Vol.2021-CE-158 No.10
2021/2/14

完全オンライン授業におけるPHPプログラミング実践と実習環境

長 慎也[1,a)]　浦上 理[1]　長島 和平[2]　兼宗 進[3]　並木 美太郎[2]

概要：
明星大学の情報学部では例年「Web プログラミング」という授業において，Web アプリケーションを作成しながら，その技術要素を習得させている．2020 年度の授業は全学的に原則非対面（オンライン）で行われたため，本授業もすべての回を非対面で行うようにした．例年の対面授業においては Ruby on rails を用いていたが，各 PC に環境のインストールが必要であったり，コマンドによるアプリケーションの管理が煩雑でありサポートが大変であったりすることから．Web ブラウザで動作する学習環境 BitArrow に PHP の実行機能を追加して，例年と同等の Web アプリケーションを作成させる実習を行った．また，授業の説明・サポートは Zoom や Slack などを活用した．本稿では，BitArrow で Web アプリケーションを学習する環境について述べるとともに，オンライン授業におけるプログラミング授業の可能性や課題についても論じる．

PHP programming practice and environment in a fully online class

CHO Shinya[1,a)]　URAKAMI Satoshi[1]　NAGASHIMA Kazuhei[2]　KANEMUNE Susumu[3]
NAMIKI Mitaro[2]

Abstract: In School of Information Science at Meisei University, we have a class "Web Programming" every year, in which students learn the technical elements of Web applications through creating small Web applications. In this year, the class (and almost all classes in the university) were conducted in a non-face-to-face (online) format. In previous years, we have used Ruby on Rails in face-to-face classes, but we expect that supporting students' programming in Ruby on Rails by online is not practical, because it requires installation of the environment on each PC, and resolving students' troubles are difficult in several reasons of characteristics of devlopment style in Ruby on Rails. Thus we decided to use PHP instead of Ruby on Rails in the year. But PHP still requires installation of environment. Therefore we decided to add the PHP execution function to BitArrow, a learning environment that runs on a Web browser, the functions allows students create Web applications in the same way as in previous years. We also used Zoom and Slack to explain and support the class. In this paper, we describe the environment for learning Web applications with BitArrow and discuss the possibilities and challenges of programming classes in online classes.

1. はじめに

明星大学の情報学部では例年「Web プログラミング」という授業において，Web アプリケーションを作成しながら，その技術要素を習得させている．

また，本授業はもともと「ソフトウェア設計論」という授業であったことから，アプリケーションを実装する前に，その大まかな振る舞いを設計してから実装を行う演習も行う．Web アプリケーションを題材として設計の演習を行う理由として，1 つのタスク（ユースケース）を，プロセス上独立した複数のページで実装するという，Web アプリ

[1] 明星大学
Meisei University, Japan
[2] 東京農工大学
Tokyo University of Agriculture and Technology, Japan
[3] 大阪電気通信大学
Osaka Electro-Communication University, Japan
[a)] cho@eplang.jp

ケーションの特徴が挙げられる[*1]．ユースケースのフロー（流れ）を画面単位で分割して設計するのは，Web アプリケーションを普段から使っている学習者にとっては比較的容易であることから，設計の題材として採用してきた．

例年の対面授業においては，Web アプリケーションのフレームワークとして Ruby on Rails を用いていた．Ruby on Rails は MVC モデルを採用しており，「モデル」「ビュー」「コントローラ」を独立して定義する仕組みがある．「ユースケース」における「フロー」を「コントローラ」として，各フローにおいて表示される画面を「ビュー」に対応づけることで，設計したものを実装に落とし込むのが簡単であるという特徴がある．

演習においては，**図 1** のように PC 教室の各 PC に Ruby の処理系および Ruby on Rails を配布し，localhost でサーバを起動して演習を行っていた．

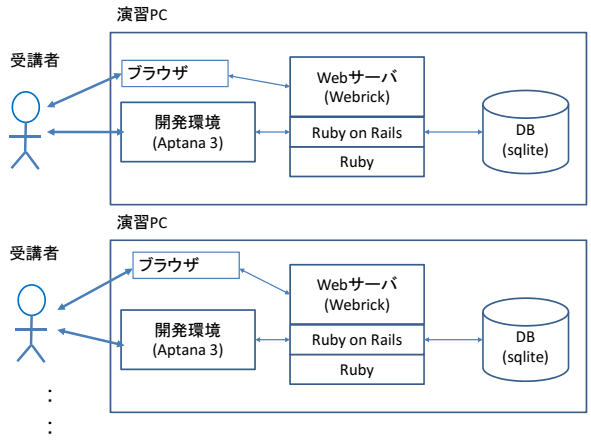

図 1 Ruby on Rails による演習システム構成

しかし，この方式では学習者のサポート面で負担が大きかった．その理由をいくつか述べる．

- **コマンドライン**
  Ruby on Rails では，必要なファイル等を生成するため，頻繁に CUI によるコマンドラインを発行する必要がある．そのため，普段コマンドラインに慣れていない学習者が，コマンド発行を行わなかったり，スペルミスをしたりすると，ソースコードが正しく書けていても正しく動作しないことがある．コマンドラインはソースコードと異なり，どのように発行したのかの履歴が正確に残らない（コマンドラインのウィンドウを閉じてしまうと履歴は消える）ため，原因を特定するのに時間がかかることがある．
- **データベースの管理**
  前述のコマンドラインの中でも特に，データベース関連のコマンドは，「モデルの作成」と「マイグレーション（DB へのテーブルの反映）」という 2 つの操作が必要であり，それらの操作の一方を行わなかったり，順序を間違えたりすると正しく動作しなくなる．また，モデル作成時にスペルミスがあると，テーブル（および追加済のデータ）を消してやり直しをする必要がある[*2]．
- **フォルダ構造の理解**
  先述した通り，データベースの状態などは，ソースコードからは判別できないが，データベースの状態を確認するためのファイル (db/schema.rb, db/migration/*.rb) は存在しているため，適切に確認をした上で修正を行うことができるが，自力で修正を行える受講者は多くない．また，MVC モデルを採用している Ruby on Rails は，ビューにあたるファイル (.html.erb) はコントローラ本体とは分離されている．本来 MVC モデルを学習するのは適してはいるが，ファイルを作り忘れると「template is missing」などのエラーが出る．このソースコード自身のエラーではないため，学習者がエラーの原因を特定することが難しい場合がある．

このような Ruby on Rails 特有の困難さは，他の実践でも同様の報告例がある [1]．本学においても，教員，TA や SA は多くのトラブルの対応に追われた．特に，TA や SA をもってしても原因がわからず，1 人あたりのサポート時間が延びてしまう傾向が見られた．

2020 年度の授業は，すべての回を非対面で行うこととなったが，各受講者が自宅等に用意している PC に Ruby on Rails の環境を入れるのが負担であること，Ruby on Rails では先ほどに述べたような理由によりオンラインでのサポートが困難であると予想されたことから，Ruby on Rails を使う代わりに PHP を使用した授業を再構築した．

ただ，PHP も個別に環境を用意させるのは負担であることから，Web ブラウザ上でのプログラミング環境 BitArrow[2] 上で PHP のプログラミングができる環境を構築した．また，授業の説明・サポートは Zoom や Slack などを活用した．

本稿では，BitArrow で PHP による Web アプリケーションを学習する環境について述べるとともに，オンライン授業におけるプログラミング授業の可能性や課題についても論じる．

2. 授業の概要

「Web プログラミング」は，明星大学情報学部情報学科コンピューター科学 (CS) コース 3 年生の選択科目であり，2020 年度の受講者は 48 人である．

[*1] JavaScript を用いれば単独ページでも実装できるが，本授業では基本的な Web アプリケーションのメカニズムを習得させるため，JavaScript は使用していない

[*2] データベースのスキーマを変更する方法も存在するが，それを行う過程に別のミスをしてしまうとデータベースの状態がさらに複雑になり，結局消したほうが早いことが多い

「Web プログラミング」のシラバスを**表 1** に示す．シラバス中には，Ruby や Ruby on Rails などの単語が入っているが，実際の授業では PHP を使ったものに置き換えた．

しかし，授業中に題材として学習者に作らせるアプリケーションは極力変えないようにした．作るアプリケーションは次の 2 つである．

- **掲示板** ログイン機能を有し，短文を投稿して他のユーザと投稿を共有する機能をもつ
- **戦闘ゲーム** ログイン機能を有し，ユーザごとに「キャラクタ」の作成，他ユーザの所持キャラクタとの戦闘を行う．

また，これらのアプリケーションを実装する演習と並行して，アプリケーションの各種機能を「ユースケース」に分割し，そのユースケースのフロー（ユーザとシステムの動作），および表示される画面についての簡単な設計演習を行う．ユースケースのフローの例を**図 2** に示す．

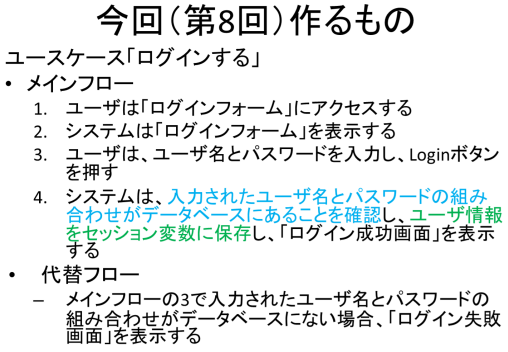

図 2 ユースケース「ログインする」の例（講義資料スライド）

3. 演習システム

本授業をすべて非対面で行うにあたり，各受講者の自宅 PC に環境をインストールすることなく，Web ブラウザのみで演習を行える BitArrow を用いて演習を行った．

BitArrow が公式にサポートをしている言語は C，ドリトル，JavaScript，Python，DNCL(どんくり)，Tonyu である [3] が，今回は試験的に PHP プログラミングを行う機能を追加した．

BitArrow のもともとの設計方針は，ユーザが記述した言語を JavaScript に変換してブラウザ上で実行する，というものであったが，Python 言語においてはサーバサイドの実行もサポートしており [4]，今回も「サーバーサイドの Web アプリケーション開発」というのを主眼に置いているため，Web サーバで PHP を実行させるようにした．しかし，PHP は任意のファイルへのアクセスが可能であり，ユーザの記述したプログラムによって Web サーバへの攻撃を許す可能性があるため，ユーザの PHP プログラムを Docker で動かす仕組みを採用した．

また，データベースについては従来の授業とは異なり，各自にテーブル（スキーマ）の作成を行わせるのではなく，教員側で用意した共通のテーブルに全受講者がアクセスさせる方式を採用した．これにより，テーブル作成ミスによるトラブルを防ぐことや，他ユーザとのデータ共有をやりやすくすることで，Web アプリケーションとしての実用性を高める効果が期待できる．

3.1 システム構成

今回作成した BitArrow 向けの PHP プログラミング環境の構成を**図 3** に示す．

(1) から (3) までが従来の BitArrow の構成に当たる．今回はここに次の 2 つの Docker コンテナを立ち上げた．

- (4) （Apache+）PHP コンテナ
- (5) mysql コンテナ

(3) ユーザプログラムフォルダは，クラスごとに別のフォルダに分類されており，クラスの中に各ユーザのフォルダが配置されている．(4)PHP コンテナは，そのフォルダの中から，今回のクラス (webpro20) に該当するフォルダだけをボリュームとして参照している．

(1) の BitArrow 本体はホスト側で動作するが，ユーザの PHP プログラムをホスト側で動作させることはせず，単純にユーザの PHP プログラムを保存する役割だけをもつ．

(3) のフォルダはホスト側の Web サーバの公開フォルダではないところにあるので，ブラウザからのリクエストで直接ホスト側で実行されることはない．その代わりリバースプロキシを使用して，ホスト側の特定の URL にアクセスしたときに，(4) 内で起動している Web サーバへのアクセスに接続するようにし，ユーザの PHP プログラムを (4) で実行させるようにしている．

このため，(4) でアクセス可能な範囲，すなわち，同一クラスの他のユーザのフォルダへのアクセスは可能ではあるが，学習者へ明示的に伝えてはいない上（本授業ではファイルアクセスのプログラムは紹介していない），万が一そのようなアクセスを行うプログラムが実行されればログに残るため，授業内での注意喚起で対応できると考えられる．また，他のクラスのフォルダはコンテナから参照していないため，ユーザプログラムからはアクセスできない．

なお，BitArrow 本体では構文エラーのチェックなどは行わず，エラーメッセージは (4) が返して来たものをそのまま表示させている．

また，(4) からは (5)mysql コンテナへのアクセスが可能である．データベースはすべてのユーザから共有され，任意の操作（読み出し，書き込み，テーブルの変更など）を可能にしている．

データベースにアクセスするユーザのプログラム例を**図 4** に示す．2〜5 行目に含まれている，データベースへの接続情報は全ユーザ共通であるので，他のユーザの作成したデータを変更，削除することも可能である．(5) のデー

表 1 「Web プログラミング」シラバス（対面授業想定時作成）

回	学習内容
1	アプリケーションの設計図： ユースケース図，ユースケースシナリオ，ER 図，画面遷移図
2	Ruby プログラミング：変数，制御構造
3	Ruby プログラミング：オブジェクト指向
4	Ruby on rails による Web アプリケーションの構築
5	アプリケーションの実装 (1) ER 図に基づくデータの構築
6	アプリケーションの実装 (2) シナリオと画面遷移図に基づくアプリケーションの実装
7	演習 - 掲示板アプリケーションの制作 - 書き込みの管理
8	演習 - 掲示板アプリケーションの制作 - ユーザの管理
9	Web ゲームアプリケーションの設計 ー おおまかな設計図の作成
10	Web ゲームアプリケーションの設計と実装 - キャラクタの作成
11	Web ゲームアプリケーションの設計と実装 - 戦闘
12	Web ゲームアプリケーションの設計と実装 - 回復
13	Web ゲームアプリケーションの設計と実装 - オリジナル機能の設計
14	Web ゲームアプリケーションの設計と実装 - オリジナル機能の実装
15	振り返り

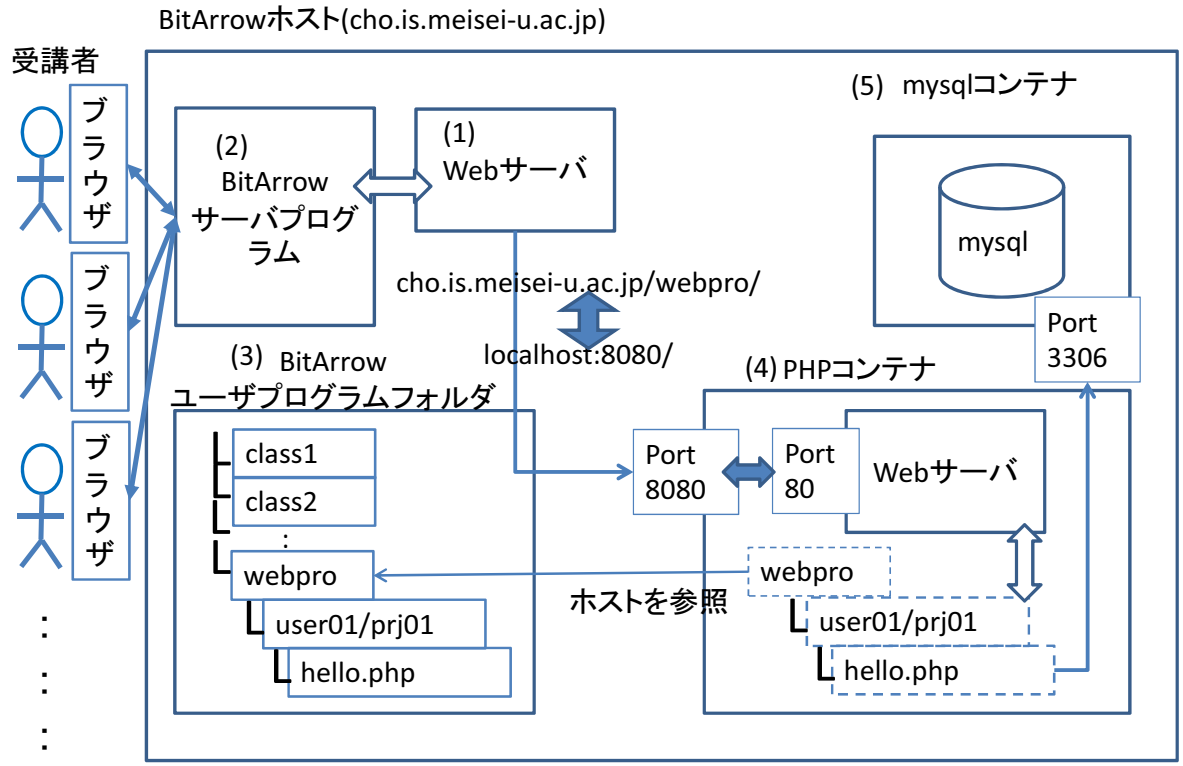

図 3 PHP による演習システム構成

タベースはコンテナ内にデータを保持しており，ユーザプログラムによってデータベースを破壊したとしても，コンテナ自身を修復すれば元通りになる．

BitArrow 上での PHP の実行画面を図 5 に示す．実行画面ダイアログには現在実行しているプログラムの URL を表示するようにした．これは，HTTP の GET メソッドによるパラメータ入力を行うとき，アドレスが変化する様子を見せるためである．アドレスにパラメータを直接入力させてパラメータを変えてデバッグをさせることも可能である．

4. 教材とカリキュラム

4.1 Web 教材

5 で後述する通り，授業は毎回各受講者に Zoom を接続

```
<?php
$h="mysql-container";//Docker コンテナ名
$dbn="xxxx";$user="xxxx";$pass="xxxx";
$dbh=new PDO('mysql:host=$h;dbname=$dbn;',
            $user, $pass);
$sth=$dbh->prepare("select * from tweet");
$sth->execute();
$results=$sth->fetchAll(PDO::FETCH_OBJ);
foreach($results as $result) {
    print_r($result);
    print("<BR>\n");
}
?>
```

図 4　データベースへのアクセス

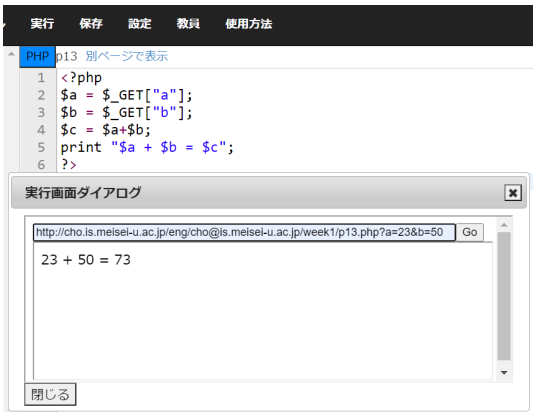

図 5　BitArrow での PHP 実行ダイアログ

させて行ったが，Zoom での内容に関する解説は最低限にし，各自が自学自習できるようにした．

この教材では，学習者自らがプログラムの振る舞いを理解してもらうために，なるべく説明を教材中に入れず，学習者に推論させるよう，次の 4 種類の Phase を導入している [5]．

- **Phase1** 教材が示したソースコードを受講者がコンピュータに実行させると，コンピュータは処理系の決めたルールに従って結果（出力）を受講者に出力する．
- **Phase2** 学習者は出力結果を教材に記入した上で，ソースコードと出力結果を見て，そこに含まれるルールを推論する．
- **Phase3** 教材がソースコードを空欄（場合によっては穴埋め）のまま，結果のみ提示する．受講者は Phase2 で推論したルールを使用して，空欄部分のソースコードを推察する．
- **Phase4** 受講者は推論したソースコードを再びコンピュータに実行させ，結果が提示されたものと同じであれば，推察したルールが正しいという確信を持つ．結果が同じでなければ，コンピュータからの出力（エラーメッセージや提示されたものとは異なる出力結果）を見て，修正を加える．
- **Phase1'** 教材は別のソースコードを提示し，以下同様に繰り返す．

教材はすべて Web ページとして配布し，解答を Web ページに直接書き込めるようにし，教員による採点結果および修正のアドバイスも表示できるようにした．また，BitArrow のファイル配布機能を使用して，すべての受講者に雛形となるファイル，必要なライブラリの配布も行った．教材 Web ページの例を**図 6** に示す．本教材の記入欄には次の種類がある．

- 空欄のない完全なプログラムの出力結果を記入する欄 (Phase2 に該当，図 6 の「出力 1-1」)
- 指定された出力結果になるようなプログラムの全体を記入する欄 (Phase3,4 に該当，図 6 の「p02.php」)
- 指定された出力結果になるようなプログラムの一部空欄になっている部分を記入する欄 (Phase3,4 に該当)
- ユースケースのフロー，および画面の一覧を記入する欄

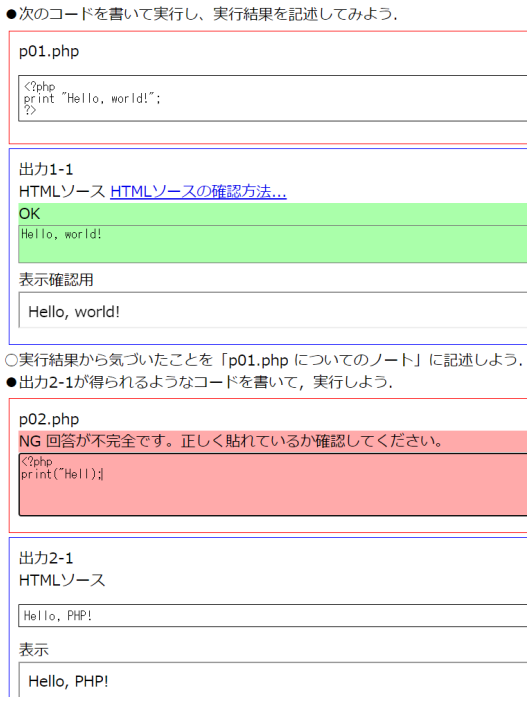

図 6　Web 教材

このうち，プログラムの出力結果を記入する欄（Phase2）においては「実行結果から気づいたことをノートに記述しよう」という指示がある．ノート機能は，BitArrow においてファイルを開いたときに，そのファイルについてのコメントを学習者が自由に記入できる仕組みである．ノート機能の画面を**図 7** に示す．ノートは自分が記入するだけでなく，他の学習者が記入した内容を参考したり，「いいね」や返信をつけたりする機能ももつ．

この教材による演習を行った後，詳細な解説を次回の授

図 7　ノート機能

業の最初に行うようにした．これによって，学習者が自らプログラムの構成要素について予め推論を行い，その上で解説を聞くことで，知識を確実に定着させることを狙った．

4.2　カリキュラムの流れ

シラバス上では 1 回目はガイダンスであるが，実際には、1 回目からプログラミングを行っている．また環境が変わったりオンラインでしたこともあり，予定は変更している．

受講者は C 言語や Java の習得歴があるので，変数や制御構造の概念はすでに知っており，PHP 特有の書き方を重点的に体験させる．

第 1 回から第 3 回は，PHP の基礎として，HTML の表示，パラメータの受け取り，クラスの定義などを学ぶ．

第 4 回では，MVC モデルにおけるコントローラの概念を導入する．PHP にも MVC フレームワークは存在するが，今回はなるべくシンプルな構成にしたかったため，**図 8** のような簡易ライブラリを配布して，**図 9** のような Ruby on Rails を模したクラス定義でコントローラのアクションを実行できるようにした．また，ファイルの管理を簡略化するため，ビューはコントローラーのアクション（メソッド）内に埋め込むようにした．

```
 <?php
function run_controller($controller) {
    $action=(isset($_GET["action"]) ?
       $_GET["action"] : "index");
    return $controller::$action();
}?>
```

図 8　コントローラ実装用簡易ライブラリ (controller.php)

第 5 回では，セッション変数の概念，第 6 回から 7 回に，データベースについて取り扱い，第 8 回，第 9 回でログインのページを作成し，掲示板を完成させた．

第 10 回からは戦闘ゲームとその設計 (ユースケース・画面の設計) を演習した．試験は行わず，最終回の「振り返り」は 3 日程度の期間で問題を解く最終レポートとした．

```
<?php class Keisan {
// GET パラメタなしの場合，index を呼ぶ
    static function index() {
        ?><h1>入力画面</h1>
        <form action="p03Keisan.php">
        <input type="hidden" name="action"
              value="show_result"/>
        <input name="a"/>
        <input type="submit" value="Calc"/>
        </form><?php
    }
// GET パラメタ action=show_result がある場合
    static function show_result() {
        ?><h1>結果表示画面</h1>
        a=<?= $_GET["a"] ?><?php
    }
}
require "controller.php";
run_controller("Keisan"); ?>
```

図 9　コントローラ実装用簡易ライブラリ使用例

表 2　データベースのテーブル

テーブル名	内容
user	認証に使う．掲示板と戦闘ゲームとで共通
tweet	掲示板で使用．「書き込み」を保存する
chara	戦闘ゲームで使用．「キャラクタ」の情報を保存する

4.3　データベース

テーブルはすべて教員側が mysql 上で CREATE TABLE を発行して準備した．作成したテーブル一覧を**表 2** に示す．これらのテーブルはすべてのユーザから共通の内容が参照される．

5.　オンラインサポート

本授業は，Zoom および Slack を用いて，授業のサポートを行った．授業時間（火曜日 2 限）になった時点で，Slack に Zoom の URL を通知し，Zoom に入室してもらった．まずは，Slack 上で学籍番号と名前を書き込んでもらい，これを出席代わりとした．Zoom では，主に前回の授業の解説を，スライドを共有した上で口頭にて行った．解説は 30 分程度で終わらせ，（その回の）Web 教材による演習を始めるようにした．

その後，教員や TA/SA により Web 教材の採点，Slack による質問，回答を随時行った．

Slack への質問手段は，次の 3 つのチャンネルを用意した．「匿名で質問」は Slack のワークフロー機能を用いて作成した．

- 授業チャンネル（全員が閲覧可能）
- ダイレクトメッセージ（送信先の 1 ユーザと送信者のみが閲覧可能）
- 「匿名で質問」（TA・SA・教員がメッセージを閲覧可能，他の受講者には見えない）

6. 実践結果

本節では，次の観点で実践内容を評価する．

- すべてオンラインで行われた授業において，受講者が問題なく授業について来られたかどうか
- オンラインでのサポートで問題が解決できたか

そのため，次のようなデータを収集した．

- 活動状況
  - Web 教材の提出数，正解数
  - BitArrow のログ
  - Slack での質問数
  - 出席数
- 授業アンケート

活動状況を**図 10** に示す．縦軸は各受講者を，Web 教材における正解した解答欄が多い順に並べたものである．横軸は左から第 1 週，右が第 13 週になり，緑色は該当週の Web 教材の 80%以上の欄において正解した週，黄色は Web 教材のうち 50%以上の欄を埋めている週，橙色は 50%未満の欄しか埋めていないが，BitArrow のログには活動した形跡がある週，赤色は BitArrow のログにも活動した形跡がない週，である．また，右端の青色と紫色のグラフは，それぞれ Slack 上での質問を行った回数（最も多かった受講者は 33 回），出席回数である．

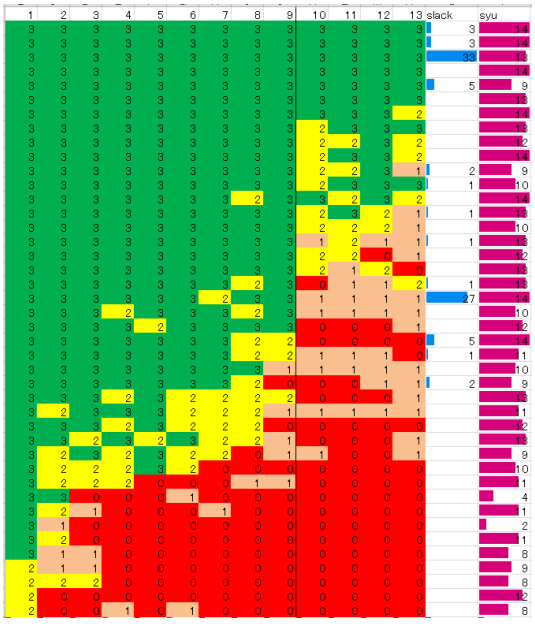

図 10 活動状況 (縦軸：受講者，横軸：週，緑＝課題完成，黄＝課題着手，橙＝ログのみ，赤＝ログなし，青棒＝ Slack 質問数，紫棒＝出席数)

表 3 Slack でのサポート状況

受講者	回数	使用チャンネル
A	33	DM
B	27	DM, WF
C	5	PUB
D	5	PUB
E	3	WF, PUB
F	3	WF
G	2	WF
H	2	DM
I	1	DM
J	1	PUB
K	1	WF
L	1	PUB
M	1	DM

6.1 受講者の活動状況

最後まで課題を終わらせているのは全受講者の 1/4 程度であった．特に，第 10 回のゲーム制作が始まったあたりから，BitArrow にアクセスしながら，課題には着手していない受講者が増えている．

6.2 Slack での質問数

授業期間を通じて，Slack を通じて何らかの質問を行った受講者は 13 人いた．**表 3** に，Slack で質問を行った受講者と，その回数，および使用したチャンネル（PUB=授業チャンネル，DM=ダイレクトメッセージ，WF=「匿名で質問」) を示す．

質問回数が 1，2 件程度の受講者については，教材の解答方法や，教材の誤りの指摘などで，プログラムの内容についての質問はほとんどなかった．また，質問が多かった学習者については，Slack にソースコードを提示させることで，提示されたプログラムを教員や TA/SA が自身のアカウントで実行することで，ほとんどの不具合の原因を，10 分程度で突き止め，質問者に回答することが可能であった．

6.3 アンケート

本授業の終了時に，LMS 上でアンケートを行った．設問および回答を**表 4** と**表 5** に示す．表 4 の設問は，すべて 1（そう思わない）から 4（そう思う）までの尺度で回答するものである．平均値の低かった 3 設問に●，高かった設問に○をつけている．

また，表 4，表 5 いずれの設問にも任意の自由記述欄が設けられている．自由記述欄には次に挙げるような趣旨の記述があった．最後の括弧は，自由記述を行った受講者が該当質問項目に尺度では何番を選択したかを表す．

- **Q1**「いつも使っているテキストエディタと機能が違う (1)」
- **Q2**「ノート機能によって編集の邪魔になることがある (3)」「何に対して使うものなのが不明 (2)」「消せる

ようにしたい (3)」
- **Q4**「あとから復習したい時に前の内容を確認するのが大変 (2)」
- **Q5**「メールより連絡のハードルが低く、使いやすい (4)」
- **Q6**「教材はオンデマンド方式にして，質疑応答を Zoom が行うのがよい (2)」[*3]
- **Q7**「テストやレポートなどのやり方や評価方法が変わったりしたため、少し対応が大変だった (3)」
- **Q8**「環境構築が不要な点を生かして、座学中心の授業（アルゴリズム等）で補助的に使えるのでは (3)」
- **Q13**「匿名で質問する機能は今回は使わなかったが，質問するなら絶対に使いたい (1)」
- **Q14**「どう聞けばよいかわからなかった (2)」「授業外に質問したいことができた時どうしても遠慮しがち (2)」「説明を受けてもうまく理解できなかった際に繰り返し同じことを聞くことができなかった (2)」「答えになってしまう部分なのか判別できなかったから (2)」
- **Q15**「少しペースが早い．一度躓くと授業の進行ペースに戻れない (2)」

7. 考察

今回作成した演習システムによって，PHP でのプログラミングを，受講者が環境を導入することなく実践することが可能になった．以前の Ruby on Rails の演習においては，図 1 のようにデータベースを各受講者がセットアップするようにしていたが，この方式であると，データベースの不具合がある場合に，TA や SA が苦労してトラブルシューティングを行う必要があった．

一方今回のシステムは図 3 のようにデータベースは共有されており，データベースは教員によってセットアップされている．このため，データベースの不具合はほとんどなく，受講者にトラブルがあった場合，PHP のソースコードを Slack に貼ってもらうことで，手元の TA や SA のアカウントで実行することで同じ不具合を再現することができ，遠隔でのサポートも問題なく行うことができた．

ただ，遠隔でのサポートは，受講者が質問などのアクションを起こしてくることを前提に行っていたため，アクションを起こさない受講者のサポートが課題となる．次の節で詳しく考察していく．

7.1 オンライン授業としての課題

活動状況から見るに，最後まで課題をやり遂げたのは全受講者の 1/4 程度であった．昨年度（2019 年度）の対面で実施された同授業では，6 割程度が最後まで課題をやり遂げているのに比べると，ドロップアウトの割合が対面より増えてしまった．

図 10 において，Slack による出席確認の状況とを比較すると，授業に出席はしているものの，演習活動，Web 教材の課題提出を行っていない受講者がいることが明らかになった．

特に，10 回目以降の戦闘ゲームを作り始めた段階で，それまではついてきた受講者（図 10 の中段あたり）が「BitArrow 上での作業を行ったが，Web 教材に解答を提出するには至っていない」という状況に陥っている．これらの受講者のログの分析は本稿執筆時にはまだ行っていないが，一つの原因としては，10 回目において，これまで構築した掲示板の一部の機能（ログイン）を，戦闘ゲームにも移築する作業が考えられる．基本的にはこれまでの機能をほぼコピーするだけでよい内容だが，そもそもどの週のどのファイルでログインの機能を作成したかを思い出せない受講者がいると考えられる．アンケートの自由記述において，**Q4**（Web 教材は使いやすいか）に対して「あとから復習したい時に前の内容を確認するのが大変」という意見があることからも，これまで作成したプログラムから，必要な機能を適切に取り出す，という技能について適切な支援を行うことが必要であったと考えられる．

10 回目以降課題提出ができてない受講者の中には，Slack に 27 回の質問をしていた受講者 B（表 3 参照）も含まれている．この受講者は 10 回目以降にも TA に質問を行っており，TA から「これまでの内容を参考に」という指示をもらいながら解いていたが，これまでのどの内容を適用すればわからない状態が続き，結局課題を提出するに至っていなかった．また，この受講者はアンケートに「少しペースが早い．一度躓くと授業の進行ペースに戻れない」という意見を寄せている．

7.2 システムとしての課題

今回作った BitArrow における PHP プログラミングの機能は，この「Web プログラミング」のために急遽作成したものなので，他のクラスでは使用できない．

もし複数クラスに対応する場合，1 つのクラスで PHP を使おうと思ったら，次の動作・設定をする必要がある．

(1) コンテナ起動（mysql と PHP2 つ）
(2) リバースプロキシ設定（空きポートを確保）
(3) 異なるクラスへの DB アクセス制限（同時に複数のクラスの mysql コンテナが起動している場合）

特に 2 番目の設定項目は，有限であるポートを適切に管理する必要があり，3 番目のアクセス制限についても，パスワードを適切に管理するなど，BitArrow に対する機能追加が必要となってくる．

さらに，mysql のコンテナは再起動するとデータベースの状態が元に戻ってしまうため，データベースのバックアップを行う仕組みも必要と考えられる．

[*3] 他の授業で行っていた方式を踏まえての回答であると推察される．本授業では質問を Zoom で行った受講者はいなかった

表 4 アンケート 1（尺度：1=そう思わない〜4=そう思う）

設問	1	2	3	4	平均
Q1.BitArrow における PHP プログラミング機能は使いやすかった	2	3	13	12	3.17
Q2.BitArrow におけるノート機能は役に立った	0	8	15	7	2.97 ●
Q3. 資料スライドは役に立った	1	4	15	10	3.13
Q4.Web 教材は使いやすかった	1	4	17	8	3.07
Q5.Slack は連絡の手段として有効であった	1	3	12	14	3.30 ○
Q6.Zoom は受講の手段として有効であった	3	5	11	11	3.00
Q7. 本授業はオンラインだが，対面に比べて特に問題はなかった	3	8	7	12	2.93 ●
Q8.[対面でも必要]BitArrow（プログラミング機能）	1	1	9	19	3.53 ○
Q9.[対面でも必要]BitArrow（ノート機能）	1	7	10	12	3.10
Q10.[対面でも必要]Slack	3	1	14	12	3.17
Q11.[対面でも必要]Zoom	6	7	9	8	2.63 ●
Q15. 本授業の進行方法は良かった	0	3	16	11	3.27 ○
Q16. 本授業に満足している	2	0	18	10	3.20

表 5 アンケート 2（授業中の質問について）

設問	1	2	3	4
Q12. この授業の内容について，教員や TA/SA にどのくらい質問したか: 1=一度もなかった，2=一度くらいはあった，3=何度もあった	13	12	5	-
Q13. わからないことがあるとき，主にどうやって質問したか: 1=質問していない，2=Slack（TA/SA に DM），3=Slack（教員に DM），4=Slack（「匿名で質問する」）	13	5	3	9
Q14. 質問したいけど，できなかったことがあったか: 1=なかった，2=あった	23	7	-	-

8. まとめ

本発表では，完全オンラインで Web アプリケーションの構築を行わせる実習を行う実践を報告した．

Web ブラウザで PHP を実習できる環境や，データベースを共有して演習できる環境を整備したことにより，Ruby on Rails を使用していた昨年度の授業に比べて，質問を受けたときのサポートはオンラインでも簡単に行えるようになった．

ただ，オンライン授業の特性上，受講者の自発的な質問を誘導するのが難しく，質問そのものの回数が減ってしまい，結果として途中で脱落してしまう受講者を増やす結果になってしまった．今後も，オンライン授業の需要は続くとみられ，受講者の活動状況をリアルタイムに分析しながら受講者のきめ細かいサポートをしていくことが重要と考えられる．

謝辞 本研究は JSPS 科研費 19K03153 の助成を受けたものです。

参考文献

[1] 高橋圭一: ログファイルと Git リポジトリを用いた Ruby on Rails の初学者の躓き要因の分析，情報教育シンポジウム論文集，Vol. 2020, pp. 69–74 (2020).
[2] 間辺広樹，長島和平，並木美太郎，長　慎也，兼宗　進: 高等学校における複数言語によるプログラミング教育の提案，情報処理学会論文誌教育とコンピュータ（TCE），Vol. 3, No. 3, pp. 29–41 (2017).
[3] 大阪電気通信大学，東京農工大学，明星大学: オンラインプログラミング環境　ビットアロー（Bit Arrow）. `https://bitarrow.eplang.jp/` (参照 2021-01-10).
[4] 長　慎也, 長島和平, 間辺広樹, 兼宗　進, 並木美太郎: オンラインプログラミング環境 Bit Arrow における Python 処理系, 情報教育シンポジウム論文集, Vol. 2019, pp. 122–129 (2019).
[5] 長　慎也，山中脩也，北島茂樹，今野貴之: 情報系初年次のプログラミング演習における，コンピュータとの対話を重視したコースデザインと支援システム，技術報告 20，明星大学, 明星大学, 明星大学, 明星大学 (2019).