pynta Documentation

Release dev

Serge Matveenko

February 11, 2015

Contents

Pynta is web framework written in Python.

Contents:

CHAPTER 1

Concepts

It is usually a kind of deal for framework developer between end application flexibility and interoperability when planning framework restrictions, project structure agreements, embedded features, etc.

Pynta is not trying to keep balancing frankly. it is flixible as much as it possible to be. You are able to plug anything on almost any level of abstraction or seamlessly connect to any other framework or use any desired storage or whatever technology you want.

But Pynta is not microframework. It has several abstractions that allow it to be flexible. There are interfaces that allow one to develop his own components to rapidly implement any imaginable feature.

1.1 Components

Pynta consits of the following logical components:

- Configuration
- Application base
- Storage
- Templates
- URLs resolver

Each of the components (except for urls resolver for now) is built on an abstract class or like application base (*PyntaApp* class) is the basis for the developer to design his project. Abstract classes stand for interface that developer could implement to allow using of any storage, template engine, etc.

1.1.1 Configuration

#TODO

1.1.2 Application base

#TODO

1.1.3 Storage

#TODO

1.1.4 Templates

#TODO

1.1.5 URLs resolver

#TODO

CHAPTER 2

Pynta reference

2.1 Configuration

CHAPTER 3

Basic Principals

Pynta will always be:

- flexible
- scalable
- complex
- PEP-oriented
- on the edge

CHAPTER 4

Goals

- Provide tools for building low volume products.
- Allow developer to choose any abstraction level to work on.
- Support most of the popular technologies out of the box.
- Allow developer to easy add support for any desired technology.
- Be compatible with any future technology.

CHAPTER 5

Main Features

- WSGI compatible on most levels (almost everything is WSGI app).
- Simple and powerful url mapping mechanism (regexes with host and url matching).
- One app could handle a number of urls for different actions.
- CRUD support out of the box (using actions mechanism).
- Different storages support (relational and non-relational):
- Anydbm,
- MongoDB via pymongo and/or MongoEngine,
- (planned) Relational via SQLAlchemy,
- (planned) Forms processing support (storage integrated).
- Sessions support.
- (planned) Authentication support.
- (planned) Flexible logging mechanism.
- (planned) Generic administration interface.
- (planned) Stateful apps support.
- (planned) Server side interface toolkit (twitter's Bootstrap and jQuery integration).

CHAPTER 6

Possible Features

These features are still subject for discuss

- Authorization support (separated from authentication).
- Django apps support.
- Visual editor (using interface toolkit).

CHAPTER 7

Indices and tables

- *genindex*
- *modindex*
- *search*