

FastAPI, Pydantic and GPTS



GenomOncology.com

FastAPI



August 2004

David Heinemeier Hansson
Basecamp & 37 Signals



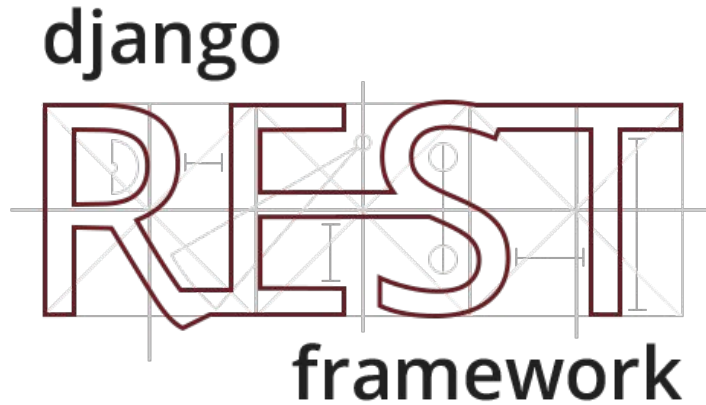
July 2005

Adrian Holovaty
Simon Willison
Jacob Kaplan Moss
Lawrence Journal World



July 2010

Armin Ronacher
Pallets Project (Pocoo)



Built on Django Web Framework

- 2011: Version 1
- 2012: Version 2
- 2014: Kickstarter
- 2015: Version 3

Key Features:

- Browsable API
- Serialization
- Authentication
- Viewsets
- Pagination

Browsable APIs

pet

Everything about your Pets

Find out more

POST

/pet

Add a new pet to the store

Parameters

Name	Description
body <i>(body)</i>	<div>Pet object that needs to be added to the store</div> <div>model - application/json - text/xml - text/html - octal-messagepack</div> <pre>{ foo: 'bar', baz: 123, flags:['f1', 'f1', 'f3'] }</pre>

Responses

Code	Description
405	Invalid input

PUT

/pet

Update an existing pet

GET

/pet/findByStatus

Finds Pets by status

Django REST framework v2.0.0

admin

User List

OPTIONS GET

API endpoint that represents a list of users.

GET /users/

HTTP 200 OK

Vary: Accept

Content-Type: text/html

Allow: GET, POST, HEAD, OPTIONS

```
{
  "count": 2,
  "next": null,
  "previous": null,
  "results": [
    {
      "email": "admin@example.com",
      "groups": [],
      "url": "http://127.0.0.1:8000/users/1/",
      "username": "admin"
    },
    {
      "email": "tom@example.com",
      "groups": [],
      "url": "http://127.0.0.1:8000/users/2/",
      "username": "tom"
    }
  ]
}
```

Username

Email

Groups

POST

PEP 3156 – Asynchronous IO Support Rebooted: the “asyncio” Module

Author: Guido van Rossum <guido at python.org>

BDFL-Delegate: Antoine Pitrou <antoine at python.org>

Discussions-To: python-tulip@googlegroups.com

Status: Final

Type: Standards Track

Created: 12-Dec-2012

Python-Version: 3.3

Post-History: 21-Dec-2012

Replaces: [3153](#)

Resolution: [Python-Dev message](#)

<https://peps.python.org/pep-3156/>

PEP 492 – Coroutines with async and await syntax

Author: Yury Selivanov <yury at edgedb.com>

Discussions-To: [Python-Dev list](#)

Status: Final

Type: Standards Track

Created: 09-Apr-2015

Python-Version: 3.5

Post-History: 17-Apr-2015, 21-Apr-2015, 27-Apr-2015, 29-Apr-2015, 05-May-2015

```
async def read_data(db):
    data = await db.fetch('SELECT ...')
    ...
```

<https://peps.python.org/pep-0492/>



Spring 2017
Async Framework
Paused 2019



Summer 2017
ASGI Server



Summer 2018
Async Framework



HTTPX

Summer 2019
Async HTTP
Requests

<https://www.encode.io/projects/>



2018 - Development Started

Key Features:

- Built on Starlette & Pydantic
- Asyncio, Annotations Support
- OpenAPI and JSON Schema
- Pydantic Validation/Serialization
- **Great** documentation
- High-Performance (for Python)
- Very popular

Key Gaps:

- No admin UI
- No ORM module

<https://fastapi.tiangolo.com/>


```
from typing import Union

from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"Hello": "World"}

@app.get("/items/{item_id}")
def read_item(item_id: int, q: Union[str, None]):
    return {"item_id": item_id, "q": q}
```

Fast API 0.100.0 0.100.0

logonapi.json

default

GET

/items/{item_id} Read Item Get

Parameters

Try it out

Name

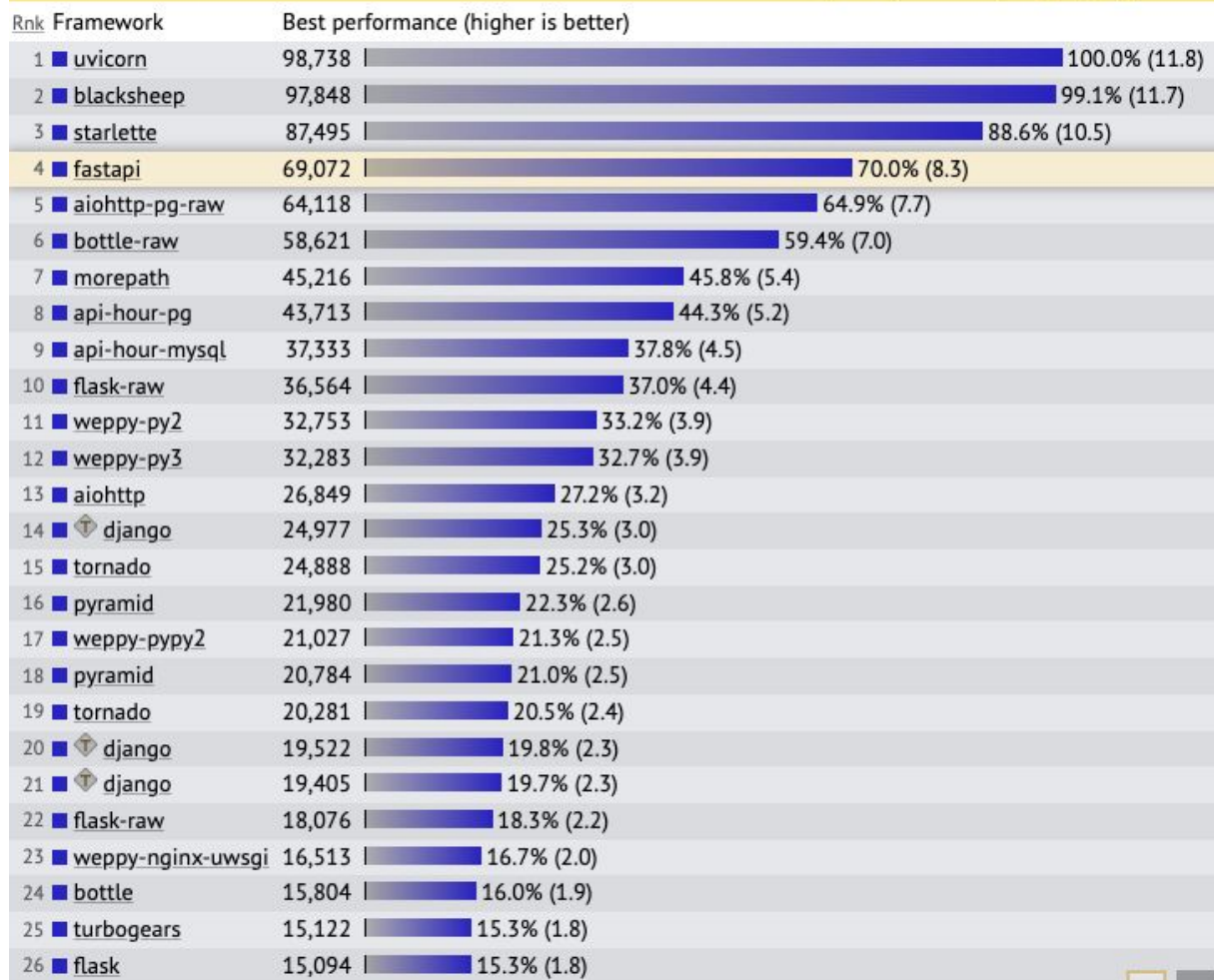
Description

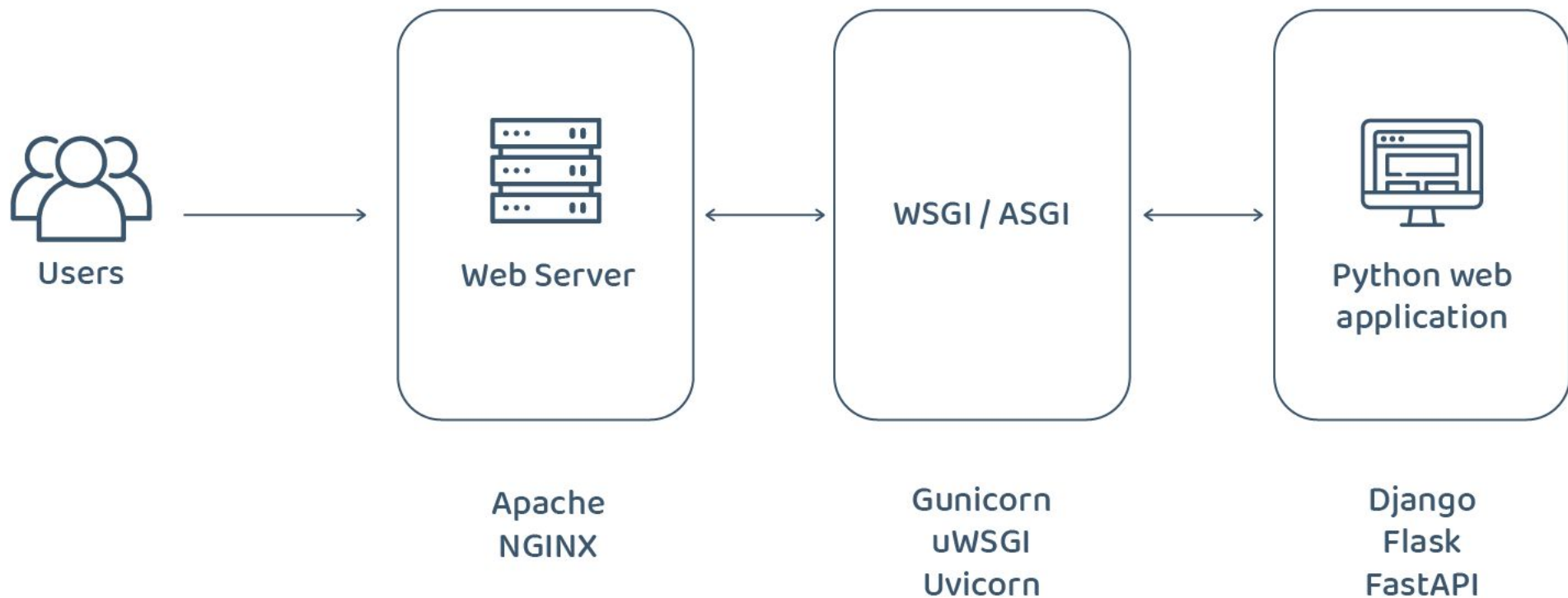
item_id * required

integer
(path)

q

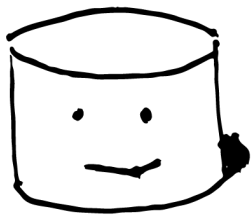
string
(query)





source: <https://www.portmind.com/blog/serving-ml-model-as-an-api-part-1>

Pydantic



marshmallow

November 2013

Steven Loria

Framework-agnostic
validation and serialization
library.



April 2015

Hynek Schlawack

Pre-cursor to dataclasses
with support for validation,
converters, extensibility.



June 2017

Eric Smith

PEP 557
*"mutable namedtuples with
defaults"*

PEP 484 – Type Hints

Author: Guido van Rossum <guido at python.org>, Jukka Lehtosalo <jukka.lehtosalo at iki.fi>, Łukasz Langa <lukasz at python.org>

BDFL-Delegate: Mark Shannon

Discussions-To: [Python-Dev list](#)

Status: Final

Type: Standards Track

Topic: [Typing](#)

Created: 29-Sep-2014

Python-Version: 3.5

Post-History: 16-Jan-2015, 20-Mar-2015, 17-Apr-2015, 20-May-2015, 22-May-2015

Resolution: [Python-Dev message](#)

<https://peps.python.org/pep-0484/>

PEP 526 – Syntax for Variable Annotations

Author: Ryan Gonzalez <rymg19 at gmail.com>, Philip House <phouse512 at gmail.com>, Ivan Levkivskyi <levkivskyi at gmail.com>, Lisa Roach <lisaroach14 at gmail.com>, Guido van Rossum <guido at python.org>

Status: Final

Type: Standards Track

Topic: [Typing](#)

Created: 09-Aug-2016

Python-Version: 3.6

Post-History: 30-Aug-2016, 02-Sep-2016

Resolution: [Python-Dev message](#)

<https://peps.python.org/pep-0526/>



Pydantic

May 2017: Development Started

October 2019: Pydantic 1.0

July 2023: Pydantic 2.0

Key Features:

- Data Validation via Type Hints
- Serialization
- JSON Schema
- Dataclass, Typed Dict support
- 2.0 Rewritten in Rust
- Ecosystem (LangChain, etc.)

Raised funding, creating FastUI

```

from datetime import datetime

from pydantic import BaseModel, PositiveInt

class User(BaseModel):
    id: int
    name: str = 'John Doe'
    signup_ts: datetime | None
    tastes: dict[str, PositiveInt]

external_data = {
    'id': 123,
    'signup_ts': '2019-06-01 12:22',
    'tastes': {
        'wine': 9,
        'cheese': 7,
        'cabbage': 1,
    },
}

user = User(**external_data)

```

```

print(user.id)
#> 123
print(user.model_dump())
"""
{
    'id': 123,
    'name': 'John Doe',
    'signup_ts': datetime.datetime(2019, 6, 1, 12, 22),
    'tastes': {'wine': 9, 'cheese': 7, 'cabbage': 1},
}
"""

```

Pydantic 2.0 just released! An overview of the most popular data validation python package

Learn what has changed in the recent 2.0 release of Pydantic



Logan Kilpatrick · Follow

5 min read · Jul 3


```
external_data = {'id': 'not an int', 'tastes': {}}

try:
    User(**external_data)
except ValidationError as e:
    print(e.errors())
    """
    [
        {
            'type': 'int_parsing',
            'loc': ('id',),
            'msg': 'Input should be a valid integer, unable to parse string as a',
            'input': 'not an int',
            'url': 'https://errors.pydantic.dev/2/v/int_parsing',
        },
        {
            'type': 'missing',
            'loc': ('signup_ts',),
            'msg': 'Field required',
            'input': {'id': 'not an int', 'tastes': {}},
            'url': 'https://errors.pydantic.dev/2/v/missing',
        },
    ]
    """
```

```
# msgspec 0.16.0
```

- encode: 176.2 us
- decode: 487.6 us
- roundtrip: 663.9 us

```
# pydantic 2.0b3
```

- encode: 3667.3 us (20.8x slower)
- decode: 5763.2 us (11.8x slower)
- roundtrip: 9430.5 us (14.2x slower)

```
# pydantic 1.10.9
```

- encode: 16389.6 us (93.0x slower)
- decode: 65533.8 us (134.4x slower)
- roundtrip: 81923.4 us (123.4x slower)



```
from typing import Union

from fastapi import FastAPI
from pydantic import BaseModel
```

```
app = FastAPI()
```

```
class Item(BaseModel):
    name: str
    price: float
    is_offer: Union[bool, None] = None
```

```
@app.get("/")
def read_root():
    return {"Hello": "World"}
```

```
@app.get("/items/{item_id}")
def read_item(item_id: int, q: Union[str, None] = None):
    return {"item_id": item_id, "q": q}
```

```
@app.put("/items/{item_id}")
def update_item(item_id: int, item: Item):
    return {"item_name": item.name, "item_id": item_id}
```

PUT

/items/{item_id} Save Item Put

Try it out

Parameters

Name

Description

item_id ★ required

integer

(path)

Request body required

application/json



Example Value | Schema

```
{
  "name": "string",
  "price": 0,
  "is_offer": true
}
```



2021 - Initial Release

Pydantic-based ORM

Built on top of SQLAlchemy

“Sensible” Defaults

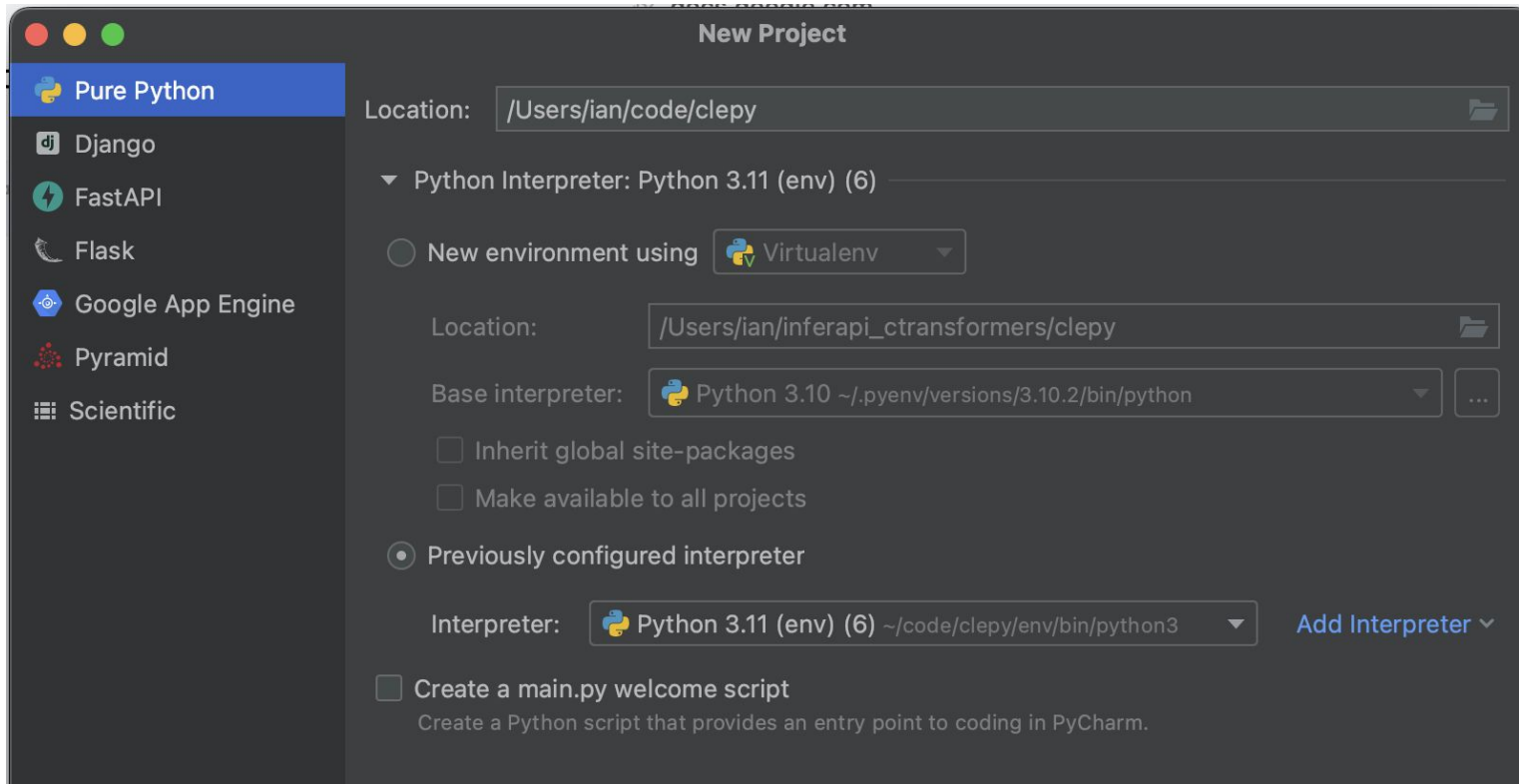
Really raw

Documentation not complete

Just released Pydantic V2 support

<https://sqlmodel.tiangolo.com/>

Project Setup



```
# pyproject.toml
```

```
dependencies = [  
    "fastapi == 0.104.1",  
    "sqlmodel == 0.0.14",  
]  
dynamic = ["version"]  
  
[project.optional-dependencies]  
test = [  
    "pytest",  
    "coverage[toml]",  
    "uvicorn",  
]
```

```
$ pip install ".[test]"  
$ pip freeze
```

```
click==8.1.7  
coverage==7.3.2  
fastapi==0.104.1  
h11==0.14.0  
idna==3.6  
iniconfig==2.0.0  
packaging==23.2  
pluggy==1.3.0  
pydantic==2.5.2  
pydantic_core==2.14.5  
pytest==7.4.3  
sniffio==1.3.0  
SQLAlchemy==2.0.23  
sqlmodel==0.0.14  
starlette==0.27.0  
typing_extensions==4.9.0  
uvicorn==0.24.0.post1
```

3 src/clepy/__init__.py

@@ -1,4 +1,7 @@

```
1 + from .app import app
2 +
3   __version__ = "0.0.1"
4
5   __all__ = (
6 +     "app",
7   )
```

8 src/clepy/app.py

@@ -0,0 +1,8 @@

```
1 + from fastapi import FastAPI
2 +
3 + app = FastAPI()
4 +
5 +
6 + @app.get("/")
7 + async def root():
8 +     return {"message": "Hello World"}
```



```
(env) ian@Ians-MBP clepy % make api
. ./activate.sh && uvicorn clepy:app --reload --port 8010
INFO: Will watch for changes in these directories: ['/Users/ian/code/clepy']
INFO: Uvicorn running on http://127.0.0.1:8010 (Press CTRL+C to quit)
INFO: Started reloader process [85176] using StatReload
INFO: Started server process [85178]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:55840 - "GET /docs HTTP/1.1" 200 OK
INFO: 127.0.0.1:55840 - "GET /openapi.json HTTP/1.1" 200 OK
INFO: 127.0.0.1:55840 - "GET / HTTP/1.1" 200 OK
```

http://127.0.0.1:8010



```
{
  "message": "Hello World"
}
```


✓ 11 src/clepy/app.py

... @@ -1,6 +1,15 @@

1 1 from fastapi import FastAPI

2 2

3 - app = FastAPI()

3 + app = FastAPI(

4 + title="ClePy 12/11",

5 + summary="Demonstration of FastAPI, Pydantic, and

6 + description="""

7 + Supports `markdown` including:

8 + - Bullets

9 + - And other things.

10 + """`.strip()`,

11 + version="0.1.0.beta12345",

12 +)

4 13

5 14

6 15 @app.get("/")



ClePy 12/11

0.1.0.beta12345

OAS 3.1

/openapi.json

Demonstration of FastAPI, Pydantic, and OpenAI GPTS.

Supports `markdown` including:

- Bullets
- And other things.

default

GET

/ Root

Parameters

Cancel

No parameters

Execute

Clear

PyPI Data Demo

PyPI Data

The contents of the PyPI JSON API for all packages, updated every 12 hours

Why?

Fetching bulk data from the PyPI API in bulk is non-trivial, and using the [BigQuery dataset](#) requires using BigQuery. The entire package dataset is not large and easily fits into the memory of most developer machines, so it's much more fluid to explore the data with Pandas than the heavyweight (and sometimes expensive) BigQuery.

Release data

Via sqlite database

Every day the contents of this repository is bundled into a sqlite database and added as a [Github Release](#). The schema can be found [in schema.sql](#) and contains release + download data, without classifiers or readme information. An example:

Latest Release

Latest

Automated bundling of release data into a sqlite database.

Updated on 2023-12-11

See the readme for more information.

▼ Assets 4

 pypi-data.parquet	743 MB	5 hours ago
 pypi-data.sqlite.gz	1.04 GB	4 minutes ago

```
import sqlite3

conn = sqlite3.connect("./pypi-data.sqlite")

cur = conn.cursor()

cur.execute("select count(*) from projects")
<sqlite3.Cursor at 0x105a7c7c0>

_.fetchone()
(5288639,)
```

```
cur.execute("select count(*) from urls")
<sqlite3.Cursor at 0x105a7c7c0>

_.fetchone()
(10123064,)
```

<https://github.com/pypi-data/pypi-json-data>

```
CREATE table projects(  
    id integer not null primary key,  
    name text,  
    version text,  
    author text,  
    author_email text,  
    home_page text,  
    license text,  
    maintainer text,  
    maintainer_email text,  
    package_url text,  
    platform text,  
    project_url text,  
    requires_python text,  
    summary text,  
    yanked int,  
    yanked_reason text,  
    classifiers text,  
    requires_dist text,  
    UNIQUE(name, version)  
);
```

```
CREATE table urls(  
    project_id int,  
    url text,  
    upload_time text,  
    package_type text,  
    python_version text,  
    requires_python text,  
    size int,  
    yanked int,  
    yanked_reason text,  
    foreign key(project_id) REFERENCES projects(id)  
);
```

<https://github.com/pypi-data/pypi-json-data/blob/main/scripts/schema.sql>

```
from typing import Optional, List
from sqlmodel import Field, SQLModel, Relationship
```

```
class Project(SQLModel, table=True):
    __tablename__ = "projects"

    id: int = Field(default=None, primary_key=True)
    name: str = Field(index=True)
    version: str
    author: Optional[str] = None
    author_email: Optional[str] = None
    home_page: Optional[str] = None
    license: Optional[str] = None
    maintainer: Optional[str] = None
    maintainer_email: Optional[str] = None
    package_url: Optional[str] = None
    platform: Optional[str] = None
    project_url: Optional[str] = None
    requires_python: Optional[str] = None
    summary: Optional[str] = None
    yanked: int
    yanked_reason: Optional[str] = None
    classifiers: Optional[str] = None
    requires_dist: Optional[str] = None
```

```
    urls: List["URL"] =
Relationship(back_populates="project")
```

```
class URL(SQLModel, table=True):
    __tablename__ = "urls"

    project_id: int = Field(
        foreign_key="projects.id",
        primary_key=True,
    )
    url: str = Field(primary_key=True)
    upload_time: str
    package_type: Optional[str] = None
    python_version: Optional[str] = None
    requires_python: Optional[str] = None
    size: int
    yanked: int
    yanked_reason: Optional[str] = None

    project: Project = Relationship(back_populates="urls")
```

```

from pathlib import Path
from typing import Type

from sqlalchemy import create_engine, Session, select, func, SQLModel

from clepy import Project

_engine = None

def get_engine():
    global _engine
    if _engine is None:
        db_path = Path( file ).parent / "pypi-data.sqlite"
        engine = create_engine( f"sqlite:/// {db_path}" )
    return _engine

def count_table( table: Type[SQLModel] ) -> int:
    engine = get_engine()
    with Session(engine) as session:
        statement = select(func.count()).select_from(table)
        result = session.exec(statement).one()
    return result

def test_counts():
    n_projects = db.count_table(Project)
    assert n_projects == 5288639

    n_urls = db.count_table(URL)
    assert n_urls == 10123064

```

```
def get_project(name: str, version: str) -> Project:
    with Session(get_engine()) as session:
        statement = select(Project).where(
            Project.name == name,
            Project.version == version,
        )
        project = session.exec(statement).first()

        # todo: figure out how to handle relationships..
        assert project.urls
        return project


def test_get_project():
    project = db.get_project(name="fastapi", version="0.104.1")

    assert project.name == "fastapi"
    assert project.author_email == "Sebastián Ramírez <tiangolo@gmail.com>"
    assert project.summary == (
        "FastAPI framework, high performance, easy to "
        "learn, fast to code, ready for production"
    )
    assert len(project.urls) == 2
```

```
from fastapi import FastAPI, Path
from clepy import Project, db

app = FastAPI(title="ClePy 12/11")

@app.get("/{name}/{version}/")
async def get_project(
    name: str = Path(...),
    version: str = Path(...),
) -> Project:
    return db.get_project(name, version)
```

default

GET `//{name}/{version}/` Get Project

Parameters Cancel

Name	Description
name * required string (path)	<input type="text" value="fastapi"/>
version * required string (path)	<input type="text" value="0.104.1"/>

Execute Clear



{

Raw

Parsed

```

"home_page": "",
"summary": "FastAPI framework, high performance, easy to learn, fast to code, ready for production",
"license": "",
"yanked": 0,
"author": "",
"maintainer": "",
"yanked_reason": "",
"version": "0.104.1",
"id": 1602347,
"maintainer_email": "",
"classifiers": ["Development Status :: 4 - Beta", "Environment :: Web Environment", "Framework :: AsyncIO", "Framework :: FastAPI", "Framework :: Pydantic", "Framework :: Pydantic :: 1", "Intended Audience :: Developers", "Intended Audience :: Information Technology", "Intended Audience :: System Administrators", "License :: OSI Approved :: MIT License", "Operating System :: OS Independent", "Programming Language :: Python", "Programming Language :: Python :: 3", "Programming Language :: Python :: 3 :: Only", "Programming Language :: Python :: 3.10", "Programming Language :: Python :: 3.11", "Programming Language :: Python :: 3.8", "Programming Language :: Python :: 3.9", "Topic :: Internet", "Topic :: Internet :: WWW/HTTP", "Topic :: Internet :: WWW/HTTP :: HTTP Servers", "Topic :: Software Development", "Topic :: Software Development :: Libraries", "Topic :: Software Development :: Libraries :: Application Frameworks", "Topic :: Software Development :: Libraries :: Python Modules", "Typing :: Typed"],
"name": "fastapi",
"package_url": "https://pypi.org/project/fastapi/",
"requires_dist": ["anyio<4.0.0,>=3.7.1", "pydantic!=1.8,!>=1.8.1,!>=2.0.0,!>=2.0.1,!>=2.1.0,<3.0.0,>=1.7.4", "starlette<0.28.0,>=0.27.0", "typing-extensions>=4.8.0", "email-validator>=2.0.0; extra == 'all'", "httpx>=0.23.0; extra == 'all'", "itsdangerous>=1.1.0; extra == 'all'", "jinja2>=2.11.2; extra == 'all'", "orjson>=3.2.1; extra == 'all'", "pydantic-extra-types>=2.0.0; extra == 'all'", "pydantic-settings>=2.0.0; extra == 'all'", "python-multipart>=0.0.5; extra == 'all'", "pyyaml>=5.3.1; extra == 'all'", "ujson!=4.0.2,!>=4.1.0,!>=4.2.0,!>=4.3.0,!>=5.0.0,!>=5.1.0,>=4.0.1; extra == 'all'", "uvicorn[standard]>=0.12.0; extra == 'all'"],

```

Opportunities for more deserialization.

```
import json
from typing import Optional, List
from pydantic import field_serializer
from sqlmodel import Field, SQLModel, Relationship
```

```
class Project(SQLModel, table=True):
    __tablename__ = "projects"
```

```
...
    classifiers: Optional[str] = None
    requires_dist: Optional[str] = None
```

```
@field_serializer('classifiers', 'requires_dist')
def serialize_json(self, value):
    if isinstance(value, str):
        return json.loads(value)
    return value
```

i http://127.0.0.1:8010/fastapi/0.104.1/

```
{
  "project_url": "https://pypi.org/project/fastapi/",
  "requires_python": ">=3.8",
  "home_page": "",
  "summary": "FastAPI framework, high performance, easy to learn, fast to code, ready for production",
  "license": "",
  "yanked": 0,
  "author": "",
  "maintainer": "",
  "yanked_reason": "",
  "version": "0.104.1",
  "id": 1602347,
  "maintainer_email": "",
  "classifiers": [
    "Development Status :: 4 - Beta",
    "Environment :: Web Environment",
    "Framework :: AsyncIO",
    "Framework :: FastAPI",
    "Framework :: Pydantic",
    "Framework :: Pydantic :: 1",
    "Intended Audience :: Developers",
    "Intended Audience :: Information Technology",
    "Intended Audience :: System Administrators",
    "License :: OSI Approved :: MIT License",
    "Operating System :: OS Independent",
    "Programming Language :: Python",
    "Programming Language :: Python :: 3",
    "Programming Language :: Python :: 3 :: Only",
    "Programming Language :: Python :: 3.10",
    "Programming Language :: Python :: 3.11",
    "Programming Language :: Python :: 3.8",
    "Programming Language :: Python :: 3.9",
    "Topic :: Internet",
    "Topic :: Internet :: WWW/HTTP",
    "Topic :: Internet :: WWW/HTTP :: HTTP Servers",
    "Topic :: Software Development",
    "Topic :: Software Development :: Libraries",
    "Topic :: Software Development :: Libraries :: Application Frameworks",
    "Topic :: Software Development :: Libraries :: Python Modules",
    "Typing :: Typed"
  ],
  "name": "fastapi",
  "package_url": "https://pypi.org/project/fastapi/",
  "requires_dist": [
    "anyio<4.0.0,>=3.7.1",
    "pydantic!=1.8,!<1.8.1,!>2.0.0,!<2.0.1,!>2.1.0,<3.0.0,>=1.7.4",
    "starlette<0.28.0,>=0.27.0",
    "typing-extensions>=4.8.0",
    "email-validator>=2.0.0; extra == 'all'",
  ]
}
```

```

from pydantic import BaseModel, ValidationError, field_validator

class UserModel(BaseModel):
    id: int
    name: str

    @field_validator('name')
    @classmethod
    def name_must_contain_space(cls, v: str) -> str:
        if ' ' not in v:
            raise ValueError('must contain a space')
        return v.title()

print(UserModel(id=1, name='John Doe'))
#> id=1 name='John Doe'

try:
    UserModel(id=1, name='samuel')
except ValidationError as e:
    print(e)
    """
    1 validation error for UserModel
    name
      Value error, must contain a space [type=value_error,
input value='samuel', input_type=str]
    """

```

<https://docs.pydantic.dev/latest/concepts/validators/#field-validators>

```

from typing import Any

from pydantic import BaseModel, ValidationError, model_validator

class UserModel(BaseModel):
    username: str
    password1: str
    password2: str

    @model_validator(mode='after')
    def check_passwords_match(self) -> 'UserModel':
        pw1 = self.password1
        pw2 = self.password2
        if pw1 is not None and pw2 is not None and pw1 != pw2:
            raise ValueError('passwords do not match')
        return self

try:
    UserModel(username='scolvin', password1='zxcvbn',
password2='zxcvbn2')
except ValidationError as e:
    print(e)
    """
    1 validation error for UserModel
      Value error, passwords do not match [type=value error,
input value={'username': 'scolvin', '... 'password2': 'zxcvbn2'}],
input type=dict]
    """

```

<https://docs.pydantic.dev/latest/concepts/validators/#model-validators>

i http://127.0.0.1:8010/fastapi/0.104.1/

```
{
  "project_url": "https://pypi.org/project/fastapi/",
  "requires_python": ">=3.8",
  "home_page": "",
  "summary": "FastAPI framework, high performance, easy to learn, fast to co
  "license": "",
  "yanked": 0,
  "author": "",
  "maintainer": "",
  "yanked_reason": "",
  "version": "0.104.1",
  "id": 1602347,
  "maintainer_email": "",
  "classifiers": [...], // 26 items
  "name": "fastapi",
  "package_url": "https://pypi.org/project/fastapi/",
  "requires_dist": [...], // 15 items
  "author_email": "Sebastián Ramírez <tiangolo@gmail.com>",
  "platform": ""
}
```

Missing URLs

```
import json
from typing import Optional, List

from pydantic import field_serializer
from sqlmodel import Field, SQLModel, Relationship
```

```
class ProjectBase(SQLModel):
    id: int = Field(default=None, primary_key=True)
    name: str = Field(index=True)
    version: str
    author: Optional[str] = None
    author_email: Optional[str] = None
    home_page: Optional[str] = None
    license: Optional[str] = None
    maintainer: Optional[str] = None
    maintainer_email: Optional[str] = None
    package_url: Optional[str] = None
    platform: Optional[str] = None
    project_url: Optional[str] = None
    requires_python: Optional[str] = None
    summary: Optional[str] = None
    yanked: int
    yanked_reason: Optional[str] = None
    classifiers: Optional[str] = None
    requires_dist: Optional[str] = None
```

```
class Project(ProjectBase, table=True):
    tablename = "projects"
    urls: List["URL"] = Relationship(back_populates="project")

    class Config:
        unique_together = [("name", "version")]

class ProjectWithURLs(ProjectBase):
    urls: List[URL] = Field(default_factory=list)
```

```
{
  "id": 1602347,
  "name": "fastapi",
  "version": "0.104.1",
  "author": "",
  "author_email": "Sebastián Ramírez <tiangolo@gmail.com>",
  "home_page": "",
  "license": "",
  "maintainer": "",
  "maintainer_email": "",
  "package_url": "https://pypi.org/project/fastapi/",
  "platform": "",
  "project_url": "https://pypi.org/project/fastapi/",
  "requires_python": ">=3.8",
  "summary": "FastAPI framework, high performance, easy to learn, fast to code, ready for production",
  "yanked": 0,
  "yanked_reason": "",
  "classifiers": [], // 26 items
  "requires_dist": [], // 15 items
  "urls": [
    {
      "upload_time": "2023-10-30T10:07:35.636232Z",
      "url": "https://files.pythonhosted.org/packages/f3/4f/0ce34195b63240b6693086496c9bab4ef239991121843c/any.whl",
      "requires_python": ">=3.8",
      "yanked": 0,
      "project_id": 1602347,
      "package_type": "bdist_wheel",
      "python_version": "py3",
      "size": 92862,
      "yanked_reason": ""
    },
    {
      "upload_time": "2023-10-30T10:07:39.105972Z",
      "url": "https://files.pythonhosted.org/packages/d9/d8/002e0ba7cf848a981b3ee92aaf5aa396c5700b0d7dec5c",
      "requires_python": ">=3.8",
      "yanked": 0,
      "project_id": 1602347,
      "package_type": "sdist",
      "python_version": "source",
      "size": 11295150,
      "yanked_reason": ""
    }
  ]
}
```

Added URLs

ClePy 12/11

0.1.0

OAS 3.1

/openapi.json

default

GET

/ {name}/{version}/ Get Project

Parameters

Name	Description
------	-------------

name * required

string
(path)

sqlmodel

version * required

string
(path)

0.0.14

http://127.0.0.1:8010/openapi.json

```
{
  "openapi": "3.1.0",
  "info": {
    "title": "ClePy 12/11",
    "version": "0.1.0"
  },
  "paths": {
    "/{name}/{version}": {
      "get": {
        "summary": "Get Project",
        "operationId": "get_project_name_version_get",
        "parameters": [
          {
            "name": "name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string",
              "title": "Name"
            }
          },
          {
            "name": "version",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string",
              "title": "Version"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "Successful Response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/ProjectWithURLs"
                }
              }
            }
          }
        }
      }
    }
  }
}
```


Use the Dependency 📌

Now let's make FastAPI execute a dependency and get its value in the *path operation*.

We import `Depends()` from `fastapi`. Then we use it in the *path operation function* in a **parameter**, the same way we declared parameters to get JSON bodies, path parameters, etc.

```
from typing import List, Optional

from fastapi import Depends, FastAPI, HTTPException, Query
from sqlmodel import Field, Session, SQLModel, create_engine, select

# Code here omitted 🙋

def get_session():
    with Session(engine) as session:
        yield session

# Code here omitted 🙋

@app.post("/heroes/", response_model=HeroRead)
def create_hero(*, session: Session = Depends(get_session), hero: HeroCreate):
    db_hero = Hero.model_validate(hero)
    session.add(db_hero)
    session.commit()
    session.refresh(db_hero)
    return db_hero

# Code below omitted 🙋
```

GPTS Demo

Introducing GPTs

You can now create custom versions of ChatGPT that combine instructions, extra knowledge, and any combination of skills.



GPTs are a new platform by OpenAI for developing AI-based applications, offering customization and integration capabilities. Key features include:

1. **Custom Configuration:** Developers can set specific parameters, such as the AI's name, description, and instructions, to create a GPT that aligns with particular use cases.
2. **Knowledge Enhancement:** The ability to upload files or incorporate additional knowledge enables a more specialized and informed AI response system.
3. **Capability Selection:** Developers can choose from existing capabilities like web browsing, DALL-E image generation, and code interpretation, to suit the application's requirements.
4. **API Integration and Action Design:** GPTs can be connected to external APIs, allowing for real-world interactions. This integration, coupled with the action design feature, facilitates actions like database queries, email interactions, or e-commerce transactions, and grants developers greater control over the model and how their APIs are called.

<https://openai.com/blog/introducing-gpts>



You

What is the latest version of it that you know about without searching the web?



ChatGPT

As of my last update in April 2023, the latest version of SQLAlchemy that I'm aware of was 0.0.4. However, software versions can change frequently with updates and improvements, so there might be a newer version available now. For the most current version, it's always best to check the official SQLAlchemy documentation or its repository on GitHub.



Search projects



Help

Sponsors

Log in

Register

sqlmodel 0.0.14



Latest version

`pip install sqlmodel`



Released: Dec 4, 2023

Files

main

sqlmodel / docs / release-notes.md

Top

Preview

Code

Blame

Raw



0.0.14

Features

- ✨ Add support for Pydantic v2 (while keeping support for v1 if v2 is not available). PR [#722](#) by [@tiangolo](#) including initial work in PR [#699](#) by [@AntonDeMeester](#).

0.0.13

Fixes

- ♻️ Refactor type generation of selects re-order to prioritize models to optimize editor support. PR [#718](#) by [@tiangolo](#).

Refactors

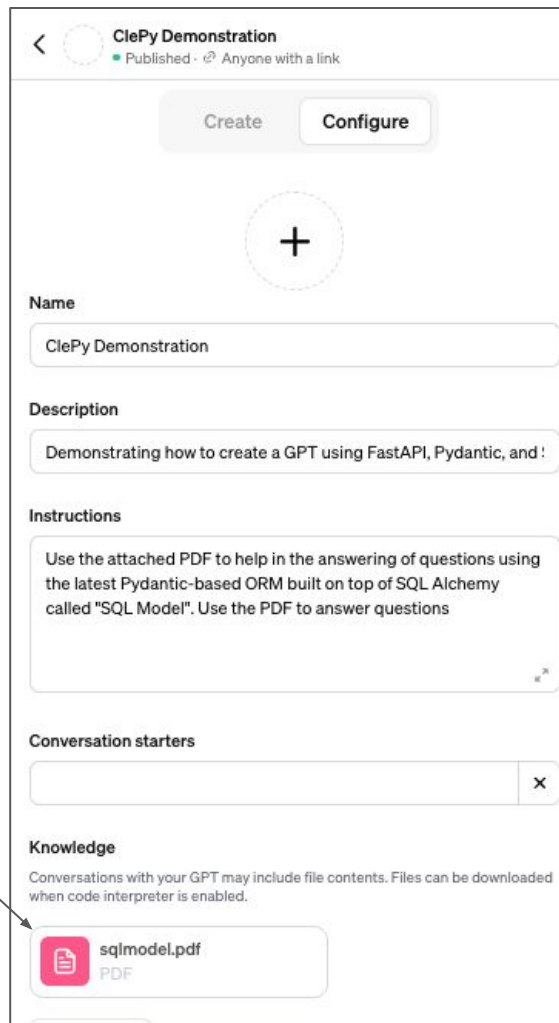
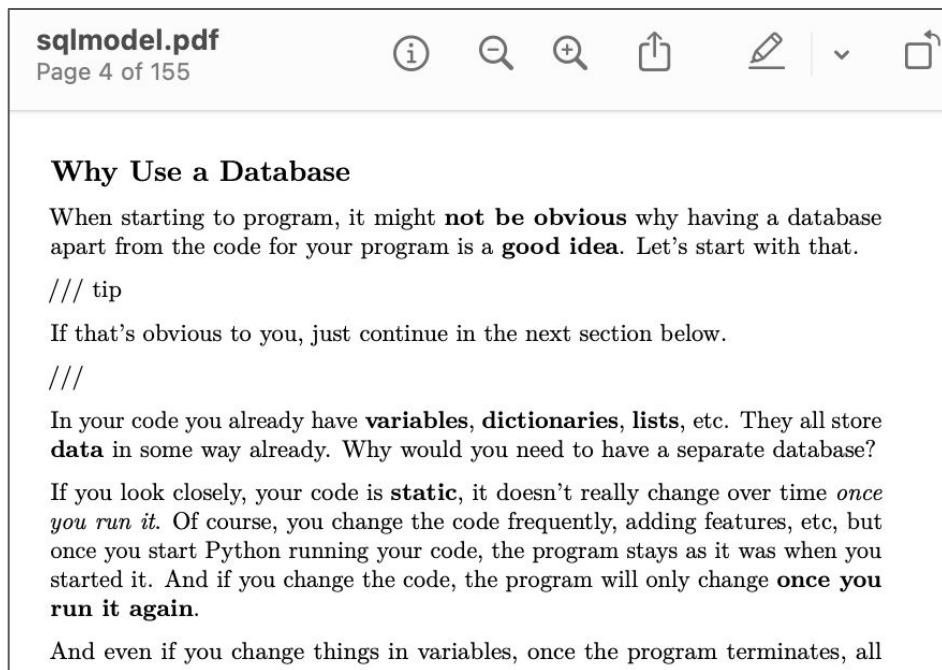
- 🚫 Do not raise deprecation warnings for execute as it's automatically used internally. PR [#716](#) by [@tiangolo](#).
- ✅ Move OpenAPI tests inline to simplify updating them with Pydantic v2. PR [#709](#) by [@tiangolo](#).

Upgrades

- ⬆️ Add support for Python 3.11 and Python 3.12. PR [#710](#) by [@tiangolo](#).

```
$ git clone git@github.com:tiangolo/sqlmodel.git
$ cd sqlmodel/docs

$ tail -n +1 *.md tutorial/*.md > /tmp/sqlmodel.md
$ pandoc /tmp/sqlmodel.md -o /tmp/sqlmodel.pdf
```





You

Please list out the changes from the Release Notes in the SQLAlchemy knowledge base starting with 0.0.5 and ending with the most recent release (0.0.14).

< 2 / 2 >



ClePy Demonstration

The changes in SQLAlchemy from version 0.0.5 to 0.0.14 are as follows:

- **0.0.5**
 - Features
 - Documentation updates
 - Internal changes
- **0.0.6**
 - Breaking Changes
 - Documentation updates
- **0.0.7**
 - Features
 - Fixes
 - Documentation updates
 - Internal changes
- **0.0.8**
 - Fixes
 - Documentation updates
- **0.0.9**
 - Breaking Changes
 - Features
 - Fixes
 - Upgrades



You

Provide a concise summary of each release.

< 2 / 2 >



ClePy Demonstration

Here's a concise summary of each SQLAlchemy release from version 0.0.5 to 0.0.14:

- **0.0.14**
 - **Features:** Added support for Pydantic v2 while maintaining compatibility with v1.
- **0.0.13**
 - **Fixes:** Refactored type generation of selects to prioritize models for better editor support.
 - **Refactors:** Stopped raising deprecation warnings for ``execute``, moved OpenAPI tests inline for Pydantic v2.
 - **Upgrades:** Added support for Python 3.11 and 3.12.
 - **Docs:** Updated and corrected various documentation sections.
 - **Internal:** Several internal updates including pre-commit auto-updates and documentation build enhancements.
- **0.0.12**
 - **Features:** Upgraded SQLAlchemy to version 2.0.
 - **Internal:** Various internal updates including pre-commit auto-updates and GitHub Action upgrades.
- **0.0.11**
 - **Features:** Introduced support for passing a custom SQLAlchemy type to ``Field()`` with ``sa_type``, improved handling of field parameters for columns and relationships.
 - **Docs:** Updated inline source examples in documentation.
 - **Internal:** Several internal updates including upgrades to dependencies and CI

```
from fastapi import FastAPI, Path
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI(title="ClePy 12/11", root_path="/clepy/")

origins = [
    "http://localhost",
    "https://chat.openai.com",
]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

```
% cat nginx.conf
events {
    worker_connections 1024;
}

http {
    server {
        listen 80;

        location /clepy/ {
            proxy_pass http://host.docker.internal:8010/;
            proxy_set_header Host $host;
        }
    }
}
```

```
% cat Dockerfile
FROM nginx:latest
COPY nginx.conf /etc/nginx/nginx.conf
```

```
% docker build -t nginx-ngrok-proxy .
% docker run -d -p 80:80 --name proxy nginx-ngrok-proxy
```


Add actions

Let your GPT retrieve information or take actions outside of ChatGPT.
[Learn more.](#)

Schema

Import from URL Examples

```
{
  "openapi": "3.1.0",
  "info": {
    "title": "ClePy 12/11",
    "version": "0.1.0"
  },
  "servers": [
    {
      "url": "https://perfect-camel-charming.ngrok-free.app/clepy"
    }
  ],
  "paths": {
    "/{name}/{version}": {
      "get": {
        "summary": "Get Project",
        "operationId": "get_project__name__version__get",
        "parameters": [
          {
            "name": "name",

```

Available actions

Name	Method	Path	
get_project__name__version__get	GET	/{name}/{version}/	Test

You
Tell me what you know about fastapi 0.104.1

ClePy Demonstration sent this info to perfect-camel-charming.ngrok-free.app

```
{
  "name": "fastapi",
  "version": "0.104.1"
}
```

ClePy Demonstration

Talked to perfect-camel-charming.ngrok-free.app

FastAPI version 0.104.1 is a high-performance web framework for building APIs with Python 3.8 and later. It is easy to learn and fast to code, and it's designed for production use. Here are some key details about this version:

- **Version:** 0.104.1
- **Author Email:** Sebastián Ramírez tiangolo@gmail.com
- **Package URL:** [FastAPI on PyPI](#)
- **Requires Python:** Version 3.8 or later
- **Summary:** FastAPI framework, high performance, easy to learn, fast to code, ready for production
- **Development Status:** Beta
- **Framework Compatibility:** AsyncIO, FastAPI, Pydantic
- **Intended Audience:** Developers, Information Technology, System Administrators
- **License:** MIT License
- **Operating System:** OS Independent
- **Programming Language:** Python (versions 3.8, 3.9, 3.10, 3.11)
- **Dependencies:** Depends on `'anyio'`, `'pydantic'`, `'starlette'`, `'typing-extensions'`, and has additional dependencies for the 'all' extra feature like `'email-validator'`, `'httpx'`, `'itsdangerous'`, `'jinja2'`, `'orjson'`, `'pydantic-extra-types'`, `'pydantic-settings'`, `'python-multipart'`, `'pyyaml'`, `'ujson'`, `'uvicorn[standard]'`.



clepy Public Pin Unwatch 1 Fork 0 Star 0

main Go to file Code

imaurer	Add ...	9935023 · 33 minutes ago	7 Commits
src/clepy	Add URLs to A...	33 minutes ago	
tests	Find a project b...	1 hour ago	
.gitignore	Create SQLMo...	2 hours ago	
Makefile	Initial check-in ...	4 hours ago	
README.md	Initial check-in ...	4 hours ago	
activate.sh	Hello World.	3 hours ago	
pyproject.toml	Hello World.	3 hours ago	

README

clepy

About

Cleveland Python User Group
FastAPI demo

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Languages

<https://github.com/imaurer/clepy>