

用Python構建API的八大流行框架

點擊關注  馬哥Linux運維 昨天



关注蓝字



收获每日技术干货

匠 心 精 神 良 心 教 育

本文將和您討論八種可將API的開發過程變得簡單且快捷的Python框架。其中，Hug和Eve等框架更適合於小型項目，而Django、Flask和Falcon則適合於大型的應用程序。

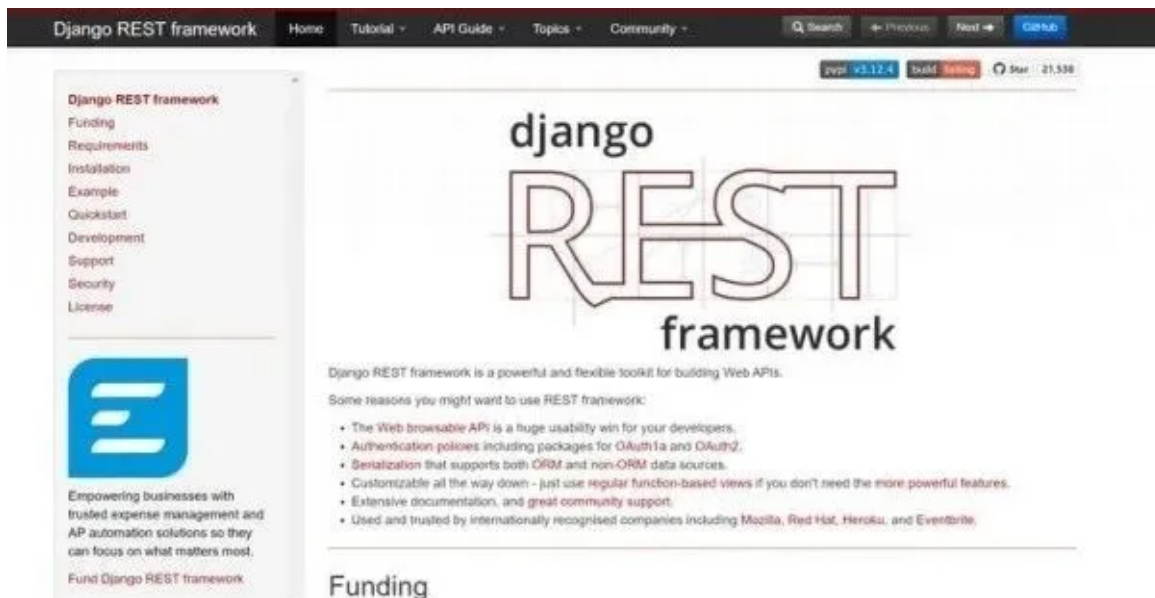
憑藉著平緩的學習曲線和簡單直接的語法，Python在全球範圍內的受歡迎程度，正在呈指數級增長。該編碼語言往往可以被用於Web開發、軟件開發、數學計算、系統腳本、以及幾乎所有其他的領域。作為開發人員的首選語言，人們除了得益於它的跨平台兼容性和代碼的壓縮能力，還可以通過Python框架，創建出強大的應用程序編程接口(API)。

什麼是API?

API可謂互聯網上所有其他平台背後的引擎。它們可以協助不同的應用程序，在後台相互通信，並保持彼此的聯繫。您可以想像自己在一家餐廳裡，手裡拿著一份後廚可以烹製的所有菜餚的菜單。那麼，接單系統在接受到您的訂單後，會將其傳遞給後端進行處理。後台完成後，系統會從後端獲取“烹飪好的食物”並發送給用戶。這便是API的基本作用：幫助用戶和系統進行溝通。

下面，讓我們來看看八種可用於構建API的優秀Python框架：

1. Django REST



Django REST为开发人员提供了丰富的功能与选择。其中，Representational State Transfer(REST)是一种基于Web的架构系统，可用于数据通信。REST的功能包括：一组可浏览的Web API、多个简化了的API开发过程、以及内置的身份验证策略。我们可以使用单个PIP命令，来轻松地安装Django REST。当然，在安装Django REST之前，请确保在您的系统上已安装了Python 3.5或更高的版本。

Django REST不但可以提供便捷的ORM和非ORM源的序列化功能，而且该框架受到了Red Hat、Mozilla和Heroku等流行组织的信任，以及各大活跃社区开发人员的持续支持。

2. Flask Restful



顾名思义，Flask Restful是为了简化和加速API开发过程，而量身定制的。作为轻量级的Python框架，它只需要几个命令，就可以完成API的构建。可以说，作为公认的API工具，Flask能提供直接易用的API开发体验。此外，Flask相对于格式化的字段模块、以及marshal_with()装饰器(decorator)等数据字段，也非常方便。

作为一个带有各种常见API特性的全栈式Python框架工具，Flask通过使用representation()装饰器，提供了诸如：XML、CSV和HTML多种数据的表示。

当然，你若想流畅地运行Flask，则需要在自己的机器上，事先安装好PyPy 2.7、或Python 3.5及其更高的版本。

3. Falcon



The Falcon Web Framework

Release v3.0 (Installation)

Falcon is a minimalist WSGI library for building speedy web APIs and app backends. We like to think of Falcon as the *Dieter Rams* of web frameworks.

When it comes to building HTTP APIs, other frameworks weigh you down with tons of dependencies and unnecessary abstractions. Falcon cuts to the chase with a clean design that embraces HTTP and the REST architectural style.

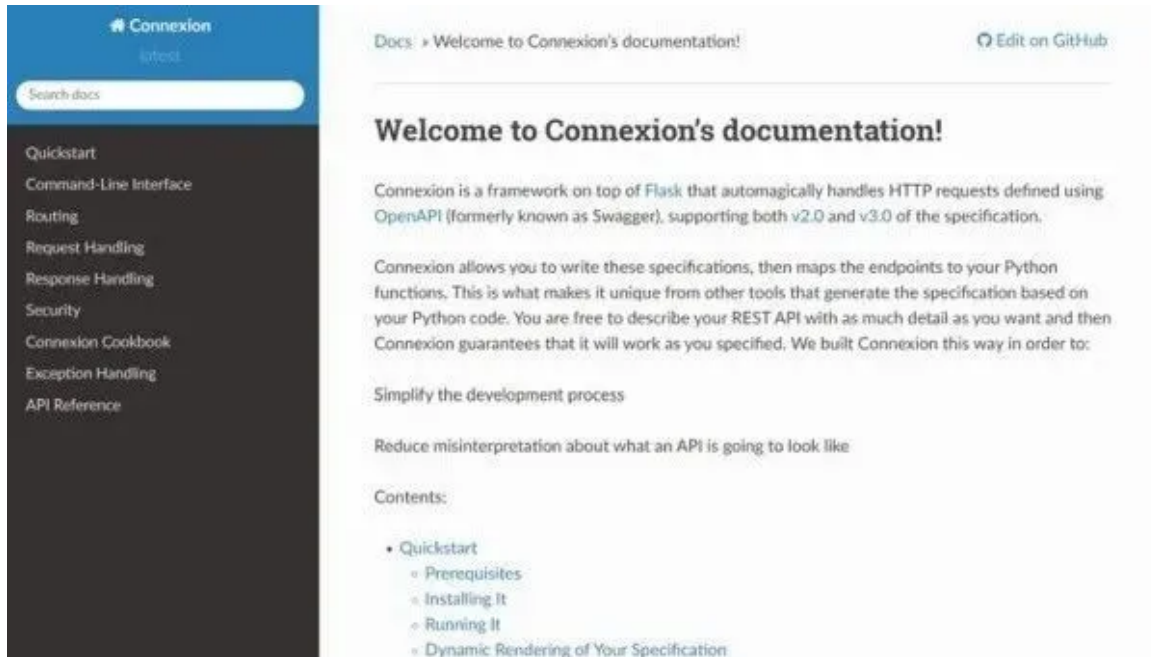
```
class QuoteResource:
    def on_get(self, req, resp):
        """Handles GET requests"""
        quote = {
            'quote': (
                "I've always been more interested in "
                "the future than in the past."
            ),
            'author': 'Grace Hopper'
        }
        resp.media = quote
```

作为开发人员的首选工具，Falcon可以被用来创建快速且高性能的API。它符合WSGI(Web服务器网关接口，Python Web Server Gateway Interface)，能够与多个服务器和平台相兼容，而且可以通过其面向对象、和基于类的界面，给用户带来无缝的转换体验。

Falcon使用HTTP和REST架构，来协助创建用户友好的设计。其REST框架通过提供开发调试器，来全面地简化了开发过程。而且，此类调试器能够与其内置的服务器实现良好的配合与协同。

此外，与其他框架不同的是，Falcon的安装过程不但非常简单、直接，并且几乎不需要用户的人工干预。

4. Connexion



Connexion能够自动处理HTTPS，并使用OpenAPI的各项规范。也就是说，您既可以基于Python代码生成API规范，又可以遵从OpenAPI规范，采取不同的路线。当然，您必须以YAML格式编写OpenAPI规范，然后映射到Python功能函数的各个端点上，以实现对请求及其端点的自动验证。

Connexion能够使用OAuth 2的授权类型，来处理基于令牌的身份验证。它既自带有Web Swagger Console UI，又允许用户调用某个API的端点。此外，Connexion框架还包含有：API版本控制、有效负载的自动序列化等其他方便的功能。

5. FastAPI

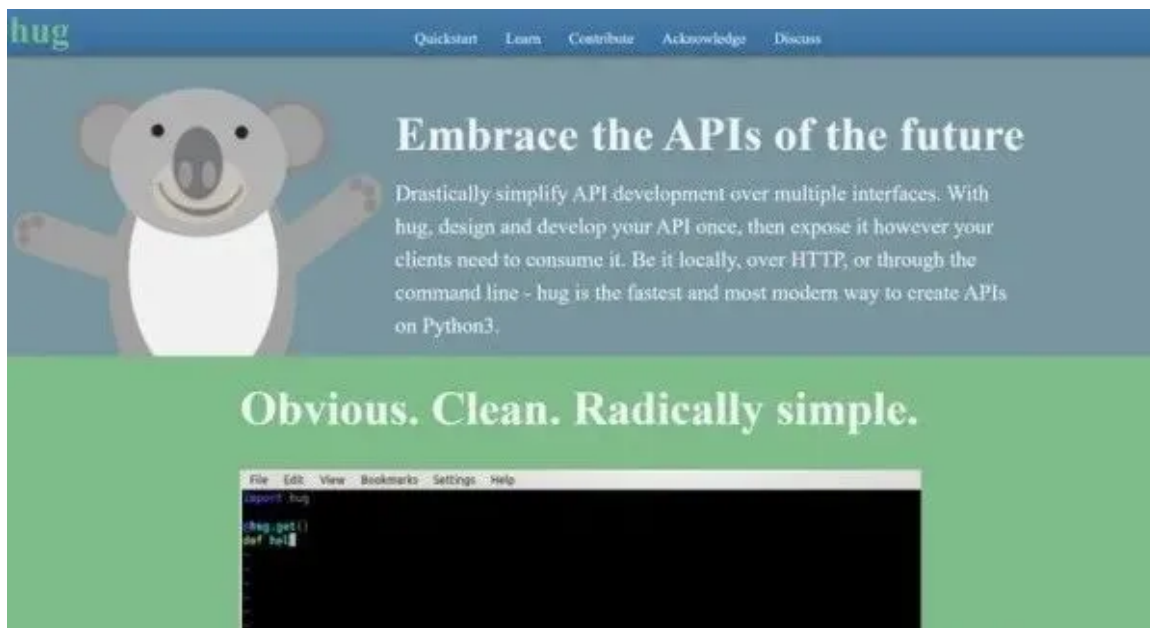


顾名思义，FastAPI是用于构建API的最快Python框架之一。根据用户的反馈，它可以将开发的速度提高200-300%。目前，FastAPI被广泛地用于构建异步类型的Web应用，并得到了200多名社区贡献者的支持。

该Web框架不但能够使得开发过程变得快速且简单，还可以提供诸如：交互式API文档和重复性代码消除等，大量成熟的功能。由于是基于Python，因此该框架能将错误率减少约40%。

在FastAPI中，您也可以使用VSCode和PyCharm之类，常见编辑器的自动完成功能。此外，FastAPI框架还具有无限插件支持、以及集成化的安全协议等不错的特性。

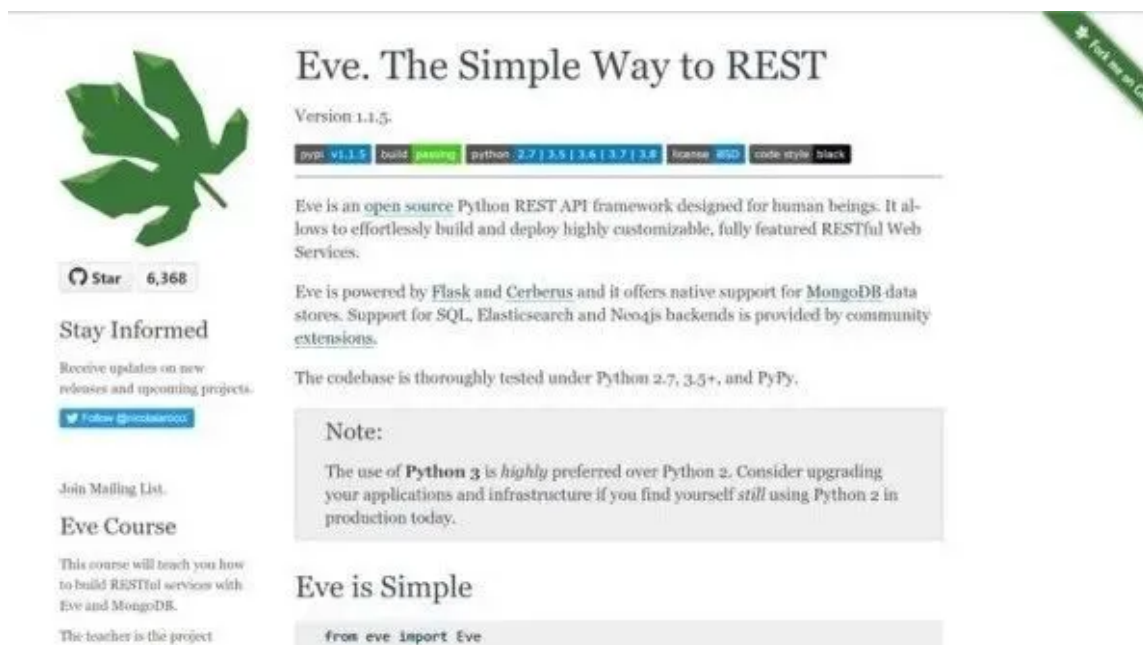
6.Hug



秉承着“一次编写，随处使用”理念的Hug，能够方便用户创建高效的API、本地包和CLI，并将其使用到代码中的任何地方。Hug的高速性能，源于它采用Cython进行编译，以及仅在必要时使用资源。

目前，Hug能够支持Python 3，您可以使用单个PIP命令去安装它。同时，Hug拥有业内最好的API文档。您可以使用内置的hug.test模块，去测试API的完整Python栈。

7. Eve



Eve的灵感来自Flask和Cerberus，并吸取了两者的精华。该工具专为那些需要让API的开发过程变得简单、快速和高效的用户，而量身定制。因此，该框架不但非常适合高效地创建中型的Web服务，而且提供了可定制的端点、分页、排序、以及过滤等功能。

总的说来，您在获得其开箱即用的数据验证支持的同时，还能调用Eve的身份验证、缓存、速率限制等高级功能。同时，该工具也增加了对于全方位的CRUD操作、以及跨源资源共享(CORS)功能的广泛支持。此外，您还可以轻松地将Eve与SQL数据库、MongoDB、Elasticsearch、以及Neo4js相集成。

8. Cornice



Cornice: A REST framework for Pyramid

Cornice provides helpers to build & document REST-ish Web Services with Pyramid, with decent default behaviors. It takes care of following the HTTP specification in an automated way where possible.

We designed and implemented cornice in a really simple way, so it is easy to use and you can get started in a matter of minutes.

Show me some code!

A full Cornice WSGI application looks like this (this example is taken from the [demoapp project](#)):

```
from collections import defaultdict
from pyramid.httpexceptions import HTTPForbidden
from pyramid.view import view_config
from cornice import Service

user_info = Service(name='users',
                    path='/({username})/info',
                    description='Get and set user data.')

_USERS = defaultdict(dict)

@user_info.get()
def get_info(request):
    """Returns the public information about a "user".

    If the user does not exists, returns an empty dataset.
    """
    username = request.matchdict['username']
    return _USERS[username]
```

作为基于REST的Pyramid框架，Cornice为构建和记录基于REST的Web服务，提供了各种帮助。当然，它也可以为各种服务添加跨域资源共享(CORS)的支持。由于Cornice能够自动使用HTTP规范，因此Pyramid可以根据应用程序的需要，自行进行扩展。例如，它可以使用Pyramid的ACL进行认证授权，并在验证过程中发现无效的数据，进而触发400类型的错误。

此外，该框架还允许您使用命令行Python工具—Tox，进行各种自动化测试。

小结

可以说，API在我们使用的各种互联网服务中持续发挥着重要的作用。上文我们讨论了八种Python框架，都能够将API的开发过程变得简单且快捷。其中，Hug和Eve等框架更适合于小型项目，而Django、Flask和Falcon则适合于大型的应用程序。

文章转载：Python编程学习圈
(版权归原作者所有，侵权)



变装天使, 我们一起变美把, 热门小游戏 推荐

用腾讯视频观看



点击下方“阅读原文”查看更多

[阅读原文](#)

喜欢此内容的人还喜欢

超讚，2W 字的Java class類文件結構詳解！

業餘草



SpringBoot巧用@Async 提升API接口並發能力

架構師社區



Java中clone()和new效率哪個更高？

Java筆記蝦

