はじめに

Web アプリケーション開発へようこそ。

本書では、HTML/CSS/JavaScript による基礎的な Web ページに Web フレームワークとデータベースを組み合わせることで、ユーザーデータを活用した、よりダイナミックな Web アプリケーションの構築方法を学びます。Web フレームワークのプログラミング言語には Python を用います。

多様な技術を利用する Web アプリケーションの開発環境は準備が面倒ですが、本書では PaaS（Platform as a Service）を利用することで、この手間を省きます。PasS は、ちょっとした設定とコードのアップロードだけでアプリケーションをインターネットに展開できる優れもののクラウドサービスです。

巷にはいろいろな PaaS サービスがありますが、本書では **PythonAnywhere** を使います。

Python のパッケージマネージャで有名な Anaconda が運営しているこのサービスは、次の環境を無料で提供してくれます。

- ホストプラットフォーム：Linux（Ubuntu）
- Web サーバ：nginx
- データベースシステム：MySQL（有料アカウントなら PostgreSQL も可）
- 開発言語：Python
- Web フレームワーク：Flask（Django、web2py、Bottle も利用できますが、本書では Flask を用います）
- ネットワークアクセス：インターネット全域からアクセス可。ドメイン名は ユーザ名 + `pythonanywhere.com`。

アカウントは誰でも作れます。大手クラウドプロバイダのように、無料枠は提供しても、ちゃっかりクレジットカードを登録させることもありません（なので中学生、高校生も使えます）。サーバ証明書も含めてセキュリティも備わっているので、安心してインターネットでサービス運用で

† PythonAnywhere ロゴは当該社より許可を得て転載。

きます。Python 公式サイトのオンラインコンソールの提供元なので、技術力も確かです。たいていのサードパーティパッケージは用意されているので、インストールの手間もかかりません。

例題には、飲食店の注文管理アプリケーションを取り上げます。架空の大学の架空のクラブの架空の学園祭模擬店で使うという設定で、注文票ページはこんな感じです。

目的は、Web アプリケーションの多様な要素技術がどのように連携しているかを学ぶところにあります。

初心者プラスアルファな読者をターゲットにしているので、凝ったプログラミングはしません。画面は HTML、CSS、JavaScript のベーシックな機能だけで作成します。バックエンドの Python プログラミングもデータベースシステム（SQL）も同様で、シンプルで短いものです。

個々の要素は簡単でも、それらの組み合わせとなると急に見通しが悪くなります。そこで、本書では簡単な技術から始めて、段階的に新技術を加えて作り込んでいきます。最初は HTML/CSS/JavaScript だけで基本画面を作成します。続いて Web フレームワークを加え、最後にデータベースの機能を加えます。どの段階でも、一部だけでもサービスは利用できます。こうすることで、利用する技術とそれが可能にするサービスの関連がつかめると思います。

これを機会に Web アプリケーション開発の基本をマスターしてもらえたら幸いです。一歩が踏み出せれば、そこから先、さらに高度な開発に抵抗なしに進むこともできるはずです。

2025 年 8 月

豊沢 聡

◆ 本書の目標

本書は、Web アプリケーションを作成しながら基盤技術を学ぶことを目標としています。各種技術を系統立てて網羅するのは分量的に無理なので、説明は例題の構築に必要な範囲にとどめています。

本書を読了できたら、Web 開発者初級コースを修了したと思ってください。

本書の章立てを次に示します。

- **第 1 章「Web アプリケーションとは」**
  Web アプリケーションとその要素技術を簡単に説明します。そのうえで、本書で作成する例題の注文管理アプリケーションの構成を示します。
- **第 2 章「PythonAnywhere を使う」**
  開発プラットフォームの PythonAnywhere の用法を示します。この章では、アカウントの作成および管理の方法、無料版と有料版の違い、Web インタフェース（ダッシュボード）の使い方を説明してから、Web アプリケーションを起動します。お仕着せのテンプレートを使った簡単なものですが、インターネットのどこからでもアクセスできる、立派な Web アプリケーションです。手持ちのスマホからでも試せます。
- **第 3 章「静的ページを作成する」**
  HTML/CSS/JavaScript を使って、クラブ紹介と注文を受け付ける注文票のページを作成します。Web 開発でいうところのフロントエンドの部分です。
- **第 4 章「フレームワークで動的ページを作成する」**
  Flask Web フレームワークを組み込むことで、注文票からのデータを受け付ける注文確認ページを作成します。また、複数の HTML ページのデザインと保守が容易になるよう、HTML ページをテンプレートを使ってまとめます。Web 開発でいうバックエンドです。
- **第 5 章「データベースを活用する」**
  データベースシステムを導入します。注文確認ページから注文データをデータベースに挿入し、注文履歴ページから閲覧できるようにします。この章では、MySQL データベースシステムの起動、データベース表の作成や挿入の方法を説明します。操作方法は、PythonAnywhere の MySQL コンソールのものを先に示してから、Python API（Python モジュール）を介したものを説明します。
- **付録**
  本文では扱えなかったトピックを取り上げます。これには、ローカルで開発・運用する方法（付録 A）、先生と生徒でスクリーンやシステムを共有する機能など PythonAnywhere のより高度なサービスの使い方（付録 B）、Flask の追加説明（付録 C）が含まれます。付録 D には、出版社ダウンロードサービスから入手できる本書コードのリストを示します。

◆ 本書の対象

本書は、Web アプリケーション開発に興味のある、技術的には初心者レベルプラスアルファな方を対象に書いています。

モデル読者は、基礎技術である HTML/CSS/JavaScript と初級レベルの Python プログラミングは学び終え、モダンでプラクティカルな開発に進もうとしている生徒・学生です。大学でいえば、2 年次の中級コース（200 番台）という位置付けですが、向学心のある中学・高校生でも理解できるレベルで書いています。

◆ 必要な事前知識

HTML、CSS、JavaScript、Python の基礎は既知であることを前提としています。`<a>` や `<form>` といった HTML タグ、`text-decoration` のような CSS のプロパティ、制御文やデータ型 / コンテナオブジェクトなど初級コースで学ぶ技術は説明しません。中級レベル以上の話題はその都度補足します。

HTTP のメッセージ交換方法（メソッド、ステータスコード、メッセージの構造など）や URL の読み方は、知っていれば本書の理解も早いでしょうが、必須ではありません。本書の例題を開発するに際して必要最小限の情報は第 1 章で説明します。

Web サーバには一切触れないので、何も知らなくても問題ありません。どのみち、PythonAnywhere ではユーザに透過的です（nginx が使われていることにすら気付きません）。

Web フレームワークの Flask は最初から説明します。ただし、本書の開発順序に沿って必要になったときに必要な事柄のみをピンポイントに説明します。Flask の体系的な説明が入用なら、簡潔にして明瞭な Flask 公式ページの Quickstart（クィックスタート）がお勧めです。A4 印刷すれば 16 ページほどです。英語ですが、機械翻訳サービスを通せば、大意はつかめます。全体像を把握したいなら、専門書を紐解いてください。本書読了後なら、高度なテクニックも、すんなりと頭に入るはずです。

MySQL データベースシステムも同様に、くどくならない程度に説明を加えます。その代わり、本書で使わない機能にはほとんど触れないので、本書でデータベースシステムの全容がつかめるかというと、そんなことはありません。コンソール（`mysql` クライアントのインタラクティブモード）を使うので、コマンドラインの使用経験があると楽ですが、必須ではありません。

PythonAnywhere は仮想 Linux プラットフォームですが、ファイル操作は Web インタフェース（GUI）から行うので、Bash など Unix の知識は不要です（付録では使いますが）。

◆ 授業モデル

本書を授業で使うときは、生徒・学生のぶんだけ PC を用意します。PythonAnywhere 上のエディタを使えばタブレットやスマホでも入力や編集ができますが、ローカルで編集し、アップロードすると効率がよいからです（Web インタフェースの応答速度は必ずしもよくはない）。

先生登録してもらえれば、先生が生徒の環境を直接操作できます。Linux 上でおかしな操作をしてしまった、データベース表が壊れたなどなど、初心者が犯しがちなミスを修正するときに便利です。スクリーンの共有もできるので、生徒の画面を教場に映すこともできます。方法は付録 B を参照してください。

PythonAnywhere の Linux 仮想環境は制限付きなので、よほど破壊的な操作をしない限り、壊れはしません。第一、Linux にもデータベースシステムにもユーザには管理者権限が与えられないので、壊すにしても限りがあります。安心して使ってください。

本書の分量は、付録を除いて半期分（15 週）程度です。

◆ サンプルコード

本書では、Web アプリケーションをシンプルなところから書き始め、章を追うごとに順次機能を追加していきます。そのため、前に作成した部分を別のものに置き換えるなど、非効率な開発をしています。これは、意図的です。最終版にストレートに進めば効率的かもしれませんが、相互に関係しあった複数の構成要素を一気に把握するのは、慣れないうちは難しいと考えたからです。

本書のコードは、見た目での読みやすさを優先して書かれています。より効率的、あるいはより慣用的なやり方があっても、初心者にわかりやすい書き方があれば、そちらを採用しています。もっとも、「読みやすい」は主観的なクライテリアなので、わかりやすいとは思えないものもあるかもしれません。読者の宥恕を請うばかりです。

「コード 2.1」のように番号が付せられたコードをパッケージ化したファイルは、出版社ダウンロードサービスからダウンロードできます。これには HTML、CSS、JavaSctript、Python、Jinja テンプレート、ファビコン画像が含まれています。パッケージの中身は、付録 D を参照してください。

1.1 Web アプリケーションの概要

Web アプリケーションとは何か、これまでの Web 技術とはどこが違うのかを簡単に説明します。

◆ Web アプリケーションとは

Web アプリケーションは、既存の Web 技術の上に構築されるアプリケーションです。

ここで「既存の Web 技術」は、Web サーバがブラウザの要求に応じて各種のコンテンツを提供するサービスです。データ搬送の通信プロトコルは必ず HTTP です。コンテンツは固定的で、いつ見ても誰が読んでも（変更されていなければ）同じ中身です。情報の流れがサーバからブラウザへの一方通行なので、ユーザとの柔軟なインタラクションは期待できません。

Web アプリケーションは、ユーザの操作や設定に応じてダイナミックに変化するコンテンツの生成とブラウザとの複雑な情報交換を Web サーバの背後で行うことで、豊かなサービスを提供するものです。

しかし、Web の基本的な枠組みはそのまま流用しています。コンテンツの搬送には HTTP を、表示や操作のインタフェースにブラウザを使うことに変わりはありません。高度なサービスを達成しているのは、Web サーバの背後で情報を処理する補助プログラムです。方法は昔からいろいろ考案されてきましたが、「Web アプリケーション」といったとき、最近では、**Web フレームワーク**†というプログラミング環境を使って構築したサービスを指します。

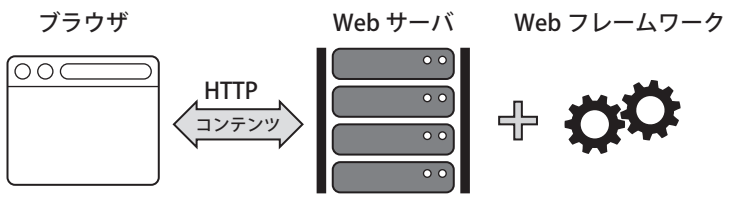

図 1.1　Web アプリケーションは従来の Web 技術と Web フレームワークとの組み合わせ

開発ツールとして見た Web フレームワークは、Web アプリケーションで共通した処理をあらかじめパーツ化して提供するライブラリです。一般的な機能ならイチから作らなくて済むので、手間が省け、ミスも減ります。半面、パーツの中身にまで立ち入った操作が難しいというデメリットもありますが、お仕着せが合わないのは、よほど細かいことが必要になるときくらいです。

† Web フレームワーク（あるいは Web アプリケーションフレームワーク）はフロントエンドとバックエンド（あるいはクライアントサイドとサーバサイド）のどちらの機能も含む語として使われることもありますが、本書ではバックエンドを指すものとしています。

◆ Web アプリケーションのメリット

Web アプリケーションのメリットは、ブラウザとリンク先の URL だけですべての用が足りるという使いやすさにあります。ソフトウェアのインストールやアップデートも不要なので、ユーザだけでなく、システム管理者の手間も省けます。

インストールして使う一般のアプリケーションと違って OS に依存しないので、プログラムの移植も必要ありません。そのまま、スマホも含めた多くのプラットフォームで使ってもらえます。

データの大半はサービス運営側に置かれるので（たとえば Google Docs の文書は Google に置かれる）、デバイスを変えても継続利用できます。ファイルや設定をコピーしなければならない一般のアプリケーションより便利です。デバイスがクラックされてもデータが漏洩しないという、セキュリティ上のメリットもあります。

◆ Web アプリケーションのデメリット

ネットがなければ使えない、というのが一番のネックでしょう。テレワーク中にネットが落ちれば、仕事になりません。そうそう落ちるものでもないですが、電波が届かなかったり、速度が出なかったりするところは、今もところどころにあります。快適につながっても、契約以上にパケットを使ってしまったら、お財布に厳しくなります。

セキュリティに関しては、運営側がクラックされたら被害が大規模になるのは、しばしばニュースで見るとおりです。

Web アプリケーションの中には、本当に必要かと疑うほど個人情報を訊いてくるサイトもあります。課金されない範囲で使おうと思っているのにクレジットカードまで登録させられるのには、気が乗らないものです。

◆ Web アプリケーション開発の複雑さ

従来型の Web サービス開発は、HTML、CSS、あるいは JavaScript でページを記述するだけのシンプルなものでした。メモ帳とブラウザさえあれば、開発と動作確認までできました。インターネットに公開するにしても、組織の Web 担当やホスティングサービスにファイルを送るだけです。今も、固定的なページだけの Web サイトはそのように運用されています。

しかし、Web アプリケーションはそう簡単ではありません。Web フレームワークを動作させる言語環境とプラットフォームが必要ですし、たいていの場合は、それにデータベースシステムが加わります。そうしたインフラストラクチャの管理運営は、それだけでプロの管理者を雇わなければならないほどハードなものです。ちょっと遊んでみる、というわけにはいきません。

◆ PythonAnywhere

もっとも、昨今では、自前でプラットフォームを用意する必要はありません。すぐに使えるサーバやデータベースシステムを提供するクラウドサービスがあるからです。とくに、PaaS（Platform as a Service）と呼ばれるサービスならネットワーク、OS、Web サーバ、Web フレームワークと言語環境、データベースをひとまとめに提供してくれます。

PaaS のプロバイダは数多くありますが、本書ではオンライン Python 開発環境とホスティングサービスを提供する PythonAnywhere を利用します。

図 1.2　PythonAnywhere のロゴ〜ヘビのマークは Python の証

開発環境は素朴ですが、これ 1 つで Web アプリケーションを開発できるところが魅力です。データベースシステムも、最も利用されているリレーショナルデータベースシステムの 1 つである MySQL が使えます。Python については、ポピュラーなサードパーティパッケージが数多くプリインストールされているので、追加でインストールする必要はほとんどありません。無料プランがあり、クレジットカード登録もなしで気軽に使い始められます。

知名度はまだ高くはありませんが、Python 公式サイトのトップに置かれたインタラクティブコンソールに採用されているくらい、Python エンジニアに信頼されています。

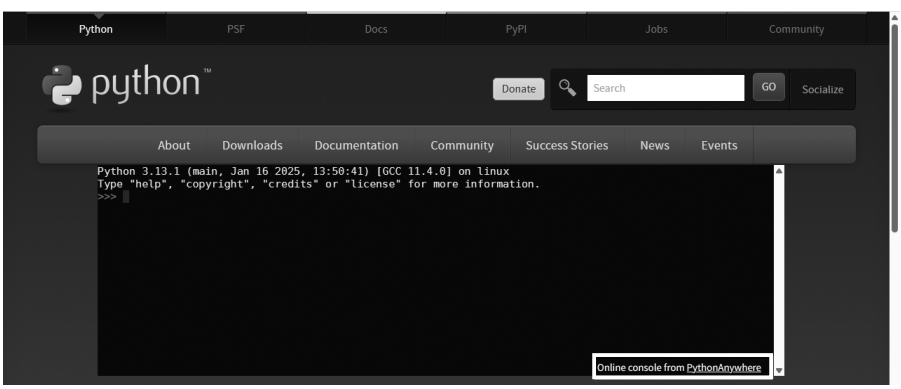

図 1.3　Python トップページの PythonAnywhere

サービス提供元は、Python のパッケージマネージャで知られている Anaconda です（2022 年に買収）。

PythonAnywhere は既存のクラウドサービス上に構築されています（Amazon EC2）。

1.2 Web アプリケーションの構成

Web アプリケーションはいろいろな技術を組み合わせて作成・運用されます。本節ではこれら技術を、本書の Web アプリケーションに必要な範囲で簡単に説明します。データベースシステムは Web システムから独立した 1 つのシステムなので、次節で扱います。

◆ Web アプリケーション技術

Web アプリケーションの構成とそこで使う要素技術を次の図に示します。

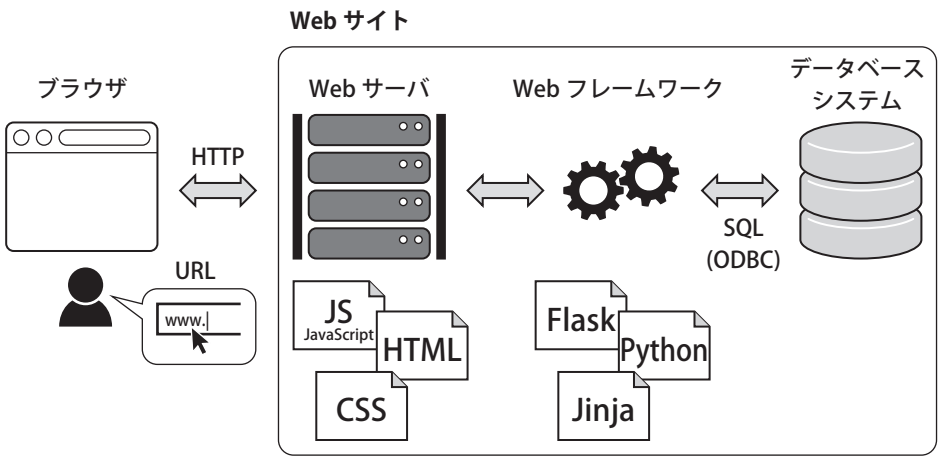

図 1.4 Web アプリケーションの技術

Web アプリケーションは、ユーザにはブラウザとアクセス先の URL です。

ブラウザは、通信メカニズムの HTTP を介して Web サイトにつながっています。

サイトの中には Web サーバ、Web フレームワーク、データベースシステムがあります。Web サーバがブラウザと直接やり取りをする、いってみれば店員さんです。Web フレームワークは直接お客さんとは関わらない、商品を手配する裏方さんです。データベースシステムはこの伝でいくと倉庫で、Web サイトはこれらをひっくるめてお店全体です。

Web サーバとフレームワークはたいていは一体となって同じコンピュータ上に置かれ、データベースシステムは少し離れた別のコンピュータで運用されます。PythonAnywhere でも同様です。もっとも、クラウドでは実体はどこにあってもよいので、所在を気にすることはまずありません。

◆ URL

URL はよく見かける `https://www.python.org/` などの文字列で、インターネット上のコンテンツの所在とアクセス方法をピンポイントに指し示す識別子です。

URL では、HTML ファイルや画像データ、Web フレームワークが自動生成するデータなど、多様な方法や表現で提供するコンテンツを**リソース**と総称します。そのことは、URL のもとの語である Uniform *Resource* Locator の真ん中の単語に現れています。

URL はスキーム、ドメイン名、URL パスの 3 つの情報で構成されています。

図 1.5　URL の構造

スキームはリソースを取得するときの通信方法を示し、Web では http か https のどちらかです。通信プロトコルはどちらも HTTP ですが、前者はデータをそのまま流す平文版、後者はデータを暗号化する暗号版です（末尾の s は安全の secure）。PythonAnywhere の Web サーバはどちらもサポートしていますが、通信の安全性が保証されない前者を使う理由はあまりありません。

スキーム直後の : と // は続くドメイン名との間を示す記号で、それ自体に意味はありませんが、必須です。

ドメイン名は Web サイトを特定するための名前で、広いインターネットで同じ名前は 1 つもないユニークなもので、たとえば `www.example.com` です。

URL パスは、Web サイト内部ローカルでのリソースの所在を示します。図の `/top/page.html` は、そのサイトのトップのディレクトリ / にある、`top` サブディレクトリの中にある、`page.html` というファイルを指します。仕様は単に「パス」と呼んでいますが、OS ファイルシステム上のローカルなパスと紛らわしいので、本書では「URL パス」と書きます。

◆ 自分のドメイン名

ドットで分けられたドメイン名の要素を**ラベル**といいます。`www.example.com` では `www`、`example`、`com` がラベルです。

ラベルで使える文字はアルファベット（a-z）、数字（0-9）、ハイフン（-）に限られます（いずれも半角）。ハイフンはラベルの先頭と末尾には置けません（たとえば `-ww` は認められない）。大文字小文字は問わないので、`www` も `WwW` も同じラベルと認識されます。最大長は 63 文字です。

PythonAnywhere は、2.1 節で作成するあなたのユーザ名をそのまま最初のラベルに使うこと