Introducing PIP Services, a new open-source toolkit for enterprise-level microservice creation

Eugenio Andrieu
PyComZA 2021
7–8 October, 2021
Agenda

1 - **Introduction**  
   (3 minutes)  
   A brief explanation of what PIP.Services is.

2 - **Demo**  
   (17 minutes)  
   This demo will show how to build a simple microservice ("Hello World").

3 - **Principles behind and architecture**  
   (10 minutes)  
   Principles behind PIP.Services  
   A brief explanation of its architecture.  
   A brief explanation of a component's lifecycle  
   Advantages of this toolkit
Part 1: Introduction
What is Pip.Services?

Pip.services is a toolkit that presents a collection of ready-to-use components used to build microservices.

- It has a large number of reusable components such as loggers, tracers, performance counters, distributed caches, distributed locks, credential stores, message queues, and configuration readers.

- At present, it supports six programming languages: .NET, Java, Node.js, Python, Golang, and Dart.
Where to find it

https://www.pipservices.org/
Brief history

Initially developed by Enterprise Innovation Consulting and launched as open-source with an MIT license in June 2021.

https://www.entinco.com/
Part 2: Demo
Where to find the code

Python:
https://github.com/pip-services-samples/service-quickstart-python

Node.js:
https://github.com/pip-services-samples/service-quickstart-node

Golang:
https://github.com/pip-services-samples/service-quickstart-go
Part 3: Principles behind and architecture
Principles behind

Symmetric implementation

It means that for every programming language it is implemented in, there are a common set of classes, methods, and method signatures.
#!/usr/bin/env python

from HelloWorldServiceFactory import HelloWorldServiceFactory
from pip_services3_container.ProcessContainer import ProcessContainer
from pip_services3_rpc.build import DefaultRpcFactory

class HelloWorldProcess(ProcessContainer):
    def __init__(self):
        super(HelloWorldProcess, self).__init__("hello-world", "HelloWorld microservice")
        self._config_path = './config.yml'
        self._factories.add(HelloWorldServiceFactory())
        self._factories.add(DefaultRpcFactory())

(exports.HelloWorldProcess = HelloWorldProcess())
Architecture

Components: reusable component definitions

Commons: cross-language primitives and common patterns

.NET Core, Java, Node.js, Python, Go
A component’s lifecycle
The life of a component
Advantages of PIP.Services
Why PIP.Services?

1. Both cross-language and cross-platform
2. Comprehensive
3. Incremental approach
5. Production-grade code
6. High-development productivity
7. Architectural flexibility

It allows businesses to create long-living, adaptable systems in a short time.
The end