〈研 究 論 文〉

Web アプリケーション開発を題材に用いた IT 教育の導入に至る道程

飯　尾　　淳

A Roadmap Leading to the Introduction of IT Education using Web Application Development

Jun Iio

Abstract

Many students in our university feel anxiety that they cannot develop applications even though they have learned the basics of programming well in the first grade. However, this is not surprising since it is necessary to learn not only programming but also various aspects of the system to develop an application. In this paper, we focus on the "interface" and describe the results of our study of the path that should be followed from the basics of programming to Web application development exercises. The five stages are discussed, from the concept of functions, through CLI, bots, and GUI, to Web application development. Finally, we discussed the effectiveness of the sequence presented in this paper, points to be noted, and points lacking in the current study.

KeyWords

IT education, Web application development, five stages, programming skills.

目　　次

第 1 章　は じ め に

インターネットが Web2.0 の時代といわれるようになり久しく，現在ではインターネットを介した Web アプリケーションは生活の隅々まで浸透しているといっても過言ではない．そのような時代において情報技術（information technology, IT）をどのように教育するか，IT 教育をどう進めるかに関して，最新の技術をキャッチアップしつつ，根本的な概念の抜けや漏れのない教育が重要であることは明らかであろう．

IT 教育の題材に Web アプリケーション開発を用いることの背景には，次のような要素が存在する．

まず，デバイスの多様化である．マルチデバイスともいわれる．コンピュータのコモディティ化が進み，また，通信端末が極度に高度化したため，いまや個人用コンピュータ（パーソナル・コンピュータ，PC）とスマートフォン（スマホ）の性能にはほ

とんど差がなくなった．のみならず，PC とスマホの間を埋めるタブレット機器も登場し，日常的に利用されるようになっている．

ハードウェアの多様化に伴い，ソフトウェアもまた多様化している．代表的なものがオペレーティング・システム（OS）であり，Windows，macOS，iOS，Linux ベースの各種 OS など，様々な OS が日常的に利用されている．システム開発においては，これらの多様性を吸収した互換性の維持が重要となる（Iio and Ohgama, 2016）．IT 教育の対象となるプラットフォームの選択においても然りである．

この多様性を吸収するフレームワークとして有望と考えられる代表的なものが Web アプリである．Web アプリの本体はサーバ側で実行されるため，サーバの種類を特定してしまえば互換性を気にする必要はない．

問題なのはエンドユーザが利用するクライアントのほうであり，それらは多様なハードウェアとソフトウェアに対して配慮が求められるが，幸いなことに多くのデバイスと OS の上で Web ブラウザが標準装備されている．かつてはこの Web ブラウザに非互換問題が存在しており，その互換性をどうするかが課題となっていたが（Iio et al., 2010），現在ではそのような軽微な非互換はソフトウェアライブラリが吸収し，開発者はあまり気にせずともよくなっている．

さらに，最近ではクラウドコンピューティングのサービス（クラウドサービス）が当たり前のものとなり，とくに Amazon Web Service（AWS）や Google Cloud Platform（GCP）といった Infrastructure as a Service（IaaS）の利用や，Heroku など Platform as a Service（PaaS）の利用が一般化している．AWS や GCP といった IaaS を利用するか，あるいは PaaS を利用するかはケースバイケースであるとしても，これらのサービスが教育用に安価に提供されていることに鑑みても，今後はますますこのようなサービスの活用が重要になってくるであろう．

いずれにしても，様々なサービスが Web アプリとして提供されている現在においては，IT 教育の一つの目標として，Web アプリを開発しデプロイ，運用できるようになること，というマイルストーンが考えられる．Web アプリの技術自体も進化を遂げており，さらに進んだ教育を行う上でも，Web アプリ開発の基礎は目標点としては適切と考えられる．

ただし，一口に Web アプリ開発といっても実際にそれを実現するためには様々な前提知識が求められるため，簡単にその導入を進められないという問題が残る．

本論文の構成は以下のとおりである．第 1 章で本研究の背景について説明した．第 2 章では関連する先行研究を示し，本研究の位置付けを明確にする．第 3 章では「インタフェース」という概念に着目し，Web アプリ開発教育への導入過程について論じる．続く第 4 章では，各段階の詳細を示す．第 5 章で，本モデルがカバーできない範囲について補足し，今後の課題について示す．最後に第 6 章でまとめる．

第 2 章　関連研究

数年来，筆者らは「人間と情報システムのインタラクション」，すなわち，人間と情報システムの間でどのような相互作用があるべきか，また，そのインタフェースはどのようなものであるべきかという課題をテーマの中心に据えて研究活動を進めている．そのなかで，Web アプリを構築して実験の道具に利用するというパターンは多い（Iio and Wakabayashi, 2020; Hanagaki and Iio, 2023 など）．また，サイバー空間における人間の行動データを収集するような場合の利用も典型的な利用形態である（Iio, 2019; Iio, 2023 など）．

このような実際の演習に向けた教育を体系的にまとめた研究はさほど多くないが，たとえば Liang and Chapa-Martell（2018）による事例がある．彼らはクラウド環境を用いた Web アプリ開発演習を，トップダウンのアプローチで提案する．そこでは，準備段階，ウェブ技術の教育段階，応用プロジェクト段階という三つのフェーズで演習を進

める．

de Oliveira Fassbinder ら（2018）は，Web アプリ開発演習に関する massive open online course（MOOC）を提案した．彼らのコースは，学習設計フレームワーク（Learning De-sign Framework for MOOCs; LDF4MOOCs）に基づいて提案されている．

Web アプリ開発の演習を実施する上では，その基盤となるクラウドコンピューティングに関する知識の教育が不可欠である．クラウド環境を用いた演習の実施例，提案例あるいは論考としては，Anglano et al.(2020）によるもの，Vakaloudis et al.(2020）によるもの，Liang and Chapa-Martell（2019）などがある．

ただし，本提案の場合は，必ずしもクラウド環境の利用を想定していない．もちろん，実サービスとして展開する場合はネットワークに接続されたサーバにデプロイして運用する必要があるが，本提案の範囲はローカルサーバで試行する段階までで完結する．

第 3 章　インタフェースという観点からの考察

Web アプリ開発を題材に IT 教育を進めようとしても，それほど簡単にはいかないことは，すでに述べたとおりである．本学部においては 1 年次に「プログラミング基礎」という必修科目が用意されているが，プログラムの書き方を覚えただけではアプリやシステムを作れるようにはならない．それは，プログラムの書き方以外に身に付けなければならない知識がたくさんあるからである．

ざっと並べてみても，適切かつ効率的なロジックの構築方法，計算精度や誤差に関する知識，適切なコード体系の選択，データの入出力と永続化（データベースの利用），エラー処理，例外処理，セキュリティ対策，テストや開発手法の共有（ソフトウェア工学的知識），プロジェクトのマネジメント方法など，様々な知識がシステム開発には求められる．

前述のとおり，この数年の筆者らの研究テーマは「人間と情報システムとのインタラクション」である．したがって，畢竟，学生が進める研究のテーマもこの分野に関連したものが中心であり，そのための教育もまたこのテーマと関係した様相を呈することになる．そこで，本研究では「インタフェース」のあり方に着目し，段階的に習得する道筋を検討する．

3.1　情報システムのモデルにおけるインタフェースの位置付け

ここで，情報システムのモデルを考える．図 1 (a) は，入出力を伴う情報システムの最も単純化されたモデルである（中央大学国際情報学部, 2021, 第 14 章）．システムのユーザは情報システムに対して何がしかの情報を「入力」し，その入力に基づきシステムが「処理」を行なった結果が「出力」される．それを受け取ったユーザは「フィードバック」の過程を経て新たな「入力」を行う．このサイクルを何回か行うことによって，最終的に希望する処理結果ないしは出力結果をユーザが得る，というものである．

1 年次の必修科目である「プログラミング基礎」では，プログラミングに関する必要な学習を網羅する．しかし，情報システムの概念に関しては，その科目のなかでさほど触れる機会は提供されていない[1)]．主に，同科目では，情報システムにおいて「処理」がどのように行われるか，「処理」をどのように実装すればよいか（プログラミングすればよいか）のみを学ぶ（図 1 (b) の点線部分）．

前述したとおり，本研究では，不足する前提知識のうち「インタフェース」に着目する（図 1 (c)）．インタフェースに関連する事項について，プログラミング基礎で学ぶ範囲から始め，Web アプリの実装に至るまでにどのような道筋で学習を進めるべきかについて論考する．

1)　ただし，他の必修科目のなかで，この概念に触れる機会があるため，1 年生がこの概念に対して全く触れないというわけではない．

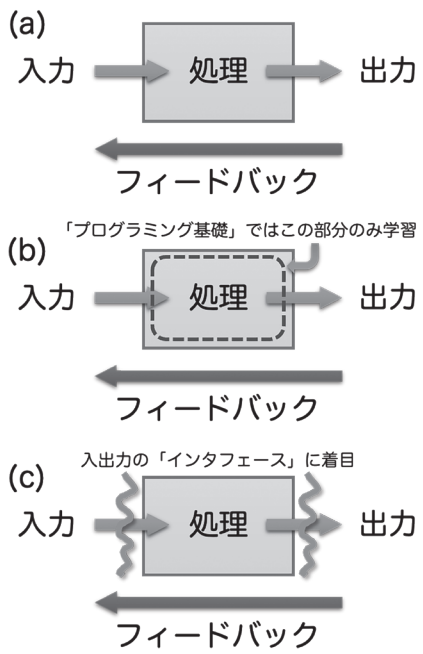

図 1　情報システムのモデル

3.2　インタフェース開発の段階

それぞれの段階について詳しく説明する前に，「プログラミング基礎」から Web アプリ開発演習に至るまでの（想定すべき）5 段階について，整理する．

表 1 に，その 5 段階を示す．

表 1　説明順序の 5 段階

レベル	内容
level 0	関数
level 1	Command Line Interface, CLI
level 2	チャットボット
level 3	Graphical User Interface, GUI
level 4	Web アプリケーション

第 4 章　各段階の詳細

本章では，先に示した 5 段階に基づき，順番にその詳細について述べる．

4.1　関数

まず「プログラミング基礎」で学ぶ範囲である「関数」から始める．同科目では入出力のための関数として scanf () や printf () を扱うが，ここではそのようなファイル入出力は取り扱わない．それ以前のプリミティブな入出力としての関数を考える．

図 2 は，Python を用いて二乗の計算を行うだけのシンプルな関数である．入力として引数 x をとり，処理として y=x × x の計算を行う．その計算結果が格納される変数 y の値は，最後の return 文で関数の呼び出し元に戻される．

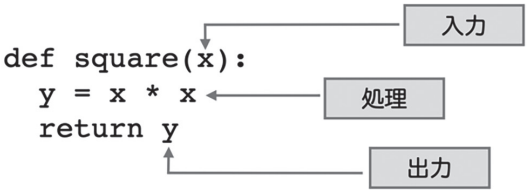

図 2　関数の入力・処理・出力（Python）

通常，これだけで完結するものではなく，それを利用したフィードバックループを考える必要はない．しかし，情報システムのモデルに当てはめると，入力から処理を経て結果が出力される，最もシンプルなモデルの具現化に相当すると理解できる．

4.2　CLI アプリケーション

次の複雑さレベルである level 1 が，コマンドラインインタフェース（Command Line Interface, CLI）[2] である．ターミナルで動作させた CLI プログラムの例を，図 3 に示す．

この例では，新たに cli () という関数を定義しており，その関数によって CLI の動作をシミュレートする．scanf () と printf () に相当する Python の関数である input () および print () を用い，ターミナル上でのユーザとのやり取りを実現する．

関数 cli () 自体は，非常にシンプルなものであ

2）キャラクター・ユーザー・インタフェース（Character User Interface, CUI）で，CUI と呼ぶこともある．

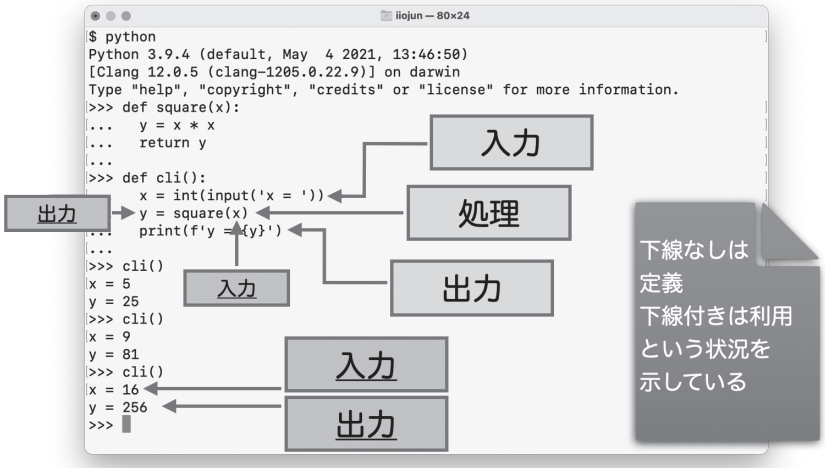

図 3　Python 関数の入力・処理・出力と関数の利用

る．その本体として先ほど定義した関数 square（）を利用している点に注目されたい．

図 3 において，下線を引いて示しているアノテーションは，入出力の「利用」を示す．すなわち，cli（）の内部で関数 input（）によりユーザから入力された値は変数 x を介して関数 square（）の入力として利用され，結果として出力された値は変数 y を介して cli（）の出力へと渡される．

Python インタプリタを介してユーザから入力された値（図 3 の下部）も同様である．これらの値は Python インタプリタによって関数 cli（）に渡されて利用される．

level 0 から level 1 に進む際にこれらの点について言及することにより，CLI と関数定義の対比の考慮が，システムの内部でサブシステムが再帰的に利用される状況の理解に繋がることを期待できる．

4.3　チャットボット

CLI の次はグラフィカルなアプリに進むべきではあるが，一足飛びには進めず，CLI と GUI を繋ぐ中間的な段階（level 2）として，チャットボット（ボット）の利用を検討する．

図 4 にボットのプログラミング例を，図 5 に，LINE bot として実装したボットアプリを利用している様子をそれぞれ示す．このボットは，入力を鸚鵡返しに答えるだけのものである．図 5 において，「Hello, JunBot.」という問いかけに対して「You mentioned “Hello, JunBot.” OK, Thank you.」と鸚鵡返しに答えていること，次の会話も同様であることを確かめられたい．

CLI からすぐ GUI に進まずにボットを挟んだ理由は，ボットであれば，「入力」と「出力」のフォーマットが固定化されているからである．図 5 に示した LINE アプリ自体はグラフィカルなユーザインタフェースを備えているが，実際にやり取りされる入出力データは単なるテキストデータであり，CLI でのやり取りとなんら変わらない．すなわち，ボットは一種の「GUI の皮を被った CLI アプリ[3)]」として考えられる．

バックエンドは Google App Script（GAS）を用いて実装する．ここでも，入力されたテキストを受け取り（17 行目），処理を行い（19～33 行目），出力する（34 行目）という手順でプログラムが定義されている．

ボット用の LINE アカウントを用意したり，同様に GAS を利用する環境を用意したりといった

3)　厳密にいうと先に示した CLI アプリの利用例も，GUI 上のターミナルエミュレータを利用しており，GUI 環境における CLI エミュレータである．

code.gs

```
// obtain access token from the script property
const ACCESS_TOKEN = PropertiesService.getScriptProperties().getProperty("ACCESS_TOKEN");
// define endpoint's URI
const line_endpoint = 'https://api.line.me/v2/bot/message/reply';

function doPost(e) {
  const json = JSON.parse(e.postData.contents);

  // get the token to respond message
  const reply_token= json.events[0].replyToken;
  if (typeof reply_token === 'undefined') {
    return;
  }

  // get message body from JSON data
  const message = json.events[0].message.text;

  // return message to the endpoint
  UrlFetchApp.fetch(line_endpoint, {
    'headers': {
      'Content-Type': 'application/json; charset=UTF-8',
      'Authorization': 'Bearer ' + ACCESS_TOKEN,
    },
    'method': 'post',
    'payload': JSON.stringify({
      'replyToken': reply_token,
  // define message content to respond
      'messages': [{
        'type': 'text',
        'text': 'You mentioned "'+message+'"\n\nOK. Thank you.', }],
    }),
  });
  return ContentService.createTextOutput(JSON.stringify({'content': 'post ok'})).setMimeType(ContentService.MimeType.JSON);
}
```

入力

処理

出力

図 4　GAS による LINE bot の実装

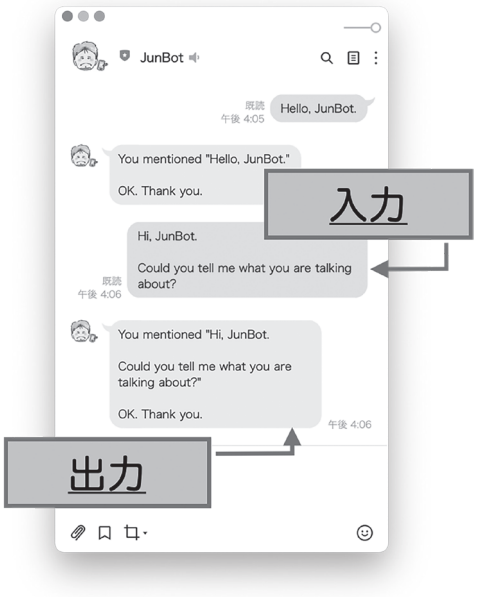

図 5　LINE bot の利用

手間がかかること，および，GAS に実装されるプログラムが（見た目上）やや複雑になっていることなどの課題があるが，ボットの実装自体に関する難易度はさほど高いものではない．

場合によっては図 4 に示すコードのうちアルゴリズムの理解に関して本質的ではない部分を隠蔽し，テキスト処理の中心部分を別の関数に括り出す．その後，その処理の中心部分だけを学生に実装させるといった工夫を加えることで，前述の課題のうち，後者の問題点は解決できるはずである．

4.4　GUI アプリケーション

続いての段階（level 3）が，グラフィカルなユーザインタフェースを持つ GUI アプリケーションである（図 6）．

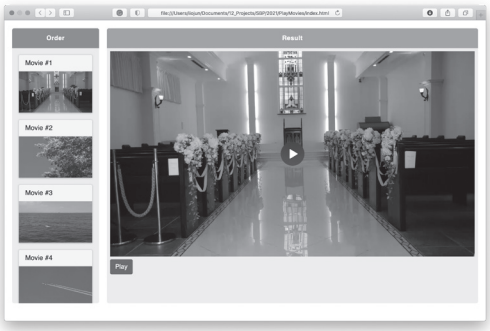

図 6　GUI アプリケーション

このアプリケーションは，ある種のシンプルな動画編集アプリである．左側の動画サムネイルをドラッグして並び順を変えることができる．さらに，右側のメイン画面で，並べられた順に動画が再生される[4]．

4)　これらの機能は muuri というライブラリを利用して実現した．動画サムネイルのみならず，左右のペインそのものもドラッグして入れ替えることができるようになっている．

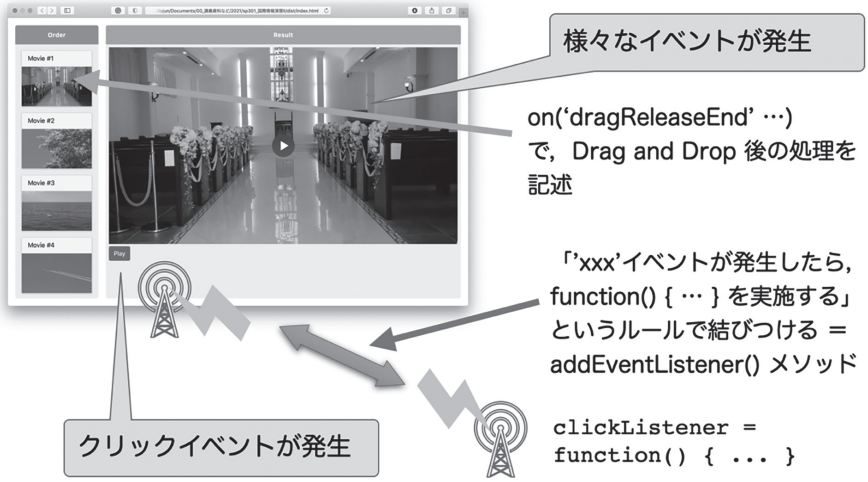

図 7　イベント駆動型プログラミング

なお，この GUI アプリも，プラットフォームの差異を吸収するために Web ブラウザを利用して実現するアプリである．ユーザインタフェースを HTML および CSS を用いてデザインし，ロジックの動作は JavaScript で実装している．

ところで，現代風の GUI アプリケーション構築にはそれなりの付加的な知識が求められる．たとえば，GUI 操作に求められる WIMP（Window, Icon, Menu, and Pointer）という概念，画面上に並べて操作画面を構築する際のグラフィック部品（ウィジェット）という構成単位に関する知識，操作対象と結果の出力が多くの場合に同じ画面上で行われることと，それに伴うページ遷移という事象の背景にある動作原理，さらには，それらを実現するためのプログラミングパラダイムであるイベント駆動型（イベントドリブン）プログラミング（図 7）という考え方などである．

4.5　GUI アプリケーションの課題

GUI アプリ構築を演習で実施する際の課題を，再度，整理する．

先に指摘したとおり，操作対象と結果の出力が多くの場合に同じ画面上で行われる，すなわち，入力と出力が判然一体となっており，情報システムのモデルに照らし合わせたときにモデルとの対応関係がわかりにくいという課題がある．

また，OS やハードウェアに依存する部分（たとえば画面のサイズや，タッチディスプレイであるか否か，ポインティングデバイスのボタンの数の違いなど）も多く環境整備がやや面倒であること，くわえて，プログラミング方法が環境に大きく依存し，特定のフレームワークやライブラリの利用に関する学習負荷が高いという問題もある．

GUI ウィジットの構成は階層的にデザインされており，オブジェクト指向プログラミングが画面設計に効果的であるとも指摘されている（Dey et al, 2019）．たしかにオブジェクト指向設計に長じた技術者であれば，その利点を享受することは容易であろう．しかし，オブジェクト指向の概念を学ぶこと自体の難易度が高いという問題があり，初学者にとってはその点が学習の障害になりかねない．

さらに，「プログラミング基礎」科目では 1 年生は構造化プログラミングについて学んでいる．そこでは，現代的な構造化プログラミングの要素である，連結，分岐，反復，呼出を学んでおり，これらを組み合わせてプログラムを作るという意識を彼ら彼女らは持つ．なかでも，上から順番に実行される「連結」の概念は単純であり，この考え方に親しんでいると，イベント駆動型プログラミ

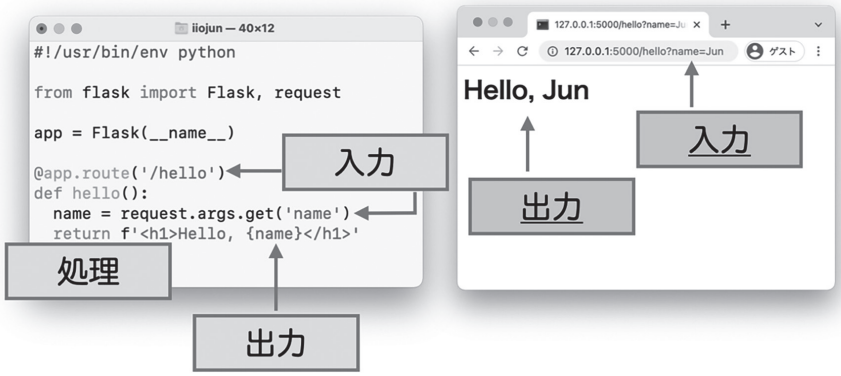

図 8　最もシンプルな Web アプリケーション

ングの動作に慣れるには発想の転換が必要となるので要注意である．

ただし，次の Web アプリも基本的にはイベント駆動型[5]で処理が進むものであり，また，これまで参照してきた情報システムのモデルそのものも「入力」をイベントと捉えるとイベント駆動型であると考えられるため，早いうちにこのプログラミングパラダイムに慣れさせることは学習者にも大きな意味を持つと考えられる．

4.6　Web アプリケーション

HTML と CSS，DOM 等を駆使して表現する Web アプリケーションは，実装の方法論は他の GUI アプリケーションと異なる．とはいえ，Web ブラウザ上でグラフィカルな表現を可能とする Web アプリケーションも，GUI プログラミングの延長上に位置付けられよう．

手元のデバイス内部で動作する一般的なアプリと異なり，Web アプリはユーザインタフェースをクライアントが，アプリケーションロジックの実行をサーバが担うという構成である．したがってネットワーク通信を介してユーザとのインタラクションが行われ，通信に関する知識も不可欠である．

しかし，昨今の Web アプリは Django や Ruby on Rails などのアプリケーション・フレームワークを用いた構築が一般的であり，このようなフレームワークを利用すればネットワーク部分への配慮はほぼ隠蔽される．図 8 は軽量フレームワーク Flask（Mufid et al. 2019）を用いた最もシンプルな Web アプリの例である．図 8 左のプログラム例を見ると，ほぼ level 0 の関数定義と変わらない程度にシンプルである．その動作状況を図 8 右に示す．

ただし，本格的な，あるいは，実用的なアプリを作ろうとすると，このような単純なものでは不十分である．具体的には，次のような知識への理解が求められる．

まず，情報の入力に対するアプローチがやや複雑だという問題がある．先に述べたように，フレームワークが抽象化して具体的な実装を隠蔽しているとはいえ，効率的なコードを書くためには，GET と POST の仕組みなど，HTTP を理解しなければならない．フレームワーク上でも，パスパラメータ，クエリパラメータ，リクエストパラメータの違いなどに現れる．

また，Web アプリの基本構造としてルーティングと呼ばれる概念の理解が求められる．これは入力と処理の対応付けの基礎をなす概念である．さらに，実際にサービスを提供する際に起こり得るインシデントを防ぐために，ネットワークセキュリティの基本的な問題と対策も学んでおく必要が

5）　クライアントからのリクエストがサーバに届くと処理が行われるという動作をするので，リクエストの受理がある種のイベントであり，そのように考えれば歴としたイベント駆動型のシステムである．

ある．

情報の入力だけでなく，出力へのアプローチも複雑である．図 8 に示したようなシンプルなものではなく，リッチな画面によるサービス提供を実現するためには，テンプレートの利用が必須である．そのためテンプレートを利用した画面デザインの知識が求められる．

そもそも，HTTP の根本的な課題とその対応が複雑であり，その対応に各種の工夫が盛り込まれてきたことに対する理解が難しい．当初の HTTP は「行ってこい」型の通信モデルであり，セッション管理などの工夫は後付けで工夫されてきた経緯がある．その課題に対応するための AJAX などを理解するにはフレームワークの利用が必須であり，実際に利用するフレームワークの作法を十分に理解しないと本格的なアプリケーションは作れない．

第 5 章　残された課題

本論文では，プログラミング入門科目で得られた知識レベルから，Web アプリ開発演習を実施できるレベルに至るまでの道程を，インタフェースのあり方に着目して考察した．本章では，本モデルがカバーできない範囲と，今後の課題について述べる．

5.1　本モデルがカバーしない範囲

インタフェースのあり方との観点から，段階的に掘り下げる IT 教育のモデルを提案した．しかし，モデルを単純化して説明を簡単にするために，これまであえて言及してこなかったが，実際には重要な概念がある．それは「エラー出力」[6] である．

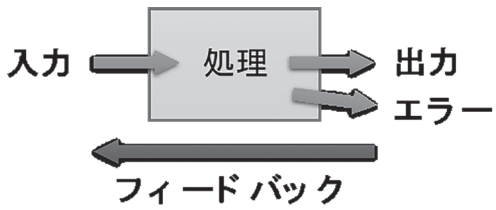

図 9　実際のアプリケーション

実際のアプリケーションにおいて，エンドユーザーは思いがけない利用の仕方をする．すなわち，入力として，想定もしない情報が入力されることはしばしば発生する．その結果，処理が対応しきれない場合はエラーとなる．エラーの状態に適切に対応し，通常の手続きに復帰するためのエラー処理の実装は，とくに 24 時間 365 日の稼働が求められる Web アプリには必須である．しかし，その点に関しては全く言及しておらず，必要に応じて指摘すべき課題であろう．

その他，すでに述べてきたような GUI アプリや Web アプリ構築に構築に必要な様々な知識も，実際には，具体的なアプリ構築事例において必要に応じて指導する必要がある．

5.2　今後の課題

本論文においては，情報技術を学ばんとするものの基礎的な知識しかない学生を対象として，現実的なアプリケーションとして活用できるレベルの Web アプリ開発演習に至るまでの道筋を検討してきたが，現在のところ，状況に依存した指導を重ねてきているのみであり，体系的な指導体系を適用しているわけではない．そこで，今後の課題として，汎用的に適用可能なカリキュラムへと具体化することが望まれる．

また，指導手順の提案に留まっているため，その効果測定も行うべきである．具体的には，多人数の学生を対象として，本指導手順を適用して，理解度を何らかの方法で計測し，その効果を測定することにより，本提案手順の有効性の確認が求められよう．現在は LMS（Learning Management System）の普及が進んでおり，その効果的な活用が期待されるところである．

第 6 章　お わ り に

インターネットでのサービス提供が一般的になっている現代においては，Web アプリ開発の演

6）　インタフェースの言葉を用いて説明すると「エラー出力」ということになるが，実際にはエラーハンドリングの処理が重要である．

習に対する要求が増大している．しかし，プログラミングの初学者がいきなり Web アプリ開発に着手することはできず，基礎的な学習から Web アプリの開発に至るまでの学習ロードマップが求められている．

本論文では，「インターフェース」に着目し，プログラミング入門科目から Web アプリケーション開発演習に至るまで，どのような道筋をたどればよいかを検討した結果を述べた．機能の概念から始め，CLI，ボット，GUI アプリケーションの開発に関する留意点の言及を経て，Web アプリ開発に至る五つの段階を考察した．最後に，本論文で示した順序の有効性，留意点，今回の研究で不足している点などについての論考も加えた．

本提案を汎用化させて具体的なカリキュラムへと落とし込むこと，および，実際の教育現場へ展開してその妥当性を評価することが今後の課題として残されている．

謝　辞

本研究は，中央大学特定課題研究費の助成を受けて実施された．

参 考 文 献

中央大学国際情報学部（編集）（2021）．国際情報学入門，ミネルヴァ書房．

Anglano, C., Canonico, M., & Guazzone, M.（2020）. Teaching Cloud Computing: Motivations, Challenges and Tools. *In 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW),* pp. 300-306. IEEE.

de Oliveira Fassbinder, A. G., Barbosa, E. F., & Fassbinder, M.（2018）. Massive open online courses on web development education: A case study. *In 2018 IEEE Frontiers in Education Conference (FIE),* pp. 1-9. IEEE.

Dey, P., Sinha, B. R., Amin, M., & Badkoobehi, H.（2019）. Best practices for improving user interface design. *Int J Softw Eng Appl,* 10（5）.

Hanagaki, T. & Iio, J.（2023）. Emotional Evaluation of Movie Posters, *The 21st International Conference on e-Society (ES2023),* pp. 428-431.

Iio, J., Shimizu, H. Sasaki, H., & Matsumoto, A.（2010）. Pirka'r: Tool for Web Designers - Supporting Development of Multi-platform Web Application, *The 6th International Conference on Web Information Systems and Technologies (WEBIST2010),* Vol. 2, pp. 159-162, Valencia, Spain.

Iio, J. & Ohgama, S.（2016）. 'Make It Possible' Study: Can LibreOffice and Apache OpenOffice Be Alternative to MS-Office from Consumer's Perspective? *The 2nd International Conference on Social, Education, and Management Engineering (SEME2016),* Paper ID: S067, Bangkok, Thailand.

Iio, J.（2019）. TWtrends ― A Visualization System on Topic Maps Extracted from Twitter Trends, *IADIS International Journal on WWW/Internet,* Vol. 17, No. 2, pp. 104-118.

Iio, J. & Wakabayashi, S.（2020）. Dialogbook: A Proposal for Simple e-Portfolio System for International Communication Learning, *International Journal of Web Information Systems,* Vol. 16, Issue. 5, pp. 611-622.

Iio, J.（2023）. How Many Tweets Describe the Topics on TV Programs: An Investigation on the Relation between Twitter and Mass Media, *IEICE Transactions on Information and Systems*. Vol. E106-D, No. 4, pp. 443-449.

Liang, Z., & Chapa-Martell, M. A.（2018, December）. A Top-Down Approach to Teaching Web Development in the Cloud. *In 2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* pp. 32-39. IEEE.

Liang, Z., & Chapa-Martell, M. A.（2019）. Mitigating resource constraints in web development education using commercial cloud services. *In The 18th Forum on Information Technology*.

Mufid, M. R., Basofi, A., Al Rasyid, M. U. H., & Rochimansyah, I. F.（2019）. Design an mvc model using python for flask framework development. In *2019 International Electronics Symposium (IES)* pp. 214-219.

Vakaloudis, A., Cahill, B., O'Leary, C., & Challa, D.（2020）. Preparation and execution of final year student projects on the cloud. *In 2020 IEEE Frontiers in Education Conference (FIE),* pp. 1-7. IEEE.