ORM methods

Keeping the context in ORM methods

In OpenObject, the context holds very important data such as the language in which a document must be written, whether function field needs updating or not, etc.

When calling an ORM method, you will probably already have a context - for example the framework will provide you with one as a parameter of almost every method. If you do have a context, it is very important that you always pass it through to every single method you call.

This rule also applies to writing ORM methods. You should expect to receive a context as parameter, and always pass it through to every other method you call..

ORM methods

class `osv.osv.osv`**(*pool*, *cr*)**
Bases: `osv.osv.osv_base`, `osv.orm.orm`

`browse`**(*cr*, *uid*, *select*, *context=None*, *list_class=None*, *fields_process=None*)**
Fetch records as objects allowing to use dot notation to browse fields and relations

Parameters:
- cr -- database cursor
- user -- current user id
- select -- id or list of ids
- context -- context arguments, like lang, time zone

Return type: object or list of objects requested

`check_access_rule`**(*cr*, *uid*, *ids*, *operation*, *context=None*)**
Verifies that the operation given by `operation` is allowed for the user according to ir.rules.

Parameters: operation -- one of `write`, `unlink`

Raises except_orm:
- if current ir.rules do not permit this operation.

Returns: None if the operation is allowed

`copy`**(*cr*, *uid*, *id*, *default=None*, *context=None*)**
Duplicate record with given id updating it with default values

Parameters:
- cr -- database cursor
- uid -- current user id
- id -- id of the record to copy
- default (*dictionary*) -- dictionary of field values to override in the original values of the copied record, e.g: `{'field_name':` `overriden_value,` `...}`
- context (*dictionary*) -- context arguments, like lang, time zone

Returns: True

`copy_data`(*cr*, *uid*, *id*, *default=None*, *context=None*)
Copy given record's data with all its fields values

Parameters:
- cr -- database cursor
- user -- current user id
- id -- id of the record to copy
- default (*dictionary*) -- field values to override in the original values of the copied record
- context (*dictionary*) -- context arguments, like lang, time zone

Returns: dictionary containing all the field values

`create`(*cr*, *user*, *vals*, *context=None*)
Create new record with specified value

Parameters:
- cr -- database cursor
- user (*integer*) -- current user id
- vals (*dictionary*) -- field values for new record, e.g {'field_name': field_value, ...}
- context (*dictionary*) -- optional context arguments, e.g. {'lang': 'en_us', 'tz': 'UTC', ...}

Returns: id of new record created

Raises:
- AccessError --
  - if user has no create rights on the requested object
  - if user tries to bypass access rules for create on the requested object
- ValidateError -- if user tries to enter invalid value for a field that is not in selection
- UserError -- if a loop would be created in a hierarchy of objects a result of the operation (such as setting an object as its own parent)

Note: The type of field values to pass in `vals` for relationship fields is specific. Please see the description of the `write()` method for details about the possible values and how to specify them.

`default_get`(*cr*, *uid*, *fields_list*, *context=None*)
Returns default values for the fields in fields_list.

Parameters:
- fields_list (*list*) -- list of fields to get the default values for (example ['field1', 'field2',])
- context -- optional context dictionary - it may contains keys for specifying certain options like `context_lang` (language) or `context_tz` (timezone) to alter the results of the call. It may contain keys in the form `default_XXX` (where XXX is a field name), to set or override a default value for a field. A special `bin_size` boolean flag may also be passed in the context to request the value of all fields.binary columns to be returned as the size of the binary instead of its contents. This can also be selectively overriden by passing a field-specific flag in the form `bin_size_XXX: True/False` where `XXX` is the name of the field. Note: The `bin_size_XXX` form is new in OpenERP v6.0.

Returns: dictionary of the default values (set on the object model class, through user preferences, or in the context)

`export_data`(*cr*, *uid*, *ids*, *fields_to_export*, *context=None*)
Export fields for selected objects

Parameters:
- cr -- database cursor
- uid -- current user id
- ids -- list of ids
- fields_to_export -- list of fields
- context -- context arguments, like lang, time zone

Return type: dictionary with a *datas* matrix

This method is used when exporting data via client menu

`fields_get`(*cr*, *user*, *fields=None*, *context=None*)

Get the description of list of fields

Parameters:
- cr -- database cursor
- user -- current user id
- fields -- list of fields
- context -- context arguments, like lang, time zone

Returns: dictionary of field dictionaries, each one describing a field of the business object

Raises AccessError:
- if user has no create/write rights on the requested object

`fields_view_get`(*cr*, *user*, *view_id=None*, *view_type='form'*, *context=None*, *toolbar=False*, *submenu=False*)

Get the detailed composition of the requested view like fields, model, view architecture

Parameters:
- cr -- database cursor
- user -- current user id
- view_id -- id of the view or None
- view_type -- type of the view to return if view_id is None ('form', tree', ...)
- context -- context arguments, like lang, time zone
- toolbar -- true to include contextual actions
- submenu -- example (portal_project module)

Returns: dictionary describing the composition of the requested view (including inherited views and extensions)

Raises:
- AttributeError --
  - if the inherited view has unknown position to work with other than 'before', 'after', 'inside', 'replace'
  - if some tag other than 'position' is found in parent view
- Invalid ArchitectureError -- if there is view type other than form, tree, calendar, search etc defined on the structure

`get_xml_id`(*cr*, *uid*, *ids*, **args*, ***kwargs*)

Find out the XML ID of any database record, if there is one. This method works as a possible implementation for a function field, to be able to add it to any model object easily, referencing it as `osv.osv.get_xml_id`.

When multiple XML IDs exist for a record, only one of them is returned (randomly).

Synopsis: `get_xml_id(cr, uid, ids) -> { 'id': 'module.xml_id' }`

Returns: map of ids to their fully qualified XML ID, defaulting to an empty string when there's none (to be usable as a function field).

`import_data`(*cr*, *uid*, *fields*, *datas*, *mode='init'*, *current_module=''*, *noupdate=False*, *context=None*, *filename=None*)

Import given data in given module

Parameters:
- cr -- database cursor
- uid -- current user id
- fields -- list of fields
- data -- data to import
- mode -- 'init' or 'update' for record creation
- current_module -- module name
- noupdate -- flag for record creation
- context -- context arguments, like lang, time zone,
- filename -- optional file to store partial import state for recovery

Return type: tuple

This method is used when importing data via client menu.

Example of fields to import for a sale.order:

```
.id,                        (=database_id)
partner_id,                 (=name_search)
order_line/.id,             (=database_id)
order_line/name,
order_line/product_id/id,   (=xml id)
order_line/price_unit,
order_line/product_uom_qty,
order_line/product_uom/id   (=xml_id)
```

`name_get`(*cr*, *user*, *ids*, *context=None*)

Parameters:
- cr -- database cursor
- user (*integer*) -- current user id
- ids -- list of ids
- context (*dictionary*) -- context arguments, like lang, time zone

Returns: tuples with the text representation of requested objects for to-many relationships

`name_search`(*cr*, *user*, *name=''*, *args=None*, *operator='ilike'*, *context=None*, *limit=100*)

Search for records and their display names according to a search domain.

Parameters:
- cr -- database cursor
- user -- current user id
- name -- object name to search
- args -- list of tuples specifying search criteria [('field_name', 'operator', 'value'), ...]
- operator -- operator for search criterion
- context (*dictionary*) -- context arguments, like lang, time zone
- limit -- optional max number of records to return

Returns: list of object names matching the search criteria, used to provide completion for to-many relationships

This method is equivalent of `search()` on name + `name_get()` on the result. See `search()` for an explanation of the possible values for the search domain specified in args.

`perm_read`(*cr*, *user*, *ids*, *context=None*, *details=True*)

Returns some metadata about the given records.

Parameters:	details -- if True, *_uid fields are replaced with the name of the user
Returns:	list of ownership dictionaries for each requested record
Return type:	list of dictionaries with the following keys: • id: object id • create_uid: user who created the record • create_date: date when the record was created • write_uid: last user who changed the record • write_date: date of the last change to the record • xmlid: XML ID to use to refer to this record (if there is one), in format `module.name`

read_group(*cr*, *uid*, *domain*, *fields*, *groupby*, *offset=0*, *limit=None*, *context=None*, *orderby=False*)
Get the list of records in list view grouped by the given `groupby` fields

Parameters:	• cr -- database cursor • uid -- current user id • domain -- list specifying search criteria [['field_name', 'operator', 'value'], ...] • fields -- list of fields present in the list view specified on the object • groupby -- list of fields on which to groupby the records • offset -- optional number of records to skip • limit -- optional max number of records to return • context -- context arguments, like lang, time zone • order -- optional `order by` specification, for overriding the natural sort ordering of the groups, see also `search()` (supported only for many2one fields currently)
Returns:	list of dictionaries(one dictionary for each record) containing: • the values of fields grouped by the fields in `groupby` argument • __domain: list of tuples specifying the search criteria • __context: dictionary with argument like `groupby`
Return type:	[{'field_name_1': value, ...]
Raises AccessError:	• if user has no read rights on the requested object • if user tries to bypass access rules for read on the requested object

search(*cr*, *user*, *args*, *offset=0*, *limit=None*, *order=None*, *context=None*, *count=False*)
Search for records based on a search domain.

Parameters:	• cr -- database cursor • user -- current user id • args -- list of tuples specifying the search domain [('field_name', 'operator', value), ...]. Pass an empty list to match all records. • offset -- optional number of results to skip in the returned values (default: 0) • limit -- optional max number of records to return (default: None) • order -- optional columns to sort by (default: self._order=id) • context (*dictionary*) -- optional context arguments, like lang, time zone • count -- optional (default: False), if True, returns only the number of records matching the criteria, not their ids
Returns:	id or list of ids of records matching the criteria
Return type:	integer or list of integers
Raises AccessError:	• if user tries to bypass access rules for read on the requested object.

Expressing a search domain (args)

Each tuple in the search domain needs to have 3 elements, in the form: ('field_name', 'operator', value), where:

- field_name must be a valid name of field of the object model, possibly following many-to-one relationships using dot-notation, e.g 'street' or 'partner_id.country' are valid values.
- operator must be a string with a valid comparison operator from this list: `=,` `!=,` `>,` `>=,` `<,` `<=,` `like,` `ilike,` `in,` `not` `in,` `child_of,` `parent_left,` `parent_right` The semantics of most of these operators are obvious. The `child_of` operator will look for records who are children or grand-children of a given record, according to the semantics of this model (i.e following the relationship field named by `self._parent_name`, by default `parent_id`.
- value must be a valid value to compare with the values of field_name, depending on its type.

Domain criteria can be combined using 3 logical operators than can be added between tuples: '&' (logical AND, default), '|' (logical OR), '!' (logical NOT). These are prefix operators and the arity of the '&' and '|' operator is 2, while the arity of the '!' is just 1. Be very careful about this when you combine them the first time.

Here is an example of searching for Partners named *ABC* from Belgium and Germany whose language is not english

```
[('name','=','ABC'),'!',('language.code','=','en_US'),'|',('country_id.code','=','be'),
('country_id.code','=','de'))
```

The '&' is omitted as it is the default, and of course we could have used '!=' for the language, but what this domain really represents is:

```
(name is 'ABC' AND (language is NOT english) AND (country is Belgium OR Germany))
```

`unlink`(*cr*, *uid*, *ids*, *context=None*)
Delete records with given ids

Parameters:
- cr -- database cursor
- uid -- current user id
- ids -- id or list of ids
- context -- (optional) context arguments, like lang, time zone

Returns: True

Raises:
- AccessError --
  - if user has no unlink rights on the requested object
  - if user tries to bypass access rules for unlink on the requested object
- UserError -- if the record is default property for other records

`view_init`(*cr*, *uid*, *fields_list*, *context=None*)
Override this method to do specific things when a view on the object is opened.

`write`(*cr*, *user*, *ids*, *vals*, *context=None*)
Update records with given ids with the given field values

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

Returns: True

Raises:
- AccessError --
  - if user has no write rights on the requested object
  - if user tries to bypass access rules for write on the requested object
- ValidateError -- if user tries to enter invalid value for a field that is not in selection
- UserError -- if a loop would be created in a hierarchy of objects a result of the operation (such as setting an object as its own parent)

Note: The type of field values to pass in `vals` for relationship fields is specific:

- For a many2many field, a list of tuples is expected. Here is the list of tuple that are accepted, with the corresponding semantics

```
(0, 0,  { values })    link to a new record that needs to be created with the
given values dictionary
(1, ID, { values })    update the linked record with id = ID (write *values*
on it)
(2, ID)                remove and delete the linked record with id = ID (call
s unlink on ID, that will delete the object completely, and the link to it as
well)
(3, ID)                cut the link to the linked record with id = ID (delete
the relationship between the two objects but does not delete the target objec
t itself)
(4, ID)                link to existing record with id = ID (adds a relations
hip)
(5)                    unlink all (like using (3,ID) for all linked records)
(6, 0, [IDs])          replace the list of linked IDs (like using (5) then
(4,ID) for each ID in the list of IDs)

Example:
   [(6, 0, [8, 5, 6, 4])] sets the many2many to ids [8, 5, 6, 4]
```

- For a one2many field, a lits of tuples is expected. Here is the list of tuple that are accepted, with the corresponding semantics

```
(0, 0,  { values })    link to a new record that needs to be created with the
given values dictionary
(1, ID, { values })    update the linked record with id = ID (write *values*
on it)
(2, ID)                remove and delete the linked record with id = ID (call
s unlink on ID, that will delete the object completely, and the link to it as
well)

Example:
   [(0, 0, {'field_name':field_value_record1, ...}), (0, 0, {'field_name':fie
ld_value_record2, ...})]
```

- For a many2one field, simply use the ID of target record, which must already exist, or `False` to remove the link.
- For a reference field, use a string with the model name, a comma, and the target object id (example: `'product.product,` `5'`)

Learn

Demo (https://www.odoo.com/saas_master/demo)

Help (https://www.odoo.com/forum/Help-1)
Documentation (https://www.odoo.com/page/documentation)

Presentations (http://www.slideshare.net/openobject)
Videos (https://www.youtube.com/user/OpenERPonline)

Education Program (https://www.odoo.com/page/education-program)

Open Source

GitHub (https://github.com/odoo)

Download (https://www.odoo.com/page/download)
Mailing Lists (https://www.odoo.com/page/odoo-community)
Community (https://www.odoo.com/page/community)

Continuous Integration (http://runbot.openerp.com/)

Report a Bug (https://github.com/odoo/odoo/issues)
Translate

Services

Pricing (https://www.odoo.com/page/pricing)

Trainings (https://www.odoo.com/event)
Point of Sales Materials (https://www.odoo.com/shop/category/Point-of-Sale-Hardware-3)

Find a Partner (https://www.odoo.com/partners)
Become a Partner (https://www.odoo.com/page/become-a-partner)

Find an Expert (https://www.odoo.com/certifications)

About us

Contact us (https://www.odoo.com/page/contactus)
About us (https://www.odoo.com/page/about-us)
Our Customers (https://www.odoo.com/page/cases)

Events (https://www.odoo.com/event)
Blogs (https://www.odoo.com/blog/OpenERP-Blog-Post-1)

Jobs (https://www.odoo.com/jobs)

Twitter (https://twitter.com/Odooapps)
Facebook (https://www.facebook.com/odooapps)
LinkedIn (https://www.linkedin.com/company/odoo)
Google+ (https://plus.google.com/u/0/100883176139028080672/posts)

Apps to Boost Sales

CRM (https://www.odoo.com/page/crm)
Point of Sales (https://www.odoo.com/page/point-of-sale)
Quote Builder (https://www.odoo.com/page/quote-builder)

Mass Mailing (https://www.odoo.com/page/mailing)
Survey (https://www.odoo.com/page/survey)
Events (https://www.odoo.com/page/events)

Community Builder (https://www.odoo.com/page/website.community-builder)
Lead Automation (https://www.odoo.com/page/lead-automation)
Live Chat (https://www.odoo.com/page/live-chat)

Apps to Build Websites

CMS (https://www.odoo.com/page/website-builder)
e-Commerce (https://www.odoo.com/page/e-commerce)
Blogs (https://www.odoo.com/page/blog-engine)

Forum (https://www.odoo.com/page/website.community-builder)

Apps to Run Operations

Project Management (https://www.odoo.com/page/project-management)
Billing (https://www.odoo.com/page/billing)
Accounting (https://www.odoo.com/page/accounting)

Warehouse Management (https://www.odoo.com/page/warehouse)
Manufacturing (https://www.odoo.com/page/manufacturing)
Procurements (https://www.odoo.com/page/purchase)

Apps to Delight Employees

Employees Directory (https://www.odoo.com/page/employees)
Enterprise Social Network (https://www.odoo.com/page/enterprise-social-network)
Recruitment (https://www.odoo.com/page/recruitment)

Expenses (https://www.odoo.com/page/expenses)
Appraisals (https://www.odoo.com/page/appraisal)
Fleet (https://www.odoo.com/page/fleet)