Day 1

Created	@January 9, 2024 10:21 PM
Status	Open
Sub-item	Potential Questions, ToDo
Updated	@September 17, 2024 5:42 PM

What is Frappe?

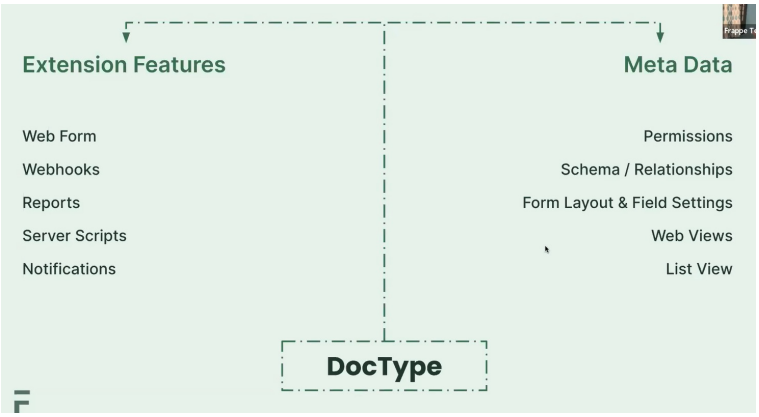
Low-code/no-code with option for full-code based on Python & JS with MIT License.

Hybrid: Typical MVC Framework + Low-code tools = Frappe

Batteries Included: Admin UI (Desk), Authentication, REST API + Webhooks, Caching Redis, Email, PDF, Realtime w/ SocketIO, Role Permissions, Scheduler, etc.

Made it possible to build a sophisticated web app without worrying about those components, just focus on the USP

How is the Frappe architecture?



Everything in Frappe is a DocType (Model + View), it also has meta data + extension features.

We can have as many sites as we want in a bench

bench —site mysite.local add-to-hosts (tell the computer that if any user wants to access mysite.local, just forward them to 127.0.0.1) **(it adds the site to /etc/hosts)**

/app is a special route where the Desk is rendered

private means only logged in users can access the file

Reload → clear cache, and then reload the page, it's different than hard refresh

Information about current user (my settings/user settings)

System settings → for system level settings (applied to all users)

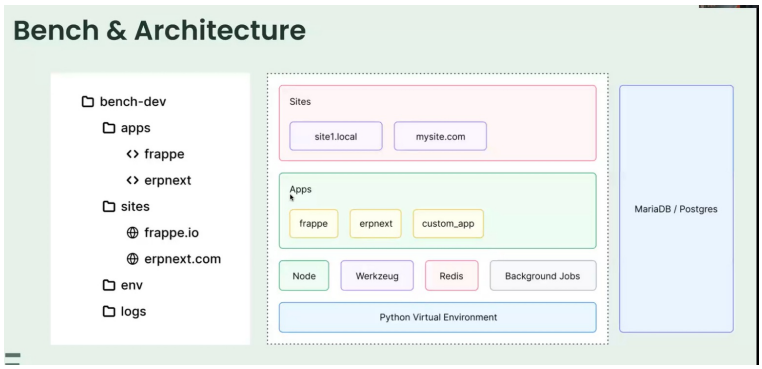
Website settings → for the portal

Custom? checkbox → When you're not in developer mode, and they live in the database of the site with no generated files for them

Doctype bts created a new database table automatically

Go into developer mode (don't do on production sites because it will start behaving differently): `bench --site [site-name] set-config developer_mode 1`. It will add `"developer_mode": 1` to the site_config.json of that site. Then reload the site to get the changes.

How is the bench directory? What is a site?



Bench is encapsulating a python venv. Bench gives you a venv and a CLI tool as well where you can create new sites apps etc.

Sites are their own instances standalone with their own DB, they also can have independent sets of apps. Werkzeug serving traffic for the website (web server). Node to build the website (css, js, socketio), it is used to run the socketio process that will push things happening on the backend to the frontend. Frappe runs on python, not node.

```
→  training-bench bench get-app NagariaHussain/tldraw_whiteboard
Getting tldraw_whiteboard
$ git clone https://github.com/NagariaHussain/tldraw_whiteboard.git
```

We can get-app using any git, as long as we provide path to the git initialized folder (so a locally unzipped folder can work, but for a local file path add "file://" precedence).

Core concept: Bench initializes a venv, it has apps, sites, every site has its own database. Apps bring code and doctypes etc.

Bench supports multitenancy, meaning multiple sites can run at once and they are distinguishable by their site name. Behind the scene, bench will figure out which site to route the traffic based on the name. It is the web server ran by the bench that does the job. In production, it is nginx. `Bench use` is used to force the localhost to point to a single site (single-tenant mode), so even if we access the localhost instead, it will be forwarded to the site.

Procfile (Process file) → used by `bench start` command, to determine what processes to start.

`watch` is only for development, it watches changes then reload automatically. `bench serve` uses Werkzeug.

The sites & site directory:

common_site_config is applied to all sites on that bench

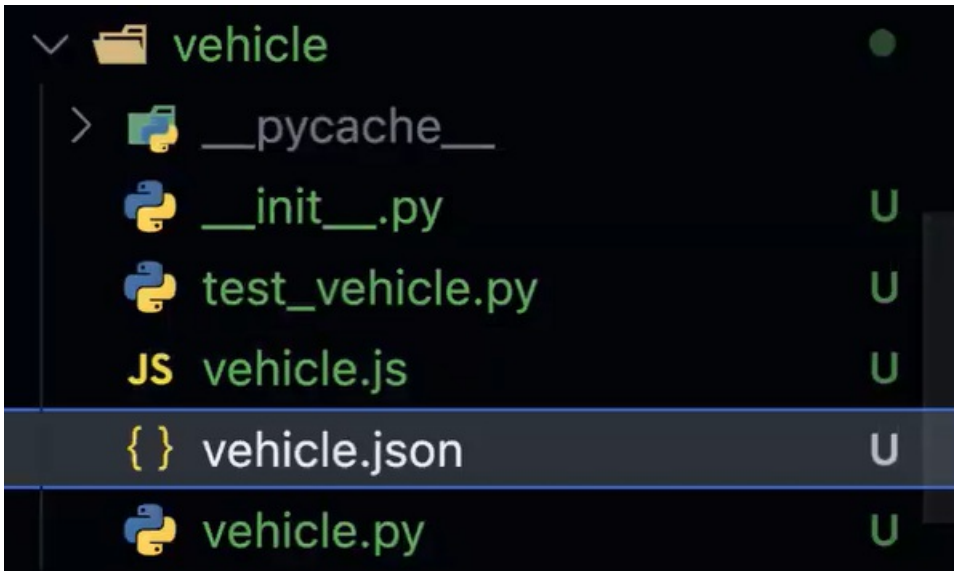
site_config is applied for just that particular site, contains db_name, db_password, db_type, and we can add more configs there, such as enable developer mode.

We can also check user-uploaded files on File doctype, website_theme is also placed on the public directory

logs contains various logs for database, scheduler, web, and errors.

There is site-level logs and bench-level logs, don't confuse them

When we create and installed a custom app on our site, there will be a default module created with the same name as the app name.



When we create a standard doctype, a `doctype` directory will appear inside the module, then a `[your-doctype-name]` directory will also appear inside the doctype directory, and it will create code files (.py, .js, .json) for that doctype. So, now doctypes are not just living in the database, they will also have their code files that are shareable to anyone.

On the database-level, when we create a doctype for the first time, frappe will create a new table on the database with the format `tab[doctype-name]`. Frappe also automatically make the fieldname based on the label (lowercased, replace spaces with _, removed special characters). It will also create a record/entry on `tabDocType` to store the newly created doctype.

.py are for the python code for logic (controllers), the .json are the schema (holds the fields, metadata, etc.) that will be used to create the db table behind the scene, the .js is for client-script (list view, form view, etc.)

test_vehicle.py is used to write unit tests.

There is also a doctype called `DocType` that stores all doctype.

`Version` doctype stores the changes/different versions of doctypes.

Form Settings → Image Field → set a fieldname that is of type "Attach Image" → Image will appear on the sidebar instead

View Settings → Title Field → set a fieldname → That field will be the title of each record instead of ID

There is a doctype called `Deleted Document` that keeps track of deleted records