

# UtilMeta

Progressive Meta Framework for API development in Python

面向服务端应用的 Python 渐进式元框架



周煦林

XULIN ZHOU



# 声明式开发?

WHY DECLARATIVE



我们来直观地比较两段 Python 接口代码

```
def signup(request):
    import re
    if request.method != 'POST':
        # 请求方法错误, 给出 405 响应
        raise MethodNotAllowed
    username = request.POST.get('username')
    password = request.POST.get('password')
    # 提取用户名密码参数
    if not isinstance(username, str) or not \
        re.match('[0-9a-zA-Z]{3,20}', username):
        # 用户名校验, 不通过返回 400 响应
        raise BadRequest('Invalid username')
    if not isinstance(password, str) or len(password) < 6:
        # 密码校验, 不通过返回 400 响应
        raise BadRequest('Invalid password')
    # 全部校验通过, 开始编写业务逻辑
```

```
from utilmeta.core import api, request

class UserAPI(api.API):
    @api.post # 定义一个 POST 方法
    def signup(
        self,
        username: str = request.BodyParam(regex='[0-9a-zA-Z]{3,20}'),
        # 声明用户名参数: 以及需要满足的正则表达式
        password: str = request.BodyParam(min_length=6),
        # 声明密码参数: 指定最低长度为 6
    ):
        # 直接在函数中编写业务逻辑
        # UtilMeta 已自动拒绝参数校验不通过的请求
```

## 过程式接口

- 代码冗余, 可读性差
- 手写校验容易遗漏或出错

## 声明式接口

- 代码即文档, 清晰简洁, 减少 bug
- IDE 类型提示 + 参数补全, 高效开发

# 声明式 ORM：高效开发 RESTful API

DECLARATIVE RESTFUL API



```
from utilmeta.core import api, orm
from .models import User, Article
from django.db import models

# 定义用户 Schema: 包括用户名与文章数字段
class UserSchema(orm.Schema[User]):
    username: str
    articles_num: int = models.Count('articles')

# 定义文章 Schema: 包括作者对象与内容字段
class ArticleSchema(orm.Schema[Article]):
    id: int
    author: UserSchema
    content: str

class ArticleAPI(api.API):
    # 异步 GET 接口, 通过 id 参数直接序列化出
    # 对应的文章对象实例
    async def get(self, id: int) -> ArticleSchema:
        return await ArticleSchema.ainit(id)
```

“what you define is what you get”

GET /article?id=1

200

```
{...} json

{
  "id": 1,
  "author": {
    "username": "alice",
    "articles_num": 3
  },
  "content": "hello world"
}
```



QUERY PARAMETERS

- id \* integer

RESPONSE application/json

- id integer

- author object

- username string

- articles\_num integer

- content string

- 大幅提高 RESTful 接口开发效率，产出代码简洁易读
- 启动时可检测大部分常见错误，轻松 debug
- 自动解析请求参数，自动生成 OpenAPI 文档



# 声明式 ORM：自动优化查询执行



UtilMeta ORM 会自动对关系查询进行执行优化，避免手写关系查询可能带来的  $N + 1$  查询问题

```
from utilmeta.core import api, orm
from .models import User, Article
from django.db import models

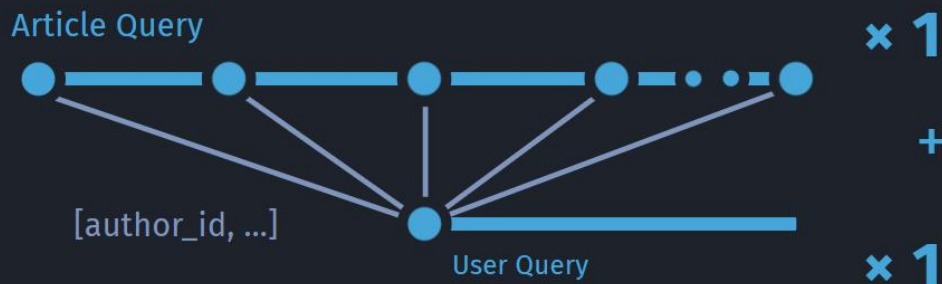
class UserSchema(orm.Schema[User]):
    username: str
    articles_num: int = models.Count('articles')

class ArticleSchema(orm.Schema[Article]):
    id: int
    author: UserSchema
    content: str

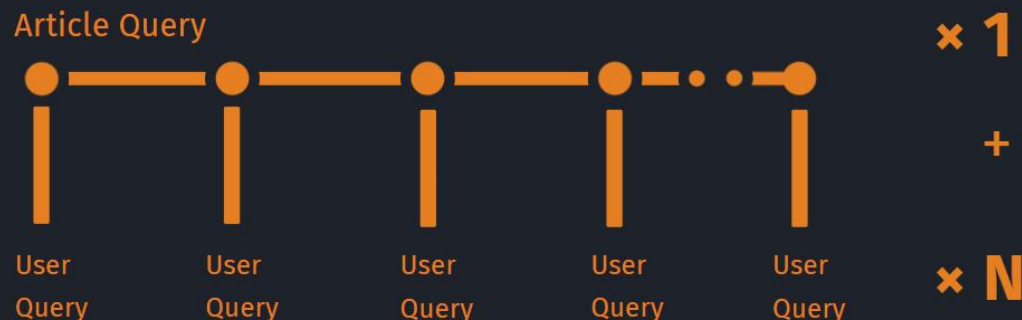
# 使用 UtilMeta ORM 高效序列化，得到一个 ArticleSchema 实例的列表
result = ArticleSchema.serialize(article_queryset)
```

```
result = []
for article in article_queryset:
    # 遍历查询集中的每个元素都运行了其他查询
    # 就会导致不少于元素个数 N 倍数的总查询次数
    author = User.objects.get(id=article.author_id)
    result.append(dict(
        id=article.id,
        content=article.content,
        author=dict(
            username=author.username,
            articles_num=Article.objects.filter(author=author).count()
        )
    ))
```

UtilMeta ORM 自动优化的高效查询：常数次查询



$N + 1$  查询问题示例： $O(N)$  次查询  $N$  为关系层数



# API 系统：路由，钩子与插件



```
from utilmeta.core import api
from .user import UserAPI

@api.route('{slug}/comments')
class CommentAPI(api.API):
    slug: str

# /api/articles
class ArticleAPI(api.API):
    # /api/articles/{slug}/comments
    comments: CommentAPI

    # /api/articles/{slug}
    @api.get('/{slug}')
    def get_article(self, slug: str):
        pass

# /api
class RootAPI(api.API):
    # /api/user
    user: UserAPI
    # /api/articles
    articles: ArticleAPI
```

UtilMeta 框架使用 [API 类挂载](#) 的方式定义树状路由  
简洁方便且便于组织代码

API 类中支持灵活指定目标函数或路由的钩子  
在 API 函数调用的 [前](#)，[后](#)，和 [出错](#) 时进行处理

[@api.before](#)

[@api.after](#)

[@api.handle](#)

支持定义接口插件，以装饰器的方式灵活选择作用范围

部分内置插件：

- Session 服务端会话插件
- CORS 跨域插件
- Retry 请求重试插件

# 声明式客户端构建



```
from utilmeta.core import api, orm
from .models import User, Article
from django.db import models

# 定义用户 Schema: 包括用户名与文章数字段
class UserSchema(orm.Schema[User]):
    username: str
    articles_num: int = models.Count('articles')

# 定义文章 Schema: 包括作者对象与内容字段
class ArticleSchema(orm.Schema[Article]):
    id: int
    author: UserSchema
    content: str

class ArticleAPI(api.API):
    # 异步 GET 接口, 通过 id 参数直接序列化出
    # 对应的文章对象实例
    async def get(self, id: int) -> ArticleSchema:
        return await ArticleSchema.ainit(id)
```

API 接口代码

客户端 SDK 代码



```
from utilmeta.core import api, cli, response, request
import utype

class UserSchema(utype.Schema):
    username: str
    articles_num: int

class ArticleSchema(utype.Schema):
    id: int
    author: UserSchema
    content: str

class article_get_response(response.Response):
    content_type = "application/json"
    result: ArticleSchema

class APIClient(cli.Client):
    @api.get("/article", tags=["article"])
    async def article_get(
        self, id: int = request.QueryParam(required=True)
    ) -> article_get_response[200]: pass
```

- 声明式语法与规则与 API 类相同
- 请求库支持 urllib, requests, httpx, aiohttp
- 可直接解析 OpenAPI 文档生成 SDK



```
import httpx
>>> client = APIClient(base_url="http://127.0.0.1:8080", backend=httpx)
>>> resp = await client.article_get(id=1)
>>> resp
article_get_response [200 OK] "GET /article?id=1"
>>> resp.result
ArticleSchema(id=1, author=UserSchema(username='alice', articles_num=1), content='Hello World')
```



# 设计理念 - 简洁，健壮，灵活，一致

BE SIMPLE, BE STRONG



- 代码即文档，好理解好维护

语法简洁清晰，让开发者降低心智负担，专注于真正重要的逻辑

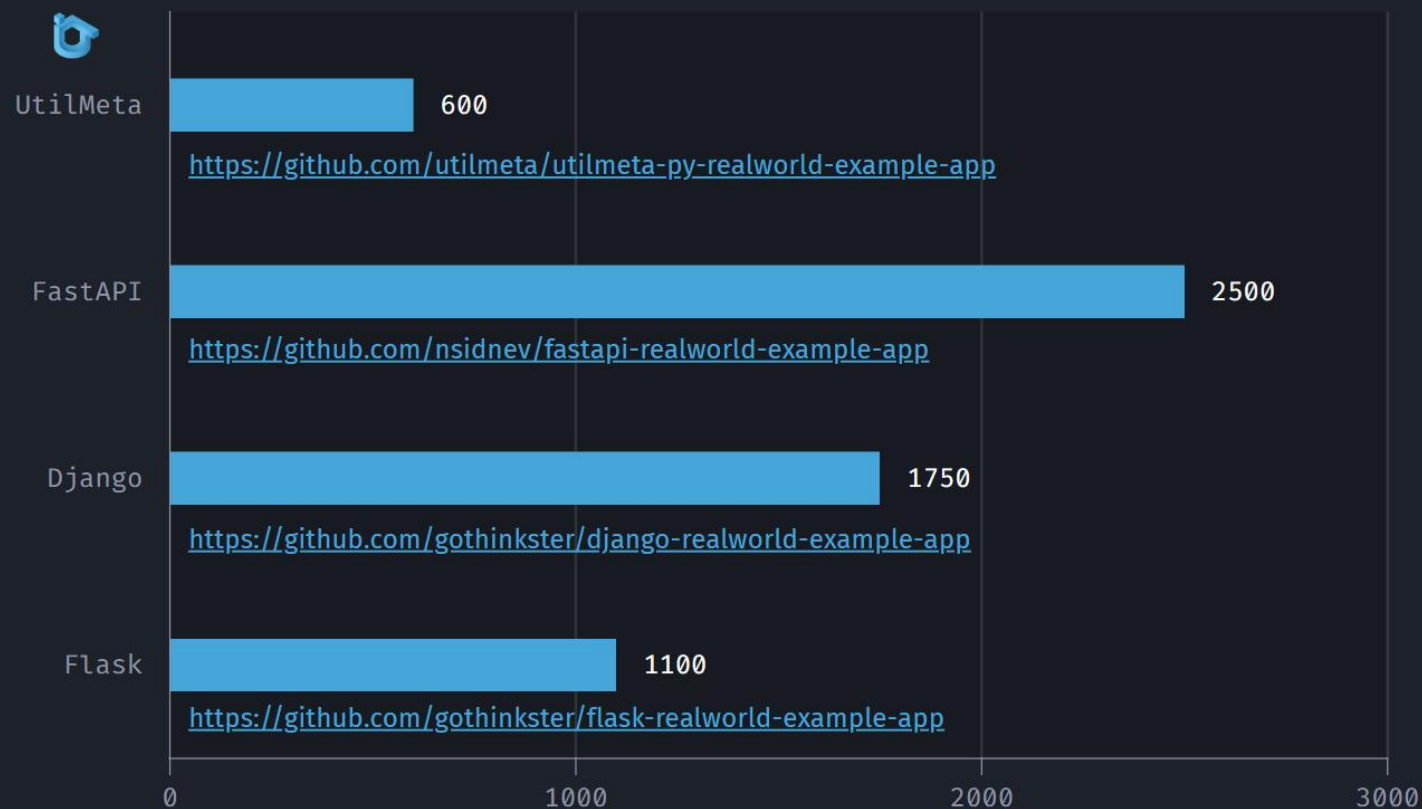
- 健壮稳定，充分优化执行效率

声明式语法为开发者屏蔽底层细节，降低 bug 概率，技术社区可以充分优化执行效率

- 贴合语言语法，拥抱开放标准

基于 Python PEP 484 类型提示 + OpenAPI 文档标准，IDE 与工具链生态完善

## 同一套接口的开发代码行数比对 - Realworld 博客项目



Authentication:

Registration:

Get Current User

Update User

Get Profile

Follow user

Unfollow user

List Articles

Feed Articles

Get Article

Create Article

Update Article

Delete Article

Add Comments to an Article

Get Comments from an Article

Delete Comment

Favorite Article

Unfavorite Article

Get Tags



# 渐进式元框架

PROGRESSIVE META FRAMEWORK



使用一套标准支持主流 Python 框架作为 HTTP 运行时实现  
切换运行时实现只需要一个参数

```
from utilmeta import UtilMeta
import starlette

service = UtilMeta(
    __name__,
    backend=starlette,
)
```

django



Flask

SANIC  
FRAMEWORK

Starlette



FastAPI



Tornado

and more ...

```
from flask import Flask
from utilmeta.core import api, response

app = Flask(__name__)

# Flask 框架的接口函数
@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"

# UtilMeta 框架的接口函数
class CalcAPI(api.API):
    @api.get
    def add(self, a: int, b: int) -> int:
        return a + b

# 使用一行代码即可将 UtilMeta API 挂载到指定路径
CalcAPI.__as__(app, route='/calc')
```



Flask

- 支持现有项目以接口粒度接入 UtilMeta
- 支持接入其他 Python 框架接口

# UtilMeta API 服务管理平台

FULL-LIFECYCLE API DEVOPS



团队权限管理  
数据 CRUD 管理  
接口文档 & 调试  
日志查询与链路追踪  
接口访问数据分析  
单测与集成测试  
拨测与性能监控  
服务端资源监控  
事件管理与报警

一键连接 API 服务  
一站式观测与管理



# API 文档

自动同步可调试的 API 文档



UtilMeta Team Realworld Demo

Filter APIs

user (2) users (2) profiles (3) articles (11) comments (3)

GET /tags service.api.RootAPI.tags

DELETE /articles/{slug}/favorite domain.article.api.ArticleAPI.unfavorite

POST /articles/{slug}/favorite domain.article.api.ArticleAPI.favorite

DELETE /articles/{slug} domain.article.api.ArticleAPI.delete\_article

PUT /articles/{slug} domain.article.api.ArticleAPI.update\_article

GET /articles/{slug} domain.article.api.ArticleAPI.get\_article

GET /articles/feed domain.article.api.ArticleAPI.feed

POST 创建文章 domain.article.api.ArticleAPI.post

GET /articles domain.article.api.ArticleAPI.get

DELETE /articles/{slug}/comments... domain.article.api.CommentAPI.delete\_comm...

POST https://realworld.utilmeta.com/api /articles

Panel Logs Tests Monitors Analytics

# 创建文章

articles

Add API Document

AUTHENTICATIONS

jwt http.bearer JWT Input key Manage credentials

Authorization

REQUEST BODY application/json object

article \* ArticleSchema\_a

body \* string Hello UtilMeta

title string length <= 255

tagList array items <= 255

Docs Debug Send

request validate passed

curl Python Javascript

```
curl -X POST 'https://realworld.utilmeta.com/api/articles' --data '{
  "article": {
    "body": "Hello UtilMeta"
  }
}'
```

快速检索过滤 API 接口

可调试并得到响应数据

支持多种 API 鉴权方式

自动生成不同语言 SDK

支持编写与执行 API 测试

# 数据管理

强大美观的 CRUD 可视化工具



UtilMeta

Team Realworld Demo

Overview

Team

Data

API

Logs

Analytics

Test

Monitor

Servers

Incidents

Settings

Filter Tables

user (3) article (3)

Comment  
domain.article.models.Comment

Article  
domain.article.models.Article

Tag  
domain.article.models.Tag

Follow  
domain.user.models.Follow

Favorite  
domain.user.models.Favorite

User  
domain.user.models.User

Article

Field Condition Value

article\_article

title	author_id	description	public
UtilMeta - a meta backend framework for Python	1	UtilMeta is a progressive meta-framework for backend applica ...242 chars	<input checked="" type="checkbox"/> true
Building an SEO-friendly responsive 118n website using Vite- ...74 chars	1	Why SPA shouldn't be used to build a project homepage? How t ...120 chars	<input checked="" type="checkbox"/> true
Build User login/signup & RESTful APIs in 100 lines of Pytho ...61 chars	1	For most applications, user is a must, we always need to dev ...212 chars	<input checked="" type="checkbox"/> true

Rows 10

Expand Value (description)

Copy Row URL

Export Row Value (JSON)

Edit Row

Delete Row

Add Model Document

Model Fields in article\_article

id  
Primary Key BigAutoField

slug\*  
SlugField - unique

integer  
1 <= value <= 2147483647

string  
length <= 255

便捷 CRUD 后台管理

支持全部字段查询与排序

支持外键数据跳转查询

支持隐藏自定义保密字段

展示表结构与表字段文档

导入导出 JSON / Excel



# 日志分析

全盘观测 API 服务的每一条请求与错误



Service Logs

Test logs

Query logs

...

Environment

https://api-sh.utilmeta.com

Levels

DEBUG

INFO

WARN

ERROR

Status Code

2XX

3XX

4XX

5XX

HTTP Methods

GET

POST

PUT

PATCH

DELETE

Order By

Time

Duration

Level	Time	Method	Path	Status	Client	Duration
INFO	2024-11-09 14:48:02	GET	/api/endpoint/credentials/list	200	CeVu ...	25 ms
INFO	2024-11-09	GET	/api/user/credentials			
INFO	2024-11-09		https://api.lnzhou.com/api			
INFO	2024-11-09		production			
INFO	2024-11-06		domain.user.credentials.CredentialsAPI.get			
INFO	2024-11-06		IP			
ERROR	2024-09-16		Agent			
ERROR	2024-09-16		Chrome Mobile 113.0.0			
ERROR	2024-09-16		Mobile			
ERROR	2024-09-16		Request Headers			
ERROR	2024-09-16		host : api.lnzhou.com			
WARN	2024-08-31		accept : application/json, text/plain, /*			

Status

401 Unauthorized

Duration

2 ms

In Traffic

634 B

Out Traffic

192 B

Response Body

application/json

{

"msg": "Unauthorized: None",

"count": 0,

"state": 0,

"result": null

}

Response Headers

5

Vary : Cookie

Access-Control-Max-Age : 604800

Access-Control-Allow-Origin : https://lnzhou.com

Access-Control-Allow-Methods : GET,POST

Access-Control-Allow-Credentials : true

多维度日志查询

详细的请求响应记录

客户端信息记录

异常调用栈记录

服务端耗时分析

微服务链路追踪

开发代码无侵入

# 服务端观测

实时观测 API 依赖的服务端资源



服务器关键指标实时观测

数据库连接与空间观测

缓存性能观测

服务实例与进程监控

# 一站式 Python 后端服务管理

自动同步 API 文档，查询服务端实时日志



Django  
& DRF



Flask  
& APIFlask



FastAPI  
& Starlette



Sanic



UtilMeta



meta connect

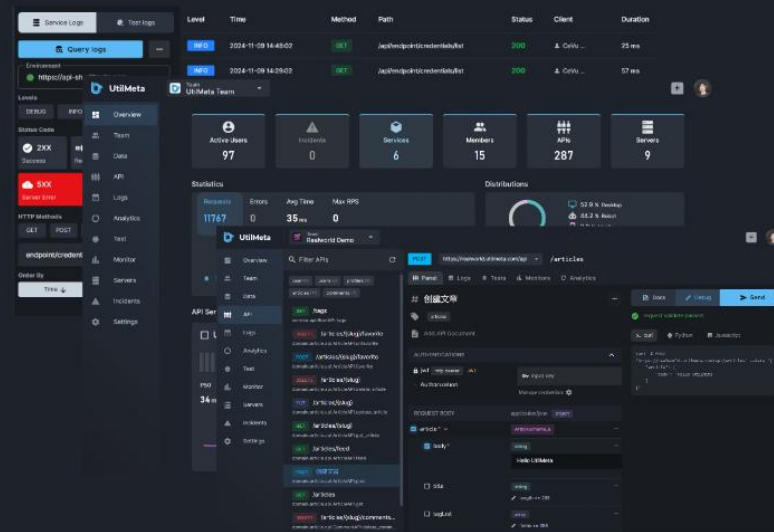


线上服务

使用平台作为客户端 UI



自动同步 OpenAPI 文档，服务元数据



UtilMeta 管理平台

# 轻松连接现有 Python 项目



pip install utilmeta

meta init 初始化项目

引入配置代码

meta connect 一键连接



连接现有 Django 项目

包括 Django REST framework



连接现有 Flask 项目

包括 APIFlask



连接现有 FastAPI 项目

包括 Starlette

```
import os
from django.core.wsgi import get_wsgi_application
application = get_wsgi_application()

from utilmeta.ops import Operations
Operations(
    route='ops',
    database=Operations.Database(
        name='operations_db',
        engine='sqlite3' # or 'postgresql'
    ),
    base_url='https://<YOUR DOMAIN>/api',
).integrate(application, __name__)
```

```
from flask import Flask

app = Flask(__name__)

from utilmeta.ops import Operations
Operations(
    route='ops',
    database=Operations.Database(
        name='operations_db',
        engine='sqlite3' # or 'postgresql'
    ),
    base_url='https://<YOUR DOMAIN>/api',
).integrate(app, __name__)
```

```
from fastapi import FastAPI

app = FastAPI()

from utilmeta.ops import Operations
Operations(
    route='ops',
    database=Operations.Database(
        name='operations_db',
        engine='sqlite3' # or 'postgresql'
    ),
    base_url='https://<YOUR DOMAIN>/api',
).integrate(app, __name__)
```



# UtilMeta 的第一个 5 年迭代旅程



# 旅程才刚刚开始

## UtilMeta

→ API 服务领域的 Git

Python 框架

- 支持声明式语法编写 Websocket / SSE / Server Push 接口
- 支持 SQLAlchemy, Peewee 等 ORM 库作为声明式 ORM 的模型
- 支持定义报警规则, 自动生成部署配置, 集群服务注册与发现, 访问控制, 限流熔断

## UtilMeta

→ API 服务领域的 Github

API 服务管理平台

- 支持 API 请求分析统计, 拨测监控, 压力测试, 事件管理
- 支持接入现有的日志与监控数据源
- 支持为项目 host 公开的 API 文档, Analytics, Status Page
- 支持开发者自定义编写 API 服务管理插件与扩展

LAUNCH



官网

[utilmeta.com](https://utilmeta.com)



扫码进入官网

或搜索

**UtilMeta**

**UtilMeta**

Python 框架

源码

[github.com/utilmeta/utilmeta-py](https://github.com/utilmeta/utilmeta-py)

安装

```
pip install utilmeta
```

**UtilMeta**

API 服务管理平台

平台地址

[beta.utilmeta.com](https://beta.utilmeta.com)

Beta 版本开放体验

样例项目

[beta.utilmeta.com/realworld](https://beta.utilmeta.com/realworld)



扫码体验管理样例项目

## ABOUT ME



周煦林

XULIN ZHOU

全栈架构师

UtilMeta 作者

元组科技创始人

### 主页



github.com/voidZXL



x.com/voidZXL

### 微信



voidZXL

### 爱好



扫码添加我的微信



验证信息: UtilMeta

我来拉你进开发者群

