Special Workshop in IT

Week 2: Developing a Library Management App

Sirasit Lochanachit, PhD

Today's Objectives

1) Creating a new app using Frappe Framework
2) Creating a new site
3) Install an app on a new site
4) Create a DocType
5) Install VS Code for code editing (optional)

Creating a New App

Creating a New App

1. To create a new app named 'library_management', go to 'frappe-bench' directory, then enter

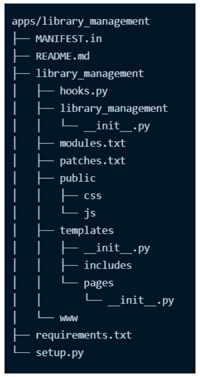
```
bench new-app library_management
```

2. Enter details of the app (can be anything)

- An app named 'library_management' will be created in the 'apps' folder

For list of bench commands: https://frappeframework.com/docs/user/en/bench/bench-commands

App Directory Structure



```
apps/library_management
├── MANIFEST.in
├── README.md
├── library_management
│   ├── hooks.py
│   ├── library_management
│   │   └── __init__.py
│   ├── modules.txt
│   ├── patches.txt
│   ├── public
│   │   ├── css
│   │   └── js
│   ├── templates
│   │   ├── __init__.py
│   │   ├── includes
│   │   └── pages
│   │       └── __init__.py
│   └── www
├── requirements.txt
└── setup.py
```

library_management: This directory will contain all the source code for your app

- public: Store static files that will be served from Nginx in production
- templates: Jinja templates used to render web views
- www: Web pages that are served based on their directory path
- library_management: Default Module bootstrapped with app
- modules.txt: List of modules defined in the app
- patches.txt: Patch entries for database migrations
- hooks.py: Hooks used to extend or intercept standard functionality provided by the framework

requirements.txt: List of Python packages that will be installed when you install this app

Library Management App Overview

We will build a simple Library Management System in which the **Librarian** can log in and manage Articles and Memberships

We will build the following models:

1. **Article**: A Book or similar item that can be rented
2. **Library Member**: A user who is subscribed to a membership
3. **Library Transaction**: An Issue or Return of an article
4. **Library Membership**: A document that represents an active membership of a Library Member
5. **Library Settings**: Settings that define values like Loan Period and the maximum number of articles that can be issued at a time

Library Management App Overview

The Librarian will log in to an interface known as **Desk**, a rich admin interface that ships with the framework.

- The Desk provides many standard views like List view, Form view, Report view, etc, and many features like Role-based Permissions.

We will also create public **Web Views** which can be accessed by the Library Members where they can browse available Articles.

Creating a New Site

Creating a New Site

1. To create a site called ‘library.test’, from the ‘frappe-bench’ directory, enter

   bench new-site library.test

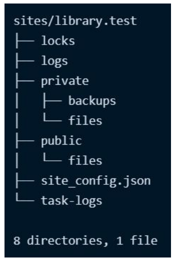
   This command will create a new DB

   Enter your MySQL password (created last week!)

   Then enter a new password for DB Admin user

- A new folder named ‘library.test’ will be created in the ‘sites’ folder

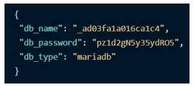
Site Directory Structure



```
sites/library.test
├── locks
├── logs
├── private
│   ├── backups
│   └── files
├── public
│   └── files
├── site_config.json
└── task-logs

8 directories, 1 file
```

site_config.json



```
{
 "db_name": "_ad03fa1a016ca1c4",
 "db_password": "pz1d2gN5y35ydRO5",
 "db_type": "mariadb"
}
```

The ‘private’ folder will contain any database backups and private files

- Private files are user uploaded files that need authentication to be accessible

The ‘public’ folder will contain files that are accessible without authentication

- This can contain website images that should be accessible without login

The site_config.json file contains configuration that is specific to this site which should not be version controlled

- This is similar to an environment variables file

Open a Recently Created Site

Now we can access our first site on http://localhost:8000 in our browser

- Also try http://library.test:8000
- ‘bench’ allows multiple sites to be created and accessed separately on the same port (Multi-tenancy)
- To allow multi-tenancy so that the specific site can be accessed, need to tell our operating system that library.test should point to localhost.
  - To do this, enter bench --site library.test add-to-hosts
  - This will map ‘library.test’ to ‘localhost’ and add an entry in ‘/etc/hosts’ file

Install Library Management App on Site

1. To install ‘Library Management’ app on our site, enter

   bench --site library.test install-app library_management

   This will install ‘library_management’ app on ‘library.test’ site

2. Check whether app was installed successfully on library.test site, run

   bench --site library.test list-apps

   ‘frappe’ and ‘library_management’ should appear as installed apps on your site

   - When a new site is created, the ‘frappe’ app is installed by default

Site Commands

Site Commands

The --site options allows access to Python and MariaDB and etc.

1. To access Python console, enter

   `bench --site library.test console`

2. To access MariaDB Console, enter

   `bench --site library.test mariadb`

3. To backup a database

   `bench --site library.test backup`

4. To see a list of all available site commands

   `bench --site library.test --help`

Creating a DocType

Login to Desk

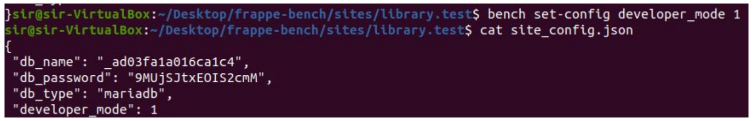
DocType is generally a database table and form (or a 'Model' in other frameworks)

- DocType define properties and behaviour of the model

1. Go to http://library.test:8000 to access a login page
   - Enter a username 'Administrator' and password that you created while building a site
2. After a successful login, the setup wizard appears to set up the site
   - Select your country and complete the wizard

Developer Mode

Before DocTypes can be created, need to enable developer mode



```
}sir@sir-VirtualBox:~/Desktop/frappe-bench/sites/library.test$ bench set-config developer_mode 1
sir@sir-VirtualBox:~/Desktop/frappe-bench/sites/library.test$ cat site_config.json
{
 "db_name": "_ad03fa1a016ca1c4",
 "db_password": "9MUjSJtxEOIS2cmM",
 "db_type": "mariadb",
 "developer_mode": 1
```

Creating a DocType

Go to Desk, type ‘doctype’ in the search bar and select the **DocType List** option

There are different types of DocTypes in the framework.

Let’s create **Article** doctype

1. Click on New
2. Enter Name as Article
3. Select Library Management Module
4. Add the following fields in the Fields table
5. Click Save

Article Name (Data)
Image (Attach Image)
Author (Data)
Description (Text Editor)
ISBN (Data)
Status (Select) - Enter two options: Issued and Available (Type Issued, hit enter, then type Available)
Publisher (Data)

Adding Articles

1. At Article doctype, click ‘Go to Article List’ at the top right of the form
   - Article List is the list view that shows the records from the DB table
2. Go to ‘Settings’ -> ‘Reload’ to clear Desk cache so that the **New** button appear
3. Click **New** button, which leads to the **Form** view of the Article doctype
   - Form view allows creating a new document of viewing an existing one

Article DocType

When a new doctype is created in the app, a number of files is generated

- article.json - JSON file that defines the doctype attributes
- article.js - Client-side controller for the Form view
- article.py - Python controller for Article
- test_article.py - Python Unit Test boilerplate for writing tests

DocType Features

1. Naming
   - https://frappeframework.com/docs/user/en/basics/doctypes/naming
2. Form Layout
3. Form Settings
4. Permissions

Controller Methods

Creating a Library Member DocType

Controller methods allow you to write business logic during the lifecycle of a document.

1. Create a 2nd doctype: Library Member, which have the following fields
   - First Name (Data, Mandatory)
   - Last Name (Data)
   - Full Name (Data, Read Only)
   - Email Address (Data)
   - Phone (Data)
2. Go to ‘Library Member list’, clear the cache, and create a new Library Member

Creating a Library Member DocType

Open the code editor and open the file ‘library_member.py’ and add the code:

```
class LibraryMember(Document):
  def before_save(self):
    self.full_name = f'{self.first_name} {self.last_name or ""}'
```

- The before_save method will run every time a document is saved
- This is one of the controller hooks provided by the **Document** class
  - More hooks - https://frappeframework.com/docs/user/en/basics/doctypes/controllers
- Try create another Library Member and see whether the Full Name showup

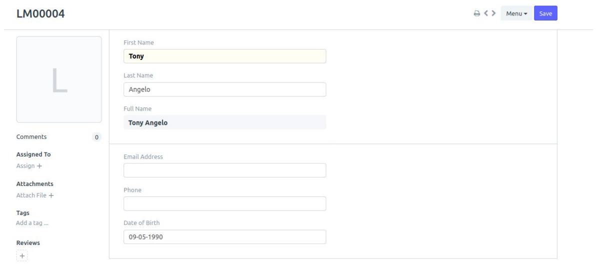
Next Week

- Another 3 DocTypes
  - Library Membership
  - Library Transaction
  - Library Settings
- Form Scripts
- Portal Pages for website visitors

Assignment #2

Customise Library Member DocType and Form to obtain the results in this screenshot

- Add your first name and lastname as a Library Member and submit your screenshot by Monday 25/01 before noon



LM00004

First Name: Tony
Last Name: Angelo
Full Name: Tony Angelo
Email Address:
Phone:
Date of Birth: 09-05-1990