Odoo Development Cookbook

Odoo Development Cookbook starts by covering Odoo installation and administration, and provides a gentle introduction to application development. It then dives deep into several of the areas that an experienced developer will need to use. You'll learn how to implement business logic, adapt the UI, and extend existing features.

These practical and easy-to-follow recipes are presented step by step, with dozens of hands-on recipes to boost your Odoo skills.

What this book will do for you...

- Install and manage Odoo environments and instances
- Use models to define your application's data structures
- Add business logic to your applications
- Implement automated tests and debug Odoo apps
- Use backend views to create a user interface
- Develop frontend website features
- Extend the web client with new widgets and features

Inside the Cookbook...

- A straightforward and easy-to-follow format
- A selection of the most important tasks and problems
- Carefully organized instructions to solve problems efficiently
- Clear explanations of what you did
- Solutions that can be applied to solve real-world problems

Odoo Development Cookbook

Build effective applications by applying Odoo development best practices

Holger Brunn
Alexandre Fayolle
Daniel Reis


Prices do not include local sales tax or VAT where applicable.

Free Sample
In this package, you will find:

- The author's biography
- A preview chapter from the book, Chapter 1 'Installing the Odoo Development Environment'
- A synopsis of the book’s content
- More information on Odoo Development Cookbook
About the Authors

Holger Brunn has been a fervent open source advocate since he came in to contact with the open source market sometime in the nineties. With an academic background in philosophy and sociology, he turned his interest to generalized logic, which proves helpful in many aspects of his IT work. Turning into a professional programmer was a side effect of his hobbyist interest, combined with a part-time job with a very open-minded mentor to whom he owes a lot of gratitude.

He has programmed for ERP and similar systems in different positions since 2001. For the last eight years, he has dedicated his time to TinyERP, which became OpenERP and evolved into Odoo. Currently, he works at Therp BV in the Netherlands as a developer and is an active member of the Odoo Community Association (OCA). He's most interested in fundamental work on technical modules, but also enjoys contributing to other projects, with a focus on UI and website widgets, CRM, and knowledge management.

Alexandre Fayolle started working with Linux and free software in the mid 1990s and quickly became interested in the Python programming language. Between 1999 and 2012, he helped manage Logilab, a company he cofounded, he specialized in Python development, and had the opportunity to work on projects for large companies such as EDF, Arcelor-Mittal, and GdF Suez (now Engie) using the Cubicweb framework.

He also tackled fun projects involving machine learning, natural language processing, and multi-agent systems. In 2012, he joined Camptocamp to share his expertise on Python, PostgreSQL, and Linux with the team implementing Odoo (OpenERP at the time). He currently manages projects for Camptocamp and is strongly involved in the Odoo Community Association. In his spare time, he likes to play the vibraphone in a jazz quartet, but has recently been known for writing a book about Odoo, which he hopes you'll enjoy.
**Daniel Reis** has been working in the IT industry for over 15 years in developer, consultant, and management roles. Most of this work was at the Capgemini multinational consultancy firm, implementing proprietary business solutions for reference companies in a variety of sectors, such as industry, telecommunications, and banking. Daniel has a BSc in applied mathematics and a master's in business administration from the ISCTE Business School.

He's worked with Odoo solutions (formerly OpenERP) since 2010, and he is an active contributor in the Odoo community association projects. He has been a speaker at the Open Days annual conference and other open source events. He is the author of the first Odoo development book: *Odoo Development Essentials*, also by Packt Publishing.

He currently works at Securitas, the global security services company, where he has introduced Python, Odoo, and other open source solutions into the company's IT applications portfolio.

I thank my wife, Maria José, for all the support and patience that made this book possible.
Preface

Odoo, formerly known as OpenERP, is a great platform for developers. The framework at its core is very rich and allows building client–server applications from scratch as well as adapting existing applications to your needs through a clever extension mechanism and a very modular design. The latest versions have brought a wealth of new possibilities with the addition of a full-featured website development stack. The scope is huge and it is easy for newcomers to feel lost.

For years, Odoo developers have been learning their craft by reading the code of the addon modules, which are built on top of the framework to provide enterprise management features. While effective, the process is long and error prone, since it is difficult to know whether the source code you are learning from is using the latest possibilities offered by the framework, or if you are looking at an older module that has not been updated to use these features. To make things worse, some code flows are intrinsically hard to follow because they're partly in the business logic layer, partly in the database layer, partly in the request handling layer, and partly in the client side code. The introduction of a new API in version 8 has made things even more confusing, since most of the code base was not immediately ported to this new API.

This book is meant to save you time by tapping in to the years of experience accumulated by long-time Odoo contributors to learn the current best practices in Odoo development by focusing on the new features of version 9, and also giving a solid base in the existing mature functionality of the framework. Since Odoo has a long tradition of guaranteeing backward compatibility, most of the presented material should still work with the upcoming versions.

What this book covers

This book contains 16 chapters. We tried hard to make each chapter as independent as possible, and to make the various recipes in each chapter self-contained.

Chapter 1, Installing the Odoo Development Environment, explains how to create a development environment for Odoo, start Odoo, create a configuration file, and activate the developer tools of Odoo.
Chapter 2, *Managing Odoo Server Instances*, is about addon installation and upgrading. It provides useful tips for working with addons installed from GitHub, and organizing the source code of your instance.

Chapter 3, *Creating Odoo Modules*, explains the structure of an Odoo addon module and gives a step-by-step guide for creating a simple module from scratch.

Chapter 4, *Application Models*, focuses on Odoo model descriptions, and explains the various field types and the different inheritance models available in Odoo.

Chapter 5, *Basic Server Side Development*, introduces the v8 API of Odoo, presents the commonly used methods of the `Model` class, and explains how to write business logic methods.

Chapter 6, *Advanced Server Side Development Techniques*, deals with more advanced topics useful when writing business methods such as writing wizards to walk the user through a process or writing onchange methods. It also covers porting code from the old API to the v8 API.

Chapter 7, *Debugging and Automated Testing*, proposes some strategies for server-side debugging and an introduction to the Python debugger. It also explains how to write and run automated tests using YAML or Python for your addon modules.

Chapter 8, *Backend Views*, explains how to write business views for your data models and how to call server-side methods from these views. It covers the usual views (list view, form view, and search view) as well as the views introduced in recent versions of Odoo (Kanban, graph, calendar, pivot, and so on).

Chapter 9, *Module Data*, shows how to ship data along with the code of your module. It also explains how to write a migration script when a data model provided by an addon is modified in a new release.

Chapter 10, *Access Security*, explains how to control who has access to what in your Odoo instance, by creating security groups, writing access control lists to define what operations are available to each group on a given model, and, if necessary, by writing record level rules.

Chapter 11, *Internationalization*, deals with the translation of the user interfaces of your addons.

Chapter 12, *Automation and Workflows*, illustrates the different tools available in Odoo to implement business process for your records. It also shows how server actions and automated rules can be used to support business rules.

Chapter 13, *Web Server Development*, deals with the core of the web server in Odoo. It explains how to map URLs to methods and how to control who can access these URLs.

Chapter 14, *CMS Website Development*, shows how to customize websites built with Odoo, by writing your own templates and providing new snippets for use in the website builder.
Chapter 15, *Web Client Development*, dives into the JavaScript part of Odoo and explains how you can provide new widgets and make RPC calls to the server. It also gives tips about debugging and testing this part of your code.

Chapter 16, *Server Deployment*, provides advice on how to install and configure Odoo for production, including setting up a reverse proxy to encrypt network communications over HTTPS and ensuring that Odoo starts when the server boots.
Installing the Odoo Development Environment

In this chapter, we will cover the following topics:

- Easy installation of Odoo from source
- Managing Odoo environments using the `start` command
- Managing Odoo server databases
- Storing the instance configuration in a file
- Activating the Odoo developer tools
- Updating Odoo from source

Introduction

There are lots of ways to set up an Odoo development environment. This chapter proposes one of these, although you will certainly find a number of other tutorials on the web explaining other approaches. Keep in mind that this chapter is about a development environment, which has different requirements from a production environment, covered in Chapter 16, Server Deployment.
Installing the Odoo Development Environment

Easy installation of Odoo from source

For Odoo deployment, it is recommended to use a GNU/Linux environment. You may be
more at ease using Microsoft Windows or Mac OS X, but the fact is that most of the Odoo
developers are using GNU/Linux and you are much more likely to get support from the
community for OS-level issues happening on GNU/Linux than on Windows.

It is also recommended to develop using the same environment (same distribution, same
version) as the one which will be used in production. This will avoid nasty surprises such as
discovering on the day of deployment that some library has a different version than expected,
with a slightly different and incompatible behavior. If your workstation is using a different OS,
a good approach is to set up a virtual machine on your workstation and to install a GNU/Linux
distribution in the VM.

To avoid copying files between your workstation where you are running
your development environment and the virtual machine which runs Odoo,
you can configure a SAMBA share inside the virtual machine and store the
source code there. You can then mount the share on your workstation in
order to edit the files easily.

This book assumes you are running Debian GNU/Linux as its stable version (Jessie at the time
of writing). Ubuntu is another popular choice, and since it is built on top of Debian, most of
the examples in this book should work unchanged. Whatever Linux distribution you choose,
you should have some notion of how to use it from the command line, and having a few ideas
about system administration will certainly not cause any harm.

Getting ready

We assume that Linux is up and running and that you have an account with root access, either
because you know the root password or because sudo has been configured. In the following
pages, we will be using $(whoami) whenever the login of your work user is required in a
command line. This is a shell command which will substitute your login in the command you
are typing.

Some operations will definitely be easier if you have a GitHub account. Go to
https://github.com and create one if you don't have one already.

How to do it...

To install Odoo from source, you need to follow these steps:

1. Run the following commands to install the main dependencies:

   $ sudo apt-get install git python2.7 postgresql nano \
   python-virtualenv
2. Download and install wkhtmltopdf:
   $ wget http://nightly.odoo.com EXTRA/wkhtmltox-0.12.1.2_linux-jessie-amd64.deb
   $ sudo dpkg -i wkhtmltox-0.12.1.2_linux-jessie-amd64.deb

   **Caution!**
   This is a package provided by the Odoo maintainer for Debian Jessie.
   If you are using another distribution, browse to http://download.gna.org/wkhtmltopdf/0.12/0.12.1/ and download the package for your operating system.

3. Now, use this to install the build dependencies:
   $ sudo apt-get install gcc python2.7-dev libxml2-dev libxslt1-dev libevent-dev libsasl2-dev libldap2-dev libpq-dev libpng12-dev libjpeg-dev

4. Configure PostgreSQL:
   $ sudo -u postgres createuser --createdb $(whoami)
   $ createdb $(whoami)

5. Configure git:
   $ git config --global user.name "Your Name"
   $ git config --global user.email youremail@example.com

6. Clone the Odoo code base:
   $ mkdir ~/odoo-dev
   $ cd ~/odoo-dev
   $ git clone -b 9.0 --single-branch https://github.com/odoo/odoo.git
   $ cd odoo

7. Create an odoo-9.0 virtual environment and activate it:
   $ virtualenv ~/odoo-9.0
   $ source ~/odoo-9.0/bin/activate

8. Install the Python dependencies of Odoo in virtualenv:
   $ pip install -r requirements.txt
9. Create and start your first Odoo instances:

   $ createdb odoo-test

   $ python odoo.py -d odoo-test --addons-path=addons --dbfilter=odoo-test$

10. Point your browser to http://localhost:8069 and authenticate using the admin account and admin as password.

You can download the example code files for this book from your account at http://www.packtpub.com. If you purchased this book elsewhere, you can visit http://www.packtpub.com/support and register to have the files e-mailed directly to you.

You can download the code files by following these steps:

   ▶ Log in or register to our website using your e-mail address and password
   ▶ Hover the mouse pointer on the SUPPORT tab at the top
   ▶ Click on Code Downloads & Errata
   ▶ Enter the name of the book in the Search box
   ▶ Select the book for which you’re looking to download the code files
   ▶ Choose from the drop-down menu where you purchased this book from
   ▶ Click on Code Download

You can also download the code files by clicking on the Code Files button on the book's webpage at the Packt Publishing website. This page can be accessed by entering the book's name in the Search box. Please note that you need to be logged in to your Packt account.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

   ▶ WinRAR / 7-Zip for Windows
   ▶ Zipge / iZip / UnRarX for Mac
   ▶ 7-Zip / PeaZip for Linux

How it works...

Dependencies come from various sources. First, you have the core dependencies of Odoo, the Python interpreter that is used to run the source code, and the PostgreSQL database server used to store the instance data. Git is used for source code versioning and getting the source code of Odoo itself.
Since we will need to edit some files as root or as postgres (the PostgreSQL administrative user) on our server, we need to install a console-based text editor. We suggest nano as it is very simple to use, but feel free to choose any editor with which you feel at ease as long as it works on the console, such as vim, e3, or emacs-nox.

**Wkhtmltopdf** is a runtime dependency of Odoo used to produce PDF reports. The version required by Odoo 9.0 is 0.12.1, which is not included in current GNU/Linux distributions. Fortunately for us, the maintainers of wkhtmltopdf provide prebuilt packages for various distributions on [http://wkhtmltopdf.org/downloads.html](http://wkhtmltopdf.org/downloads.html) (in the archive section). However, Debian Jessie is not there, so the Odoo maintainers provide their own version of the package on [http://nightly.odoo.com/extra/](http://nightly.odoo.com/extra/).

There are lots of other runtime dependencies that are Python modules, which we can install using pip in a virtual environment. However, some of these Python modules can feature some dependencies on native C libraries for which the Python bindings need to be compiled. We therefore install the development packages for these C libraries as well as the Python development package and a C compiler. Once these build dependencies are installed, we can use `pip -r requirements.txt` (a file which comes from the Odoo source code distribution) to download, compile, and install the Python modules.

### Virtual environments

Python virtual environments, or virtualenv for short, are isolated Python workspaces. They are very useful to Python developers because they allow different workspaces with different versions of various Python libraries installed, possibly on different Python interpreter versions.

You can create as many environments as you wish using the command `virtualenv path/to/newenv`. This will create a `newenv` directory in the specified location, containing a `bin/` subdirectory and a `lib/python2.7` subdirectory.

In `bin/` you will find several scripts:

- **activate**: The script is not executed, it is sourced using the built-in `source` shell. This will activate the environment by adjusting the `PATH` environment variable to include the `bin/` directory of the `virtualenv`. It also installs a shell function called `deactivate`, which you can run to exit the `virtualenv`, and changes the shell prompt to let you know which `virtualenv` is currently activated.

- **pip**: This is a special version of the `pip` command which acts inside the `virtualenv` only.

- **python**: This is a wrapper around your system Python interpreter which uses the packages installed in the `virtualenv`.

---

---
Installing the Odoo Development Environment

The built-in `source` shell is also available (as a single dot, followed by a space, and the path to the file to source). The shortcut form is perfectly fine, but we will stick to `source` in this book for readability.

There are two main ways of using a `virtualenv`. You may activate it as we show in the recipe (and call `deactivate` when you're done) or you may use the scripts in the `bin/` directory of the environment explicitly by calling them with their full path, in which case you don't need to activate the `virtualenv`. This is mainly a matter of taste, so you should experiment and find out which style suits you better for which case.

You may have executable Python scripts with the first line looking like the following:

```
#!/usr/bin/env python
```

These will be easier to use with an activated `virtualenv`. This is the case with the `odoo.py` script, which you can therefore call in the following way:

```
$ ./odoo.py -d odoo-test --addons-path=addons --db-filter=odoo-test$
```

**PostgreSQL configuration**

On a GNU/Linux system, Odoo works very well with the default values of `psycopg2`, the Python module used to access a PostgreSQL database:

- Passwordless authentication if the database user has the same name as the current user on local connections
- Local connection uses Unix domain sockets
- The database server listens on port 5432

In that case, there is nothing special to do: we use the `postgres` administrative user to create a database user which shares our login name and give it the right to create new databases. We then create a new database with the same name as the new user, which will be used as a default database when using the `psql` command.

When on a development server, it is OK to give the PostgreSQL user more rights and to use the `--superuser` command-line option rather than just `--createdb`. The net effect is that this user can then also create other users and globally manage the database instance. If you feel `--superuser` is too much, you may still want to use `--createrole` in addition to `--createdb` when creating your database user. Avoid doing this on production servers as it would give additional leverage to an attacker exploiting a vulnerability in some part of the deployed code (see Chapter 16, Server Deployment).

If you want to use a database user with a different login, you will need to provide a password for the user. This is done by passing the `--pwprompt` flag on the command line when creating the user, in which case the command will prompt you for the password.
If the user has already been created and you want to set a password (or modify a forgotten password) you can use the following command:

```
$ psql -c "alter role $(whoami) with password 'newpassword'"
```

If this command fails with an error message saying that the database does not exist, it is because you did not create a database named after your login name in step 3. That's fine; just add the `--dbname` option with an existing database name such as `--dbname template1`.

**Git configuration**

At some point in the book, you will need to use `git commit`. This will fail unless some basic configuration is performed; you need to provide Git with your name and email address. Git will remind you to do this with a nice error message, but you may as well do it now.

This is also something to keep in mind if you are using a service such as Travis for continuous integration, and your test scripts need to perform some `git` merges: you have to provide a dummy name and e-mail for the merging to succeed.

**Downloading the Odoo source code**

Downloading the Odoo code base is done by performing a `git clone` operation. Be patient as this will take some time. The options `--branch 9.0 --single-branch` avoid downloading other branches and save a little time. The `--depth` option can also be used to avoid downloading the whole repository history, but the downside of that option is that you will not be able to explore that history when looking for issues.

The Odoo developers also propose nightly builds, which are available as tarballs and distribution packages. The main advantage of using a `git clone` is that you will be able to update your repository when new bug fixes are committed in the source tree. You will also be able to easily test any proposed fixes and track regressions, so you can make your bug reports more precise and helpful for the developers.

**Starting the instance**

Now comes the moment you've been waiting for. To start our first instance, we first create a new empty database and then use the `odoo.py` script with the following command-line arguments:

- `-d database_name`: Use that database by default.
Installing the Odoo Development Environment

- `--db-filter=database_name$`: Only try to connect to databases matching the supplied regular expression. One Odoo installation can serve multiple instances living in separate databases and this argument limits the available databases. The trailing `$` is important as the regular expression is used in match mode; this avoids selecting names starting with the specified string.

- `--addons-path=directory1,directory2,...`: This is a comma separated list of directories in which Odoo will look for addons. This list is scanned at the instance creation time to populate the list of available add-on modules in the instance.

If you are using a database user with a database login different from your Linux login, you need to pass the following additional arguments:

- `--db_host=localhost`: use a TCP connection to the database server
- `--db_user=database_username`: use the specified database login
- `--db_password=database_password`: the password to use to authenticate against the PostgreSQL server

To get an overview of all the available options, use the `--help` argument. We will see much more about the odoo.py script in this chapter as well as in Chapter 2, Managing Odoo Server Instances.

When Odoo is started on an empty database, it will first create the database structure needed to support its operations. It will also scan the addons path to find the available addon modules, and insert some the initial records in the database. This includes the admin user with the default password admin which you will use to authenticate with.

Odoo includes an HTTP server. By default, it listens on all local network interfaces on TCP port 8069 so pointing your web browser to `http://localhost:8069/` leads you to your newly created instance.

There is more...

In the recipe, we downloaded the latest stable version of Odoo using the following command:

$ git clone -b 9.0 --single-branch https://github.com/odoo/odoo.git

This uses the official branch maintained by Odoo. One issue with this branch is that bug fixes contributed by the community are not always merged in a timely fashion. The Odoo Community Association (OCA) maintains a parallel branch in which fixes and improvements are peer-reviewed by the community and tend to be merged faster than on the official branch. It is not a fork of Odoo, and the latest version of Odoo is merged back into that branch daily. You may want to use it for your developments and deployments, in which case you need to clone Odoo like this:

$ git clone -b 9.0 --single-branch https://github.com/OCA/OCB.git odoo
Managing Odoo environments using the start command

We will often want to use custom or community modules with our Odoo instance. Keeping them in a separate directory makes it easier to install upgrades to Odoo or troubleshoot issues from our custom modules. We just have to add that directory to the addons path and they will be available in our instance, just like the core modules are.

It is possible to think about this module directory as an Odoo environment. The Odoo start command makes it easy to organize Odoo instances as directories, each with its own modules.

Getting ready

For this recipe we need to have already installed Odoo. We assume that it will be at ~/odoo-dev/odoo, and that the virtualenv is activated.

This means that the following command should successfully start an Odoo server:

$ ~/odoo-dev/odoo/odoo.py

How to do it...

To create a work environment for your instance, you need to follow these steps:

1. Change to the directory where Odoo is:
   
   $ cd ~/odoo-dev

2. Choose a name for the environment and create a directory for it:

   $ mkdir my-odoo

3. Change to that directory and start an Odoo server instance for that environment:

   $ cd my-odoo/
   $ ../odoo/odoo.py start

How it works...

The Odoo start command is a shortcut to start a server instance using the current directory. The directory name is automatically used as the database name (for the -d option), and the current directory is automatically added to the addons path (the --addons-path option) as long as it contains an Odoo addon module. In the preceding recipe you won't see the current directory in the addons path because it doesn't contain any modules yet.
Installing the Odoo Development Environment

There’s more

By default the current directory is used, but the --path option allows you to set a specific path to use instead. For example, this would work from any directory:

$ ~/odoo-dev/odoo/odoo.py start --path=~/odoo-dev/my-odoo

The database to use can also be overridden using the usual -d option. In fact, all the other usual odoo.py command-line arguments, except --addons-path, will work. For example, to set the server listening port, use the following command:

$ ../odoo/odoo.py start --xmlrpc-port=8080

As we can see, the Odoo start command can be a convenient way to quickstart Odoo instances with their own module directory.

Managing Odoo server databases

When working with Odoo, all the data of your instance is stored in a PostgreSQL database. All the standard database management tools you are used to are available, but Odoo also proposes a web interface for some common operations.

Getting ready

We assume that your work environment is set up and you have an instance running. Do not start it using the odoo.py start command shown in the previous recipe, as it configures the server with some options which interfere with multi-database management.

How to do it...

The Odoo database management interface provides tools to create, duplicate, remove, back up, and restore a database. There is also a way to change the master password which is used to protect access to the database management interface.

Access the Database Management interface

To access the database, the following steps need to be performed:

1. Go to the login screen of your instance (if you are authenticated, log out).
2. Click on the Manage Databases link. This will navigate to http://localhost:8069/web/database/manager (you can also point your browser directly to that URL).
Set or change the master password

If you've set up your instance with default values, and not yet modified it as explained in the following section, the database management screen will display a warning telling you that the master password is not set, and advising you to set one, with a direct link:

1. To set the Master Password, you can click on that link. You will get a dialog box asking you to provide the new password:

![Set Master Password dialog box]

2. Type in a non-trivial new password and click on Continue.
3. When the master password is already set, click the Set Master Password button at the bottom of the screen to change it.
4. In the displayed dialog box, type the previous master password and the new one, and then click on Continue.
The master password is in the server configuration file under the admin_password key. If the server was started without specifying a configuration file, a new one will be generated in ~/.openerp_serverrc. See the next recipe for more information about the configuration file.

**Creating a new database**

This dialog box can be used to create a new database instance which will be handled by the current Odoo server:

1. In the database management screen, click on the Create Database button at the bottom of the screen.

2. Fill the form in as follows:
   - **Master password**: The master password for this instance.
   - **Database name**: Input the name of the database you wish to create.
   - **Language**: Select the language you wish to be installed by default in the new database.
   - **Password of admin user**: Type the password you want to set for the admin user of the new instance.
Load demonstration data: Check this box to have demonstration data. This is useful to run tests or set up a demonstration for a customer, but should not be checked for a database meant to contain production data.

3. Click on the Continue button, and wait a little until the new database is initialized. You will then be redirected to the instance, connected as the Administrator.

Troubleshooting
If you are redirected to a login screen, this is probably because the option --db-filter was passed to Odoo and the new database name did not match the new database name. Note that the odoo.py start command does this silently, making only the current database available. To work around this, simply restart Odoo without the start command, as shown in the first recipe of this chapter. If you have a configuration file (see the Storing the instance configuration in a file recipe later in this chapter), then check that the db_filter option is unset or set to a value matching the new database name.

Duplicating a database
Very often you will have an existing database and you want to experiment with it to try a procedure or run a test, but without modifying the existing data. The answer is simple: duplicate the database and run the tests on the copy. Repeat as many times as required:

1. In the database management screen, click on the Duplicate link next to the name of the database you wish to clone.
Installing the Odoo Development Environment

2. Fill in the form:
   - **Master Password**: the master password of the Odoo server
   - **New Name**: the name you want to give to the copy

3. Click on the **Continue** button.

4. You can then click on the name of the newly created database in the database management screen to access the login screen for that database.

**Removing a database**

When you have finished your tests, you will want to clean up the duplicated databases. To do this, perform the following steps:

1. In the database management screen, click on the Delete link next to the name of the database you want to remove.

   ![Delete Database Form](image)

   2. Fill in the form; enter the **Master Password**, which is the master password of the Odoo server.

   3. Click the **Delete** button.

   **Caution! Potential data loss!**
   If you selected the wrong database, and have no backup, there is no way to recover the lost data.
Back up a database

For creating a backup, the following steps need to be performed:

1. In the database management screen, click the Backup link next to the database you want to back up.

![Backup Database Form]

2. Fill in the form:
   - **Master Password**: the master password of the Odoo server.
   - **Backup Format**: always use zip for a production database, as it is the only real full backup format. Only use the pg_dump format for a development database where you don't really care about the file store (admin by default).

3. Click the Backup button. The backup file will be downloaded to your browser.
Restoring a database backup

If you need to restore a backup, this is what you need to do:

1. In the database management screen, click the Restore Database button at the bottom of the screen.

2. Fill in the form:
   - **Master Password**: the master password of the Odoo server.
   - **File**: a previously downloaded Odoo backup.
   - **Database Name**: provide the name of the database in which the backup will be restored. The database must not exist on the server.
   - **Generate a new database uuid**: leave unchecked if you are installing a database which has been deleted from the server; otherwise check the box. There is little difference between them, and if in doubt, leaving it unchecked is a safe choice.

3. Click the Continue button.

**Note**: It is not possible to restore a database on top of itself. If you try to do this, you will get an error message *(Database restore error: Database already exists)*. You need to remove the database first.
Chapter 1

How it works...

These features, apart from the Change master password screen, run postgresql administration commands on the server and report back through the web interface.

The master password is a very important piece of information which only lives in the Odoo server configuration file and is never stored in the database. There used to be a default value of admin, but using this value is a security liability as it is well known. In Odoo v9, this is identified as an "unset" master password and you get urged to change it when accessing the database administration interface. Even if it is stored in the configuration file under the admin_passwd entry, this is not the same as the password of the admin user; these are two independent passwords: the master password is set for an Odoo server process, which itself can handle multiple database instances, each of which has an independent admin user with his own password.

Security considerations: Remember that we are considering a development environment in this chapter. The Odoo database management interface is something which needs to be secured when you are working on a production server as it gives access to a lot of sensitive information, especially if the server hosts Odoo instances for several different clients. This will be covered in Chapter 16, Server Deployment.

To create a new database, Odoo uses the PostgreSQL createdb utility and calls the internal Odoo function to initialize the new database in the same way as when you start Odoo on an empty database.

To duplicate a database, Odoo uses the --template option of createdb passing the original database as an argument. This essentially duplicates the structure of the template database in the new database using internal and optimized PostgreSQL routines, which is much faster than creating a backup and restoring it (especially when using the web interface which requires downloading the backup file and uploading it again).

Backup and restore operations use the pg_dump and pg_restore utilities respectively. When using the .zip format, the backup will also include a copy of the file store which contains a copy of the documents when you configure Odoo to not keep these in the database, which is the default in 9.0. Unless you configure it otherwise, these files live in ~/.local/share/Odoo/filestore.
Installing the Odoo Development Environment

If the backup gets large, downloading it may fail, either because the Odoo server itself is not able to handle the large file in memory or if the server runs behind a reverse proxy (see Chapter 16, Server Deployment) because there is a limit to the size of HTTP responses set in the proxy. Conversely, for the same reasons, you will likely experience issues with the database restore operation. When you start running into these issues, it is time to invest in a more robust external backup solution.

There is more...

Experienced Odoo developers generally don't use the database management interface, and perform the operations from the command line. To initialize a new database with demo data for instance, the following one-liner can be used:

```
$ createdb testdb && odoo.py -d testdb
```

The additional bonus of this command line is that you can request installation of addons while you are at it using for instance `-i sale,purchase,stock` (more on this in Chapter 2, Managing Odoo Server Instances).

To duplicate a database, stop the server, and run the following command:

```
$ createdb -T dbname newdbname
$ cd ~/.local/share/Odoo/filestore # adapt if you have changed the data_dir
$ cp -r dbname newdbname
$ cd -
```

Note that in the context of development, the file store is often omitted.

The use of `createdb -T` only works if there are no active sessions on the database, which means you have to shut down your Odoo server before duplicating the database from the command line.

To remove an instance, run the following command:

```
$ dropdb dbname
$ rm -rf ~/.local/share/Odoo/filestore/dbname
```

To create a backup (assuming the PostgreSQL server is running locally), use the following command:

```
$ pg_dump -Fc -f dbname.dump dbname
$ tar cjf dbname.tgz dbname.dump ~/.local/share/Odoo/filestore/dbname
```
To restore the backup, run the following command:

```
$ tar xf dbname.tgz
$ pg_restore -C -d dbname dbname.dump
```

**Caution!**
If your Odoo instance uses a different user to connect to the database you need to pass `-U username` so that the correct user is the owner of the restored database.

### Storing the instance configuration in a file

The `odoo.py` script has dozens of options, and it is tedious to remember them all and to remember to set them properly when starting the server. Fortunately, it is possible to store them all in a configuration file and to only specify by hand the ones you want to alter, for example, for development.

#### How to do it...

To generate a configuration file for your Odoo instance, run the following command:

```
$ odoo.py --save --config myodoo.cfg --stop-after-init
```

You can add additional options, and their values will be saved in the generated file. All the unset options will be saved with their default value set. To get a list of possible options, use:

```
$ odoo.py --help | less
```

This will provide you with some help about what the various options perform. To convert from the command line form to the configuration form, use the long option name, remove the leading dashes, and convert the dashes in the middle to underscores: `--without-demo` becomes `without_demo`. This works for most options, but there are a few exceptions listed in the next section.

Edit the file `myodoo.cfg` (use the table in the following section for some parameters you may want to change). Then to start the server with the saved options, run the following command:

```
$ odoo.py -c myodoo.cfg
```

**Note:** The `--config` option is commonly abbreviated as `-c`
Installing the Odoo Development Environment

How it works...

At start up, Odoo loads its configuration in three passes. First a set of default values for all options is initialized from the source code. Then the configuration is parsed, and any value defined in the file overrides the defaults. Finally, the command-line options are analyzed and their values override the configuration obtained from the previous pass.

As mentioned earlier, the names of the configuration variables can be found from the names of the command-line options by removing the leading dashes and converting the middle dashes to underscores. There are a few exceptions, notably:

<table>
<thead>
<tr>
<th>Command line</th>
<th>Configuration file</th>
</tr>
</thead>
<tbody>
<tr>
<td>--db-filter</td>
<td>dbfilter</td>
</tr>
<tr>
<td>--no-xmlrpc</td>
<td>xmlrpc = True / False</td>
</tr>
<tr>
<td>--database</td>
<td>db_name</td>
</tr>
<tr>
<td>--debug</td>
<td>debug_mode = True / False</td>
</tr>
<tr>
<td>--i18n-import / --i18n-export</td>
<td>Unavailable</td>
</tr>
</tbody>
</table>

Here is a list of options commonly set through the configuration file:

<table>
<thead>
<tr>
<th>Option</th>
<th>Format</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>without_demo</td>
<td></td>
<td>Prevents module demo data from being loaded.</td>
</tr>
<tr>
<td>addons_path</td>
<td>Comma separated list of paths</td>
<td>A list of directory names in which the server will look for addons (see Chapter 2, Managing Odoo Server instances).</td>
</tr>
<tr>
<td>admin_passwd</td>
<td>Text</td>
<td>The master password (see previous recipe).</td>
</tr>
<tr>
<td>data_dir</td>
<td>Path to a directory</td>
<td>A directory in which the server will store session information, addons downloaded from the Internet, and documents if you enable the file store.</td>
</tr>
<tr>
<td>db_host</td>
<td>Host name</td>
<td>The name of the server running the PostgreSQL server. Use False to use local Unix Domain sockets, and localhost to use TCP sockets locally.</td>
</tr>
<tr>
<td>db_user</td>
<td>Database user login</td>
<td></td>
</tr>
<tr>
<td>db_password</td>
<td>Database user password</td>
<td>This is generally empty if db_host is False and db_user has the same name as the user running the server. Read the man page of \texttt{pg_hba.conf} for more information on this.</td>
</tr>
</tbody>
</table>
Option | Format | Usage
--- | --- | ---
database | Database name | Used to set the database name on which some commands operate by default). This does not limit the databases on which the server will act. See the following `dbfilter` option for this.
dbfilter | A regular expression | The expression should match the name of the databases considered by the server. If you run the website, it should match a single database, so it will look like `^databasename$`. More information on this is in Chapter 16, Server Deployment.
xmlrpc_interface | IP address of a network interface | Defaults to `0.0.0.0`, meaning the server listens on all interfaces.
xmlrpc_port | Port number | The ports on which the Odoo server will listen. You will need to specify both to run multiple Odoo servers on the same host. `longpolling_port` is only used if `workers` is not `0`.
longpolling_port | Port number | The ports on which the Odoo server will listen. You will need to specify both to run multiple Odoo servers on the same host. `longpolling_port` is only used if `workers` is not `0`.
logfile | Path to a file | The file in which Odoo will write its logs.
log_level | Log verbosity level | Specify the level of logging. Accepted values (in increasing verbosity order): `critical`, `error`, `warn`, `info`, `debug`, `debug_rpc`, `debug_rpc_answer`, `debug_sql`.
workers | Integer | The number of worker processes. See Chapter 16, Server Deployment, for more information.
no_database_list | True / False | Set to `True` to disable listing of databases. See Chapter 16, Server Deployment, for more information.

There is more...

The parsing of the configuration file by Odoo is done using the Python `ConfigParser` module. This module supports defining values for variables from the values of other variables using the `{section.variable}`s notation. You can omit `section` if the value comes from the same section or if it is defined in the special `[DEFAULT]` section.

For instance, if you want to define the database login to be the same as the database name, you can write the following in your Odoo configuration file:

```
[options]
db_name = projectname
db_user = %(options.db_name)s
```
Installing the Odoo Development Environment

A very common use is to define a common prefix for the paths of the addons:

```
[DEFAULT]
project = /home/odoo/projects/project1
env = dev
prefix = %(project)s/%(env)s

[options]
addons-path = %(prefix)s/odoo/addons,%(prefix)s/OCA/server-tools
data_dir = %(prefix)s/data_dir
```

Activating the Odoo developer tools

When using Odoo as a developer, you need to know how to activate Developer Mode in the web interface to access the advanced settings menu and developer information.

**How to do it...**

To activate Developer Mode in the web interface:

1. Connect to your instance and authenticate (not necessarily as admin; this function is available to all users, but the Administrator has more tools available).
2. Click on the down arrow next to your user name in the top right corner of the page.
3. In the drop-down menu, click on About.
4. In the dialog box which is displayed, click on **Activate the developer mode** in the top right corner.

It is also possible to activate the developer mode by editing the URL: before the # sign, insert `?debug=`. For instance, if you are starting from: `http://localhost:8069/web#menu_id=102&action=94`, then you need to change this to: `http://localhost:8069/web?debug=#menu_id=102&action=94`. 
5. To exit developer mode, you can edit the URL and remove that string, close your browser tab and open a new one, or use the Leave Debug Mode option at the bottom of the debug drop-down menu next to the user menu in the top right of the screen.

![Debug Menu](image)

**How it works...**

When in developer mode, three things happen:

- The JavaScript and CSS code sent to the browser is not minified, which means that the web development tools of your browser are easy to use for debugging the JavaScript code (more on this in Chapter 15, Web Client Development).

- You get tooltips when hovering over a field in a form view or over a column in list view providing technical information about the field (internal name, type, and so on).

- A drop-down menu with a Bug icon is displayed next to the user's menu in the top right corner giving access to technical information about the model being displayed, the various related view definitions, the workflow, custom filter management, and so on.

**Caution!**

Test your addons both with and without developer mode, as the unminified versions of the JavaScript libraries can hide bugs which only bite you in the minified version.
**Updating Odoo from source**

We saw in the first recipe how to install Odoo from source by using the git repository. The main benefit of this setting is being able to update the source code of Odoo using git to get the latest bug fixes.

### Getting ready

Stop any instance currently running with the Odoo source you are about to update.

Make a backup of all the databases you care about in case something goes bad. This is obviously something you need to do for production databases. See the *Managing Odoo server databases* recipe for instructions.

Then make a note of the current version of the source you are running. The best way is to create a lightweight tag using the following command:

```
$ cd ~/odoo-dev/odoo
$ git checkout 9.0
$ git tag 9.0-before-update-$date
```

### How to do it...

To update the source code of Odoo, use the following command:

```
$ git pull --ff-only
```

This will fetch the latest version of the source code committed to the current branch.

To update an instance running on this code, run the following command:

```
$ odoo.py -c myodoo.cfg --stop-after-init -u base
```

- `-u` is the shortcut notation for the `--update` option of `odoo.py`.

If you don't have a database set in the configuration file, you will have to add the `-d database_name` option. That command is to be repeated for all the instances running with this version of the source code.
Installing the Odoo Development Environment

If the update fails, don't panic, you have backups.

1. Read the error message carefully. Save it to a file, as it will be useful to make a bug report later.
2. If you cannot figure out what the problem is, restore the service; restore the Odoo source code to the previous version, which is known to work using the tag you set before updating the source version:
   $ git reset --hard 9.0-before-update-$\{(date --iso}\)
3. Drop the broken databases and restore them from the backups you made (see the Managing Odoo server databases recipe for instructions).
4. Restart your instances and tell your users that the upgrade has been postponed.

Note that in real life, this should never happen on a production database, because you would have tested the upgrade beforehand on a copy of the database, fixed the issues, and only done the upgrade on the production server after making sure that it runs flawlessly. But sometimes, you still get surprises, so even if you are really sure, make a backup.

How it works...

Updating the source code is done by making sure we are on the correct branch using git checkout, and then fetching the new revisions using git pull. The --ff-only option will cause a failure if you have local commits not present in the remote repository. If this happens and you want to keep your changes, you can use git pull (without --ff-only) to merge the remote changes with yours; otherwise, use git reset --hard origin/9.0 to force the update, discarding your local modifications.

The update command uses the following options:

- `-c`: specify the configuration file
- `--stop-after-init`: stop the instance when the update is over
- `-u base or --update base`: requests the update of the base module
When updating a module, Odoo does the following:

- It updates the database structure for the models defined in the module for which the structure changes. For updates on the stable branch of Odoo, there should be no such changes, but this can happen for your own addons or third party addons.
- It updates the database records stored in data files of the module, most notably the views. It then recursively updates the installed modules which have declared a dependency on the module.

Since the base module is an implicit dependency of all Odoo modules, updating it will trigger an update of all the installed modules in your instance. To update all installed modules, the alias all can be used instead of base.
Where to buy this book

You can buy Odoo Development Cookbook from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.