Odoo Quick Reference Guide

Your handy companion for navigating and developing within the Odoo ERP framework, covering core concepts, shortcuts, ORM, and view basics.

Odoo Essentials & Development

Basic Navigation & UI

Main Menu	Access different Odoo Apps (Sales, Inventory, etc.) via the top-left icon.
Breadcrumbs	Track your navigation path, located below the main menu.
Views (Kanban, List, Form)	Switch between data representations using icons in the top-right (below search).
Search Bar	Perform quick searches across records. Use facets for advanced filtering.
Filters & Group By	Refine record lists using predefined or custom filters and grouping options.
Pager	Navigate through multiple pages of records in List view.
User Menu	Access Preferences, Documentation, Shortcuts, Log out (top-right corner).
Create Button	Initiate the creation of a new record in the current model.
Form View Buttons	`Edit`, `Save`, `Discard`, `Action` buttons manage record state.

Keyboard Shortcuts (General)

`Alt + C`	Create a new record.
`Alt + S` or `Ctrl + S`	Save the current record (Form View).
`Alt + J` or `Ctrl + ArrowDown`	Navigate to the next record.
`Alt + K` or `Ctrl + ArrowUp`	Navigate to the previous record.
`Alt + Q`	Discard changes (Form View).
`Alt + E`	Switch to Edit mode (Form View).
`Alt + V`	Open View Switcher.
`Alt + X`	Toggle the Debug Mode (if activated).
`Alt + P`	Open Pager options.
`Alt + F`	Open Filters menu.

Common Odoo Models

- `res.partner`: Contacts, Companies, Addresses
- `res.users`: System Users
- `product.product`: Storable Products, Services, Consumables
- `product.template`: Product Base Model (defines variants)
- `sale.order`: Quotations and Sales Orders
- `purchase.order`: Purchase Orders
- `account.move`: Journal Entries, Invoices, Bills
- `stock.picking`: Inventory Transfers (Receipts, Deliveries)
- `hr.employee`: Employee Information
- `ir.ui.view`: View Definitions (XML)
- `ir.actions.act_window`: Window Actions

Developer Mode

Activation:
Go to `Settings` -> `General Settings` -> Activate the developer mode (usually at the bottom).

With Assets: Reloads frontend assets (JS, CSS). Slower.

Without Assets: Faster activation, doesn't reload assets.

URL Activation: Append `?debug=1` or `?debug=assets` to the Odoo URL.

Debug Menu: Appears as a bug icon in the top menu bar.

Features:

- View Metadata (Fields, Views)
- Edit Actions, Views, Menus
- Run Scheduled Actions manually
- Access Technical Settings
- View Field information on hover

View Metadata: Open a record or view, click the Debug Menu -> `View Fields` / `Edit FormView` / `Edit ListView`, etc.

Technical Menu: `Settings` -> `Technical` (visible only in debug mode).

Deactivation: Click the Debug Menu -> `Leave the Developer Tools` or go back to General Settings.

Basic ORM Methods

`self.env['model.name']`

Access the environment for a specific model.

`.search(domain, limit, order)`

Find records matching the domain (list of tuples). Returns a recordset.
Example:
`partners = self.env['res.partner'].search([('is_company', '=', True)])`

`.search_count(domain)`

Count records matching the domain.
Example:
`count = self.env['sale.order'].search_count([('state', '=', 'sale')])`

`.browse(ids)`

Get a recordset from a list or tuple of IDs.
Example:
`user = self.env['res.users'].browse(self.env.uid)`

`.create(vals)`

Create a new record.
Example:

```
new_partner = self.env['res.partner'].create({'name': 'New Customer'})```
```

`.write(vals)`

Update records in the recordset.
Example:
`record.write({'field_name': 'new_value'})`

`.unlink()`

Delete records in the recordset.
Example:
`records_to_delete.unlink()`

`.read(fields)`

Read specific fields for records in the recordset. Returns a list of dictionaries.

`.filtered(lambda)`

Filter a recordset based on a Python function.
Example:
`active_partners = partners.filtered(lambda p: p.active)`

`.mapped('field_name')`

Extract values of a specific field from a recordset.
Example:
`names = partners.mapped('name')`

XML View Basics

Basic Structure:

```
<odoo>
  <data>
    <record model="ir.ui.view"
id="view_mymodel_form">
      <field
name="name">mymodel.form</field>
      <field
name="model">my.model</field>
      <field name="arch" type="xml">
        <form string="My Model Form">
          <sheet>
            <group>
              <field name="name"/>
              <field
name="description"/>
            </group>
          </sheet>
        </form>
      </field>
    </record>
  </data>
</odoo>
```

Common View Types: `form`, `tree` (list), `kanban`, `search`, `graph`, `pivot`

`<record>` Tag: Defines Odoo data records (views, actions, menus, etc.).

`model` Attribute: Specifies the Odoo model the record belongs to (e.g., `ir.ui.view`).

`id` Attribute: Unique identifier for the record within the module.

`<field name="arch" type="xml">`: Contains the actual view architecture.

`<form>`, `<tree>`, `<kanban>`: Root tags for respective view types.

`<field name="field_name"/>`: Displays a model field.

`<group>`, `<notebook>`, `<page>`: Structure elements within forms.

Attributes: `string`, `invisible`, `readonly`, `required`, `widget`, `options`, `domain`, `context` modify field behavior and appearance.

Odoo Server Commands (CLI)

`odoo-bin -c <config_file>`
Start Odoo server using a configuration file.
`odoo-bin --addons-path=<paths>`
Specify comma-separated paths to addon directories.
`odoo-bin -d <database_name>`
Specify the database to use.
`odoo-bin -i <module_name>`
Install a module. Requires `-d`.
`odoo-bin -u <module_name>`
Update a module. Requires `-d`.
`--init=all` or `--update=all`
Install/Update all modules found in the addons path.
`--stop-after-init`
Stop the server after initialization/update is complete.
`--xmlrpc-port=<port>`
Specify the XML-RPC port (default: 8069).
`--log-level=<level>`
Set the logging level (e.g., `info`, `debug`, `warn`, `error`).
`--test-enable`
Run tests after module installation/update.

Search Domain Syntax

Format	List of tuples: `[('field_name', 'operator', 'value'), ...]`
`=`	Equals
`!=`	Not Equals
`>`	Greater than
`>=`	Greater than or Equal to
`<`	Less than
`<=`	Less than or Equal to
`in`	Value is in a list. `('id', 'in', [1, 2, 3])`
`not in`	Value is not in a list.
`ilike`	Case-insensitive 'like'. Use `%` as wildcard. `('name', 'ilike', '%erp%')`
`like`	Case-sensitive 'like'.
`child_of`	For hierarchical relationships (parent/child).
Logical Operators	Default is AND (`&`). Use `'|'` for OR, `'!'` for NOT as prefixes. Example: `['|', ('state', '=', 'draft'), ('user_id', '=', uid)]`

Module Structure (`__manifest__.py`)

`__manifest__.py`: (Required) Dictionary describing the module.

`'name': 'My Module'` (Required)

`'version': '16.0.1.0.0'` (Required)

`'summary': 'Short description'`

`'description': 'Longer description (can be RST)'`

`'author': 'Your Name/Company'`

`'website': 'Your Website URL'`

`'category': 'Category/Subcategory'` (e.g., 'Sales/CRM')

`'depends': ['base', 'sale_management']` (List of dependencies)

`'data': ['security/ir.model.access.csv', 'views/my_view.xml']` (List of data files to load)

`'installable': True`

`'application': False` (Set to `True` if it's a main application)

`'license': 'LGPL-3'` (Or other license like OEEL-1)

Basic Field Types (Models)

`fields.Char`
String field.
`fields.Text`
Multi-line text field.
`fields.Html`
HTML formatted text field.
`fields.Integer`
Integer number field.
`fields.Float`
Floating-point number field.
`fields.Boolean`
True/False field.
`fields.Date`
Date field.
`fields.Datetime`
Date and Time field.
`fields.Selection`
Dropdown selection list. Requires a list of tuples `[('value', 'Label'), ...]` or a method name returning it.
`fields.Many2one`
Link to another model (foreign key). Requires `comodel_name='other.model'`.
`fields.One2many`
Inverse relationship for a Many2one. Requires `comodel_name='other.model'`, `inverse_name='m2o_field_on_other_model'`.
`fields.Many2many`
Many-to-many relationship. Requires `comodel_name='other.model'`.