

GlassOnion Framework

Extending PyGeoAPI for Seamless Feature Addition & Modification

Nitheesh Chandra

Date : 2024/11/18 (v1.0)

nitheesh.research@gmail.com

GitHub nitheesh-me/pygeoapi-glassonion-auth

Based on a PoC developed during OGC Stack TechSprint 2024

GlassOnion Framework

- 1 Problem Statement ↗
- 2 Vision ↗
- 3 Architecture Overview ↗
- 4 Key Features ↗
- 5 Use Cases ↗
- 6 Try it, Break it, Know it ↗
- 7 Impact & Next Steps ↗
- 8 Thank you ↗

Problem Statement



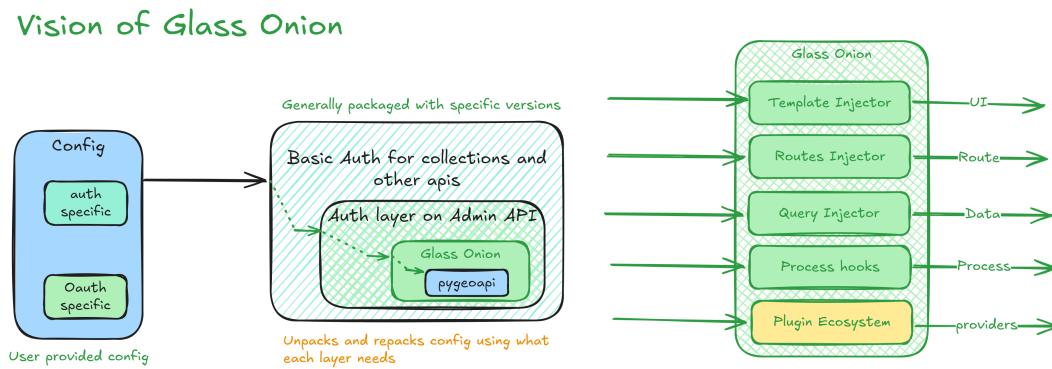
Why GlassOnion?

- **pygeoapi** is a great tool for serving geospatial data.
- But, it lacks a simple way to add or modify features
- The Current framework layout lacks intuitive extendability for downstream feature modification.
- Transparency in workflows and operations is hard to achieve.

So?

- **GlassOnion** is a framework that aims to solve this problem
- It extends **pygeoapi** and adds an abstraction layer for feature addition and modification.

Vision

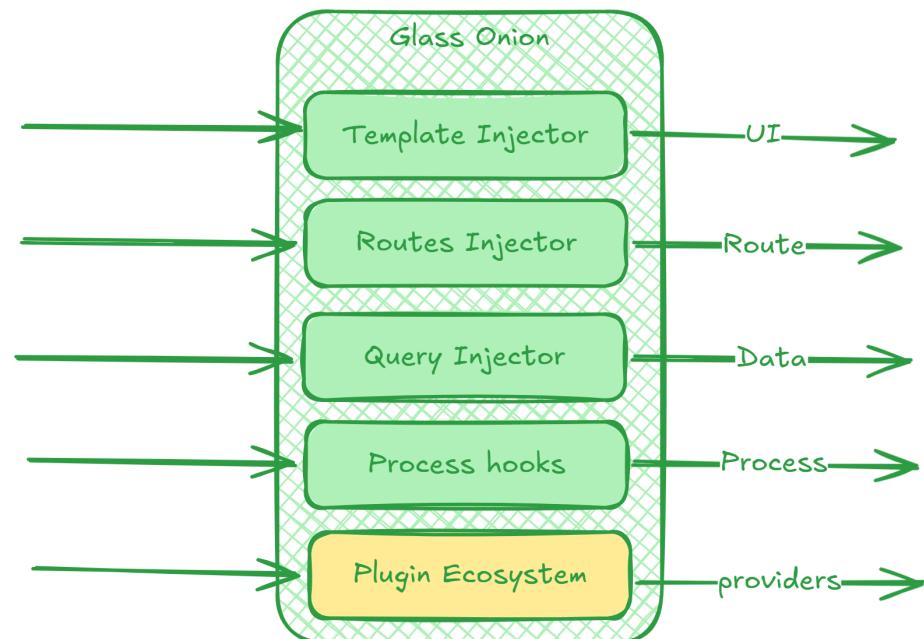
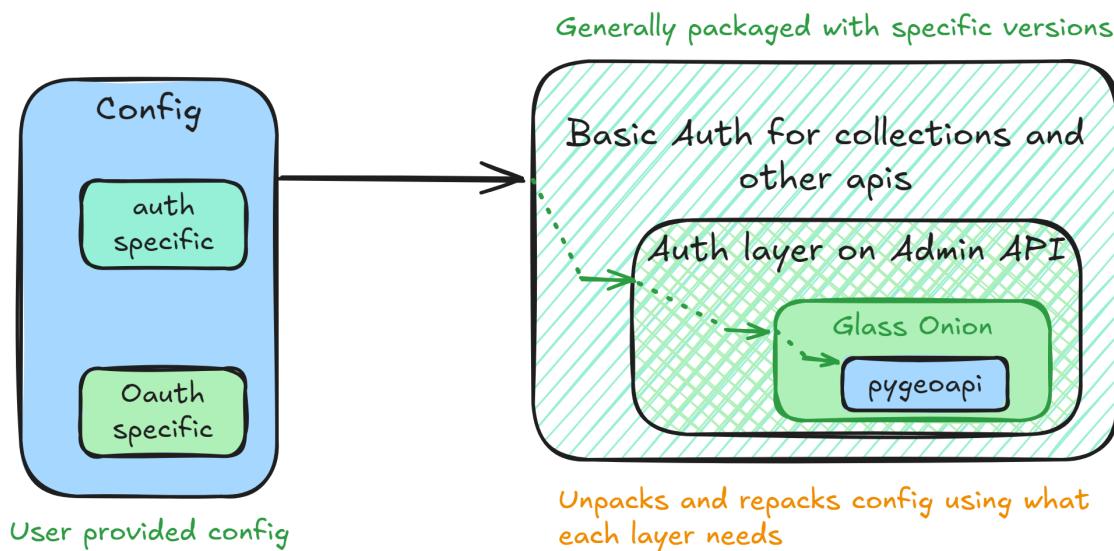


Peeling Back the Layers

- **Transparency:** Like a glass onion, operations must be easy to inspect at each layer.
- **Extendability:** New features can be added as extensions to the core.
- **Interoperability:** Smooth integration with existing tools in the geo-ecosystem and the server implementations.

Architecture Overview

Vision of Glass Onion



Key Features

What Sets GlassOnion Apart

1. (Hope) Transparent API calls for feature management.
2. (Hope) Modular plugin-based architecture for downstream services.
3. (Hope) Lightweight configuration for easy scalability.
4. (Hope) Real-time monitoring and diagnostics.
5. (Hope) Easy integration with existing tools.
6. (Hope) Extendable Configuration for each tool/layer under similar conventions of pygeoapi.
7. (Hope) Layers of functionality, just like an onion.

Use Cases

Who Benefits?

- **GIS Developers:** Seamless feature additions.
 - Authentication, Authorization, and Logging.
 - Monitoring and Observability.
 - Real-time data processing.
 - Extendable services.
- **Data Scientists:** Transparent workflows for geospatial data processing.
 - Integration with workflow management tools.
 - Real-time data processing.
- **Urban Planners:** Real-time service modifications for city modeling.

Try it, Break it, Know it

To verify how extensible and transparent pygeoapi currently is, I have developed an Auth wrapper plugin for pygeoapi.

This plugin adds a layer of authentication to the pygeoapi server.

Observations

- **Well designed for flexibility:** The configuration standards and choices are well thought out.
- **Not designed for extensibility***: Adding a new feature requires a lot of boilerplate code. Some of the core functionalities are not easily extendable. One has to dig deep into the codebase to understand the flow.
- **Documentation:** The documentation is well-written and easy to follow.
- **Understanding the codebase:** The codebase is well structured and easy to understand.
- **Plugin system:** Plugin system is constrained to a few functionalities. It is not easy to add new features.

Features as part of PoC

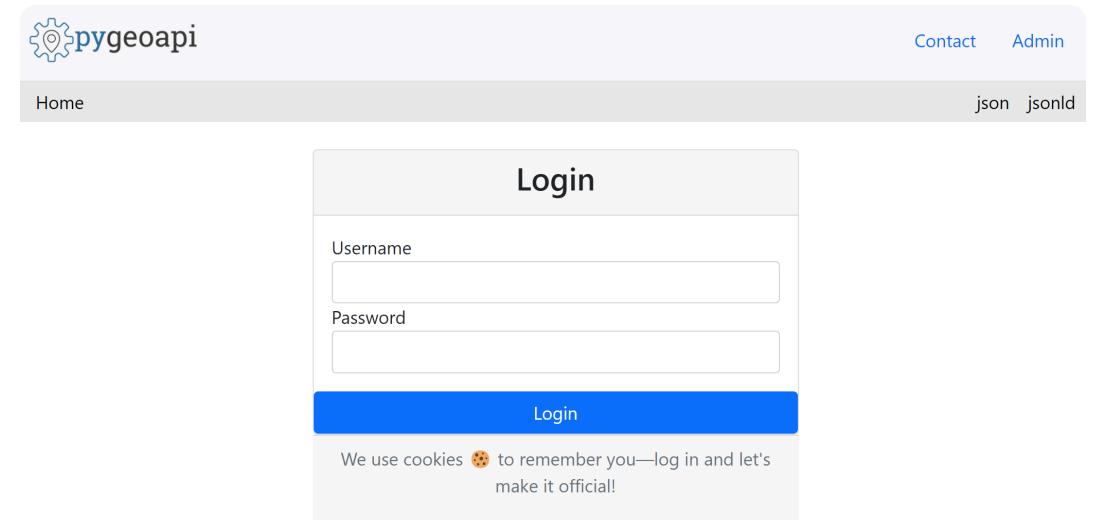
- Focused on the `flask_server`'s extensions.
- Added a new `auth` for routes in `pygeoapi`.
- Extended templates for `auth` in `pygeoapi`. (*no clear support for extensions to add templates, while still allowing the final user to customize the templates*)
- Creating "injectors" to add support to include new features in the `pygeoapi` server.

```
from .injectors.route import apply_decorator_on_views
apply_decorator_on_views(
    APP, login_required,
    include={
        'pygeoapi.*', 'account.settings', 'account.logout',
        '!pygeoapi.landing_page', '!pygeoapi.openapi', 'pygeoapi.static',
    }
)
```

Implementation Details

- **pygeoapi**: For the server configuration and data management.
- **Docker**: For containerization and deployment, same as pygeoapi, in fact, I am extending pygeoapi.
- **Flask middleware**: Flask middleware hooks to add new features.
- **Flask-login**: For user authentication.
- **Flask-Dance**: For OAuth2 authentication. (With Github)
- **Flask-Authorize**: For role-based access control.
- **Config Addition**: Extract the strings for ACL to config.

Allowing natively support login and OAuth2 authentication for the specific routes in the `pygeoapi` server.



Impact & Next Steps

Beyond the Onion

- Standardize the onion framework for different use cases.
- Open-source community adoption.
- Contribute to the pygeoapi project.
- Create core layers for the onion framework.
 - auth, logging, monitoring, etc.
 - support for open telemetry.
 - UI setup for the onion framework.
- Make all the layers configurable and pass-through.
- Community layers.



With Passion and Dedication

Explore at [GitHub nitheesh-me/pygeoapi-glassonion-auth](https://GitHub/nitheesh-me/pygeoapi-glassonion-auth)