Microsoft

改めて考えるWeb API

Internet Week 2017

日本マイクロソフト株式会社
マイクロソフトテクノロジーセンター
吉田 雄哉

自己紹介

日本マイクロソフト株式会社
マイクロソフトテクノロジーセンター
Azure テクニカルアーキテクト
吉田 雄哉（吉田パクえ）

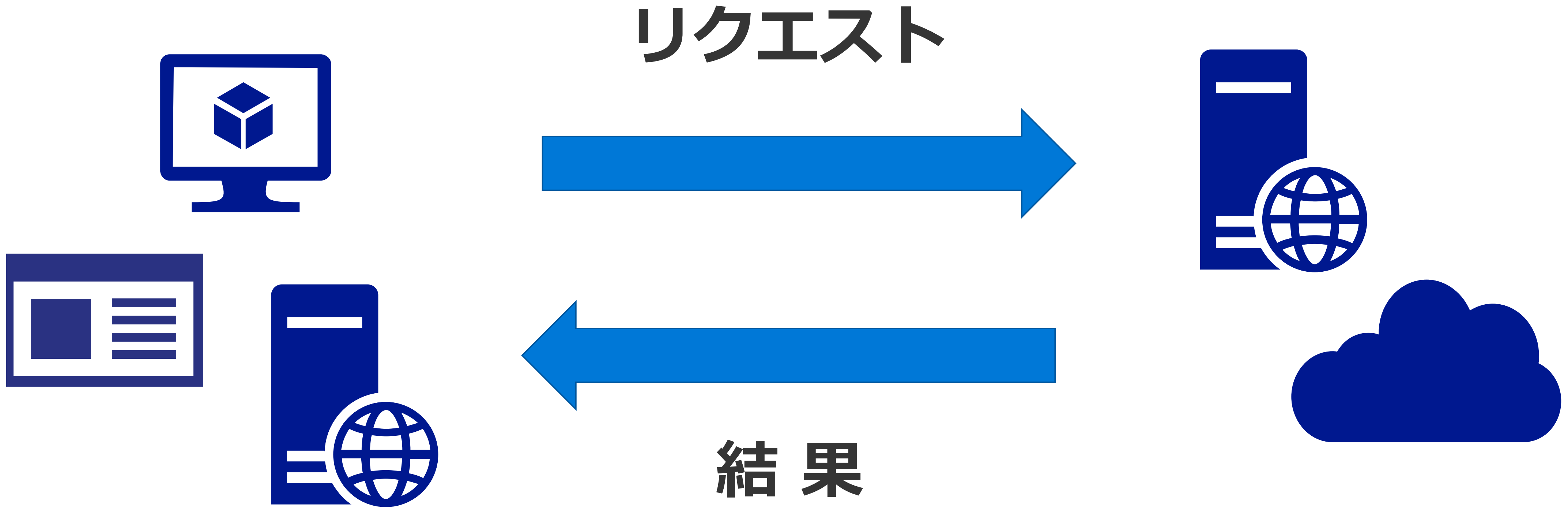
2015年1月1日入社
得意分野　オープンソース系

yuyalush　yuya_lush

Web APIって？

Web API

Web Application Programming Interface



Web APIが登場する場面

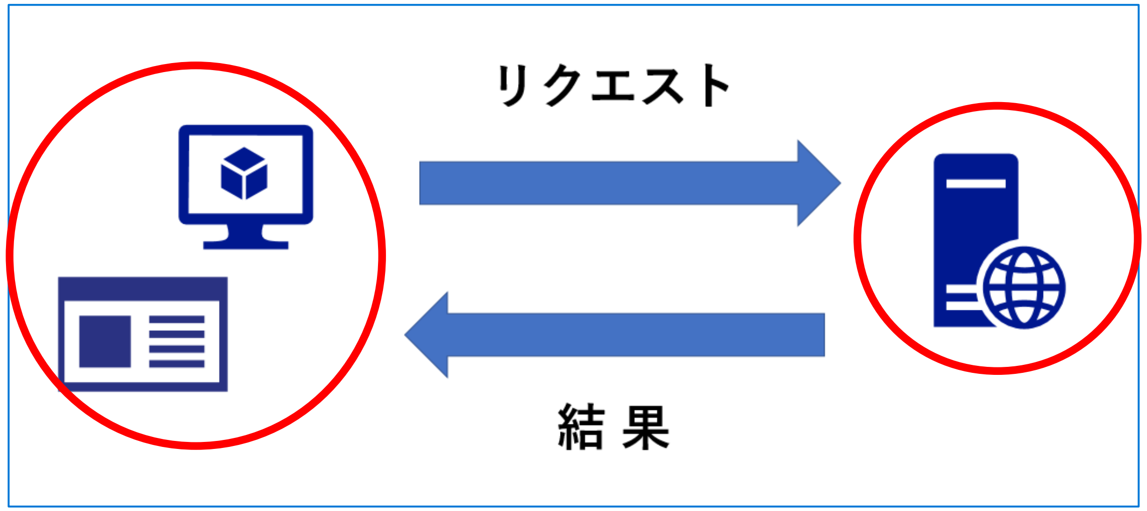
リクエストを送る側

ブラウザやスマホアプリ、サーバーサイドのアプリケーション

リクエストを受ける側

サーバサイドのアプリケーション

クラウドのサービス

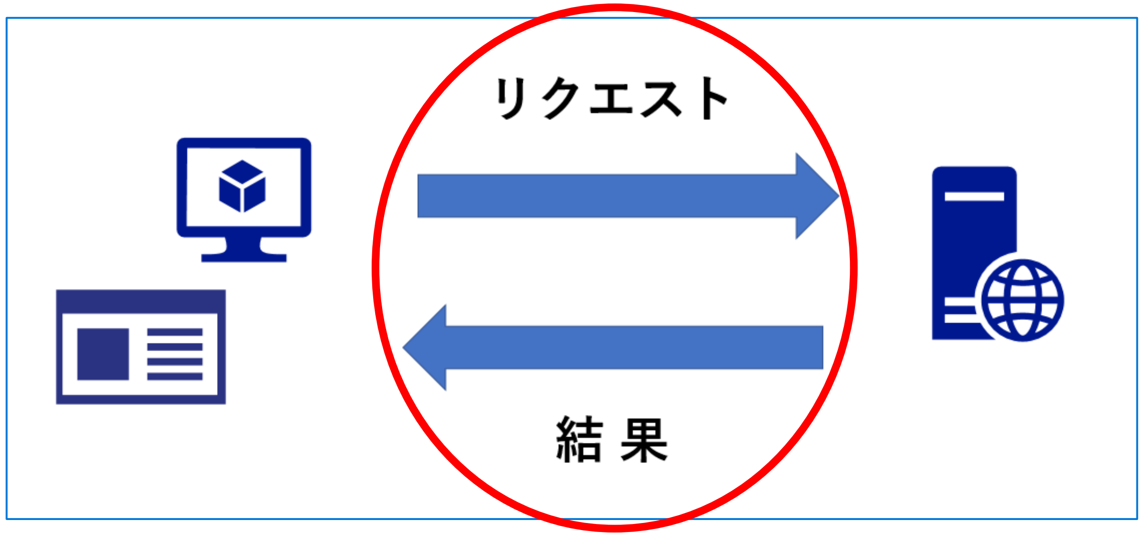


Web APIが登場する場面

通信の方法
　HTTP/HTTPSといった通常のWebで使われる技術

結果のフォーマット
　JSONやXMLが一般的

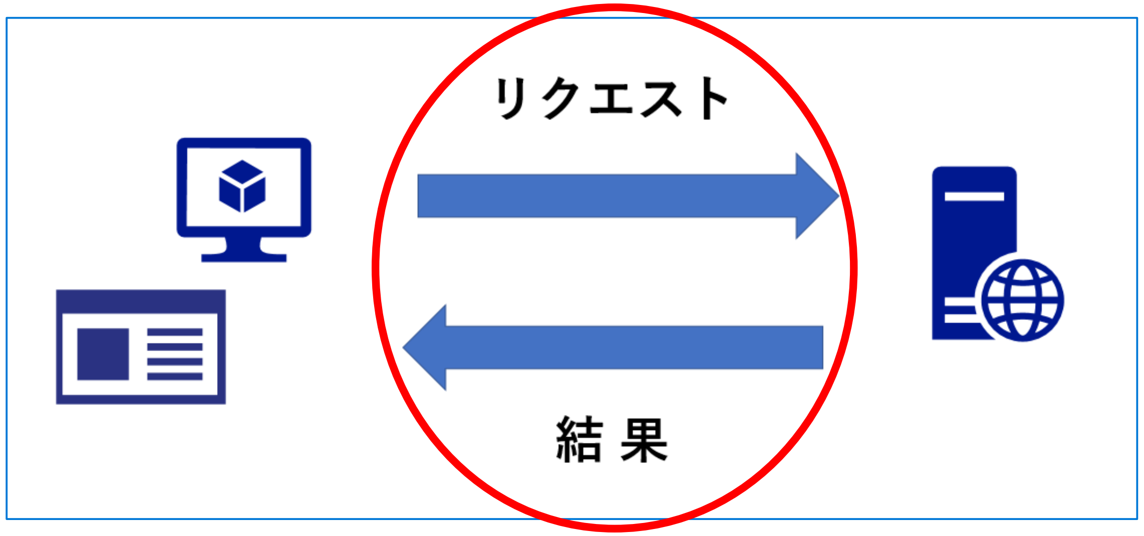
ユーザのビューが必要なく、あくまでプログラム間でのやり取り



Web APIが登場する場面

なぜ、やり取りが必要か

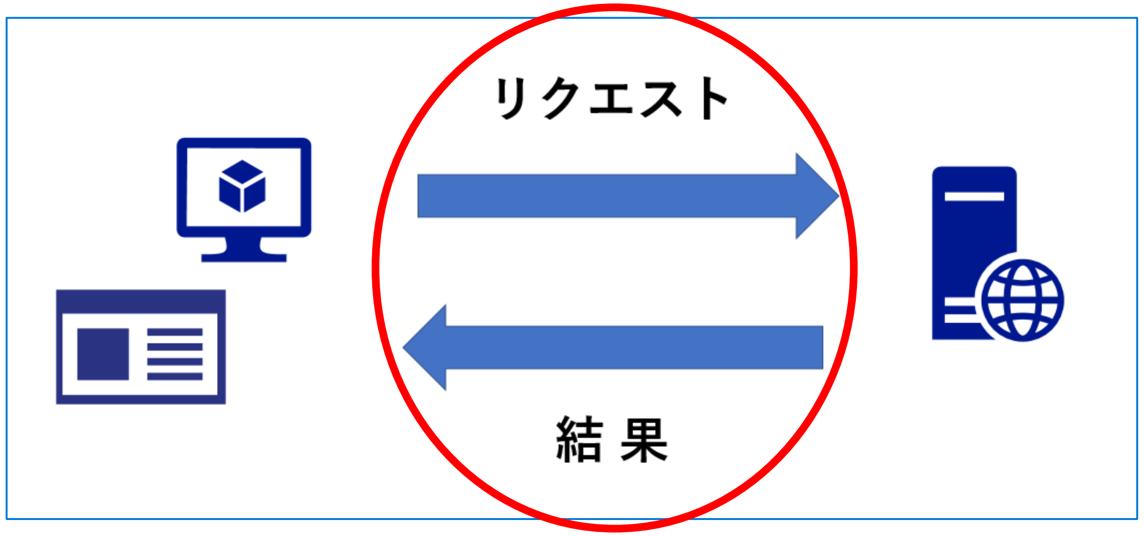
- 処理をしてもらい結果を得たい
- データを取得したい



Web APIが登場する場面

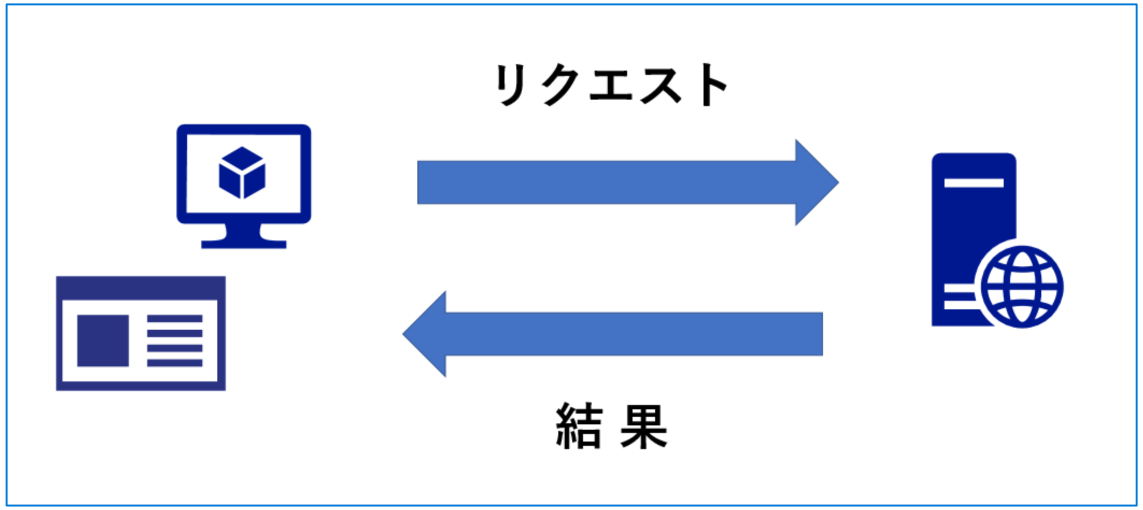
さらに
両者がどこにいるのか
どのようなビジネスメリットを求めているのか

- 社内システム連携
- APIによるエコシステム
- クラウド上でのサービス連携



色々と必要になる

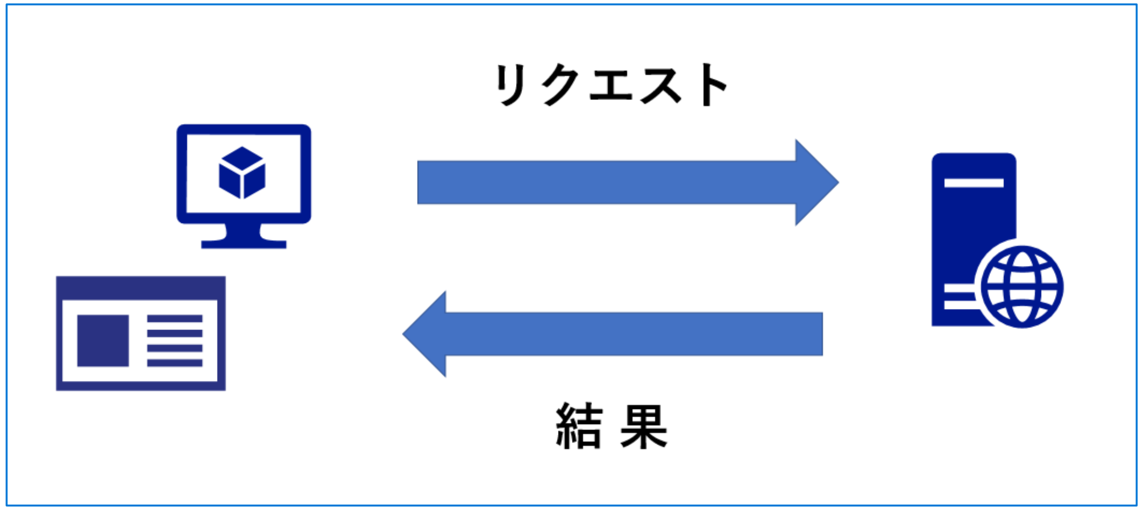
認証：リクエスト元を特定する
認可：アクセスを許可する
監査：活動内容、適正なリクエストか
流量制限：リソースの保護
提供：APIに関する情報を公開
請求：利用負担を求める
開示：利用状況を見せる



さらに

スケーラビリティ
セルフサービス化（トークンの発行、ポータルサイト）
機能強化（バージョン管理）
SDKの提供

などなど



ビジネス要件を含めて考えるととても大変だ

非機能要件が実は非常に多い

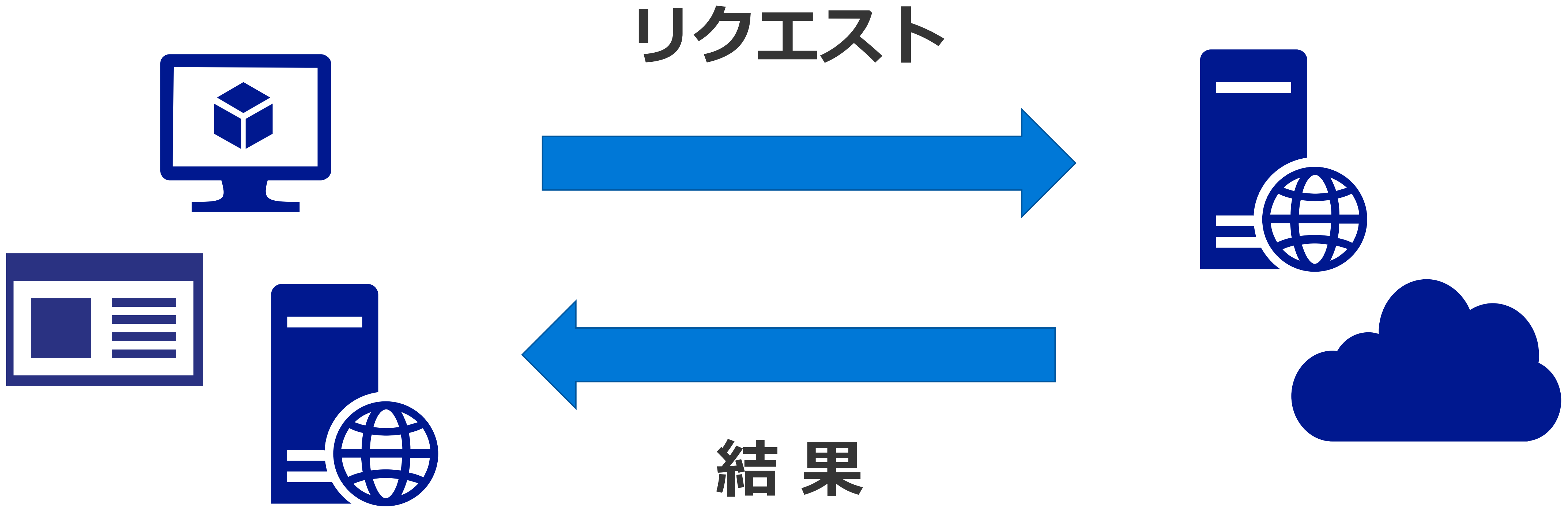
機能要件を満たすスピードをどう出すか

標準化は無視できない

Web APIの動向

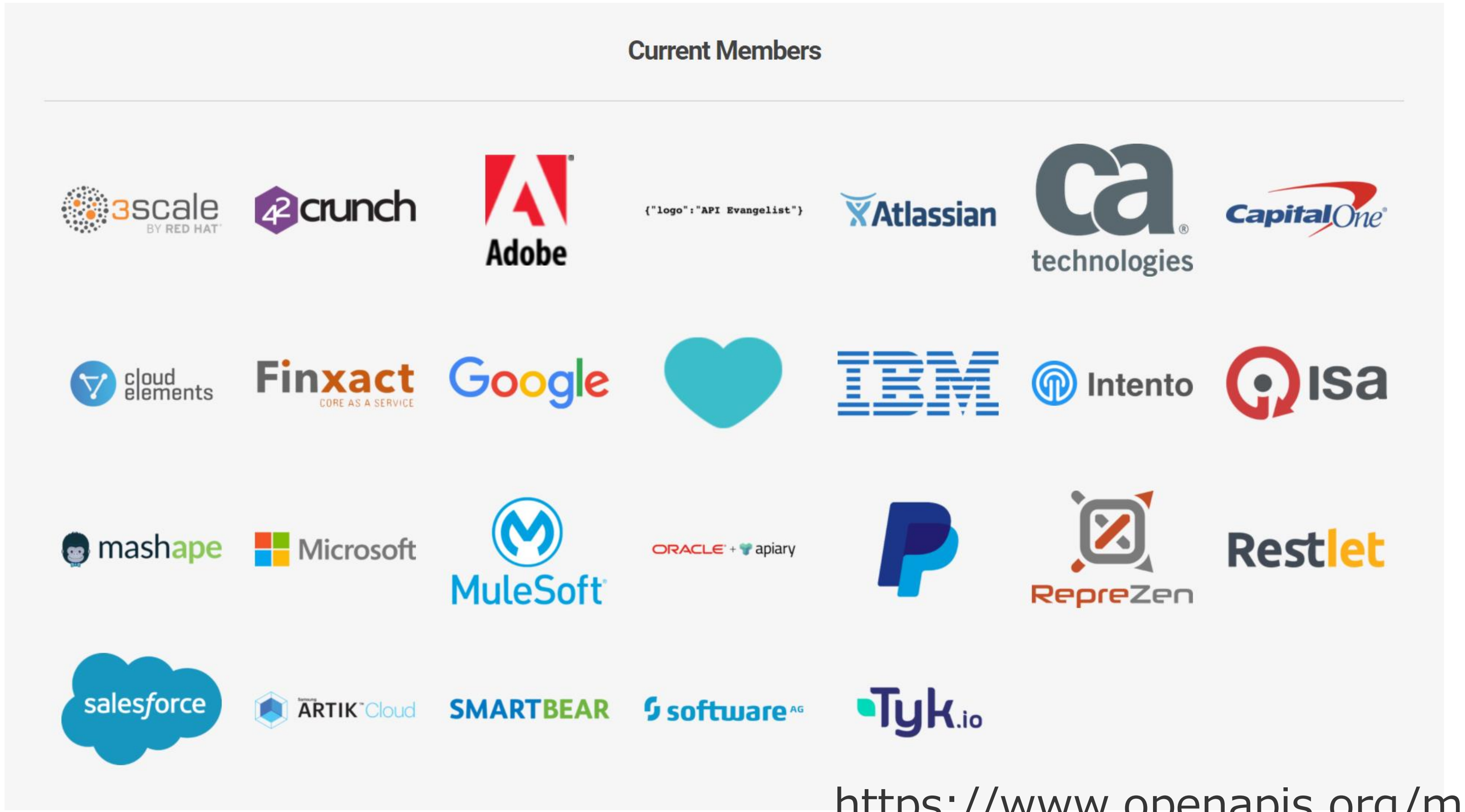
Web API

たいへん“緩い“ & かなり”大変”

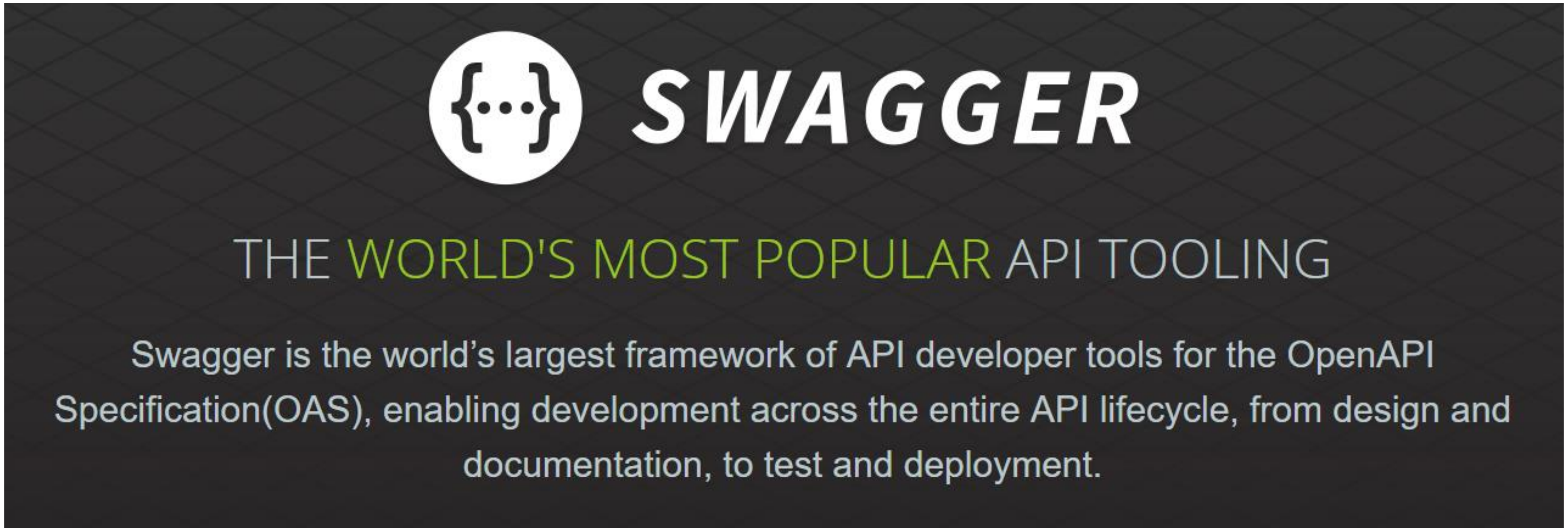


OPEN API INITIATIVE

The Open API Initiative (OAI) was created by a consortium of forward-looking industry experts who recognize the immense value of standardizing on how REST APIs are described.
As an open governance structure under the Linux Foundation, the OAI is focused on creating, evolving and promoting a vendor neutral description format.



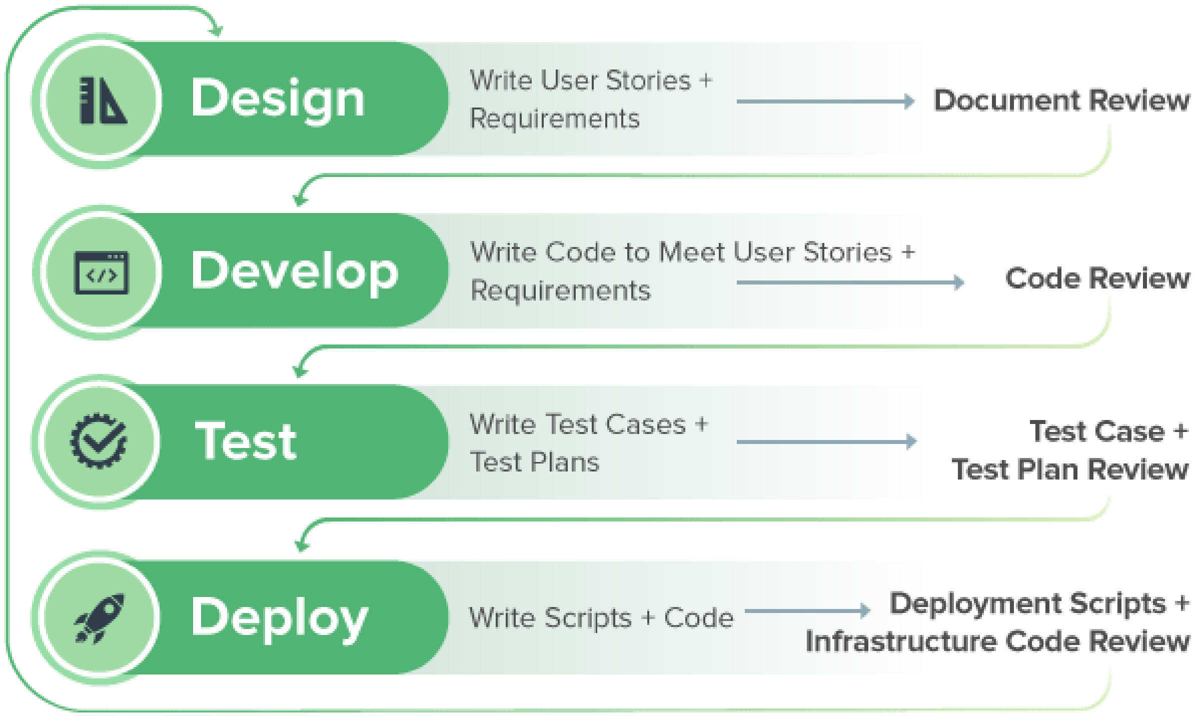
https://www.openapis.org/membership/members



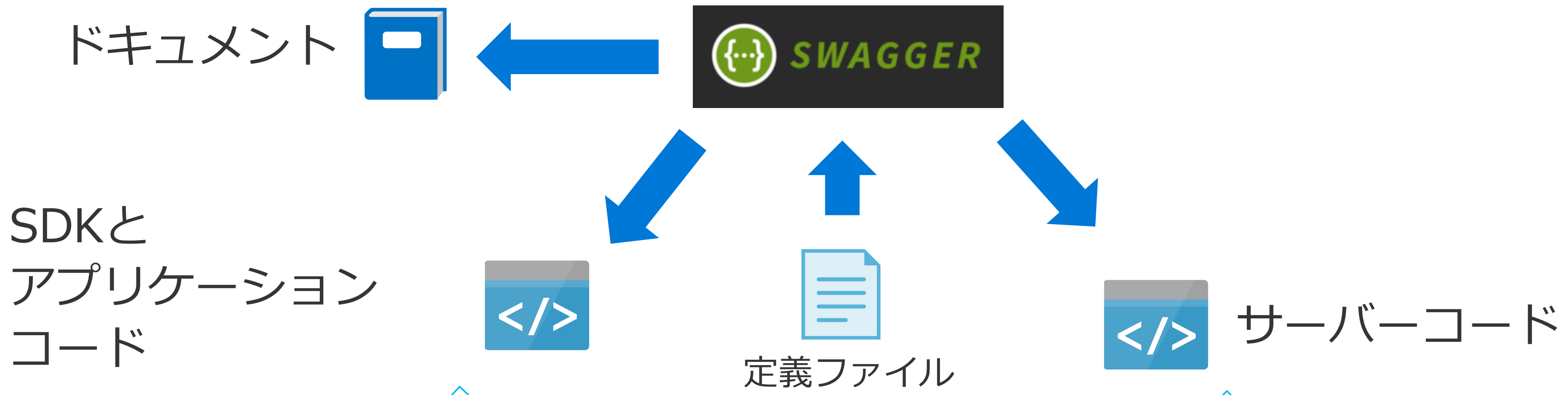
SWAGGER

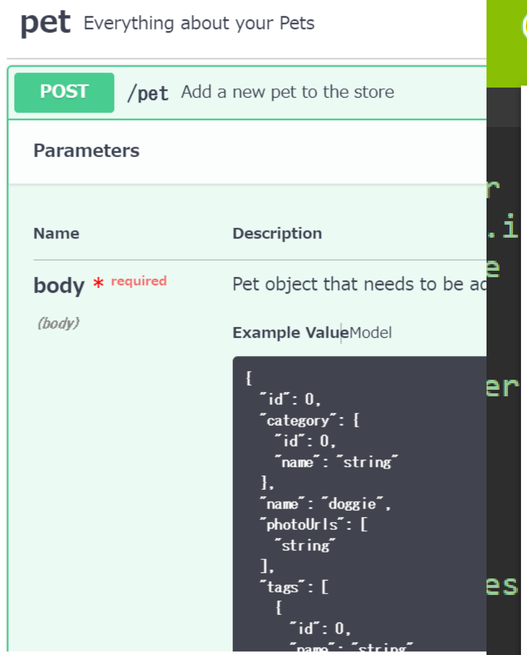
THE WORLD'S MOST POPULAR API TOOLING

Swagger is the world's largest framework of API developer tools for the OpenAPI Specification(OAS), enabling development across the entire API lifecycle, from design and documentation, to test and deployment.


Design
Write User Stories + Requirements
Document Review
Develop
Write Code to Meet User Stories + Requirements
Code Review
Test
Write Test Cases + Test Plans
Test Case + Test Plan Review
Deploy
Write Scripts + Code
Deployment Scripts + Infrastructure Code Review

https://swagger.io/





pet Everything about your Pets

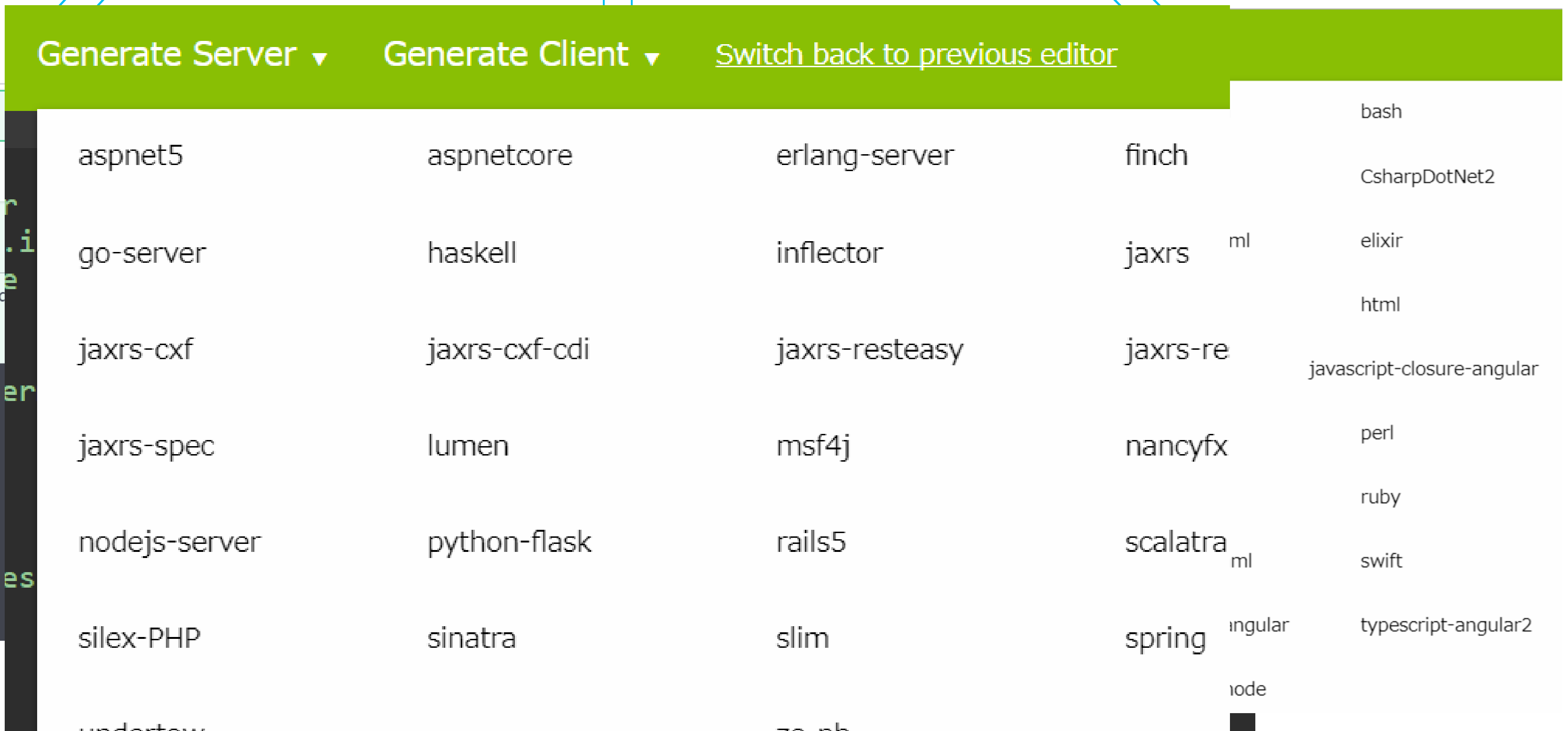
POST /pet Add a new pet to the store

Parameters

Name	Description
body * required (body)	Pet object that needs to be ad

Example Value Model

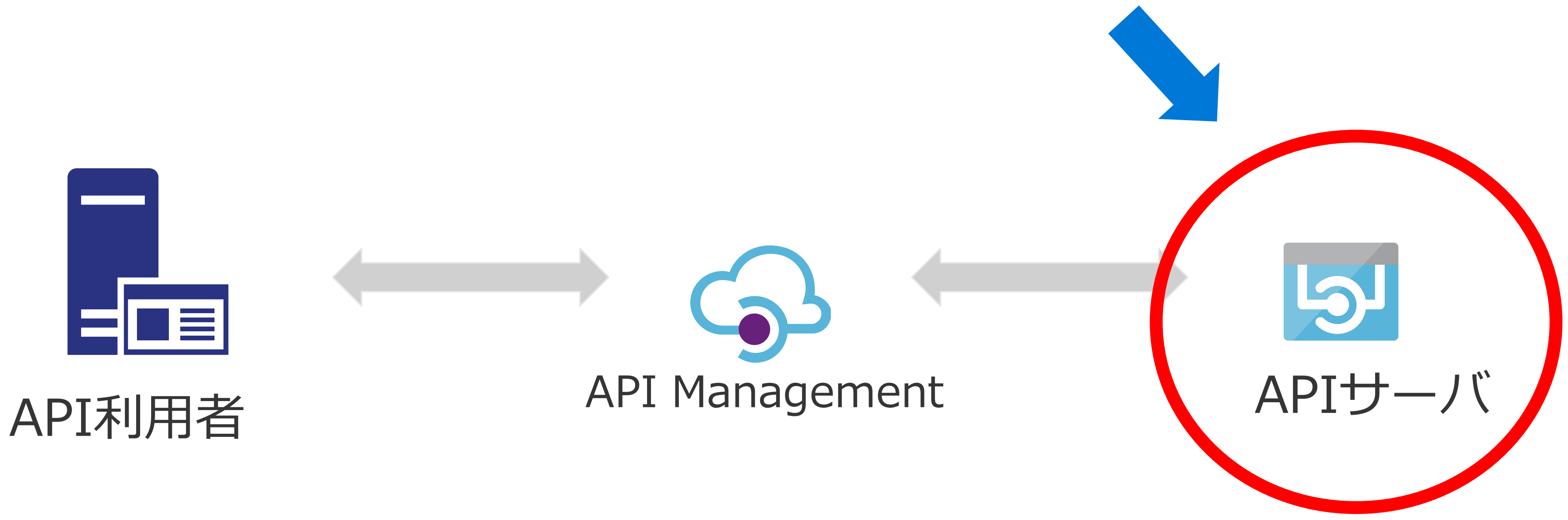
```
{
  "id": 0,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "doggie",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
```



aspnet5	aspnetcore	erlang-server	finch
go-server	haskell	inflector	jaxrs
jaxrs-cxf	jaxrs-cxf-cdi	jaxrs-resteasy	jaxrs-re
jaxrs-spec	lumen	msf4j	nancyfx
nodejs-server	python-flask	rails5	scalatra
silex-PHP	sinatra	slim	spring

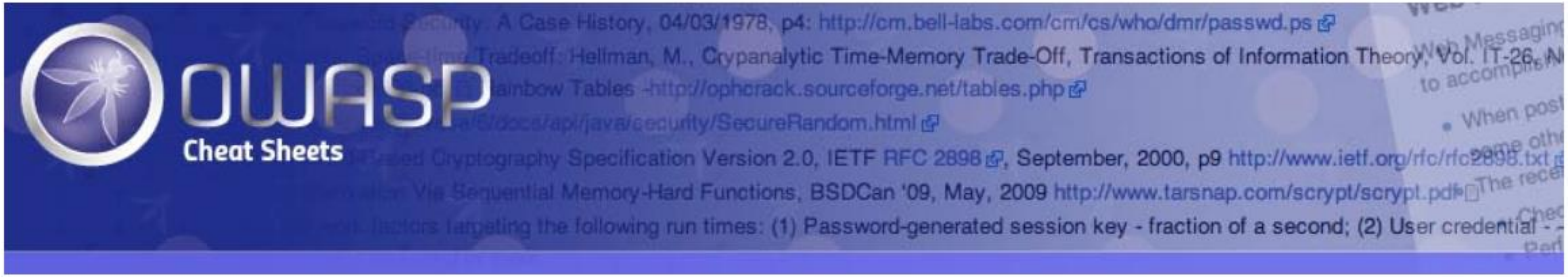
bash
CsharpDotNet2
elixir
html
javascript-closure-angular
perl
ruby
swift
typescript-angular2

Serverlessはここの実装手法の一つ



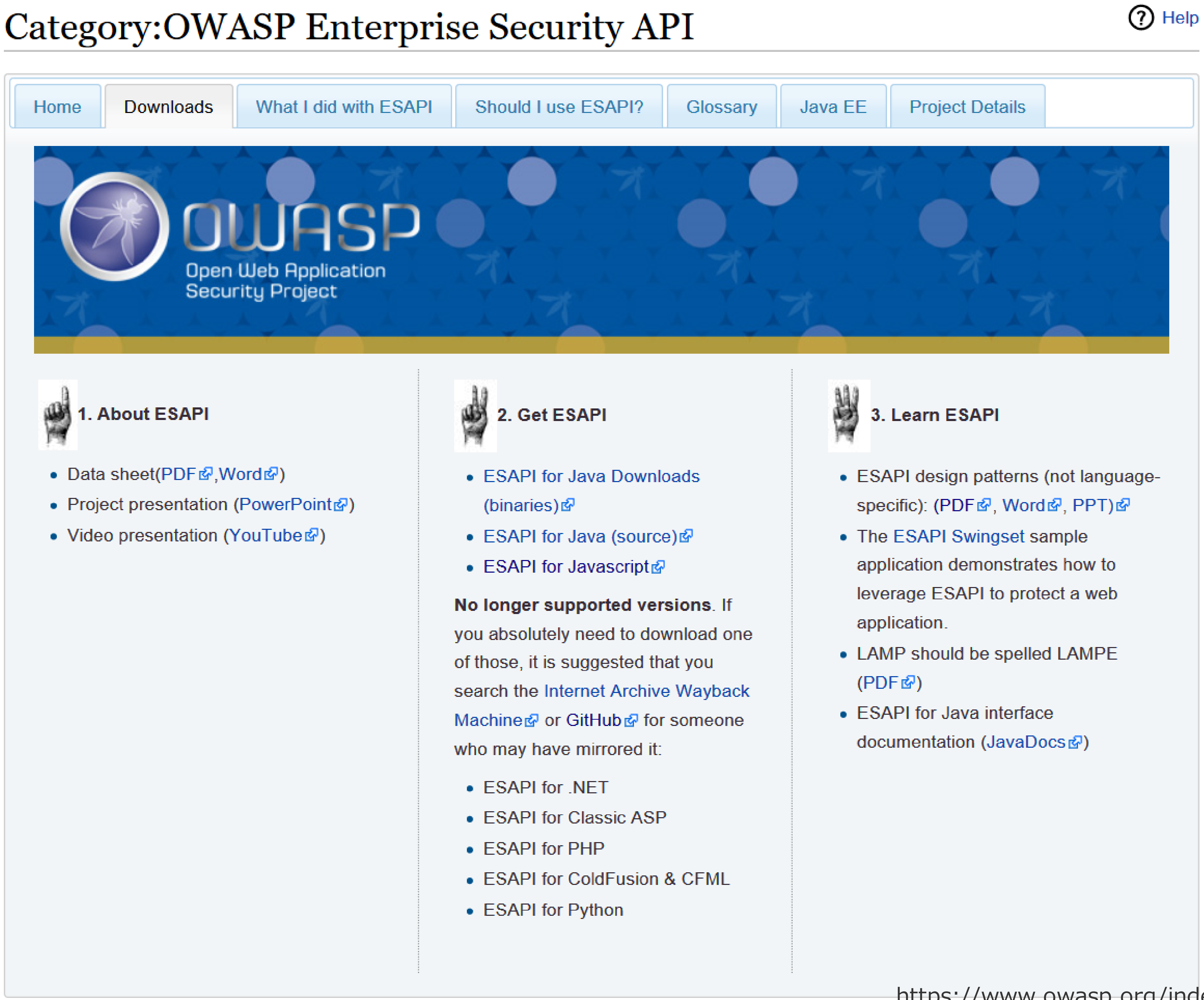
Web APIのセキュリティ

REST Security Cheat Sheet



Last revision (mm/dd/yy): **11/13/2017**

https://www.owasp.org/index.php/REST_Security_Cheat_Sheet



Category:OWASP Enterprise Security API

Help

Home | Downloads | What I did with ESAPI | Should I use ESAPI? | Glossary | Java EE | Project Details

1. About ESAPI

- Data sheet(PDF,Word)
- Project presentation (PowerPoint)
- Video presentation (YouTube)

2. Get ESAPI

- ESAPI for Java Downloads (binaries)
- ESAPI for Java (source)
- ESAPI for Javascript

No longer supported versions. If you absolutely need to download one of those, it is suggested that you search the Internet Archive Wayback Machine or GitHub for someone who may have mirrored it:

- ESAPI for .NET
- ESAPI for Classic ASP
- ESAPI for PHP
- ESAPI for ColdFusion & CFML
- ESAPI for Python

3. Learn ESAPI

- ESAPI design patterns (not language-specific): (PDF, Word, PPT)
- The ESAPI Swingset sample application demonstrates how to leverage ESAPI to protect a web application.
- LAMP should be spelled LAMPE (PDF)
- ESAPI for Java interface documentation (JavaDocs)

https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API

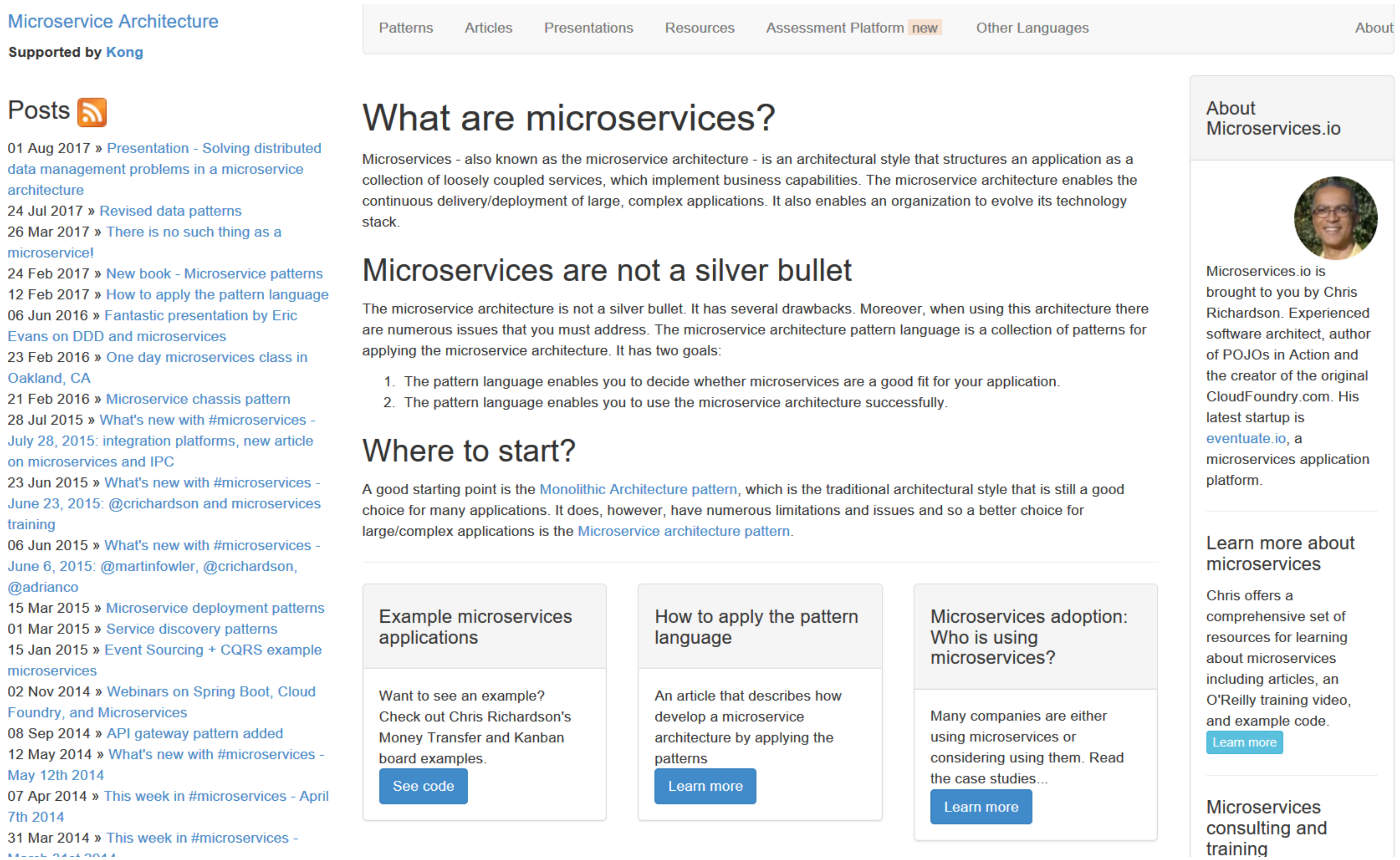
ベストプラクティス

API design
https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design

API implementation
https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-implementation

API Security Checklist
https://github.com/shieldfy/API-Security-Checklist

参考：RESTとHATEOAS
http://postd.cc/sprinkle-some-hateoas-on-your-rails-apis/



Microservice Architecture

Supported by Kong

Patterns | Articles | Presentations | Resources | Assessment Platform new | Other Languages | About

Posts

01 Aug 2017 » Presentation - Solving distributed data management problems in a microservice architecture
24 Jul 2017 » Revised data patterns
26 Mar 2017 » There is no such thing as a microservice!
24 Feb 2017 » New book - Microservice patterns
12 Feb 2017 » How to apply the pattern language
06 Jun 2016 » Fantastic presentation by Eric Evans on DDD and microservices
23 Feb 2016 » One day microservices class in Oakland, CA
21 Feb 2016 » Microservice chassis pattern
28 Jul 2015 » What's new with #microservices - July 28, 2015: integration platforms, new article on microservices and IPC
23 Jun 2015 » What's new with #microservices - June 23, 2015: @crichardson and microservices training
06 Jun 2015 » What's new with #microservices - June 6, 2015: @martinfowler, @crichardson, @adrianco
15 Mar 2015 » Microservice deployment patterns
01 Mar 2015 » Service discovery patterns
15 Jan 2015 » Event Sourcing + CQRS example microservices
02 Nov 2014 » Webinars on Spring Boot, Cloud Foundry, and Microservices
08 Sep 2014 » API gateway pattern added
12 May 2014 » What's new with #microservices - May 12th 2014
07 Apr 2014 » This week in #microservices - April 7th 2014
31 Mar 2014 » This week in #microservices -

What are microservices?

Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of loosely coupled services, which implement business capabilities. The microservice architecture enables the continuous delivery/deployment of large, complex applications. It also enables an organization to evolve its technology stack.

Microservices are not a silver bullet

The microservice architecture is not a silver bullet. It has several drawbacks. Moreover, when using this architecture there are numerous issues that you must address. The microservice architecture pattern language is a collection of patterns for applying the microservice architecture. It has two goals:

1. The pattern language enables you to decide whether microservices are a good fit for your application.
2. The pattern language enables you to use the microservice architecture successfully.

Where to start?

A good starting point is the Monolithic Architecture pattern, which is the traditional architectural style that is still a good choice for many applications. It does, however, have numerous limitations and issues and so a better choice for large/complex applications is the Microservice architecture pattern.

Example microservices applications

Want to see an example? Check out Chris Richardson's Money Transfer and Kanban board examples.

See code

How to apply the pattern language

An article that describes how develop a microservice architecture by applying the patterns

Learn more

Microservices adoption: Who is using microservices?

Many companies are either using microservices or considering using them. Read the case studies...

Learn more

About Microservices.io

Microservices.io is brought to you by Chris Richardson. Experienced software architect, author of POJOs in Action and the creator of the original CloudFoundry.com. His latest startup is eventuate.io, a microservices application platform.

Learn more about microservices

Chris offers a comprehensive set of resources for learning about microservices including articles, an O'Reilly training video, and example code.

Learn more

Microservices consulting and training

Deployment patterns

- Multiple service instances per host
- Service instance per host
- Service instance per VM
- Service instance per Container
- Serverless deployment
- Service deployment platform

http://microservices.io/index.html

まとめ

Web APIとして捉える

- サービス連携をデザインしよう
  - 実装方法は多岐にわたり、選択によって効果も異なる
- アプリケーションのインターフェースである
  - 従来からのセキュリティ対策
  - APIとしての特徴的な対策
  - 特性を理解して最適な対策を行おう
- サービスも活用（してみて下さい）