報道各位

2020年9月3日
株式会社インプレスR&D
https://nextpublishing.jp/

RISC-V ハードウェア構築言語 Chisel が手を動かしてわかる！

# 『クリーンアーキテクチャとサーバレスで実装する WebAPI』発行

技術の泉シリーズ、9 月の新刊

インプレスグループで電子出版事業を手がける株式会社インプレス R&D は、『クリーンアーキテクチャとサーバレスで実装する WebAPI』(著者:内山 浩佑)を発行いたします。

最新の知見を発信する『技術の泉シリーズ』は、「技術書典」や「技術書同人誌博覧会」をはじめとした各種即売会や、勉強会・LT 会などで頒布された技術同人誌を底本とした商業書籍を刊行し、技術同人誌の普及と発展に貢献することを目指します。

**『クリーンアーキテクチャとサーバレスで実装するWebAPI』**
https://nextpublishing.jp/isbn/9784844379058


著者:内山 浩佑
小売希望価格:電子書籍版 1600 円(税別)／印刷書籍版 2000 円(税別)
電子書籍版フォーマット:EPUB3／Kindle Format8
印刷書籍版仕様:B5 判／カラー／本文 66 ページ
ISBN:978-4-8443-7905-8
発行:インプレス R&D

**＜＜発行主旨・内容紹介＞＞**

本書では、すでにある程度実装されているサンプルプログラムを通して、サーバレス開発とクリーンアーキテクチャの理解を深めます。例として、サンプルプログラムをAWSにデプロイして、実際に動作しているところを確認できる状態にする、サンプルプログラムの構造を理解する、サンプルプログラムの改修すべきポイントを理解して、必要な機能を追加するといったものを取り上げます。

(本書は、次世代出版メソッド「NextPublishing」を使用し、出版されています。)

# サンプルプログラムのデプロイを解説

### Cloud Formationの画面で、スタックの内容を確認する

CloudFormationの画面に移動して、スタックが作成されていることを確認してみましょう。"clean-serverless-book-sample"という名前で、スタックが作成されていると思います。

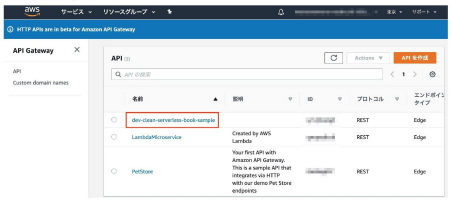
図2.3: CloudFormationの画面

### APIの動作確認する

API Gatewayの画面でエンドポイントを確認する

API Gatewayの画面に移動してください。以下のような画面が表示されます。

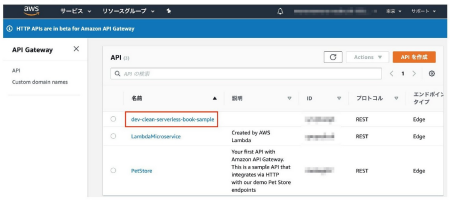
図2.5: API Gateway画面



先程のデプロイにより"clean-serverless-book-sample"というAPIが作成されていると思いますので、それをクリックして移動します。以下のような画面になります。



図2.5: APIリソース画面



左メニューの「ステージ」をクリックし、「dev」をクリックして、移動します。すると、APIのエンドポイントが表示されます。

図2.6: APIエンドポイント

このあと環境変数にセットする値ですので、コピーしておいてください。

### 環境変数にエンドポイントを設定する

curlコマンドで動作確認を行う際に、エンドポイントを参照するので、環境変数にセットしておきます。



# 新しい API の追加を解説

## 第4章　新しいAPIを追加する

### 概要

この章では、新しいAPIを追加する方法について、解説していきます。
新しいAPIの仕様は、次のとおりです。
・POST /v1/helloを受け付けること
・JSONで名前(name)を受け取ること`{"name":"Taro"}`
・名前(name)は、必須項目とすること
・名前(name)を含めたメッセージをJSONで返すこと`{"message":"HelloTaro"}`

### UseCaseを実装

UseCaseは、アプリケーションがどういった機能を持っているのかを表現したものでした。今回のアプリケーションは、「名前(name)を受け取って、メッセージを作成する」という機能を持っているとします。UseCaseの実装としては、あくまで「アプリケーションは何ができるのかを表現する」ことを表すために、interfaceで定義します。

**usecase/create_hello_message.go**という名前のファイルを作成し、以下のように実装します。

リスト4.2: Helloメッセージを作成するインターフェース(usecase/create_hello_message.go)

```
package usecase

// CreateHelloMessage Helloメッセージ作成
type CreateHelloMessage interface {
        Execute(req *CreateHelloMessageRequest)
           (*CreateHelloMessageResponse, error)
}

// CreateHelloMessageRequest Helloメッセージ作成リクエスト
type CreateHelloMessageRequest struct {
        Name string
}

// CreateHelloMessageResponse Helloメッセージ作成レスポンス
type CreateHelloMessageResponse struct {
        Message string
}
```



UseCaseでは機能の実装はせずに、インターフェースを定義します。また、引数や返り値も構造体で定義します。Helloメッセージ作成機能は、どのようなインターフェースになるのかが明確になっていますね。

### Interactorを実装

Interactorは、UseCaseで定義したインターフェースを実装します。先ほどのインターフェースを実装してみましょう。

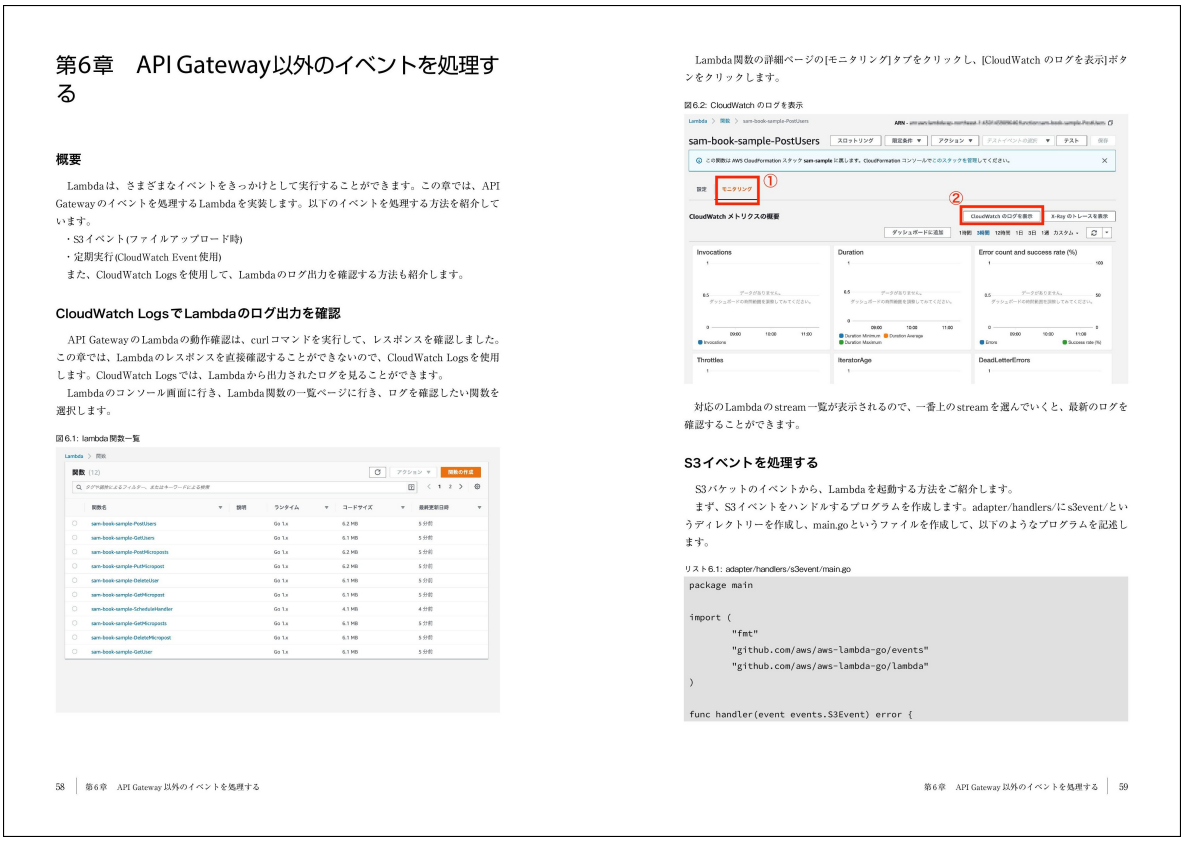
**interactor/create_hello_message.go**という名前のファイルを作成し、以下のように実装します。

リスト4.2: Helloメッセージを作成する機能を実装(create_hello_message.go)

```
package interactor

import (
        "clean-serverless-book-sample-draft/usecase"
        "fmt"
)

// CreateHelloMessage Helloメッセージ作成
type CreateHelloMessage struct {
}

// NewCreateHelloMessage CreateHelloMessageインスタンスを生成
func NewCreateHelloMessage() *CreateHelloMessage {
        return &CreateHelloMessage{}
}

// Execute 実行
func (c *CreateHelloMessage)
 Execute(req *usecase.CreateHelloMessageRequest)
   (*usecase.CreateHelloMessageResponse, error) {
        msg := fmt.Sprintf("Hello!%s", req.Name)
        return &usecase.CreateHelloMessageResponse{
          Message: msg,
        }, nil
}
```

Executeメソッドの実装を見てみると、どういった処理を行っているのか分かりますね。アプリケーションが実行される際は、このメソッドが実際に実行されます。

## 新しい Model の追加や APIGateway 以外のイベントの処理も解説



第6章　API Gateway以外のイベントを処理する

**概要**

Lambdaは、さまざまなイベントをきっかけとして実行することができます。この章では、API Gatewayのイベントを処理するLambdaを実装します。以下のイベントを処理する方法を紹介していきます。

・S3イベント(ファイルアップロード時)
・定期実行(CloudWatch Event使用)

また、CloudWatch Logsを使用して、Lambdaのログ出力を確認する方法も紹介します。

**CloudWatch LogsでLambdaのログ出力を確認**

API GatewayのLambdaの動作確認は、curlコマンドを実行して、レスポンスを確認しました。この章では、Lambdaのレスポンスを直接確認することができないので、CloudWatch Logsを使用します。CloudWatch Logsでは、Lambdaから出力されたログを見ることができます。

Lambdaのコンソール画面に行き、Lambda関数の一覧ページに行き、ログを確認したい関数を選択します。

図6.1: lambda関数一覧

58 | 第6章　API Gateway以外のイベントを処理する

Lambda関数の詳細ページの[モニタリング]タブをクリックし、[CloudWatchのログを表示]ボタンをクリックします。

図6.2: CloudWatchのログを表示

対応のLambdaのstream一覧が表示されるので、一番上のstreamを選んでいくと、最新のログを確認することができます。

**S3イベントを処理する**

S3バケットのイベントから、Lambdaを起動する方法をご紹介します。

まず、S3イベントをハンドルするプログラムを作成します。adapter/handlers/にs3event/というディレクトリーを作成し、main.goというファイルを作成して、以下のようなプログラムを記述します。

リスト6.1: adapter/handlers/s3event/main.go

```
package main

import (
        "fmt"
        "github.com/aws/aws-lambda-go/events"
        "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.S3Event) error {
```

第6章　API Gateway以外のイベントを処理する | 59

## ＜＜目次＞＞



## ＜＜著者紹介＞＞


内山 浩佑

1983 年埼玉県生まれ。愛知県岡崎市在住。全社員フルリモートワークの会社で、ソフトウェアエンジニアとして勤務。最近は試行錯誤しながらサーバレス開発を行う。AWS Certified Solutions Architect – Professional、AWS Certified DevOps Engineer – Professional 保持。


## ＜＜販売ストア＞＞

電子書籍:

Amazon Kindle ストア、楽天 kobo イーブックストア、Apple Books、紀伊國屋書店 Kinoppy、Google Play Store、honto 電子書籍ストア、Sony Reader Store、BookLive!、BOOK☆WALKER

印刷書籍:

Amazon.co.jp、三省堂書店オンデマンド、honto ネットストア、楽天ブックス

※　各ストアでの販売は準備が整いしだい開始されます。
※　全国の一般書店からもご注文いただけます。

**【インプレス R&D】** https://nextpublishing.jp/

株式会社インプレスR&D（本社：東京都千代田区、代表取締役社長：井芹昌信）は、デジタルファーストの次世代型電子出版プラットフォーム「NextPublishing」を運営する企業です。また自らも、NextPublishing を使った「インターネット白書」の出版など IT 関連メディア事業を展開しています。

※NextPublishing は、インプレス R&D が開発した電子出版プラットフォーム（またはメソッド）の名称です。電子書籍と印刷書籍の同時制作、プリント・オンデマンド（POD）による品切れ解消などの伝統的出版の課題を解決しています。これにより、伝統的出版では経済的に困難な多品種少部数の出版を可能にし、優秀な個人や組織が持つ多様な知の流通を目指しています。

**【インプレスグループ】** https://www.impressholdings.com/

株式会社インプレスホールディングス（本社：東京都千代田区、代表取締役：松本大輔、証券コード：東証 1 部 9479）を持株会社とするメディアグループ。「IT」「音楽」「デザイン」「山岳・自然」「モバイルサービス」「学術・理工学」「旅・鉄道」を主要テーマに専門性の高いメディア&サービスおよびソリューション事業を展開しています。さらに、コンテンツビジネスのプラットフォーム開発・運営も手がけています。

**【お問い合わせ先】**

株式会社インプレス R&D NextPublishing センター
TEL 03-6837-4820
電子メール: np-info@impress.co.jp