MorpCC Documentation

Release 0.1.0

Izhar Firdaus

Dec 02, 2022

CONTENTS

Morp Control Center (MorpCC) is a meta information management system (meta-IMS) built on top of Morp Framework (morpfw) & Morepath. It is designed to provide common components needed for the the development of IMSes while allowing flexibility for developers to customize and override the components.

CHAPTER

ONE

FEATURES

- Responsive default UI based on Gentelella project
- Pluggable auth system
  - User, group & API key management system (SQLAlchemy based)
  - REMOTE_USER based authentication
- Content type framework and CRUD UI
- Pluggable CRUD storage backend
  - SQLAlchemy (default)
  - ElasticSearch
  - Dictionary based, in-memory
- Listing / search interface with JQuery DataTables server-side API
- Pluggable blob storage backend
  - Filesystem store (default)
- REST API through morpfw content type API engine with JWT based token
- Statemachine engine using PyTransitions through morpfw
- Overrideable components and templates through morepath & dectate app inheritance

CHAPTER

TWO

DOCUMENTATION

2.1 Introduction

MorpCC aims to solve several common challenges when doing enterprise web application development, which tend to require capabilities such as:

- Enterprise directory service (LDAP/AD) integration
- User, group and permission management
- Task scheduling for background jobs
- Customizable business rules logic
- Customizable / overrideable components and views to cater to sub/similar use-cases.
- Data might be stored in remote systems or APIs, not necessarily a database.
- Scalability to handle large data processing workload
- Corporate theming
- Mobile-ready / mobile integration
- State tracking and state management
- Fast turnaround time from business requirements to prototype application
- Activity tracking & analytics
- Messages & notifications

MorpCC, and its underlying framework, MorpFW attempts to assist the challenges above through leveraging the component engine provided by Morepath and Dectate to provide:

- Default admin+user UI which can be overridden easily, enabling agility in development by allowing developers to focus more on the data domain model first, rather than the repetive application bootstrapping tasks.
- CRUD with pluggable storage engine, allowing flexibility in writing your own storage implementation
- State engine support, simplifying task of developing stage management of your data objects
- Pluggable user, group, permission and API key management.
- Standardized REST API interface for external integration.
- (TBD) Powerful theming capability through diazo.

morpcc and morpfw design is highly influenced by Plone project. Unlike frameworks such as Django, Pyramid, and Flasks routing which routes to views, morpcc/morpfw routing routes to an object/model publisher. Views are attached to model and goes around with the model. This design gives the framework certain benefits:

- Views and view templates are highly reusable because as long as the model implements the attributes and methods the view queries, the view can be attached to the model.

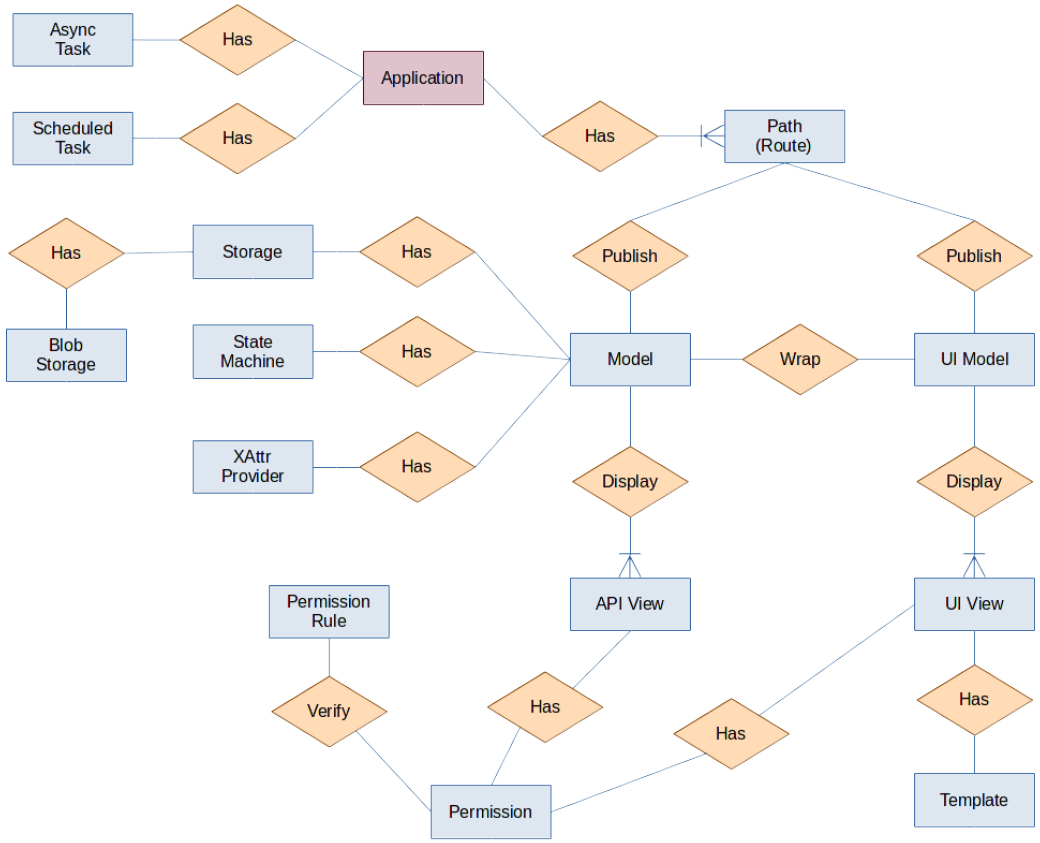
- Views can be inherited by sub-models. You can create mixin interface classes and attach the views to it. Any models which inherits from the mixin will get the view.
- Views follows model and its sub-models. Whatever path the model is mounted, the views for the model will follow.

2.1.1 Underlying Frameworks & Libraries

MorpCC was built using the following frameworks & libraries to power it. So if you need more detailed documentation on specific components that are not covered here, please head to their documentation

- Morepath - Python web framework with superpowers
- MorpFW - A REST API web framework built on top of Morepath.
- Reg - Dispatching library similar to Zope Component Architecture
- Dectate - Decorator based configuration system
- SQLAlchemy - SQL abtraction library & ORM, used as the default storage engine.
- Chameleon - Templating engine that implements Zope Template Attribute Language (TAL).
- Rulez - JSON based rules engine
- PyTransitions - State machine engine for Python.
- Celery - Distributed task queue and job scheduling library for Python.
- Gentelella - Bootstrap 3 based admin template.

2.1.2 Entity Model



2.2 Quick Start Tutorial

2.2.1 Dependencies

MorpCC requires following services for it to function correctly:

- postgresql database. 3 databases are needed, for following purpose:
  - main database - for MorpCC application tables
  - warehouse database - MorpCC provides a Through-The-Web (TTW) data model manager which allows creation of tables and managing data using the Web UI. Tables created by this feature will store its data in this database.
  - cache database - used by `beaker` for caching and session
- rabbitmq message queue - used by background processing engine

2.2.2 Bootstrapping new project

MorpCC requires Python 3.7 or newer to run. Python 3.6 is also supported but you will need to install `dataclasses` backport into your environment.

The recommended way to install morpfw is to use buildout, skeleton that is generated using `mfw-template`. Please head to mfw-template documentation for tutorial.

2.2.3 Bootstrapping without `mfw-template`

If you prefer to use `virtualenv`, or other methods, you can follow these steps.

First, lets get `morpfw` & `morpcc` installed

```
$ pip install morpfw morpcc
```

If you are using buildout, version locks files are available at `mfw_workspace` repository: https://github.com/morpframework/mfw_workspace/tree/master/versions

Lets create an `app.py`.

```
import morpcc
import morpcc.permission as ccperm
import morpfw
from morpfw.authz.pas import DefaultAuthzPolicy
from morpfw.crud import permission as crudperm


class AppRoot(morpcc.Root):
    pass


class App(morpcc.App):
    pass


@App.path(model=AppRoot, path="/")
def get_approot(request):
    return AppRoot(request)
```

(continues on next page)

(continued from previous page)

```
@App.template_directory()
def get_template_directory():
    return "templates"
```

morpcc is built on morpfw which boot up application using a settings.yml file, so lets create one. You will need a fernet key which have to be generated using following python code:

```
$ python -c "from cryptography.fernet import Fernet; print(Fernet.generate_key().
↪decode())"
```

Then lets create a settings.yml

```
application:
   title: My First App
   class: app:App
   factory: morpcc.app:create_morpcc_app

configuration:
   morpfw.authn.policy: morpcc.app:AuthnPolicy
   morpfw.secret.fernet_key: '<fernet-key>'
   morpfw.storage.sqlstorage.dburi:  'postgresql://postgres:postgres@localhost:5432/
↪morpcc'
   morpfw.storage.sqlstorage.dburi.warehouse: 'postgresql://
↪postgres:postgres@localhost:5432/morpcc_warehouse'
   morpfw.blobstorage.uri: 'fsblob://%(here)s/blobstorage'
   morpfw.beaker.session.type: ext:database
   morpfw.beaker.session.url: 'postgresql://postgres:postgres@localhost:5432/morpcc_
↪cache'
   morpfw.beaker.cache.type: ext:database
   morpfw.beaker.cache.url: 'postgresql://postgres:postgres@localhost:5432/morpcc_
↪cache'
   morpfw.celery:
      broker_url: 'amqps://guest:guest@localhost:5671/'
      result_backend: 'db+postgresql://postgres:postgres@localhost:5432/morpcc_cache'
```

You will then need to initialize database migration:

```
$ morpfw migration init migrations
```

Default alembic Afterwards, you can then start the application using:

```
$ morpfw -s settings.yml register-admin -u admin -e admin@localhost.local
$ morpfw -s settings.yml start
```

This will start your project at http://localhost:5000/

2.2.4 Understanding core framework functionalities

MorpCC is built on top of Morepath, so we suggest you head to Morepath Documentation for guide on how to register your own views, etc.

CRUD engine, resource type system and REST API engine for MorpCC is provided by MorpFW. Head to MorpFW documentation to understand more on the type system used in MorpCC.

The templating language used is TAL, and we extensively use METAL for template inheritance. Head to Chameleon TAL/METAL Language Reference and Zope Page Template Reference to understand more about TAL and METAL.

2.3 Community

Our community is still in infancy, and we hangout on Discord. Come join us at MorpFW Discord Server if you have any questions.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search