コンピュータネットワーク
第12回
Web(2/2): サーバサイドアプリケーション: PHP, JavaScript(Node.js), Python(Flask)

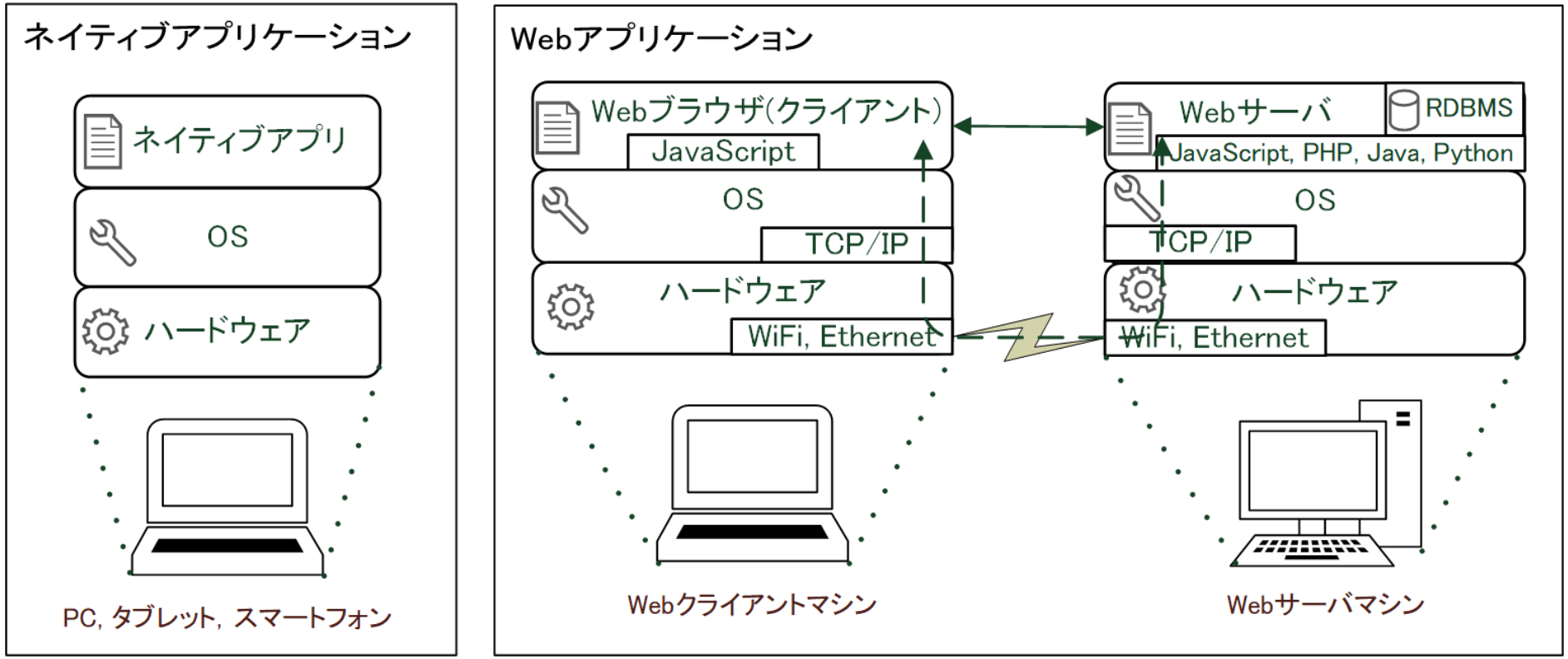
静岡理工科大学
情報学部　コンピュータシステム学科
幸谷智紀

https://na-inet.jp/compnet/

本日の内容

- Webサーバサイドのアプリケーション
- PHP
- Python (Flask)
- JavaScript (Node.js)

(復習) Webアプリケーションとは？



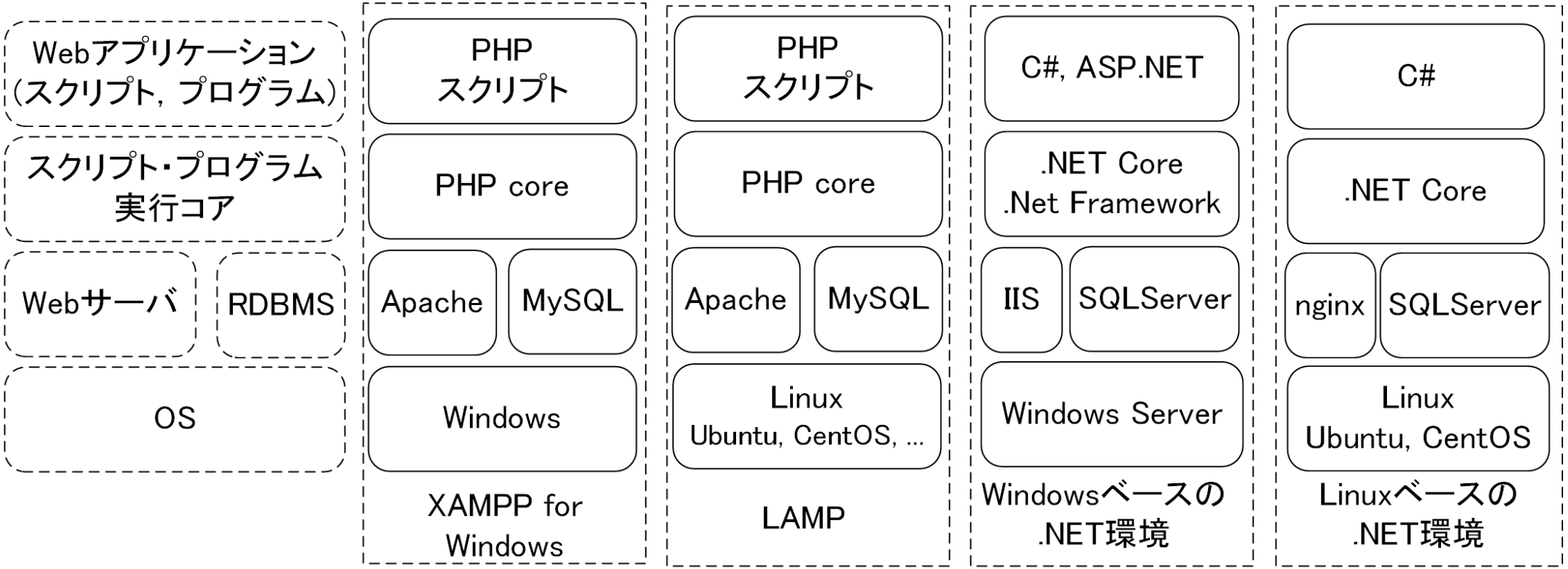
- Webシステム上で動作するアプリケーションの総称
  - Webサーバ・クライアント(ブラウザ)
  - クライアントサイドの技術要素: HTML/CSS/JavaScript
  - サーバサイドの技術要素: PHP/Python/JavaScript/C#
- ネイティブアプリ化したWebアプリケーションも存在する

Webサーバサイドのアプリケーション

- データ送出の拠点を一か所に集約しておくことの長所は？
  1. インターフェース(HTML, CSS)やデータの更新を一か所で済ますことができ，迅速な情報伝達が可能となる
  2. アクセスしてきたユーザの情報を一か所に集約でき，データのチェックや統計解析が素早くできる（例: MS Forms等）

- データ送出の拠点を一か所に集約しておくことの短所は？
  1. データの流出があった場合，全ての秘密情報が漏洩してしまう危険性が高まる（分散しておけば一部流出で済む）
  2. アクセスが集中するため，サーバの性能やネットワークの容量が十分でないときにはWebサーバの応答が遅くなる（CDNはここを解決）

(復習) Webサーバ側のソフトウェア階層



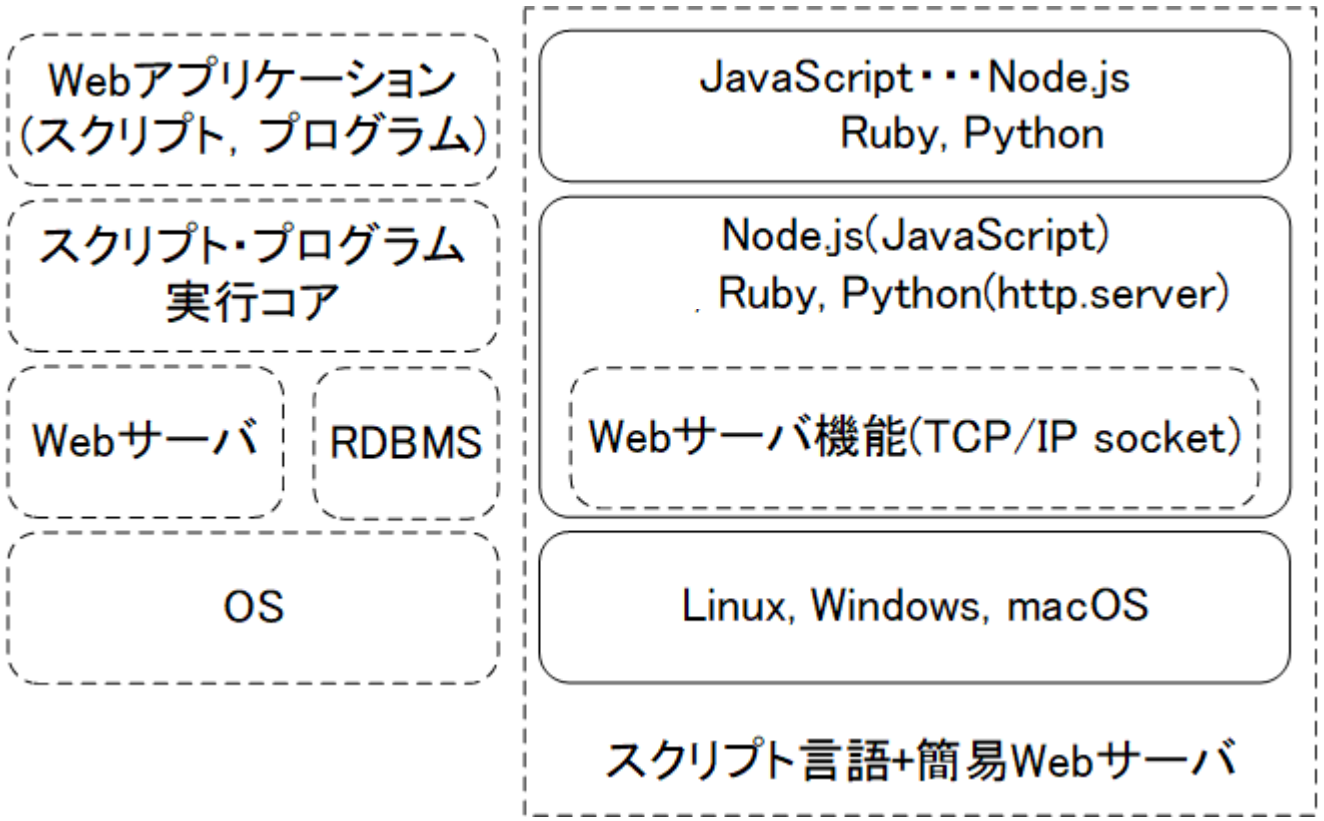
- Weサーバ側で動作するアプリケーション用のインタプリタやデータベース(ファイル入出力代わり)を組み込んでいる

代表的なサーバサイドの環境(1/2)

- PHP・・・Zent社が開発。ApacheやNginxからPHPインタプリタを呼び出して使用する。WordPress（BlogベースのCMSアプリ）が代表的なPHPアプリケーション。

- Java・・・Tomcat等のJava Servlet環境。官公庁システムではいまだ現役らしい（セキュリティ大丈夫か？）
- C#・・・Microsoft .NET Core環境では主流（？）の開発言語。Javaに構造が似ている。

代表的なサーバサイドの環境(2/2)



- Python・・・Python財団が管理。代表的なフレームワークを利用する。
  - Flask・・・簡易なアプリを作るには向ている
  - Django・・・大規模アプリを作るには向いている
- JavaScript・・・Node.jsベースのサーバサイドプログラミング環境
  - Node.js = レンダリングエンジン込みのサーバサイドJavaScript開発環境

PHP, Python (Flask), JavaScript(Node.js)について

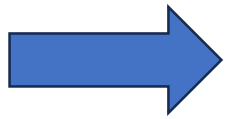
- プログラミング言語として，かなり厚みのある内容になるので，全ては扱わない

Webアプリ開発教材事例（高性能計算研究室提供）

- Node.js + MySQL（実践演習２）：https://cs-tklab.na-inet.jp/nodejs/
- PHP + MySQL （実践演習２，2019年まで）：https://cs-tklab.na-inet.jp/phpdb/
- Python + SQLite（実践演習１）：https://cs-tklab.na-inet.jp/flask/

PHPの例

```
1  <!DOCTYPE html>↓
2  <html>↓
3  <head>↓
4    <meta charset="UTF-8" />↓
5    <title>"..."と'...'の違い</title>↓
6  </head>↓
7  <body>↓
8  ↓
9  <h1>"..."と'...'の違い</h1>↓
10 <?php↓
11   $mojiretsu = "文字列"; // '文字列'でも同じ↓
12 ↓
13   print "これは $mojiretsu です。<br>¥n";↓
14   print 'これは $mojiretsu です。<br>' . "¥n";↓
15 ↓
16 ?>↓
17 ↓
18 <p><a href="index.html">インデックスに戻る</a></p>↓
19 </body>↓
20 </html>↓
```



"..."と'...'の違い

これは 文字列 です。
これは $mojiretsu です。

インデックスに戻る

- 172.16.123.0のpublic_htmlフォルダにsample.phpとして作成。
- アクセス： https://172.16.123.30/~自分のUser ID/sample.php

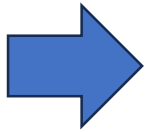
Python (Flask)の例

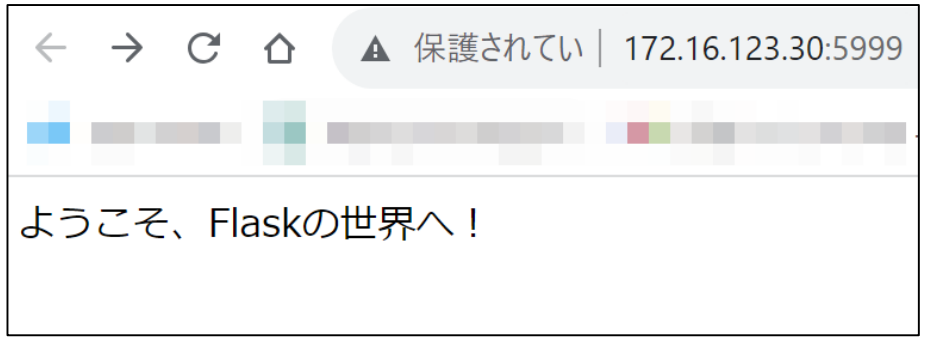
```python
# hellow.py: 最初のFlaskアプリ
from flask import Flask

app = Flask(__name__)

# トップ
@app.route('/')
def hellow_world():
    mojiretsu = 'ようこそ，Flaskの世界へ！'
    print(mojiretsu)
    return mojiretsu

```





- Flaskモジュールを呼び出し，Flaskクラスのインスタンスをappとして作成
- アクセス先は修飾子(7行目)で指定。この場合はトップ(/)のアクセスを意味する。

Python (Flask)の例：動作方法

1. hellow.pyをテキストエディタで作成し，172.16.123.30のホームディレクトリに保存する。
2. 次のコマンドでFlaskを起動する。

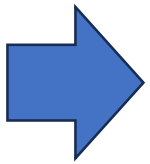
   $ export FLASK_APP=‘hellow.py’←ソースコード名を指定

   $ export FLASK_ENV='development'←開発モードを指定

   $ flask run --port=5MMM --host=0.0.0.0←ポート番号5MMMで外部アクセス許可してflask起動
3. 上記が動かない場合はpython -m flask runと指定
4. 起動したら，適当なブラウザ(Google ChromeかEdgeを推奨)でhttp://172.16.123.30:5MMM/(MMMは自分の学籍番号下3桁)を閲覧し，文字列が表示されていることを確認する。

JavaScript (Node.js)の例

```
1  // app_first.js: 最初のNode.jsアプリ
2  // expressインスタンス生成
3  const express = require('express');
4  const app = express();
5  const port = 5999; // ポート番号
6
7  // トップ
8  app.get('/', (req, res) => {
9      res.send('Hellow, World!');
10 });
11
12 // Webサーバ待機
13 app.listen(port, () => {
14     console.log(`Express server http://localhost:${port}`);
15 });
```



1. app_first.jsをテキストエディタで作成し，172.16.123.30のホームディレクトリに保存する。
2. 次のコマンドで起動する。
   $ npm install express ←expressをインストール
   $ node app_first.js ←Node.js起動
3. 起動したら，適当なブラウザ(Google ChromeかEdgeを推奨)でhttp://172.16.123.30:5MMM/(MMMは自分の学籍番号下3桁)を閲覧し，文字列が表示されていることを確認する。

[復習] 本日の内容

- Webサーバサイドのアプリケーション
- PHP
- Python (Flask)
- JavaScript (Node.js)

本日の課題

1. PHP，Python(Flask)，JavaScript(Node.js)の例を次のように変更してそれぞれ実行し，ブラウザでスナップショットを取れ。
   - PHP：「文字列」を「学籍番号　氏名」に書き換え。
   - Python：「ようこそ・・・」を「学籍番号　氏名」に書き換え。
   - JavaScript：「Hellow, World」を「学籍番号　氏名」に書き換え。
2. 本日の課題フォームにアクセスし，全ての設問に回答し，上記のスナップショットをアップロードせよ。

回答フォーム：
https://forms.office.com/r/gxRbMFZKTi