Responder: a Familar HTTP Service Framework

2018

2 min read • 414 words

Themes: Human Centered

Introduction

Responder was introduced as a modern HTTP service framework for Python that drew inspiration from the rich history of Python web development while aiming to innovate and dramatically simplify the web development experience. This framework represented both an evolution of existing patterns and a fresh perspective on what Python web development could become in the modern era.

Historical Context

The presentation traced **Python's web development evolution** from its earliest days. In **1999**, Zope and Plone established Python as a viable web development platform, particularly in government sectors where Python's clarity proved valuable. The **2003** introduction of WSGI created a crucial standardization moment.

> WSGI (Web Server Gateway Interface) standardized the interface between Python web applications and web servers, enabling interoperability and the ecosystem of middleware we see today.

Key frameworks emerged in rapid succession: **Django** (2005) as a comprehensive framework for content applications, **Pylons** (2006) with a component-oriented approach, and **Pyramid** (2007) as a balanced alternative.

Flask deserves special recognition for its unique origin and impact.

> Armin Ronacher's April Fool's joke in 2010 became one of the most influential Python web frameworks, demonstrating how simplicity and good design can triumph over comprehensive feature sets.

Initially an April Fool's joke, Flask's elegant simplicity and intuitive API made it extraordinarily popular—developers could often guess correct usage patterns without consulting documentation.

The Future: Responder

Responder's vision positioned it as a future-forward framework for **2019 and beyond**, designed to address modern web development requirements including WebSockets, Server-Sent Events (SSE), and ASGI support.

> ASGI (Asynchronous Server Gateway Interface) represents the evolution beyond WSGI, enabling support for WebSockets, HTTP/2, and other modern protocols that require asynchronous handling.

The **design philosophy** embodied Kenneth's "for Humans" approach by including **Requests** as the standard HTTP client and modeling Request/Response objects after Requests' beloved interface patterns. This ensured immediate familiarity for developers already using Requests.

The ambitious goal was explicitly stated: create "the world's best web framework." This represented a serious commitment to excellence, serving as both community interest gauge and platform for cutting-edge web development patterns.

Conclusion

Responder represented an ambitious synthesis of Python web development's rich history with a clear vision for its future. By building on proven strengths while introducing innovative, user-friendly features, the project aimed to provide Python developers with a robust, modern tool maintaining the community's core values of simplicity and practicality.

This framework embodied the evolutionary approach characterizing Kenneth's broader work: respect for the past, clear assessment of current limitations, and bold innovation toward a more human-centered future.

Generated from kennethreitz.org • 2025