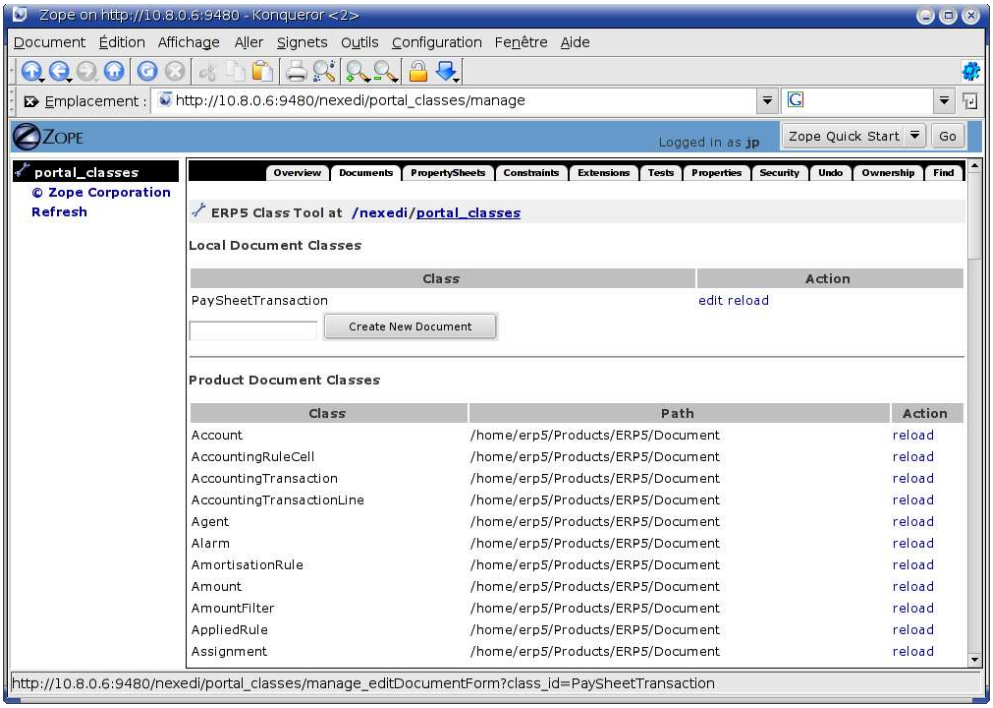
Here is what the file `PaySheetTransaction.py` looks like after editing:

```
from AccessControl import ClassSecurityInfo
from Products.ERP5Type import Permissions, PropertySheet, Constraint, Interface

from Products.ERP5.Document.Invoice import Invoice

class PaySheetTransaction(Invoice):
  """
    A paysheet will store data about the salary of an employee
  """

  meta_type = 'ERP5 Pay Sheet Transaction'
  portal_type = 'Pay Sheet Transaction'
  add_permission = Permissions.AddPortalContent
  isPortalContent = 1
  isRADContent = 1

  # Declarative security
  security = ClassSecurityInfo()
  security.declareObjectProtected(Permissions.View)

  # Default Properties
  property_sheets = ( PropertySheet.Base
                    , PropertySheet.SimpleItem
                    , PropertySheet.CategoryCore
                    , PropertySheet.Task
                    , PropertySheet.Arrow
                    , PropertySheet.Delivery
                    , PropertySheet.Movement
                    , PropertySheet.Amount
```

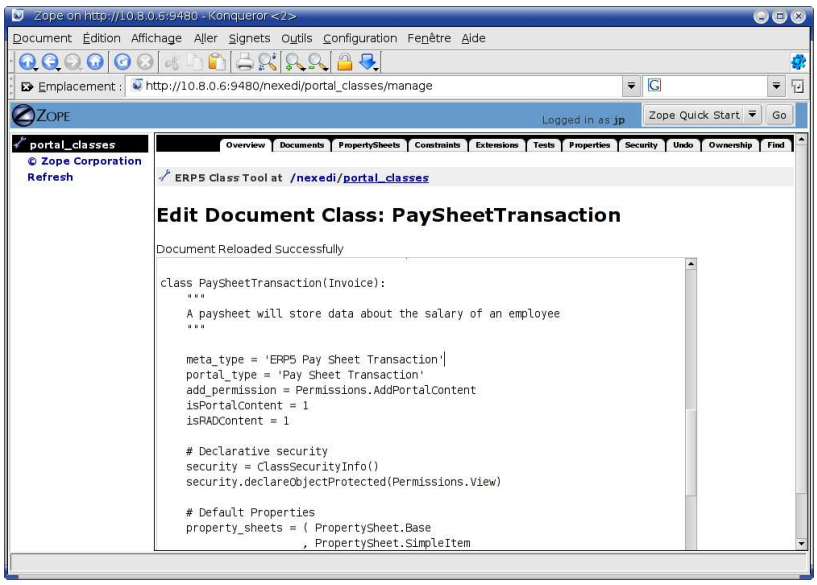
```
                    , PropertySheet.XMLObject
                    , PropertySheet.PaySheetTransaction
                    )

# Declarative Interface
__implements__ = ( )
```

3.4. Fast Web Development with portal_classes

It is possible to accomplish all the operations of creation of classes or editing of the code via the Web interface of ERP5, thanks to the tool `portal_classes`:



The editing of python code of an ERP5 class also makes it possible to reload the only modified code without starting Zope again, and to gain in productivity.



The tool `portal_classes` is very useful in a prototyping phase or for fast development. It also facilitates ERP5 training. It is, however, not advised to activate it on a system in production, because it involves high security risk. Also, it is deactivated by default in ERP5 Live CD or ERP5 packages for Mandriva Linux. To activate it, it is thus necessary to create an empty file `ALLOW_CLASS_TOOL` in the `ERP5Type` product repository.

3.5. The structure of a Document class in ERP5

The files of the type `Document` generally look the same, and the values specific to our new class would be guessed easily.

Each file of the Zope/ERP5 instance uses `PropertySheet` which contains Python scripts similar to those which one finds in the directory `PropertySheet` of the product `ERP5`. These scripts with the evocative names may define attributes of classes, and, by extension, the diagram of data of ERP5. The classes usually refer to them via the list `property_sheets`, as you can see it in the preceeding Python code.

To allow each payroll to store a gross salary, we have to refer to the property sheet `PaySheetTransaction` which will contain the definition of the attribute `gross_salary` of the type `float`. We should now create it:

```
[root@localhost /]# cd /var/lib/zope/PropertySheet/
[root@localhost PropertySheet]# vi PaySheetTransaction.py
(…)
[root@localhost PropertySheet]# chown zope:zope PaySheetTransaction.py
[root@localhost PropertySheet]# chmod 0644 PaySheetTransaction.py
```

Here is the contents of the property sheet `PaySheetTransaction.py`:

```
class PaySheetTransaction:
  """
```

```
    Properties for PaySheet Transaction objects
  """

  _properties = (
    { 'id'          : 'gross_salary'
    , 'description' : 'This variable contain the gross salary of the paysheet'
    , 'type'        : 'float'
    , 'mode'        : 'w'
    },
  )

  _categories = ( 'source'
                , 'destination'
                , 'source_section'
                , 'destination_section'
                )
```

The syntax is rather intuitive, because all the attributes are declared with a list of Python dictionaries in the variable `_properties`. Pay attention to the immutable lists (tuple) of one element in Python, and do not forget the comma before the ending bracket.

```
[kevin@localhost Products]$ python
Python 2.3.3 (#2, Feb 17 2004, 11:45:40)
[GCC 3.3.2 (Mandrake Linux 10.0 3.3.2-6mdk)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> (5)
5
>>> (5,)
(5,)
>>> (5,10)
(5, 10)
>>>
```

Thus `(5)` is interpreted as an integer `5`, and `(5,)` as an immutabe list containing only one element equal to `5`.

The list `_categories` contains the list of virtual categories. We will use those to create a relation among a payroll, an employer, and an employee. The relation `source_section` will refer to an employer (who is the organization or the company which generates a payroll), and the relation `destination_section` will refer to an employee (the person for whom a payroll is generated).

If we need to define more complex attrbutes later, we will begin with the `PropertySheets` preexistent in the product `ERP5`. This method is valid for all the approaches in this article: since the source code is open, it should not be deprived to reuse or to take existing things to build new things gradually. It is even an obligation to contribute to the code of ERP5: two similar properties must, by principle, have the same name in ERP5.

Now that our new class is built, it is necessary either to reload its code (with

portal_classes on the class of PropertySheet then on the class of Document), or to restart the Zope server, so that it can take them into account. Once this is done, the server will be available, and you will be able to reach the ZMI:

```
[root@localhost kevin]# zopectl
program: /usr/bin/runzope
program running; pid=4450
zopectl> restart
. daemon process restarted, pid=4773
zopectl> logtail
------
2004-05-12T18:50:03 INFO(0) Z2 Caught signal SIGTERM
------
2004-05-12T18:50:03 INFO(0) Z2 Shutting down fast
------
2004-05-12T18:50:03 INFO(0) ZServer closing HTTP to new connections
------
2004-05-12T18:50:03 INFO(0) ZServer closing FTP to new connections
------
2004-05-12T18:50:03 INFO(0) Zope Shutting down with exit code 0
( )
------
2004-05-12T18:50:42 INFO(0) Zope Ready to handle requests

zopectl>
```

It is possible that, at the time of the restarting, Zope stops, and obstinately refuses to start. It is also possible to observe a strange behavior of Zope: looping stopping and starting of the server every 10 seconds.

The causes of this blocking are mostly bad permissions on scripts or bad ownership on the files which you have just created. A series of chmod and chown will fix this problem.

If the problem persists in spite of these corrections, zopectl integrates a command debug, which stops the server and starts it by posting the log of starting and a report of the execution (traceback) in case of an error.

For example:

```
[root@localhost kevin]# zopectl
program: /usr/bin/runzope
daemon manager not running
zopectl> debug
Starting debugger (the name "app" is bound to the top-level Zope object)
(…)
Traceback (most recent call last):
  File "<string>", line 1, in ?
  File "/usr/lib/python2.3/site-packages/PIL/__init__.py", line 50, in app
  File "/usr/lib/python2.3/site-packages/PIL/__init__.py", line 46, in startup
  File "/usr/lib/zope/./build/build-base/python-2.3/build-
lib/Zope/App/startup.py", line 45, in startup
```

```
  File "/usr/lib/zope/./build/build-base/python-2.3/build-
lib/OFS/Application.py", line 631, in import_products
  File "/usr/lib/zope/./build/build-base/python-2.3/build-
lib/OFS/Application.py", line 654, in import_product
  File "/usr/lib/zope/lib/python/Products/ERP5/__init__.py", line 50, in ?
    from Tool import Category, CategoryTool, SimulationTool, RuleTool, IdTool,
TemplateTool, TestTool
  File "/usr/lib/zope/lib/python/Products/ERP5/Tool/Category.py", line 35, in ?
    from Products.ERP5.Document.MetaNode import MetaNode
  File "/usr/lib/zope/lib/python/Products/ERP5/Document/__init__.py", line 3145,
in ?
    import PaySheetTransaction as ERP5PaySheetTransaction
  File "/usr/lib/zope/lib/python/Products/ERP5/Document/PaySheetTransaction.py",
line 62
    __implements__ = ( )
                  ^
SyntaxError: invalid syntax
>>>
```

The `traceback` shows that there is an invalid syntax error in line 62 of the file `/usr/lib/zope/lib/python/Products/ERP5/Document/PaySheetTransaction.py`. While publishing, it says that the developer forgot to close the list `property_sheets` with a bracket. Once this small mistake is corrected, the Zope server will start fine.

Caution: to use the debug command, one should not forget to turn on the option `debug-mode` in the file `/etc/zope.conf`. It is not surprising to have to use `debug` 30 times per day when one is a beginner and does not undestand the syntax of Python, or when one works on the CVS of an ERP5 project in a team and conflicts emerge.

Based on the same principle, we built the new classes `PaySheetTransactionLine`, `PaySheetLine` and `PaySheetCell`. If you are interested, take a look at the definitions of the classes:

```
[root@localhost /]# cd /usr/lib/zope/lib/python/Products/ERP5/Document/
[root@localhost Document]# vi PaySheetTransactionLine.py
[root@localhost Document]# vi PaySheetLine.py
[root@localhost Document]# vi PaySheetCell.py
```

You might have noticed that the code of the payroll module is sometimes located in the directory `/usr/lib/zope/lib/python/Products/ERP5/Document/` and sometimes in the directory `/var/lib/zope/Document`. An explanation is essential. The directory `/var/lib/zope/Document` is designed for the rapid development of code of extension to the ERP5 *framework*. So It is useful to start to work in this directory when one conceives a new set of classes for a customer or a future extension of ERP5. If this set of classes seems sufficiently generic, and likely to correspond to a universal function of an ERP, it would be better to integrate it into the project ERP5, in order to share it with the whole community. This is the case with the payroll. On the other hand, if this set of classes is related to a trade or the specific needs for a company, there is no reason to integrate it into the core of ERP5.

As the payroll mode is a part of the core of ERP5, it often refers to files located in the

directory `Document` of ERP5. However, for the needs of the tutorial, we creates all the new classes in the Zope/ERP5 directory, i.e. generally `/var/lib/zope`.

3.6. Creating a module in EPR5

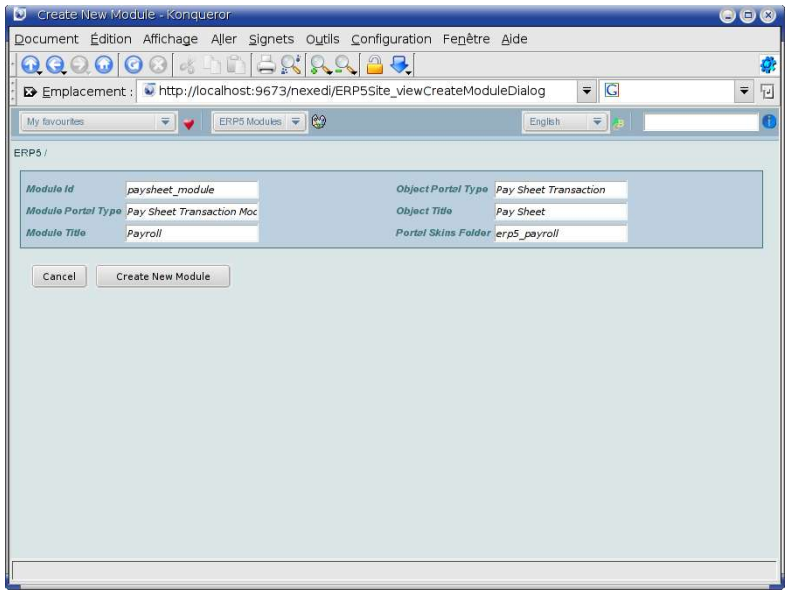
During the introduction of this section, we mixed up the Business Templates with the modules of ERP5. We now make the difference, and more precisely describe the role of a module, within the context of ERP5, then create one.

A module is simply a place in the user interface of ERP5 to gather the functionalities suitable for an aspect of the management of a company. A Business Template contains all scripts and objects which realizes a module function.

The fast creation of a new module can be carried out by going to the URL:

```
http://localhost:9480/erp5/ERP5Site_viewCreateModuleDialog
```

where appears the following dialogue:



This dialogue consists of 6 parameters:

- Module ID : The ID of the folder (`Folder`) created in the root of the ERP5 site which shall contain all the payrolls.
- Module Portal Type : The name of the document type which will be created to contain all the payrolls.
- Module Title : The title which will appear in the menu of modules for the payroll module.

- Object Portal Type : The name of the document type which will be created to define a payroll.
- Object Title : The title which will appear for each payroll by default.
- Portal Skins Folder : The folder in which the forms will be created by default.

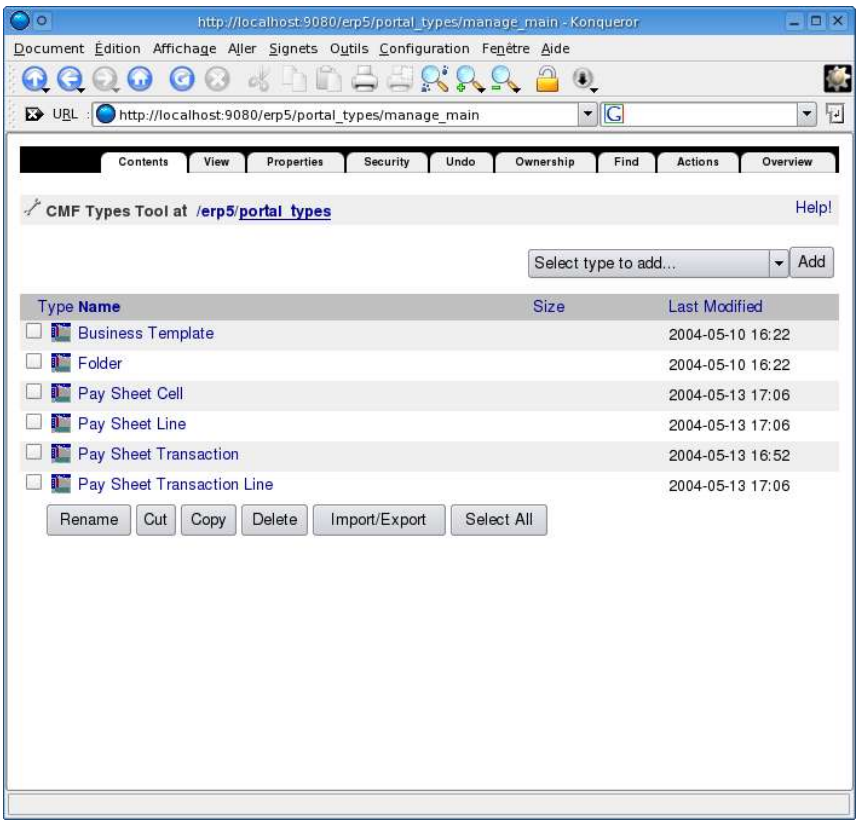
On clicking on « Create New Module », all the document types are automatically created, and the menu is updated. An « empty » payroll module is created. Now the question is how to configure it.

3.7. Integrating the classes, data, menus and appearance: the tool portal_types

ERP5 is constructed upon the architecture Zope/CMF. One of the characteristics of the CMF is the concept of tool placed at the root of a portal, and designed to provide various services: translation of formats of documents, management of menus associated with a document, internationalization, synchronization, etc.

The tool `portal_types` can create generic types of documents on the basis of our classes defined in the preceding part, and can associate menus to them as well as elements of appearance. We will make the distinction thereafter between `portal_types` (the tool) and portal types (types of documents defined by the tool `portal_types`).

The contents of the tool `portal_types` corresponds to the modules and the document types available in the ERP5 instance.



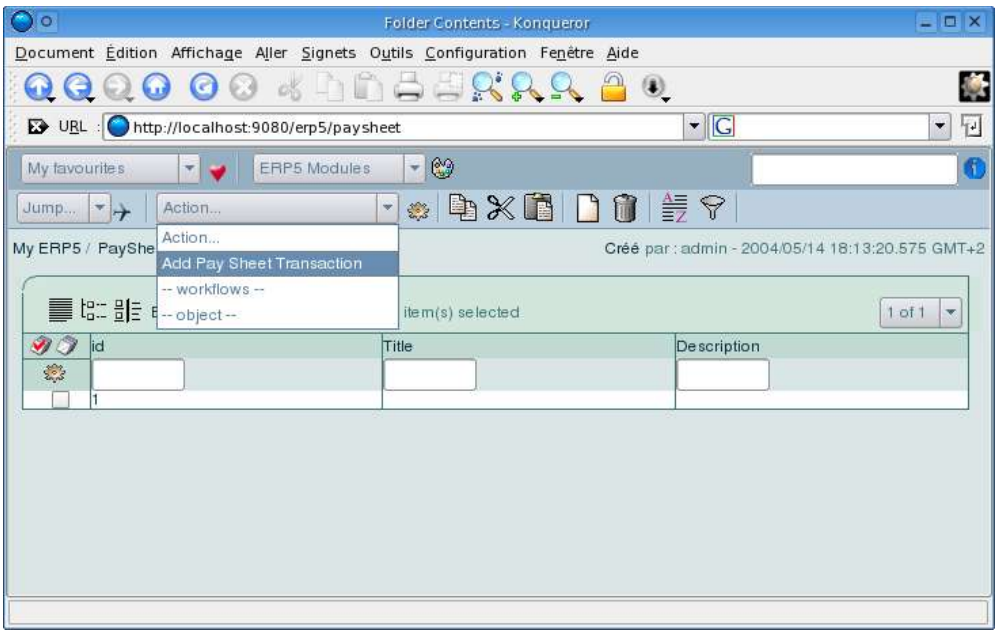
In short, all our portal types `PaySheet*` are images of our classes defined by our Python scripts in Zope.

The reality is more subtle, because we can have two different portal types which are based on the same class. This is because their fundamental role is to create document types. For example, from the same class `WrittenDocument`, we can create two different portal types: `Article` and `PressRelease`. Both are described by the same structure of data and the same properties. The interest in creating two types of documents is that we can apply a different workflow or an adapted appearance of a page to each type.

3.8. The user interface with ERP5Form

Here, we shall exploit the ability of RAD (Rapid Application Development environment) of ERP5 via the product ERP5Form, in order to define the user interface of our module. ERP5Form is the generator of forms of ERP5. It is derived from Formulator, a Zope product created by Infrae.

In the paysheet module, we can add a `PaySheetTransaction`:



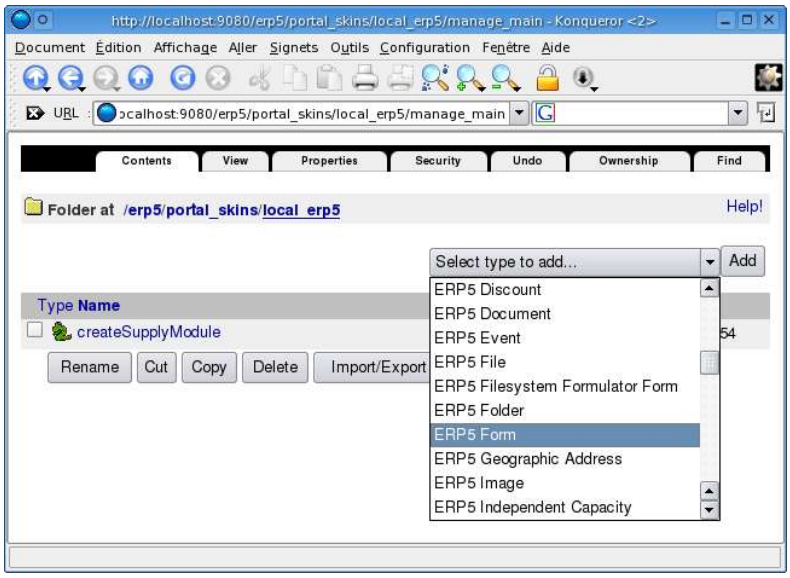
But our action finishes on an error:

However, the method of creating a new payroll has functioned successfully, as shown in the ZODB a new instance of the portal type `Pay Sheet Transaction` which is created in the `paysheet` module:

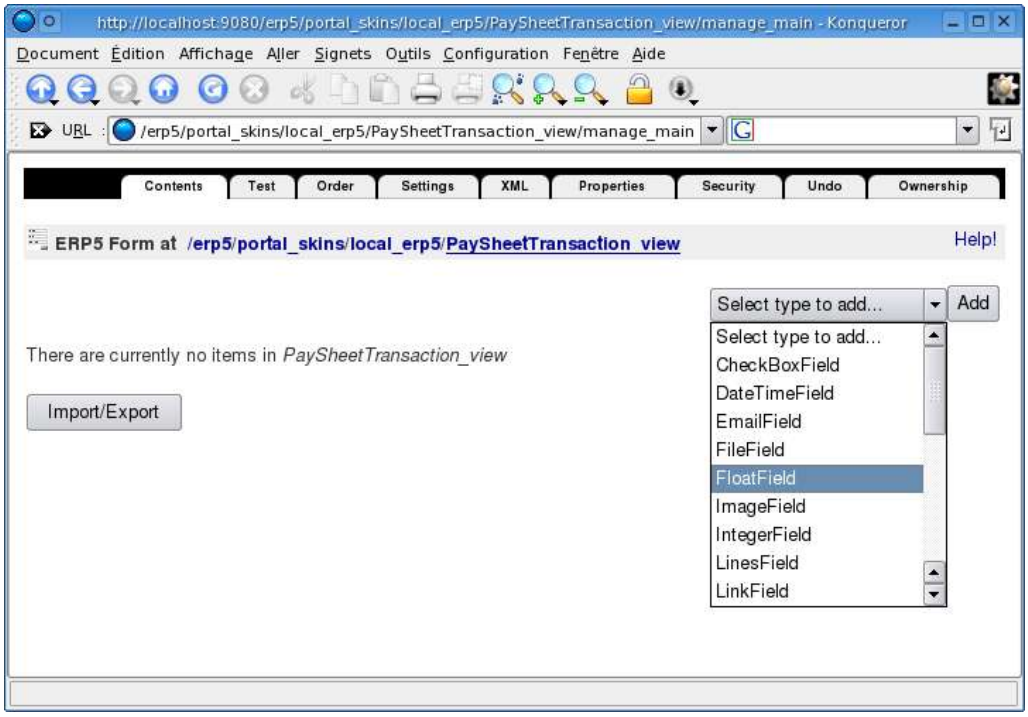
It is thus only about a problem of display. To solve it, we must add a user interface to the payroll, so that the action of creation does not lead into an error, when it tries to get an object created.

So we create a new form. Like all the forms and the other presentation objects, it is defined in a folder under of the tool `portal_skins`. Let us go, for example, in the folder `/erp5/portal_skins/local_erp5`.

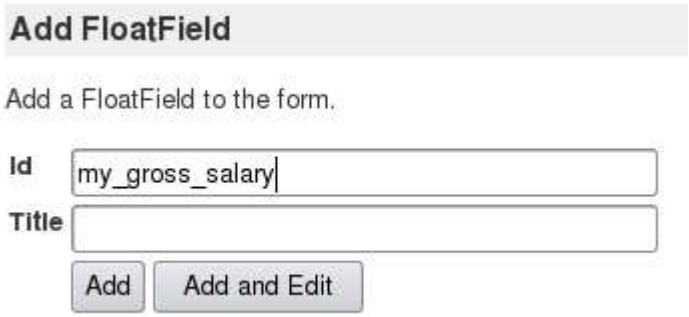
A form is created, like any other zope object, via the ZMI: just choose the type « ERP5 Form ». Name our form `PaySheetTransaction_view` :



Begin to edit it by adding a field of the type float to represent a gross salary:

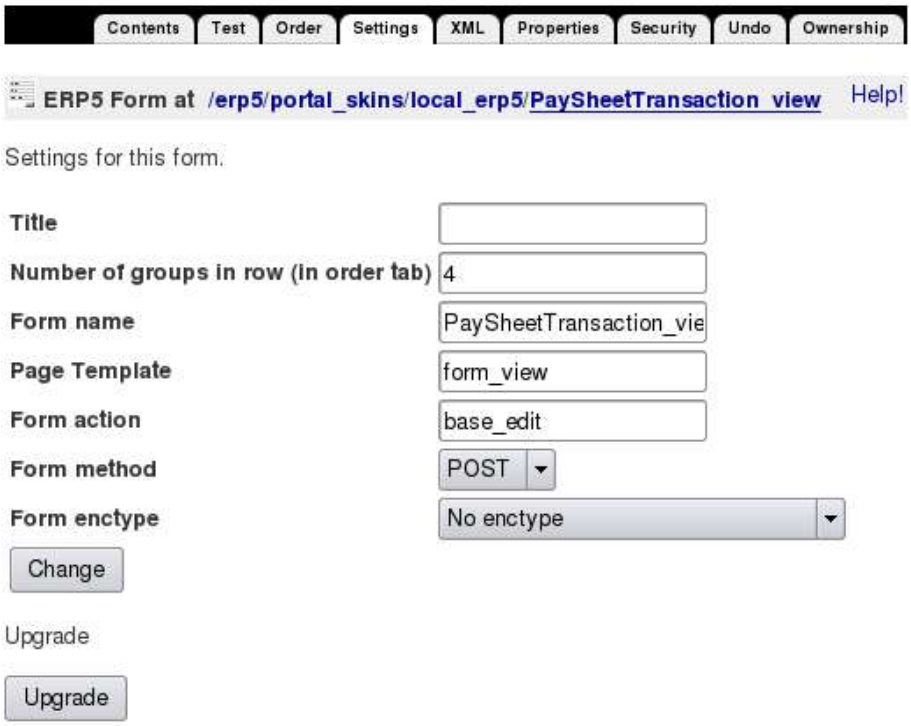


To state implicitly that this field corresponds to the attribute `gross_salary` of our object, it is enough to name it `my_gross_salary` :



The use of the prefix `my_` is one of the naming conventions specific to ERP5. If a field of a form is named `my_something`, it will get, by default, a value obtained from calling the method `getSomething` on a document to which it is applied. In the event of validation of the form, the method `setSomething` will be called on the document to which the form was applied, to update the value of the property `something`.

To be complete, the form must be configured, so that a gross salary can be modified by defining the property `Form action`, as on the screenshot:

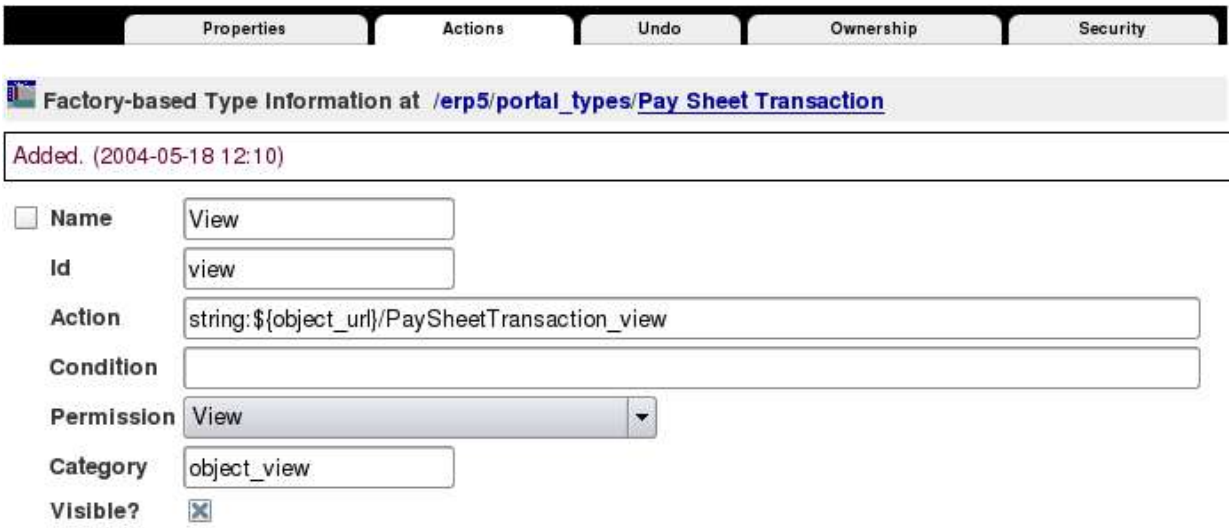


The action « Base_edit » is called when one validates the form and takes into account the data given by the user to update the document to which the form is applied.

To test that our form functions, directly call this URL:

```
http://localhost:9080/erp5/paysheet/1/PaySheetTransaction_view
```

Let us go back to the tool `portal_types`, so that this form is used by default to display a payroll, We go to the tab `Actions` of the portal type corresponding to a payroll. One can then define a new action:



The choice of the category « `object_view` » corresponds to the concept of action in ERP5. The choice of the ID « `view` » corresponds to the default display. From now on,

when we add a new payroll from the drop-down menu `Actions` of the paysheet module, the form automatically will be displayed with information of a gross salary:

Note that both URLs:

```
http://localhost:9080/erp5/paysheet/2/view
```

and

```
http://localhost:9080/erp5/paysheet/2/PaySheetTransaction_view
```

give the same result, because of the association of `PaySheetTransaction_view` to the ID `view` in the actions of the portal type.

It may be interesting, at this stage, to test some URLs typical of ERP5, in order to understand the concept of accessor. For example, The definition of the property « `gross_salary` » in the PropertySheet of payroll results in the automatic generation of a method « `getGrossSalary` » by ERP5. This method can be tested directly by calling an URL on the payroll:

```
http://localhost:9080/erp5/paysheet/2/getGrossSalary
```

This URL displays the value of `gross_salary` converted in the form of string in the browser.

By using the properties of the form and the fields which constitute it, one can quickly obtain a form more developed and more useful. One begins to add fields for properties which one wishes to manage with the form:

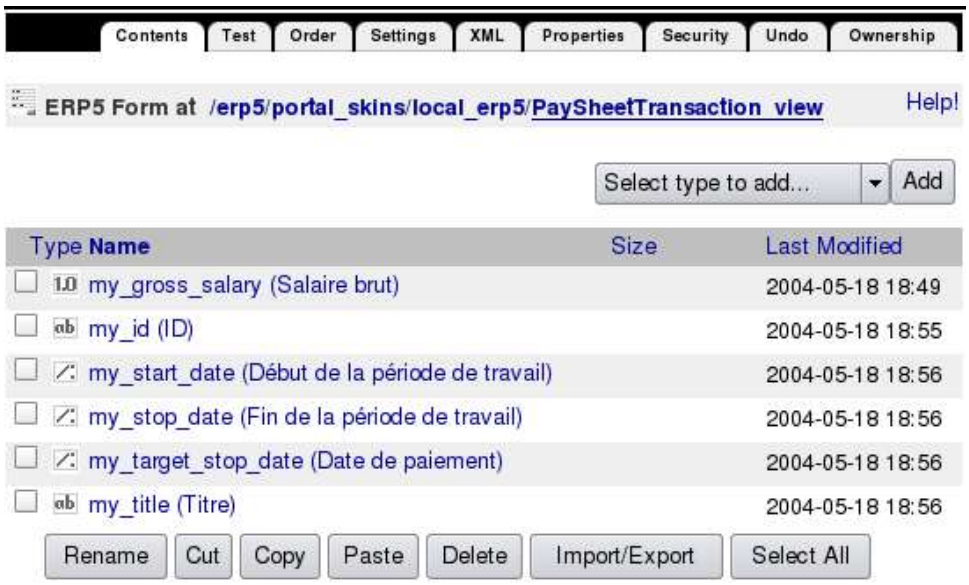


Illustration 4: New fields for attributes of the object

The tab « `Order` » can collect the fields and to define their appearance:

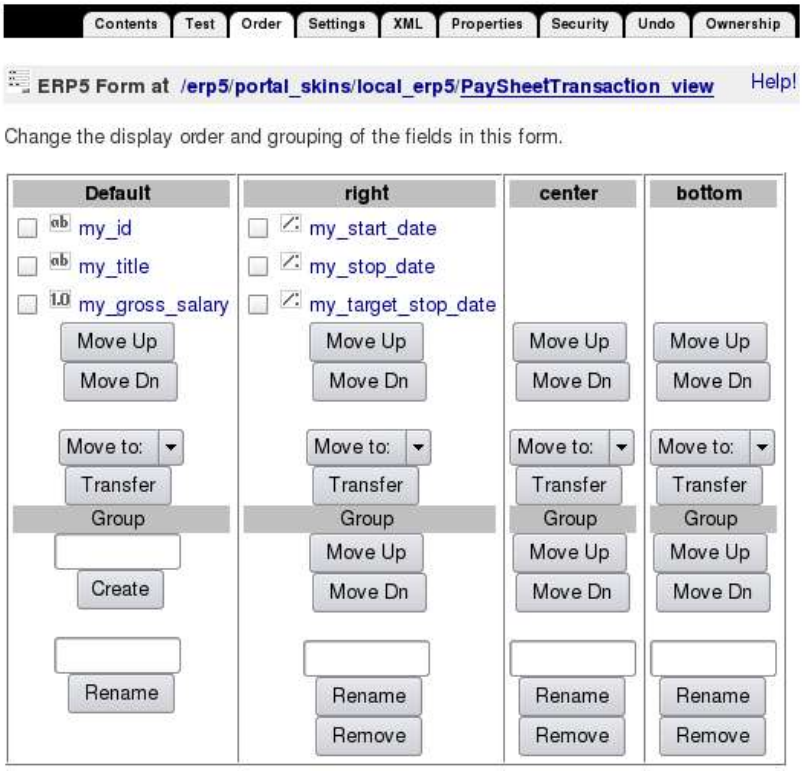


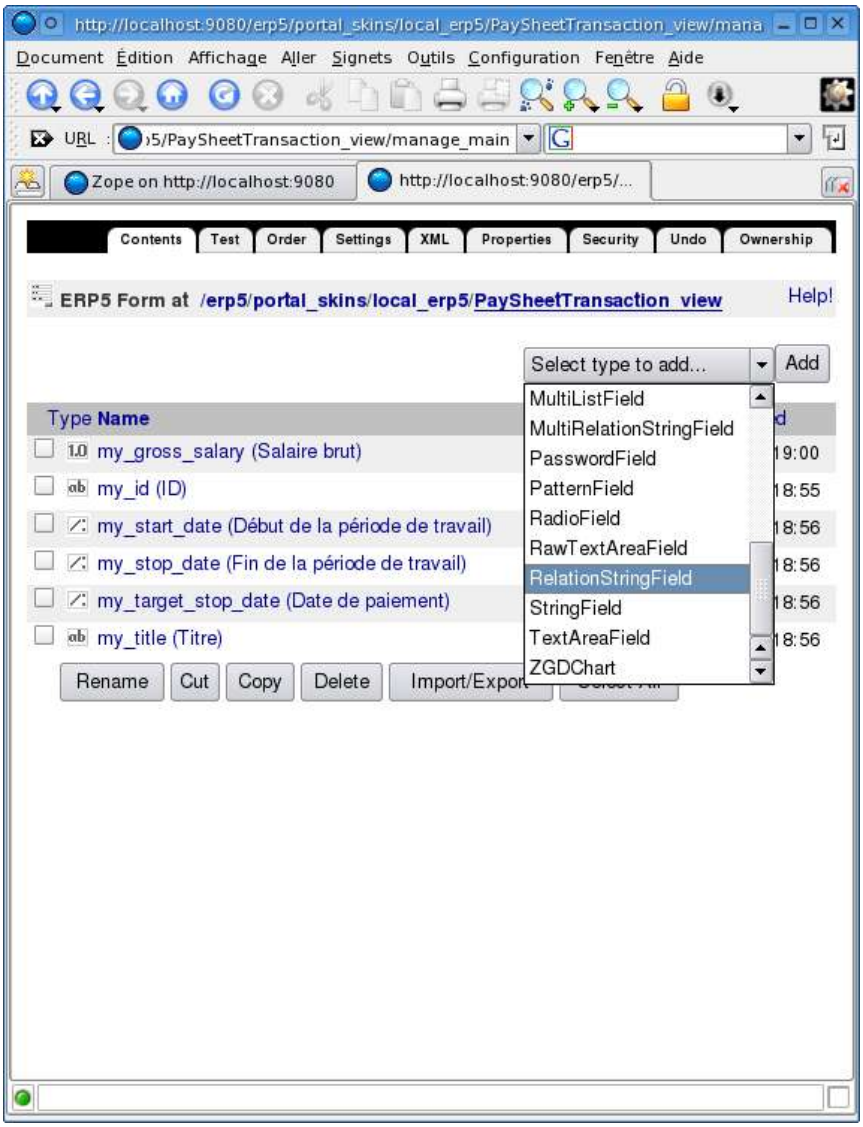
Illustration 5: Use of groups for the representation of the form

The style of ERP5, by default, uses 4 groups: `left` (or `Default`), `right`, `center` and `bottom`. We have used only the first two groups at the moment, which results in a display with two columns:

Illustration 6: The form appearance in use

Now let us add two fields to associate a payroll to an employer and to an employee.

The fields to be added are of the type `RelationStringField` :



We add two fields of this type, `my_destination_section_title`, which points to an employee for whom a payroll is intended, and `my_source_section_title`, which is for an employer who generates a payroll:

Type	Name
ab	my_destination_section_title (Salarié)
1.0	my_gross_salary (Salaire brut)
ab	my_id (ID)
ab	my_source_section_title (Employeur)
	my_start_date (Début de la période de travail)
	my_stop_date (Fin de la période de travail)
	my_target_stop_date (Date de paiement)
ab	my_title (Titre)

It is then necessary to modify the properties of both widgets to make them functional:

Widget properties

Name	Value	Field
Title*	Salarié	StringField
Description		TextAreaField
Default		StringField
CSS class		StringField
Alternate name		StringField
Hidden	☐	CheckBoxField
Display width*	20	IntegerField
Maximum input		IntegerField
Extra		StringField
Update Method*	base_update_relation	StringField
Jump Method*	base_jump_relation	StringField
Base Category*	destination_section	StringField
Portal Type*	Person | Person	ListTextAreaField
Catalog Index*	Title	StringField
Default Module*	person	StringField

Widget properties

Name	Value	Field
Title*	Employeur	StringField
Description		TextAreaField
Default		StringField
CSS class		StringField
Alternate name		StringField
Hidden	☐	CheckBoxField
Display width*	20	IntegerField
Maximum input		IntegerField
Extra		StringField
Update Method*	base_update_relation	StringField
Jump Method*	base_jump_relation	StringField
Base Category*	source_section	StringField
Portal Type*	Organisation | Organisation	ListTextAreaField
Catalog Index*	Title	StringField
Default Module*	organisation	StringField

As one can guess this with the preceding captures, `Base Category` may indicate the basic category on which the relation is based. Also remember that this category has been explicitly declared in the adequate PropertySheet, to precisely be able to be used at this place.

The relation of an employer is now definable by a virtual category of the form `source_section/organisation/nexedi`. One can check the integrity of the data by directly calling the getter of the attribute on the object, via this URL:

```
http://localhost:9080/erp5/paysheet/2/getSourceSection
```

To understand how this information is managed by ERP5, it can be useful to call the

method `Base_viewDict` which can inspect the contents of an object recorded in the ZODB:

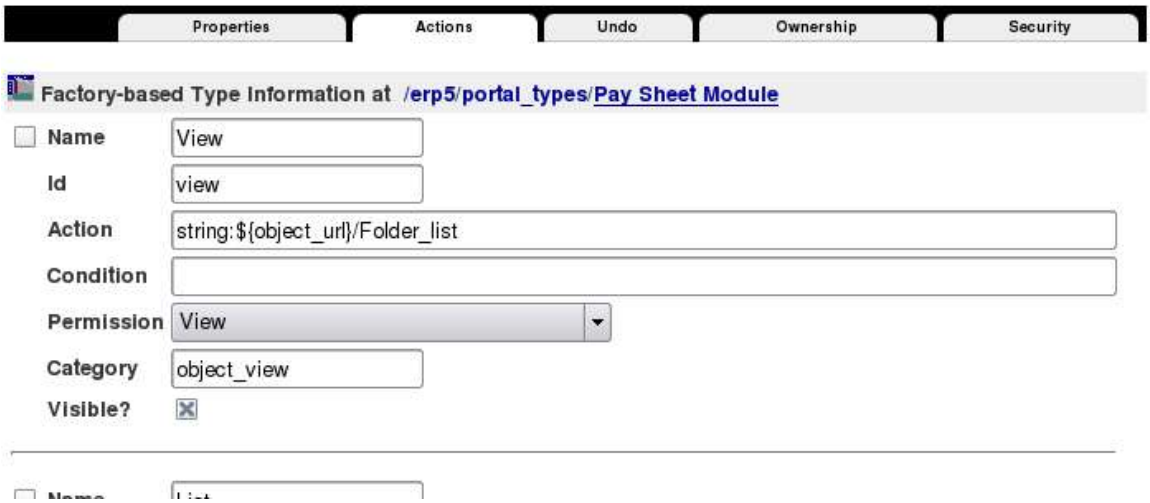
```
http://localhost:9080/erp5/paysheet/2/Base_viewDict
```

The property `Portal Type` of this type of widget may specify the types of objects which will be accepted as a relation by the field. One will find this property in different widgets of the product Formulator as well.

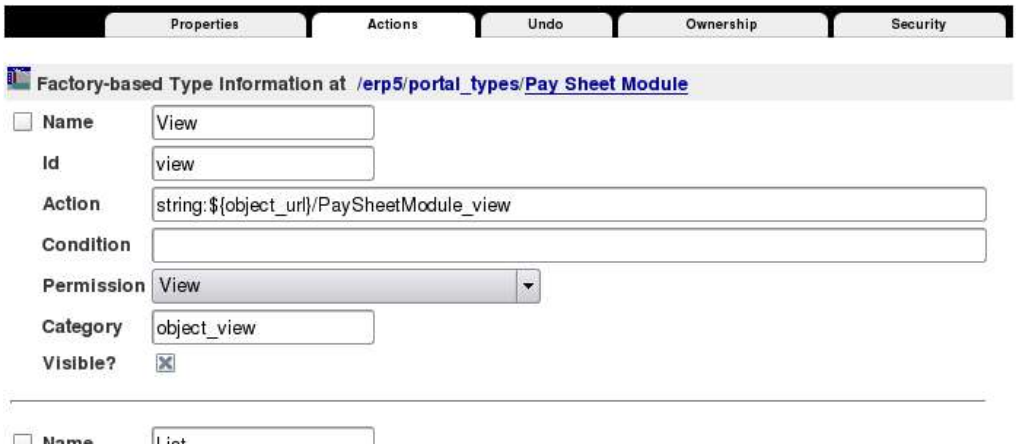
The other properties of this type of field are rather intuitive.

We are gettting close to the final form:

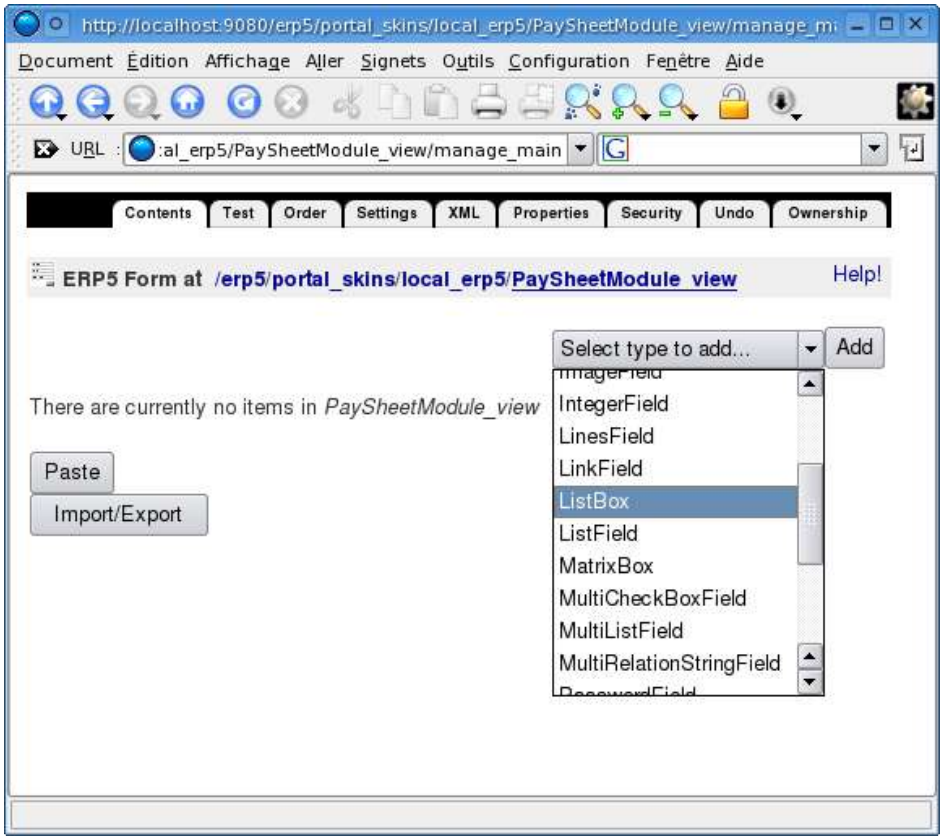
Just as there is an action `view` connected to the form `PaySheetTransaction_view` defined in the portal type `Pay Sheet Transaction`, there exists an action `view` on the portal type of the module which is `Folder_list` by default:



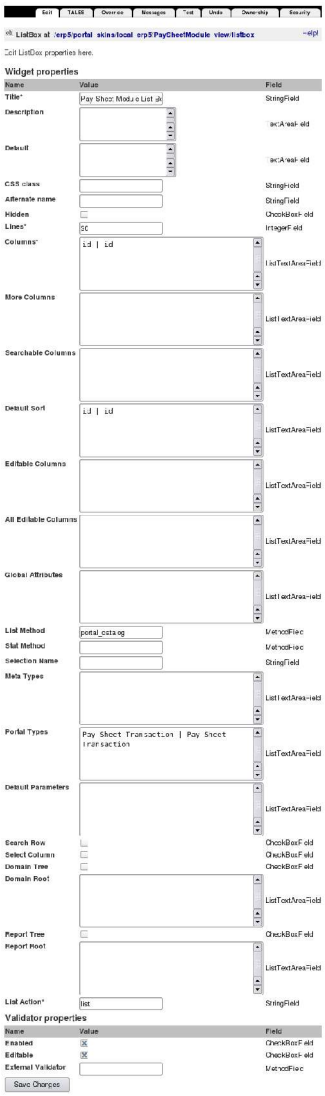
We create our own form `PaySheetModule_view` which will displays the contents of the module. For this, it is necessary to change the actions of the module, and to create a new form:



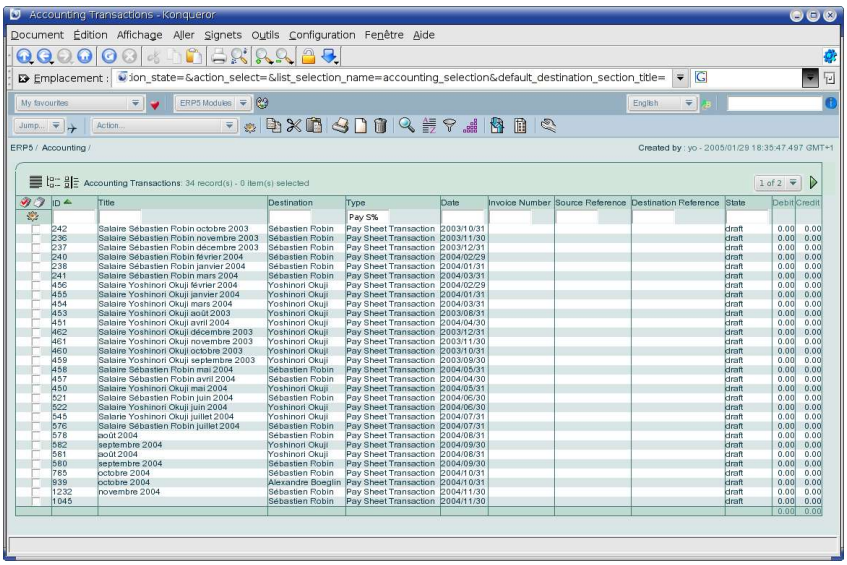
This form needs to contain one widget of the type `ListBox`:



By convention, we always call it `listbox`. ListBox is very flexible, and we use it to display a summary of the contents of the paysheet module. The following screenshot shows you the properties of ListBox:



From point of view of users, this form has the following look:



The important properties of ListBox are `List Action` and `List Method`. The latter takes the value `portal_catalog`, to indicate that the method used for the generation of the list is based on the catalogue.

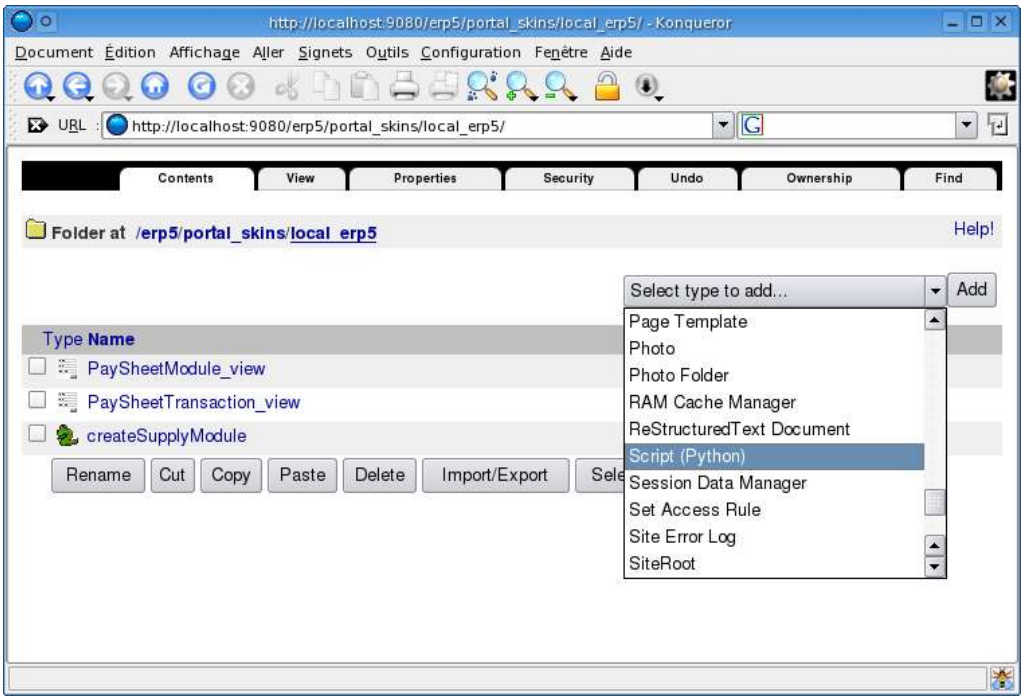
The property `Columns` defines the order and the data of the columns of the list.

The rest of the properties makes it possible to manage an advanced behavior of the list, such as sorting, the selection of lines, filtering, etc. We skip them over, and leave them to you, to explore the possibilities offered by ERP5Form. This should be rather easy, because the names of the properties are transparent, and can be understood by trial and error.

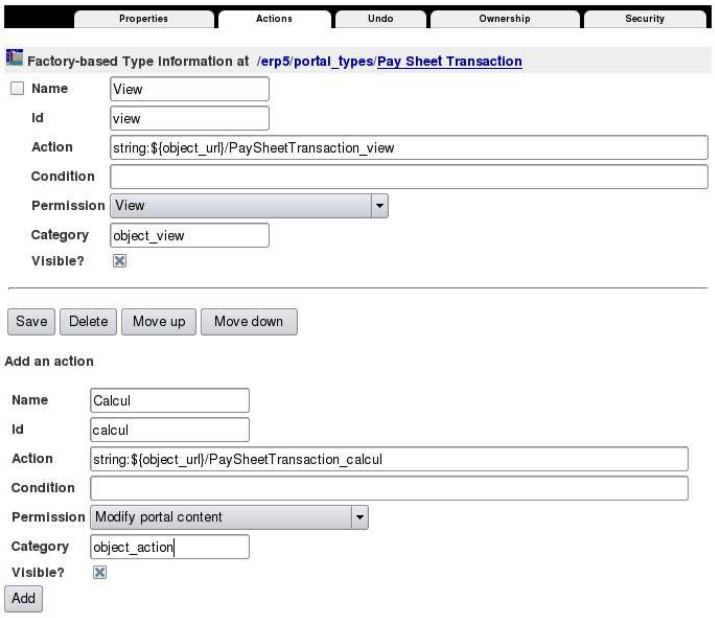
3.9. Python scripts for calculation

The following section aims at creating a script which produces all the data of a payroll from a gross salary.

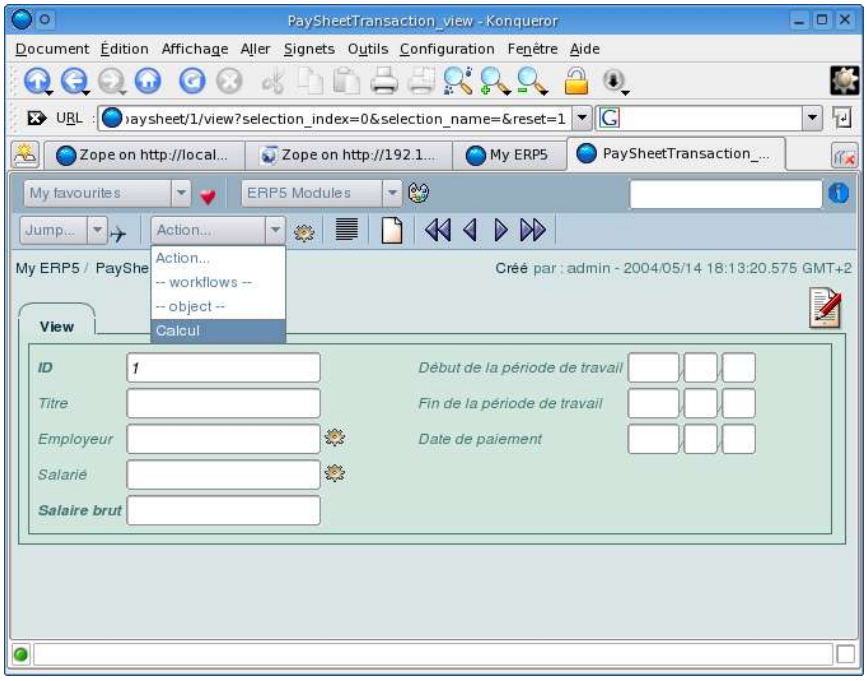
So we create a new Python script named `PaySheetTransaction_calcul` :



We also update the actions of the portal type `Pay Sheet Transaction`, so that it takes into account our script:



By giving the value `object_action` to the attribute `Category`, an entry is added into the menu of `Action` of the view of `Pay Sheet Transaction`:



As the script is empty, nothing will happen when one selects `Calcul` in the list of the actions.

Now we begin to code the script, so that it:

- collects the sufficient data (gross salary, employer, etc.),
- calculates the rates and the tax bases and
- distributes the amounts, according to the good categories and to the good social institutions.

The items 1 and 2 do not require any explanation particularly, apart from the fine knowledge of the method of calculation of the amounts. For this, we will simply generate some lines of a payroll as an example.

The third requires some explanations. We will detail the behavior of the structure `PaySheetLine/PaySheetCell` from the point of view of the developer.

The cells can be represented like tables or matrices, represented by two basic categories: `assiette` (tax base) and `categorie_taxe` (tax category)[6].

To illustrate our matter, let us take the old-age insurance (the rates are those of April 2003). The salary share is equal to the value of the ceiling salary multiplied by 6.55%. The employers' share is equal to the sum of 1.60% of the gross salary and 8.20% of the ceiling salary.

Here is the representation of these figures within the `PaySheetLine` for a salary of 3000 euros:

6 The base categories are written in French, because of the original document. This will be addressed in next revision.

```
portal_type                  = 'Pay Sheet Line'
title                        = 'Assurance Vieillesse'
variation_base_category_list = ['assiette', 'categorie_taxe']
categories                   = ( 'categorie_taxe/secu/part_employeur'
                               , 'categorie_taxe/secu/part_salariale'
                               , 'assiette/salaire_brut'
                               , 'assiette/salaire_plafonne'
                               , 'destination/organisation/urssaf'
                               )
```

variation_base_category_list		categorie_taxe	
	categories	categorie_taxe/secu/part_employeur	categorie_taxe/secu/part_salariale
Assiette	assiette/salaire_brut	portal_type = 'Pay Sheet Cell' domain_base_category_list = ('assiette', 'categorie_taxe') categories = ('categorie_taxe/secu/part_employeur', 'assiette/salaire_brut') quantity = -3000.0 price = 0.082	portal_type = 'Pay Sheet Cell' domain_base_category_list = ('assiette', 'categorie_taxe') categories = ('categorie_taxe/secu/part_salariale', 'assiette/salaire_brut') quantity = price =
	assiette/salaire_plafonne	portal_type = 'Pay Sheet Cell' domain_base_category_list = ('assiette', 'categorie_taxe') categories = ('categorie_taxe/secu/part_employeur', 'assiette/salaire_plafonne') quantity = -2432.0 price = 0.016	portal_type = 'Pay Sheet Cell' domain_base_category_list = ('assiette', 'categorie_taxe') categories = ('categorie_taxe/secu/part_salariale', 'assiette/salaire_plafonne') quantity = -2432.0 price = 0.0655

The objects are identified by a red frame, and their most significant attributes are noted inside.

The expense "Old-age insurance" is represented by a `PaySheetLine`, which contains a matrix of 4 cells of the type `PaySheetCell`. Since the categories and the base categories are defined in the level of `PaySheetLine`, the latter will be able to automatically generate the table of cells at its creation. Thanks to this, all we have to do is choose our cell according to the categories and assign its values to it. A cell has two attributes: `quantity` and `price`. `quantity` represents the tax base, and `price` does the rate. To calculate the amount, it is enough to call `getTotalPrice()`, which will carry out the multiplication `quantity` * `price`.

The old-age insurance is a sub-part of the expenses of social security, and, to represent this hierarchy, we created it under the category `secu` (an acronym of social security). The latter is subdivided into the salary share and the employer's share, to distribute the amounts finely.

Because the old-age insurance is calculated from the gross salary and the ceiling salary, we have created the corresponding subcategories in the base category `assiette`.

A good definition of the categories will allow us thereafter to easily generate detailed reports. Since we will generate, in a script, the expenses of sickness insurance, old-age insurance, and unemployment insurance, here are the categories which we have added in the tool `portal_categories`:

```
categorie_taxe
categorie_taxe/secu
categorie_taxe/secu/part_employeur
categorie_taxe/secu/part_salariale
categorie_taxe/chomage
categorie_taxe/chomage/part_employeur
categorie_taxe/chomage/part_salariale
assiette
assiette/salaire_brut
```

```
assiette/salaire_plafonne
assiette/tranche_a
assiette/tranche_b
assiette/tranche_c
```

Here is the Python script which will generate some lines of payroll:

```


# Ce script est donné à titre d'exemple
# Les constantes ont été codées en dur pour faciliter la lecture

global paysheet
paysheet                      = context.getObject()
paysheet_type                 = paysheet.getPortalType()
paysheet_cell_type            = 'Pay Sheet Cell'
paysheet_transactionline_type = 'Pay Sheet Transaction Line'


employer = paysheet.getSourceSection()
employee = paysheet.getDestinationSection()

if employer in ('', None):
  return context.REQUEST.RESPONSE.redirect(context.absolute_url() +
'?portal_status_message=The+employer+is+required')

if employee in ('', None):
  return context.REQUEST.RESPONSE.redirect(context.absolute_url() +
'?portal_status_message=The+employee+is+required')

employer_object = paysheet.getSourceSectionValue()

if employer_obj.getDefaultAddress().getZipCode() in ('', None):
  return context.REQUEST.RESPONSE.redirect(context.absolute_url() +
'?portal_status_message=The+employer+must+have+a+zip+code')

gross_salary = abs(paysheet.getGrossSalary())

employer_region = employer_object.getDefaultAddress().getZipCode()[:2]

# limited salary = salaire plafonné
if gross_salary < 2432:
  limited_salary = gross_salary
else:
  limited_salary = 2432

# "Char" slice type
if gross_salary <= 2432:
  char_slice = 'A'
elif gross_salary <= 9728:
  char_slice = 'B'
elif gross_salary <= 19456:
  char_slice = 'C'
else:
  char_slice = ''
```

```
##########################################################################
# This part of the script implement functions to register all pay sheet
# informations from an ERP5 point of view.
##########################################################################

def createPaySheetItem(title='', dest_org='', cells=[]):
  global paysheet
  # get all variation categories used in cells
  var_cat_list = []
  for cell in cells:
    var_cat_list.append(cell["x"])
    var_cat_list.append(cell["y"])
  # add a new Pay Sheet Line
  payline = paysheet.newContent( portal_type                  = 'Pay Sheet Line'
                               , title                        = title
                               , destination                  = dest_org
                               , variation_base_category_list = ('tax_category',
'salary_range')
                               , variation_category_list      = var_cat_list
                               )
  # fill each cell with values
  for cell in cells:
    paycell = payline.getCell(cell["x"], cell["y"], base_id = 'movement')
    paycell.edit(quantity=-cell["base"], price=cell["rate"]/100.0)


##########################################################################
# This part of script describe the behaviour of the calculation process
# from accountant point of view.
##########################################################################

# social organism
org_urssaf  = 'organisation/urssaf'
org_assedic = 'organisation/assedic'

# variation categories
cat_social_salary_share         = 'tax_category/social/salary_share'
cat_social_employer_share       = 'tax_category/social/employer_share'
cat_unemployment_salary_share   = 'tax_category/unemployment/salary_share'
cat_unemployment_employer_share = 'tax_category/unemployment/employer_share'
cat_gross_salary                = 'salary_range/france/salaire_brut'
cat_limited_salary              = 'salary_range/france/salaire_plafonne'
cat_slice_a                     = 'salary_range/france/tranche_a'
cat_slice_b                     = 'salary_range/france/tranche_b'
cat_slice_c                     = 'salary_range/france/tranche_c'


# sickness insurance = assurance maladie
if employer_region in ('57', '67', '68'):
  salary_share_rate = 1.70
else:
  salary_share_rate = 0.75
createPaySheetItem( title    = 'Assurance maladie'
                  , dest_org = org_urssaf
                  , cells    = [ { "x"    : cat_social_salary_share
```

```
                                  , "y"    : cat_gross_salary
                                  , "base" : gross_salary
                                  , "rate" : salary_share_rate
                                  },
                                  { "x"    : cat_social_employer_share
                                  , "y"    : cat_gross_salary
                                  , "base" : gross_salary
                                  , "rate" : 12.80
                                  }
                                ]
                    )

# old-age insurance = assurance vieillesse
createPaySheetItem( title    = 'Assurance vieillesse'
                  , dest_org = org_urssaf
                  , cells    = [ { "x"    : cat_social_salary_share
                                  , "y"    : cat_limited_salary
                                  , "base" : limited_salary
                                  , "rate" : 6.55
                                  },
                                  { "x"    : cat_social_employer_share
                                  , "y"    : cat_gross_salary
                                  , "base" : gross_salary
                                  , "rate" : 1.60
                                  },
                                  { "x"    : cat_social_employer_share
                                  , "y"    : cat_limited_salary
                                  , "base" : limited_salary
                                  , "rate" : 8.20
                                  }
                                 ]
                    )

# unemployment insurance = assurance chomage
if char_slice in ('A', 'B'):
  if char_slice == 'A':
    cat_slice = cat_slice_a
  else:
    cat_slice = cat_slice_b
  createPaySheetItem( title    = 'Assurance chomage'
                    , dest_org = org_assedic
                    , cells    = [ { "x"    : cat_unemployment_salary_share
                                     , "y"    : cat_slice
                                     , "base" : gross_salary
                                     , "rate" : 2.4
                                     },
                                     { "x"    : cat_unemployment_employer_share
                                     , "y"    : cat_slice
                                     , "base" : gross_salary
                                     , "rate" : 4.0
                                     }
                                   ]
                    )
```

If your favorite editor supports the WebDAV protocol, you can edit Zope objects

directly. By commenting out some lines in the file `/etc/zope.conf`, and restarting the Zope, WebDAV will be activated, and you will be able to browse the contents of the ZODB then edit Python scripts very easily in your favorite development environment.

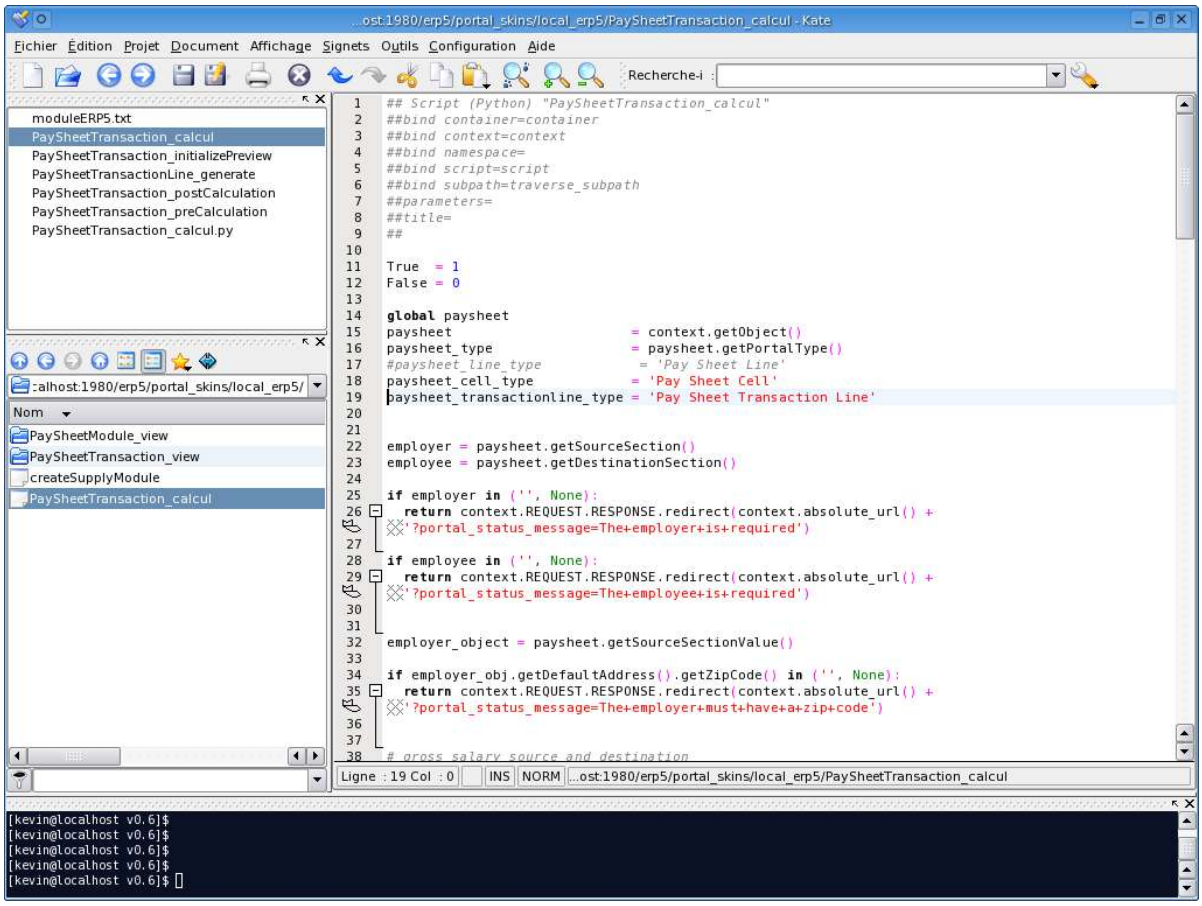


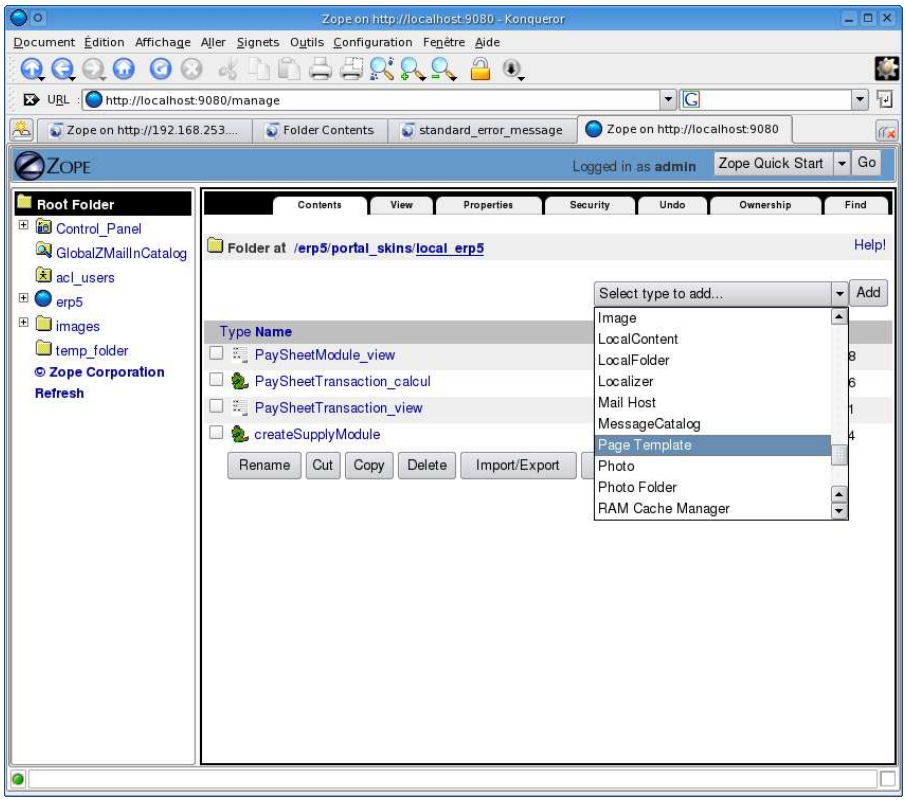
Illustration 7: Use of WebDAV with KATE

Before testing the script, do not forget to create the organizations Assedic and Urssaf with the same IDs used in the script (`organisation/urssaf` and `organisation/assedic`), so that it can function.

3.10. The reporting

You will see; in this section, how to use the data generated by our script, to create reports and the printable versions of payrolls.

The latter will be produced through a `PDF Template`, which we create in `local_erp5` with the name of `PaySheetTransaction_print`:



The `PDF Templates` are, in fact, just simple HTML (or XML) files, in which we will use special attributes in the tags. This language named TAL (Template Attribute Language) specific to Zope can describe a dynamic behavior of the pages. In addition, ERP5 integrates a script which automatically converts an OpenOffice document into TAL, in order to facilitate the rapid creation of PDF reports.

In the case of our payroll, we will create a Python script which will be given the responsibility to preformat the data of a payroll, and we will call it from the `PDF Template`.

The script, which we call `PaySheetTransaction_getDetails`, is responsible for the collection of the employers' share and the salary share for each tax, and gathering them in a dictionary:

```
# this dict contain all paysheet details
paysheet_details = {}
```

```
# initialize the employee and employer share total
total_employee_share          = 0.0
total_employer_share          = 0.0
total_taxable_employee_share  = 0.0

# get the gross salary
gross_salary = context.getGrossSalary()
if gross_salary == None:
  gross_salary = 0.0

paysheet_cat = {}
object_list = []
for object in context.objectValues():
  object_list += [object]
# Sort the list by id since lines are already ordered by id.
object_list.sort(lambda x, y: cmp(int(x.getId()), int(y.getId())))
for pay_sheet_line in object_list:
  variation_list =  pay_sheet_line.getVariationCategoryList()
  range_variation = []
  for variation in variation_list:
    if variation.find('salary_range')==0:
      if not variation in range_variation: # Extra checking because
                                           # get VariationCategoryList returns
                                           # the same 1 items 2 times
        range_variation += [variation]
  for range in range_variation:
    pay_sheet_dict = {}
    #pay_sheet_dict['range']=range[range.rfind('/')+1:]
    pay_sheet_dict['id'] = pay_sheet_line.getId()
    pay_sheet_dict['title'] = pay_sheet_line.getResourceTitle()
    for cell in pay_sheet_line.objectValues():
      predicate_list = cell.getMembershipCriterionCategoryList()
      if range in predicate_list:
        pay_sheet_dict['base_name'] = context.portal_categories.resolveCategory
(range).getTitleOrId()
        for predicate in predicate_list:
          if cell.getTotalPrice() != 0:
            if predicate.find('employee_share')>=0:
              pay_sheet_dict['base']= - cell.getQuantity()
              pay_sheet_dict['employee_share'] = cell.getTotalPrice()
              pay_sheet_dict['employee_share_rate'] = cell.getPrice() * 100
              total_employee_share += float(-pay_sheet_dict['employee_share'])
              # here we decide if a resource is taxable or not
              if str(pay_sheet_line.getResource())[-14:] == 'non_deductible' or
str(pay_sheet_line.getResource())[-4:] == 'crds' or str
(pay_sheet_line.getResource())[-7:] == 'taxable':
                pay_sheet_dict['taxable']='yes'
              elif str(pay_sheet_line.getResource())[-10:] == 'deductible':
                pay_sheet_dict['taxable']='no'
              else:
                pay_sheet_dict['taxable']='no'
              if pay_sheet_dict['taxable'] == 'yes':
                total_taxable_employee_share += float(-pay_sheet_dict
['employee_share'])
            elif predicate.find('employer_share')>=0:
              pay_sheet_dict['base'] = - cell.getQuantity()
```

```
              pay_sheet_dict['employer_share'] = cell.getTotalPrice()
              pay_sheet_dict['employer_share_rate'] = cell.getPrice() * 100
              total_employer_share += float(-pay_sheet_dict['employer_share'])
    for key in
('employee_share','employee_share_rate','employer_share','employer_share_rate'):
      if not (pay_sheet_dict.has_key(key)):
        pay_sheet_dict[key]='' # so that we can display nothing

    # find the category of the current pay sheet line
    cat_id = None
    cat_path = None
    for var in variation_list:
      sub_cat = var.split('/')
      if sub_cat[0] == 'tax_category':
        cat_id = sub_cat[1]
        cat_path = sub_cat[0] + '/' + sub_cat[1]
        break
    if cat_id == None:
      cat_id = 'no_cat'
    # add the current pay sheet line to its category
    if not paysheet_cat.has_key(cat_id):
      paysheet_cat[cat_id] = {}
      paysheet_cat[cat_id]['lines'] = []
    if cat_path != None:
      paysheet_cat[cat_id]['title'] = context.portal_categories.resolveCategory
(cat_path).getTitleOrId()
    paysheet_cat[cat_id]['lines'].append(pay_sheet_dict)

# get all paysheet transaction to calculate the sum of different value in a year
accounting_folder = context.aq_parent
paysheet_transactions = accounting_folder.contentValues(filter=
{'portal_type':'Pay Sheet Transactionss'})

# initialize every yearly variable
yearly_net_salary           = 0.0
yearly_gross_salary         = 0.0
yearly_employee_share       = 0.0
yearly_employer_share       = 0.0
yearly_taxable_net_salary   = 0.0

# get the current paysheet start date and employee
start_date  = context.getStartDate()
from DateTime import DateTime
start_date  = DateTime("%i/01/01" % start_date.year())
stop_date  = context.getStopDate()
employee    = context.restrictedTraverse
(context.getDestinationSectionRelativeUrl())

#start_date = start_date.strftime('%Y-%m-%d')
#stop_date = start_date.strftime('%Y-%m-%d')
#yearly_employee_share = -float(context.PaySheetTransaction_zGetDetailedTotal
(start_date=start_date,stop_date=stop_date,tax_category='employee_share')[0].
total)
#yearly_employer_share = -float(context.PaySheetTransaction_zGetDetailedTotal
(start_date=start_date,stop_date=stop_date,tax_category='employer_share')[0].
total)
```

```
yearly_employee_share  = 0
yearly_employer_share  = 0
try:
  yearly_employer_share = -float(context.PaySheetTransaction_zGetDetailedTotal
(start_date=start_date,stop_date=stop_date,tax_category='employer_share')[0].
total)
except:
  pass

# browse through paysheet transaction
for paysheet_obj in paysheet_transactions:
  # ignore the current paysheet to avoid infinite loop
  if paysheet_obj.getId() != context.getId():
    # the paysheet must have the same employee
    if (employee==None) or (employee!=None and context.restrictedTraverse
(paysheet_obj.getDestinationSectionRelativeUrl())==employee):
      # check the date
      if (start_date==None) or (start_date!=None and paysheet_obj.getStartDate
()!=None and start_date.year()==paysheet_obj.getStartDate().year() and
paysheet_obj.getStartDate()<= start_date):
        # get all detailed values of the paysheet
        ps_details = paysheet_obj.PaySheetTransaction_getDetails()
        # sum of yearly values
        yearly_net_salary           += float(ps_details['net_salary'])
        yearly_gross_salary         += float(ps_details['gross_salary'])
        yearly_employee_share       += float(ps_details['total_employee_share'])
        yearly_employer_share       += float(ps_details['total_employer_share'])
        yearly_taxable_net_salary   += float(ps_details['taxable_net_salary'])

# save the total share values in the exported dict
paysheet_details['net_salary']                    = gross_salary -
total_employee_share
paysheet_details['gross_salary']                  = gross_salary
paysheet_details['paysheet_categories']           = paysheet_cat
paysheet_details['total_employee_share']          = total_employee_share
paysheet_details['taxable_net_salary']            = paysheet_details
['net_salary'] + total_taxable_employee_share
paysheet_details['total_employer_share']          = total_employer_share
paysheet_details['total_taxable_employee_share']  = total_taxable_employee_share

# don't forget to add the current values to the yearly sum
paysheet_details['yearly_net_salary']         = yearly_net_salary +
paysheet_details['net_salary']
paysheet_details['yearly_gross_salary']       = yearly_gross_salary +
paysheet_details['gross_salary']
paysheet_details['yearly_employee_share']     = yearly_employee_share +
paysheet_details['total_employee_share']
paysheet_details['yearly_employer_share']     = yearly_employer_share +
paysheet_details['total_employer_share']
paysheet_details['yearly_taxable_net_salary'] = yearly_taxable_net_salary +
paysheet_details['taxable_net_salary']

return paysheet_details
```

A first version of our `PDF Template` is like this:

```
<?xml version="1.0" encoding="iso-8859-1" ?>

<document filename="report01.pdf"
   xmlns:tal="http://xml.zope.org/namespaces/tal"
   tal:define="employee python: here.getDestinationSectionValue();
                   employer python: here.getSourceSectionValue();
                   paysheet_details python: here.PaySheetTransaction_getDetails
();
                   paysheet_categories python: paysheet_details
['paysheet_categories'];
                   urssaf python: employer.getDestinationSectionValue();
                   start_date python: here.getStartDate();

                   boldstyle python:'(\'FONT\', \'Helvetica-Bold\', 7)'">


    <title>VPN</title>
    <author>Nexedi</author>
    <subject>VPN List</subject>
    <content>
       <table splitbyrow="1" rowheight='0.4cm' repeatrows="1" repeatcols="0"
style="decompte">
         <tr>
           <td colwidth="4.318cm">Nature</td>
           <td colwidth="3.81cm">Assiette</td>
           <td colwidth="1.524cm">Montant</td>
           <td colwidth="2.77cm">Taux part patronale</td>
           <td colwidth="2.288cm">Part patronale</td>
           <td colwidth="2.397cm">Taux part salariale</td>
           <td colwidth="1.959cm">Part salariale</td>
         </tr>

         <tr tal:attributes="stylecmd boldstyle"> <!--stylecmd="('FONT',
'Helvetica-Bold', 7)"> -->
            <td>Salaire brut</td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td tal:content="python: '%.2f' % paysheet_details['gross_salary']"
tal:condition="python: paysheet_details['gross_salary'] not in (None,
'')">???</td>
         </tr>

         <tal:block tal:condition="python: paysheet_categories.has_key
('no_cat')">
           <tal:block define="no_cat_lines python: paysheet_categories['no_cat']
['lines']">
             <tr tal:repeat="line no_cat_lines">
               <td><tal:block content="python: line['title']"/> </td>
               <td content="python: line['base_name']"> </td>
               <td content="python: '%.2f' % line['base']"
tal:condition="python: line['base'] not in (None, '')"> </td>
```

```
                <td content="python: '%.3f %%' % line['employer_share_rate']"
tal:condition="python: line['employer_share_rate'] not in (None, '')"> </td>
                <td content="python: '%.2f' % line['employer_share']"
tal:condition="python: line['employer_share'] not in (None, '')"> </td>
                <td content="python: '%.3f %%' % line['employee_share_rate']"
tal:condition="python: line['employee_share_rate'] not in (None, '')"> </td>
                <td content="python: '%.2f' % line['employee_share']"
tal:condition="python: line['employee_share'] not in (None, '')"> </td>
              </tr>
            </tal:block>
          </tal:block>

          <tal:block repeat="category paysheet_categories">
            <tr tal:attributes="stylecmd boldstyle">
              <td tal:content="python: paysheet_categories[category]['title']">
</td>
              <td> </td>
              <td> </td>
              <td> </td>
              <td> </td>
              <td> </td>
              <td> </td>
            </tr>
            <tal:block define="paysheet_lines python: paysheet_categories
[category]['lines']">
              <tr tal:repeat="line paysheet_lines" stylecmd="('LEFTPADDING',8)">
                <td><tal:block tal:content="python: '    ' + line['title']"/>
</td>
                <td><tal:block tal:content="python: line['base_name']"/> </td>
                <td> <tal:block content="python: '%.2f' % line['base']"
tal:condition="python: line['base'] not in (None, '')"></tal:block></td>
                <td> <tal:block content="python: '%.3f %%' % line
['employer_share_rate']" tal:condition="python: line['employer_share_rate'] not
in (None, '')"></tal:block></td>
                <td> <tal:block content="python: '%.2f' % line['employer_share']"
tal:condition="python: line['employer_share'] not in (None,
'')"></tal:block></td>
                <td> <tal:block content="python: '%.3f %%' % line
['employee_share_rate']" tal:condition="python: line['employee_share_rate'] not
in (None, '')"></tal:block></td>
                <td> <tal:block content="python: '%.2f' % line['employee_share']"
tal:condition="python: line['employee_share'] not in (None,
'')"></tal:block></td>
              </tr>
            </tal:block>
          </tal:block>

          <tr tal:attributes="stylecmd boldstyle">
            <td>Total des cotisations</td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> <tal:block replace="python: '-%.2f' % paysheet_details
['total_employer_share']" tal:condition="python: paysheet_details
['total_employer_share'] not in ('', None)"> </tal:block></td>
```

```
            <td> </td>
            <td> <tal:block replace="python: '-%.2f' % paysheet_details
['total_employee_share']" tal:condition="python: paysheet_details
['total_employee_share'] not in ('', None)"> </tal:block></td>
          </tr>
          <tr tal:attributes="stylecmd boldstyle">
            <td>Salaire Net</td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> <tal:block replace="python: '%.2f' % paysheet_details
['net_salary']" tal:condition="python: paysheet_details['net_salary'] not in
('', None)"> </tal:block></td>
          </tr>

          <tal:block repeat="category paysheet_categories">
            <tal:block define="paysheet_lines python: paysheet_categories
[category]['lines']">
              <tal:block repeat="line paysheet_lines">
                <tr tal:condition="python: line.has_key('taxable') and line
['taxable']=='yes'">
                  <td> <tal:block content="python: line['title']"/></td>
                  <td> <tal:block content="python: line['base_name']"/></td>
                  <td> <tal:block replace="python: '%.2f' % line['base']"
tal:condition="python: line['base'] not in ('', None)"></tal:block></td>
                  <td> </td>
                  <td> </td>
                  <td> <tal:block replace="python: '%.3f %%' % line
['employee_share_rate']" tal:condition="python: line['employee_share_rate'] not
in ('', None)"></tal:block></td>
                  <td> <tal:block replace="python: '+%.2f' % abs(float(line
['employee_share']))" tal:condition="python: line['employee_share'] not in ('',
None)"></tal:block></td>
                </tr>
              </tal:block>
            </tal:block>
          </tal:block>
          <tr tal:attributes="stylecmd boldstyle">
            <td>Salaire Net Imposable</td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> </td>
            <td> <tal:block replace="python: '%.2f' % paysheet_details
['taxable_net_salary']" tal:condition="python: paysheet_details
['taxable_net_salary'] not in ('', None)"> </tal:block></td>
          </tr>
        </table>

        <table rowheight="0.6cm"><tr><td> </td></tr></table>

        <table style="cumul_conges">
```

```
         <tr>
           <td colwidth="11cm"> <tal:block tal:replace="python: 'Cumuls annuels
(%s)' % start_date.year()" tal:condition="python: start_date not in ('',
None)"/></td>
           <td colwidth="3cm"> </td>
           <td colwidth="5cm"> <!-- <tal:block tal:replace="python: 'Durée des
congés payés : ???'"/> --> </td>
         </tr>
       </table>


       <table style="cumul_conges_corps">
         <tr>
           <td colwidth="2cm">Salaire brut</td>
           <td colwidth="3cm">Cotisations salariales</td>
           <td colwidth="2cm">Salaire net</td>
           <td colwidth="2cm">Net imposable</td>
           <td colwidth="2cm">Part patronale</td>
           <td colwidth="3cm"> </td>
           <td colwidth="5cm"> <tal:block tal:replace="python: 'Durée des délais
de préavis : ' + context.PaySheetTransaction_getPreavis()"/></td>
         </tr>
         <tr>
           <td> <tal:block replace="python: '%.2f' % paysheet_details
['yearly_gross_salary']" tal:condition="python: paysheet_details
['yearly_gross_salary'] not in ('', None)"></tal:block></td>
           <td> <tal:block replace="python: '%.2f' % paysheet_details
['yearly_employee_share']" tal:condition="python: paysheet_details
['yearly_employee_share'] not in ('', None)"></tal:block></td>
           <td> <tal:block replace="python: '%.2f' % paysheet_details
['yearly_net_salary']" tal:condition="python: paysheet_details
['yearly_net_salary'] not in ('', None)"></tal:block></td>
           <td> <tal:block replace="python: '%.2f' % paysheet_details
['yearly_taxable_net_salary']" tal:condition="python: paysheet_details
['yearly_taxable_net_salary'] not in ('', None)"></tal:block></td>
           <td> <tal:block replace="python: '%.2f' % paysheet_details
['yearly_employer_share']" tal:condition="python: paysheet_details
['yearly_employer_share'] not in ('', None)"></tal:block></td>
           <td> </td>
           <td> </td>
         </tr>
       </table>

    </content>

</document>
```

Let us add then an action into the portal type `Pay Sheet Transaction` with the value `object_print` for the category:

Name	Print
Id	print
Action	string:${object_url}/PaySheetTransaction_print
Condition	
Permission	View
Category	object_print
Visible?	☒

Thanks to this small manipulation, Simply clicking on the printer icon in the view of a payroll generates the printable version:

The `PDF Template` is called in the current context, and the printable payroll is displayed on the screen in the form of PDF document:

nexedi

FEUILLE DE SALAIRE

Nexedi SARL
943, avenue de la République
59700 Marcq-en-Baroeul
Etablissement : nexedi
Siret : 440047504 00012
NAF (ex APE) : 722Z

N°URSSAF : 592 190071289 1013
URSSAF de Roubaix-Tourcoing
28, avenue de la Marne
59208 Tourcoing

Convention collective de branche : Syntec

Paulo Lavabo
4, rue de la Plomberie
99555 St-Amand-les-Eaux
N°Secu.Soc. : 3 70 07 42 555 777 05

Emploi : Directeur Technique
Qualification : ingenieur
Echelon : 1
Coefficient : 1

Periode : du 2003/02/01 au 2003/02/28
Date de paiement : 2003/03/04

Nature	Assiette	Montant	Taux part patronale	Part patronale	Taux part salariale	Part salariale
Salaire brut						**2000.00**
Assurance décès						
Assurance décès	Tranche A	2000.00	1.500 %	-30.00		
Assurance chômage						
Assurance chômage	Tranche A	2000.00	4.000 %	-80.00	2.400 %	-48.00
CRDS						
CRDS	Salaire brut CRDS	1900.00			0.500 %	-9.50
CSG						
CSG non déductible	Salaire brut CSG	1900.00			2.400 %	-45.60
CSG déductible	Salaire brut CSG	1900.00			5.100 %	-96.90
Taxes et participations						
Formation professionnelle	Salaire brut	2000.00	0.150 %	-3.00		
Apprentissage	Salaire brut	2000.00	0.500 %	-10.00		
Mutuelle Convention Syntec						
Mutuelle Syntec	Salaire plafonné SYNTEC	2432.00	0.915 %	-22.25	0.915 %	-22.25
Sécurité sociale						
Assurance maladie	Salaire brut	2000.00	12.800 %	-256.00	0.750 %	-15.00
Aide au logement	Salaire plafonné	2000.00	0.100 %	-2.00		
Accidents du travail	Salaire brut	2000.00	1.000 %	-20.00		
Assurance veuvage	Salaire brut	2000.00			0.100 %	-2.00
Assurance vieillesse	Salaire brut	2000.00	1.600 %	-32.00		
Assurance vieillesse	Salaire plafonné	2000.00	8.200 %	-164.00	6.550 %	-131.00
Allocations familiales	Salaire brut	2000.00	5.400 %	-108.00		
AGS (FNGS)						
AGS	Tranche A	2000.00	0.350 %	-7.00		
Retraite complémentaire						
AGFF	Tranche A	2000.00	1.200 %	-24.00	0.800 %	-16.00
ARRCO	Tranche A	2000.00	4.500 %	-90.00	3.000 %	-60.00
CET	Tranche A	2000.00	0.220 %	-4.40	0.130 %	-2.60
Total des cotisations				**-852.65**		**-448.85**
Salaire Net						**1551.15**
CRDS	Salaire brut CRDS	1900.00			0.500 %	+9.50
CSG non déductible	Salaire brut CSG	1900.00			2.400 %	+45.60
Salaire Net Imposable						**1606.25**

Cumuls annuels (2003)				
Salaire brut	Cotisations salariales	Salaire net	Net imposable	Part patronale
4000.00	897.71	3102.29	3212.49	1705.31

Durée des congés payés : 2 jours
Durée des délais de préavis : 3 mois

Dans votre intérêt et pour vous aider à faire valoir vos droits, conservez ce bulletin de paie sans limitation de durée.

Illustration 8: A payroll generated by ERP5

The technology utilized for the generated PDF is reportlab. It is a library written in Python, used by famous financial institutes, and able to generate several hundreds of pages of PDF per minute on a low-end server.

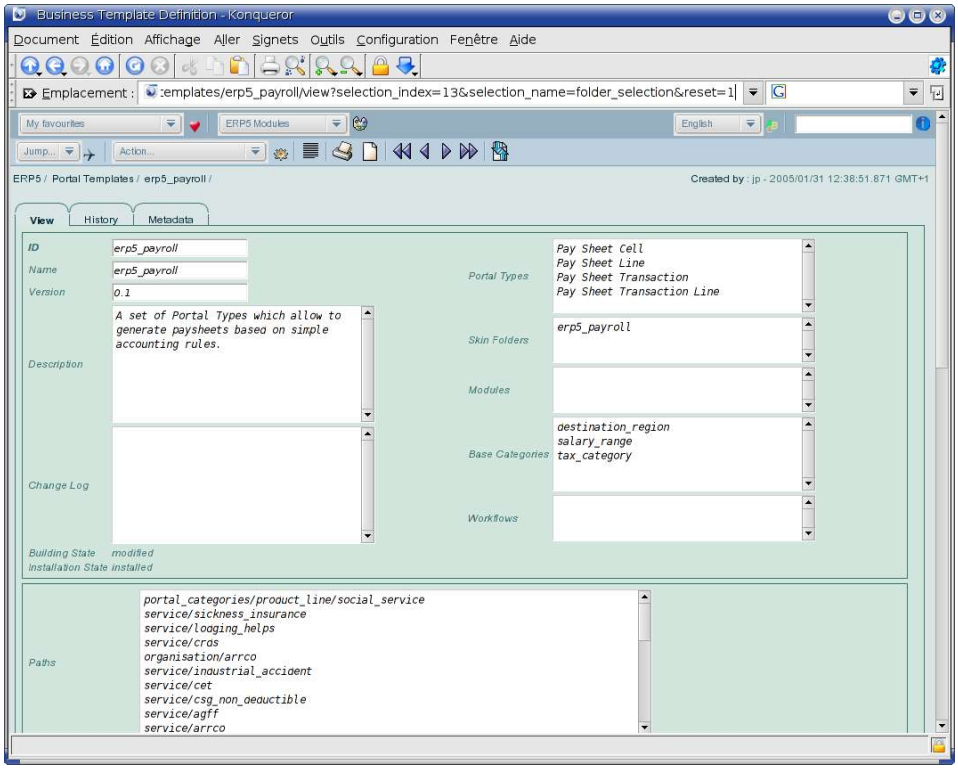
3.11. The creation of the Business Template

Now that the module fully functions, we will assemble it into a `Business Template`, in order to be able to distribute it and install it easily.

The procedure of creation of a `Business Template` is rather fast, if one knows exactly all the components, which are necessary to the correct operation of the module one wants to package.

It is necessary to initially go back to the tool `portal_templates`, and to add a new `Business Template` by using the menu « `Action` ». We name this `Business Template` `erp5_payroll`, and edit it to list up the objects, which must be included, in the various fields:

```
http://localhost:9080/erp5/portal_templates/erp5_payroll/view
```



The modified fields are:

- ID : The ID of the `Business Template`
- Name : The name of the `Business Template`, which must be unique for each group of functionalities
- Portal Types : The list of the document types defined by the `Business Template`
- Skin Folders : The folder in `portal_skins` where the elements of presentation are stored
- Base Categories : The categories and the relations necessary for the operation of the module

- Paths : The documents necessary for the operation of the module (here, the organizations URSSAF, etc. necessary for the calculation of payrolls)

Now that the `Businnes Template` knows all the components of our module, we can build the `Business Template` finely via the action « `Build Business Template` » of the menu « `Action` », then export it via the action « `Export Busines Template` » of the same menu.

We can now test our new `Business Template` on a new instance of Zope, to make sure that our module functions, before distributing it.

3.12. A point on cataloguing

Cataloguing is, in fact, a means of storing certain information of an object (such as the ID, the title, the description, etc.) in a relational database. The catalogue makes it possible to select objects instantly, without searching all the database of Zope. It can execute fast requests on large databases. Nexedi, for example, has deployed a system containing more than 2,000,000 documents and more than 10,000,000 catalogue records.

The technology of cataloguing, developed by Nexedi for ERP5, is named « ZSQLCatalog ». It is a zope product which can carry out SQL queries with no limitation of complexity on a catalogue, of which the structure itself is completely adaptable. The objective is to construct a catalogue, according to a model of indexing likely to offer very short response time. The approach is similar to that used by the OLAP engines or the datawarehousing for the statistics treatment of consolidated data.

4. Conclusion

In this article, we learned how to install an ERP5 platform and its modules, thanks to the system of Business Template of ERP5.

We then created a simplistic version of the payroll module, which already exists in ERP5. There exist many differences between these two versions. In particular, the payroll module of ERP5 is based on the planning engine of ERP5, in order to allow forecasts of floating assets. However, the structure of data and the basic principles remain the same.

We can, from now on, exploit ERP5 in a flash to build a functional ERP, to manage a great number of aspects of a company or an organization. The current library of Business Templates, which is destined for growing and enriching, offers both building basic components and a source of inspiration.

If, however, the existing Business Templates do not meet your need, you are now ready to create your own ones. In this case, you are encouraged to share your work with us on the web site `www.erp5.org` and on the mailing lists `erp5-dev` and `erp5-users`.

5. TODO

This is specific to the English translation:

- The examples are in French.
- The screenshots are in French.
- Some sentences are too literally translated.
- The page arragement is not always appropriate.

This is for the original:

Améliorations à envisager sur ce document.

Question à se poser:

* faut-il supprimer certaines parties trop détaillée ? (variantage, category, construction des classes)

* indiquer les conventsion de nommage utilisées par ERP5

* créer avec le lecteur un nouveau folder dans les portal_skins et lui faire changer les settings pour lui expliquer comment erp5 va chercher les formulaire ?

* dire à l'utilisateur qu'il est prudent de donner un nom à chaque formulaire

* ne pas evoquer les paysheet transaction line ? (compression de l'article)

TODO:

* couper les parties des images qui ne sont pas utiles (surtout celle avec des screnshot de dropdown list/combo)

* mettre en italique les babarismes (getter / setter / packages / etc) ou trouver une version française

* intégrer une bibliographie/webographie à la fin de l'article

* intégrer "Lors du développement du véritable Business Template, que vous trouverez dans la section `download` du site erp5.org, nous n'avons pas eu besoin de créer de module. En réalité le produit final des feuilles de paie est rassemblé dans le Business Template `Human Resources` (HR) qui contient aussi les fonctionnalités de gestion de carrières et s'intègre dans différents modules pré-existants (les feuilles de paie dans le module `Accounting` et la gestion de carrières dans le module `Person`). "

* simplifier les scripts et templates de rapports

* mettre à jour les recopies d'écran qui mentionnent base_edit, base_update_relation, etc. en utilisant le nouveau nommage (Base_edit, Base_updateRelation, etc.)