サーバシステムテスト対策

シラバス到達目標 :

第 01 回

講義概要

- Web アプリケーションの概念形成と、実践的なシステム構築。
- 複数の言語、複数の部品を組み合わせ、統合する能力の習得。
- 実践的な応用力の養成。

到達目標

- 学習した技術をベースとして、これらを統合した実践的な web 開発の皇帝を習得する所。
- システム開発の考え方、進め方などを知ること。
- システムエンジニアの仕事の工程を体験的に知り、即戦力としての基本能力を養成すること。

第 02 回

Xrdp

- xrdp とは?
    - xrdp は、Microsoft RDP プロトコルのフリーかつオープンソースのサーバである。Apache License, Version 2.0 でライセンスされ、主に Linux で利用可能である。RDP によるグラフィカルログインを提供する。
    - Linux へのリモートデスクトップの接続を可能とするもの。

Apache

- Apache とは?
    - オープンソースの Web サーバソフトウェア。HTTP サーバとして広く利用されており、高いカスタマイズ性と拡張性を持つ。

Linux 基本コマンド・Bash Script について

- 基本コマンド

    - `ls` - ディレクトリの内容を一覧表示
    - `cd` - ディレクトリの移動
    - `cp` - ファイルやディレクトリをコピー
    - `mv` - ファイルやディレクトリを移動または名前を変更
    - `rm` - ファイルやディレクトリを削除
    - `chmod` - ファイルの権限を変更
    - `chown` - ファイルの所有者を変更
    - `ps` - 実行中のプロセスを表示
    - `grep` - 文字列検索
    - `find` - ファイル検索

- Bash Script
    - シェルスクリプトの基本構文

```
#!/bin/bash
echo "Hello, World!"
```

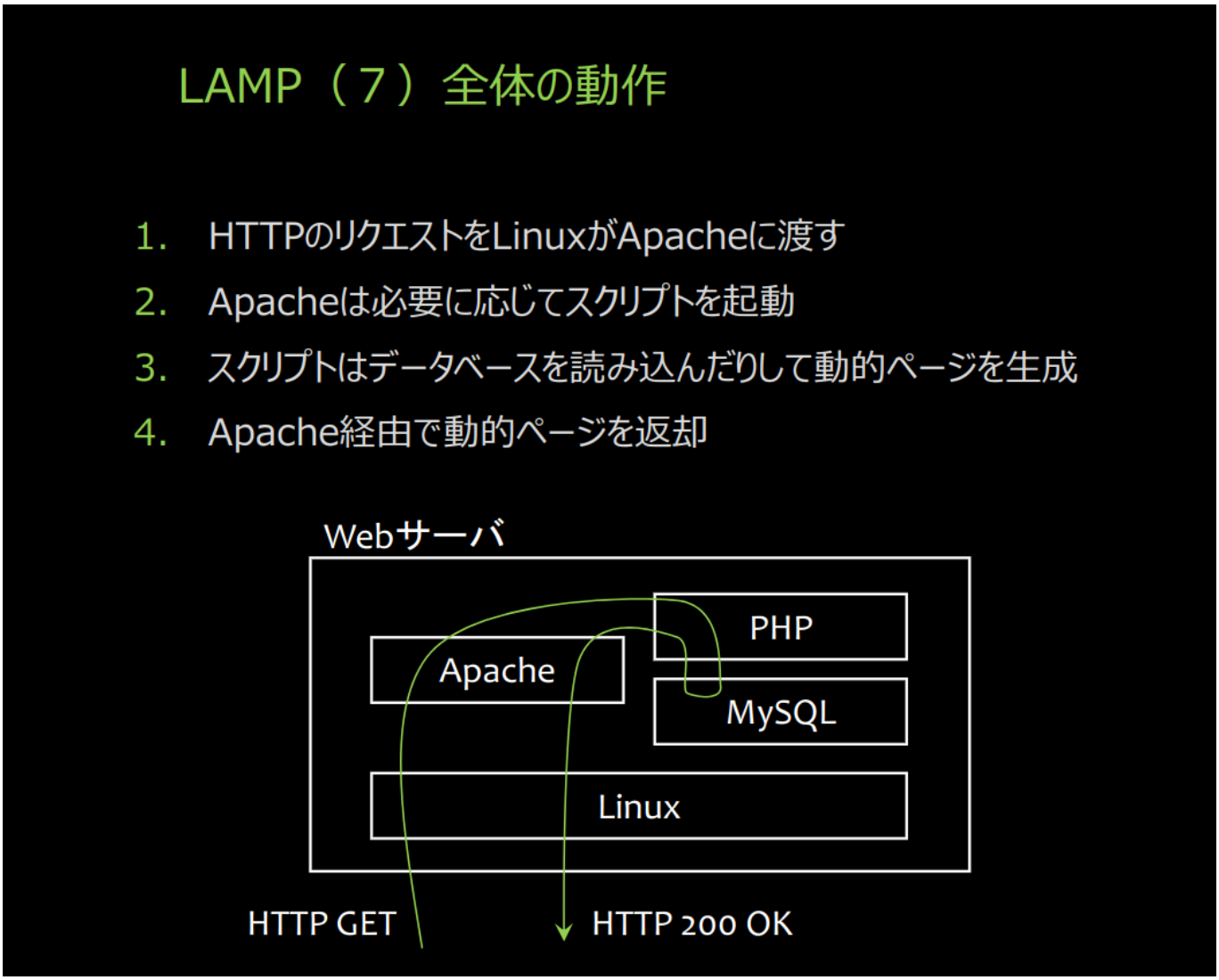
Unicode と UTF-8 の違い

- Unicode
    - 世界中の文字を一意に表すための標準文字コード。
    - 各文字に対して一意の番号（コードポイント）を割り当てる。
- UTF-8
    - Unicode をエンコードするための可変長文字エンコーディング方式。
    - 1〜4 バイトで 1 つの文字を表す。

第 03 回

LAMP

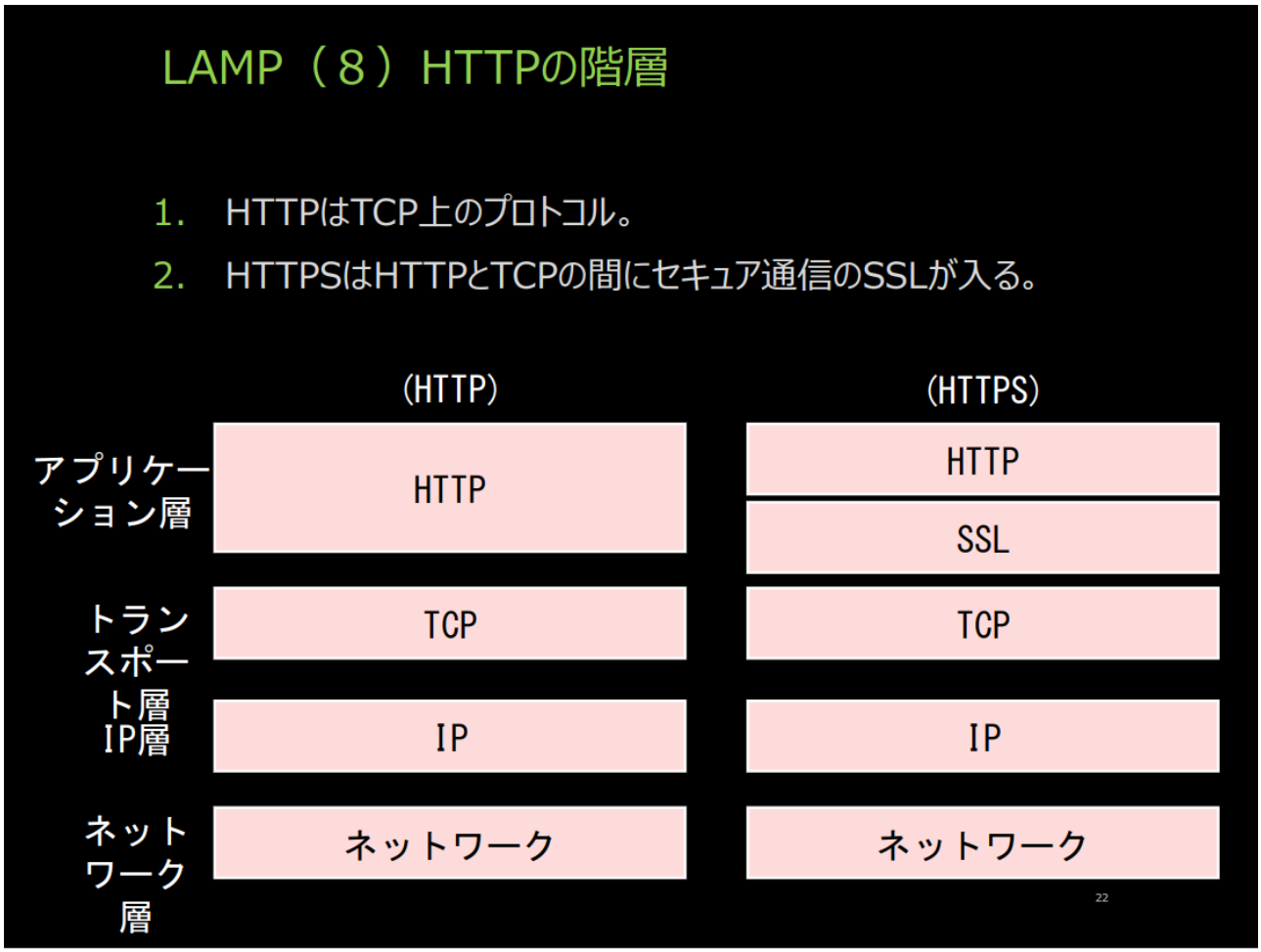
- LAMP とは?
    - Linux + Apache + MySQL + P 言語(php, perl, python 等) の頭文字を取ったもので、オープンソースソフトウェアを組み合わせた Web サービスの構築環境を指す。
- LAMP 全体の動作
    1. HTTP リクエストを Linux が Apache に渡す。(HTTP GET)
    2. Apache は必要に応じてスクリプトを実行。
    3. スクリプトはデータベースを読み込んだりして動的ページを作成。
    4. Apache 経由で動的ページを返却。(HTTP 200 OK)



LAMP HTTP の階層

- HTTP と HTTPS の階層
    - HTTP は TCP 上のプロトコル
    - HTTPS は HTTP と TCP の間にセキュア通信の SSL が入る

層	HTTP	HTTPS
アプリケーション層	HTTP	HTTP
		SSL
トランスポート層	TCP	TCP
IP 層	IP	IP
ネットワーク層	ネットワーク	ネットワーク



通信プロトコル

- 通信プロトコルの基本
  - 通信プロトコルとは、データ通信を行うためのルールセット。
  - 代表的なプロトコル：HTTP, FTP, SMTP, TCP/IP, UDP

第 04 回

JavaScript

1. 変数 (let, const, var)

宣言の種類	再代入	再宣言	スコープ
const	NG	NG	ブロック
let	NG	OK	ブロック
var	OK	OK	関数

2. if 文

```
if (条件) {
  // 条件が真の場合の処理
} else {
```

```
  // 条件が偽の場合の処理
}
```

3. 変数と if 文の例

```
let score = 75;
if (score >= 60) {
  console.log("合格");
} else {
  console.log("不合格");
}
```

4. 関数定義

```
function greet(name) {
  return "Hello, " + name;
}
let message = greet("Taro");
console.log(message); // "Hello, Taro"
```

- オブジェクト型の例

```
let person = {
  name: "Yamada",
  age: 30,
};
console.log(person.name); // "Yamada"
```

CSS と JavaScript のインクリメントの仕方

- CSS ファイルの呼び出し

```
<link rel="stylesheet" href="styles.css" />
```

- JavaScript ファイルの呼び出し

```
<script src="script.js"></script>
```

静的・動的 Web ページについて

- 静的 web ページとは

    - 既に出来上がっている HTML ファイルを送り返す方式を静的 Web ページという。
- 動的 web ページとは
    - リクエストの都度、ダイナミックに HTML ファイルを作り出す方式を動的 Web ページという。

クライアントサイド技術とは?

- クライアントサイド技術
    - ユーザーのブラウザ上で動作する技術。
    - 代表的な技術: HTML, CSS, JavaScript

問題

- 四則演算
    - 加算、減算、乗算、除算の基本演算子を使用。

```
let a = 10;
let b = 20;
console.log(a + b); // 30
console.log(a - b); // -10
console.log(a * b); // 200
console.log(a / b); // 0.5
```

- `if`, `else`
    - 条件分岐に使用。

```
if (条件) {
  // 条件が真の場合の処理
} else {
  // 条件が偽の場合の処理
}
```

- `this.xxx`
    - オブジェクトのプロパティを指す際に使用。

```
function Person(name) {
  this.name = name;
}
let p = new Person("Yamada");
console.log(p.name); // "Yamada"
```

- JavaScript を用いたアニメーションの作成

- JavaScript と CSS を組み合わせてアニメーションを実現。

```
let box = document.getElementById("box");
box.style.transition = "transform 2s";
box.style.transform = "translateX(100px)";
```

第 05 回

PHP

CGI プログラム

Web サーバシステムプログラムを実行させる仕掛けのことを CGI (Common Gateway Interface) という。また、これにより動作するプログラムを CGI プログラムという。一般的に次の条件を満たす言語が使われる。

- Web サーバプログラムからの呼び出しが簡単に行えるもの
- 文字列の扱いが容易である

文字コードの指定

PHP で日本語を生成する場合には、次のヘッダーを含める。

```
header('Content-Type: text/html; charset=UTF-8');
```

スクリプトの作成・実行方法

作成

```
<?php
echo date("Y/m/d");
?>
```

実行

```
php date.php
```

なお、HTML ファイルに直接書き込むこともできる。

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8" />
```

```
    <title>PHP Example</title>
  </head>
  <body>
    <?php echo date("Y/m/d"); ?>
  </body>
</html>
```

変数

1. `floor()`: 少数以下を切り捨てる。
2. `ceil()`: 少数を切り上げる。
3. `sqrt()`: 平方根を求める。
4. `array_fill()`: 配列を指定した値で埋める。
5. `isset()`: 配列に含まれているか調べる。
6. `unset()`: 配列から取り除く。
7. `array_keys()`: 配列のキーを返す。

Java との違い

- Java とは違い、クラスが無くても動かせる。
- 変数を宣言なしで使える。型は推論で決まる。
- オブジェクトのメソッドやメンバへのアクセスが「->」で行われる。
- 文字列の連結は「.」で行う。

文字の連結

PHP では、文字列の連結に「.」を使用する。

```
// firstName 変数に "John" を代入
$firstName = "John";

// lastName 変数に "Doe" を代入
$lastName = "Doe";



## サーバシステムテスト

// fullName 変数に firstName と lastName を空白区切りで連結した文字列を代入
$fullName = $firstName . " " . $lastName;

// fullName 変数の値を出力
echo $fullName; // John Doe
```

連想配列

連想配列について詳しく説明する。

連想配列とは？

連想配列（Associative Array）は、キーと値のペアでデータを管理する配列の一種である。通常の配列がインデックス（0, 1, 2, ...）をキーとして使用するのに対し、連想配列では任意の文字列や数値をキーとして使用できる。

連想配列の基本

連想配列の定義

連想配列は `array()` 関数を使って定義する。キーと値のペアを指定する。

```
// 連想配列の定義
// array() 関数を使って連想配列を定義する。
// キーと値のペアを指定する。
$studentGrades = array(
    "John" => 85, // "John" というキーに 85 という値を割り当てる
    "Jane" => 90, // "Jane" というキーに 90 という値を割り当てる
    "Jim" => 78  // "Jim" というキーに 78 という値を割り当てる
);
```

連想配列の要素にアクセスする

```
// キーを指定して、対応する値にアクセスできる。
echo $studentGrades["Jane"]; // 90
```

連想配列に新しい要素を追加する

```
// 新しいキーと値のペアを追加する。
$studentGrades["Jake"] = 88;
```

連想配列の要素を更新する

```
// 既存のキーに対して新しい値を割り当てる。
$studentGrades["Jim"] = 82;
```

連想配列の要素を削除する

```

```

```php
// `unset()` 関数を使って要素を削除する。
unset($studentGrades["John"]);
```

連想配列の例

```php
// 以下は、連想配列を使用して商品の在庫を管理する例である。
$inventory = array(
    "apple" => 50,
    "banana" => 30,
    "orange" => 40
);

// 在庫を表示
foreach ($inventory as $item => $quantity) {
    echo $item . ": " . $quantity . " items\n";
}

// 在庫を追加
$inventory["grape"] = 25;

// 在庫を更新
$inventory["apple"] += 20; // 20個追加

// 在庫を削除
unset($inventory["banana"]);

// 更新後の在庫を表示
foreach ($inventory as $item => $quantity) {
    echo $item . ": " . $quantity . " items\n";
}
```

連想配列の使用例

フォームデータの処理

```php
// 連想配列は、フォームデータの処理に便利である。
$formData = array(
    "name" => "John Doe",
    "email" => "john@example.com",
    "age" => 25
);

echo "Name: " . $formData["name"] . "\n";
echo "Email: " . $formData["email"] . "\n";
echo "Age: " . $formData["age"] . "\n";
```

JSON データの処理

// 連想配列は JSON データの処理にも使用される。JSON データを PHP の連想配列に変換するには、`json_decode()` 関数を使用する。

```
$jsonData = '{"name": "John Doe", "email": "john@example.com", "age": 25}';
$arrayData = json_decode($jsonData, true);

echo "Name: " . $arrayData["name"] . "\n";
echo "Email: " . $arrayData["email"] . "\n";
echo "Age: " . $arrayData["age"] . "\n";
```

インデックスキー、バリュー等のペアの例示

```
$ages = array(
    "Peter" => 35,
    "Ben" => 37,
    "Joe" => 43
);
echo $ages["Peter"]; // 35
```

関数の定義

// PHP では関数を定義することで、コードの再利用性を高めることができる。

```
function greet($name) {
    return "Hello, " . $name . "!";
}

echo greet("World"); // Hello, World!
```

```
<?php
// エラトステネスの篩を使って素数を求める関数
function eratosthenes($max) {
    $sqrt = floor(sqrt($max)); // 最大値の平方根を求める
    $lists = array_fill(2, $max-1, true); // 2から$maxまでの数をtrueで初期化した配列を作成

    // 2から最大値の平方根までの数をループ
    for ($i=2; $i<=$sqrt; $i++) {
        if (isset($lists[$i])) { // 現在の数がリストに存在する場合
            for ($j=$i*2; $j<=$max; $j+=$i) { // 現在の数の倍数をループ
                unset($lists[$j]); // 倍数をリストから削除
            }
        }
```

```
    }
    return array_keys($lists); // リストに残った数（素数）のキーを返す
}

// ヘッダーを設定し、文字コードをUTF-8にする
header('Content-Type: text/html; charset=UTF-8');

// 素数を求める最大値を設定
$m = 100;
echo '<p>' . $m . "までの素数を求めます。</p>\n";

// エラトステネスの篩を使って素数を求め、結果を表示
foreach (eratosthenes($m) as $key => $value) {
    echo '<p>' . ($key+1) . '番目の素数は' . $value . "です。</p>\n";
}
?>
```

各行に対するコメントと説明

```
<?php
// エラトステネスの篩を使って素数を求める関数
function eratosthenes($max) {
    $sqrt = floor(sqrt($max)); // 最大値の平方根を求める
    $lists = array_fill(2, $max-1, true); // 2から$maxまでの数をtrueで初期化した配列を作成

    // 2から最大値の平方根までの数をループ
    for ($i=2; $i<=$sqrt; $i++) {
        if (isset($lists[$i])) { // 現在の数がリストに存在する場合
            for ($j=$i*2; $j<=$max; $j+=$i) { // 現在の数の倍数をループ
                unset($lists[$j]); // 倍数をリストから削除
            }
        }
    }
    return array_keys($lists); // リストに残った数（素数）のキーを返す
}

// ヘッダーを設定し、文字コードをUTF-8にする
header('Content-Type: text/html; charset=UTF-8');

// 素数を求める最大値を設定
$m = 100;
echo '<p>' . $m . "までの素数を求めます。</p>\n";

// エラトステネスの篩を使って素数を求め、結果を表示
foreach (eratosthenes($m) as $key => $value) {
    echo '<p>' . ($key+1) . '番目の素数は' . $value . "です。</p>\n";
}
?>
```

説明

1. **関数 `eratosthenes` の定義**
    - `eratosthenes` は、エラトステネスの篩を使って素数を求める関数である。
    - 引数 `$max` は、素数を求める範囲の最大値である。
2. **平方根の計算**
    - `$sqrt` は、最大値 `$max` の平方根を求める。
    - `floor()` 関数を使って、小数点以下を切り捨てる。
3. **配列の初期化**
    - `$lists` は、2 から最大値 `$max` までの数値をキーに持つ配列で、すべての値を `true` に初期化する。
4. **篩の処理**
    - 2 から平方根 `$sqrt` までの数をループする。
    - その数がリストに存在する場合、その数の倍数をリストから削除する。
5. **結果の返却**
    - リストに残った数（素数）のキーを配列として返す。
6. **ヘッダーの設定**
    - `header('Content-Type: text/html; charset=UTF-8');` は、出力する HTML の文字コードを UTF-8 に設定する。
7. **最大値の設定**
    - `$m` に素数を求める最大値を設定する（この例では 100）。
8. **結果の表示**
    - `echo` を使って、素数を求める範囲を表示する。
    - `foreach` ループで、エラトステネスの篩を使って求めた素数を一つずつ表示する。

このコードは、エラトステネスの篩アルゴリズムを使って指定された範囲内の素数を求め、結果を HTML 形式で表示する簡単な PHP スクリプトである。 了解しました。以下に、HTML の`<table>`タグを使用して簡単な表を作成する方法を説明します。

`<table>`タグの基本構造

HTML で表を作成する際には、`<table>`タグを使用します。基本的な構造は以下の通りです。

```
<table>
  <caption>
    表のタイトル
  </caption>
  <tr>
```

```
      <th>ヘッダ1</th>
      <th>ヘッダ2</th>
    </tr>
    <tr>
      <td>データ1</td>
      <td>データ2</td>
    </tr>
  </table>
```

各タグの説明

1. **<table>タグ**
    - 表全体を囲むタグです。
2. **<caption>タグ**
    - 表のタイトルを指定します。省略可能です。
3. **<tr>タグ**
    - 表の行（row）を定義します。<table>タグの中に複数配置します。
4. **<th>タグ**
    - 表のヘッダーセルを定義します。通常、最初の行に配置します。
5. **<td>タグ**
    - 表のデータセルを定義します。<tr>タグの中に配置します。

実際の例

以下に、簡単な表の例を示します。

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8" />
    <title>簡単な表の例</title>
  </head>
  <body>
    <table border="1">
      <caption>
        学生の成績表
      </caption>
      <tr>
        <th>名前</th>
        <th>数学</th>
        <th>英語</th>
      </tr>
      <tr>
```

```
        <td>山田太郎</td>
        <td>90</td>
        <td>80</td>
      </tr>
      <tr>
        <td>鈴木一郎</td>
        <td>75</td>
        <td>85</td>
      </tr>
    </table>
  </body>
</html>
```

各行に対する説明

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8" />
    <title>簡単な表の例</title>
  </head>
  <body>
    <!-- 表を定義する -->
    <table border="1">
      <!-- 表のタイトル -->
      <caption>
        学生の成績表
      </caption>
      <!-- ヘッダー行 -->
      <tr>
        <th>名前</th>
        <!-- ヘッダーセル1 -->
        <th>数学</th>
        <!-- ヘッダーセル2 -->
        <th>英語</th>
        <!-- ヘッダーセル3 -->
      </tr>
      <!-- データ行1 -->
      <tr>
        <td>山田太郎</td>
        <!-- データセル1 -->
        <td>90</td>
        <!-- データセル2 -->
        <td>80</td>
        <!-- データセル3 -->
      </tr>
      <!-- データ行2 -->
      <tr>
        <td>鈴木一郎</td>
        <!-- データセル1 -->
        <td>75</td>
```

```
        <!-- データセル2 -->
        <td>85</td>
        <!-- データセル3 -->
      </tr>
    </table>
  </body>
</html>
```

PHP で表を動的に生成する例

以下に、PHP を使用して同様の表を動的に生成する例を示します。

```
<?php
// ヘッダーを設定し、文字コードをUTF-8にする
header('Content-Type: text/html; charset=UTF-8');
?>
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <title>簡単な表の例</title>
</head>
<body>
    <?php
    // 表を生成する
    echo "<table border=\"1\">\n";
    echo "<caption>学生の成績表</caption>\n";
    echo "<tr><th>名前</th><th>数学</th><th>英語</th></tr>\n";
    echo "<tr><td>山田太郎</td><td>90</td><td>80</td></tr>\n";
    echo "<tr><td>鈴木一郎</td><td>75</td><td>85</td></tr>\n";
    echo "</table>\n";
    ?>
</body>
</html>
```

この PHP スクリプトは、上記の HTML と同じ表を動的に生成します。echo文を使って HTML コードを出力することで、表を作成しています。

各行に対する説明

```
<?php
// ヘッダーを設定し、文字コードをUTF-8にする
header('Content-Type: text/html; charset=UTF-8');
?>
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <title>簡単な表の例</title>
```

```
</head>
<body>
    <?php
    // 表を生成する
    echo "<table border=\"1\">\n"; // <table>タグの開始
    echo "<caption>学生の成績表</caption>\n"; // <caption>タグ
    echo "<tr><th>名前</th><th>数学</th><th>英語</th></tr>\n"; // ヘッダー行
    echo "<tr><td>山田太郎</td><td>90</td><td>80</td></tr>\n"; // データ行1
    echo "<tr><td>鈴木一郎</td><td>75</td><td>85</td></tr>\n"; // データ行2
    echo "</table>\n"; // <table>タグの終了
    ?>
</body>
</html>
```

この例では、PHP コードを使って HTML の表を動的に生成し、ブラウザに表示する方法を示しています。

第 06 回

HTTP

HTTP 概要

特徴	説明
アプリケーションレイヤ	通信プロトコルの一つであり、ウェブブラウザとウェブサーバ間のデータ転送に使用される。
ステートレス	各リクエストとレスポンスは独立しており、通信が終了する度に状態がリセットされる。
セッション管理	Cookie やセッション ID などを用いて、ステートレスな特性を補完する。
テキストデータ	主にテキスト形式のデータをやり取りする。
暗号化なし	通常の HTTP 通信は暗号化されていない (HTTPS で暗号化が可能) 。
メソッド	クライアントがサーバに要求する動作 (GET、POST など) が規定されている。
ステータスコード	通信の結果を示すコード (例：200 OK、404 Not Found) 。

リクエストとレスポンスの構成

- **HTTP リクエスト**
    - クライアントからサーバへの要求
    - 構成: メソッド、URL、バージョン、ヘッダー、ボディ

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html
```

- **HTTP レスポンス**
  - サーバからクライアントへの応答
  - 構成: バージョン、ステータスコード、ステータスメッセージ、ヘッダー、ボディ

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 123

<html>
<body>Hello, World!</body>
</html>
```

メソッド

メソッド	説明
GET	リソースの取得
POST	データの送信 (新規作成)
PUT	データの更新
DELETE	リソースの削除
HEAD	ヘッダー情報の取得 (ボディなし)
OPTIONS	利用可能なメソッドの取得
TRACE	リクエストのループバックテスト
CONNECT	トンネルを確立するためのプロキシ使用

URL パラメータ

- URL に含まれるパラメータを使って、サーバにデータを送信する方法
- `?` の後にキーと値を指定し、複数のパラメータは `&` で区切る

```
http://www.example.com/page?name=John&age=30
```

  - **例**:
    - `name=John`
    - `age=30`

まとめ

HTTP はウェブ通信の基盤であり、リクエストとレスポンスの仕組みを理解することが重要です。各メソッドやステータスコードの役割、そして URL パラメータの使い方を把握することで、効果的なウェブ開発が可能となる。

第 07 回

MySQL

MySQL 概要

MySQL は、オープンソースのリレーショナルデータベース管理システム (RDBMS) であり、ウェブアプリケーションのデータベースとして広く使用されています。SQL (Structured Query Language) を使用してデータの操作や管理を行います。

実行方法

1. **MySQL サーバの起動**

```
sudo service mysql start
```

2. **MySQL へのログイン**

```
mysql -u ユーザー名 -p
```

`-u`オプションでユーザー名を指定し、`-p`オプションでパスワード入力を促します。

データベース操作

1. **データベースの作成**

```
CREATE DATABASE データベース名;
```

2. **データベースの選択**

```
USE データベース名;
```

3. **データベースの削除**

```
DROP DATABASE データベース名;
```

テーブル操作

1. **テーブルの作成**

```
CREATE TABLE テーブル名 (
    カラム名1 データ型1,
```

```
    カラム名2 データ型2,
    ...
);
```

例:

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100)
);
```

2. **テーブルの削除**

```
DROP TABLE テーブル名;
```

3. **テーブルの表示**

```
SHOW TABLES;
```

4. **テーブルの構造表示**

```
DESCRIBE テーブル名;
```

レコード操作

1. **レコードの挿入**

```
INSERT INTO テーブル名 (カラム名1, カラム名2, ...)
VALUES (値1, 値2, ...);
```

2. **レコードの選択**

```
SELECT カラム名1, カラム名2, ...
FROM テーブル名
WHERE 条件;
```

例:

```
SELECT * FROM users;
```

3. **レコードの更新**

```
UPDATE テーブル名
SET カラム名1 = 新しい値1, カラム名2 = 新しい値2, ...
WHERE 条件;
```

4. **レコードの削除**

```
DELETE FROM テーブル名
WHERE 条件;
```

PHP からのデータベースアクセス

1. **データベースへの接続**

```
<?php
$servername = "localhost";
$username = "ユーザー名";
$password = "パスワード";
$dbname = "データベース名";

// 接続を作成する
$conn = new mysqli($servername, $username, $password, $dbname);

// 接続をチェックする
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

2. **データの挿入**

```
<?php
$sql = "INSERT INTO users (name, email) VALUES ('John Doe',
'john@example.com')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
```

```
}
?>
```

3. **データの選択**

```
<?php
$sql = "SELECT id, name, email FROM users";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // 出力データを各行に対して
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["name"]. " -
Email: " . $row["email"]. "<br>";
    }
} else {
    echo "0 results";
}
?>
```

4. **データの更新**

```
<?php
$sql = "UPDATE users SET email='newemail@example.com' WHERE id=1";

if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}
?>
```

5. **データの削除**

```
<?php
$sql = "DELETE FROM users WHERE id=1";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}
?>
```

6. **接続の終了**

```php
<?php
$conn->close();
?>
```

第 08 回

JSP(1)

Java サーブレット

Java サーブレットとは?

- サーバ上で動作する Java プログラム
- 動的 Web アプリケーションを構築するためのサーバサイド技術 (クライアントサイド技術は JavaScript 等)
- Java サーブレットはサーバ上のサーブレットコンテナという実行環境上で実行される

サーブレットコンテナ Tomcat

サーブレットコンテナとは?

- Java サーブレット実行環境

Apache Tomcat

- Apache Software Foundation が開発しているオープンソースのサーブレットコンテナ
- Apache Tomcat はコンテナと HTTP サーバから構成される
    - **コンテナ**: Catalina
        - **Catalina**: サーブレットの実行環境を提供するコンポーネント。リクエストの受信、サーブレットのライフサイクル管理、セッション管理を行う。
    - **HTTP サーバ**: Coyote
        - **Coyote**: HTTP リクエストとレスポンスを処理するコンポーネント。HTTP/1.1 プロトコルのサポートを提供し、ブラウザからのリクエストを受け取る。

JSP の基本

JSP とは

- Java Server Pages
- 動的 Web アプリケーションを構築するためのサーバサイドの技術
- **HTML の中に断片的な Java プログラムを埋め込む簡易なサーバサイド処理技術**

ディレクティブ

- JSP ページの処理方式をコンテナに伝える。
- 書式 : `<%@ ... %>`
- 原則として同じ属性を 2 回以上指定することはできない。

- 代表的なディレクティブ: `@page`

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
```

- **属性の例**
  - `contentType`: JSP ページの出力コンテンツタイプを指定する。
  - `language`: 使用する言語を指定する (通常は "java") 。

ディレクティブの具体例

- `@include`ディレクティブを使用して他のファイルをインクルードする例：

```
<%@ include file="header.jsp" %>
```

- `@taglib`ディレクティブを使用してタグライブラリを宣言する例：

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

スクリプトレット

- HTML に埋め込む小さなプログラム
- 書式 : `<% ... %>`
- 一つ以上の命令文で、命令文はセミコロン`;`で終わる。
- 暗黙オブジェクト : インスタンス化手続きなしで使用できるオブジェクト (例: `out`, `request`, `response`)

スクリプトレットの具体例

```
<html>
<body>
    <h2>現在の日時</h2>
    <%
        java.util.Date date = new java.util.Date();
        out.println(date.toString());
    %>
</body>
</html>
```

- **暗黙オブジェクトの例**
  - `out`: `JspWriter` オブジェクトで、レスポンスに出力するために使用する。
  - `request`: クライアントのリクエスト情報を含む `HttpServletRequest` オブジェクト。
  - `response`: サーバのレスポンス情報を含む `HttpServletResponse` オブジェクト。

その他の JSP タグ

- **宣言タグ**: メソッドや変数を定義する。

```
<%!
private String getGreeting() {
    return "Hello, JSP!";
}
%>
```

- **式タグ**: 式を評価して結果を出力する。

```
<%= new java.util.Date() %>
```

JSP ページの完全な例

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
    <title>JSPの基本</title>
</head>
<body>
    <h1>ようこそ、JSPの世界へ！</h1>
    <p>現在の日時: <%= new java.util.Date() %></p>
    <h2>ユーザーリスト</h2>
    <%
        String[] users = {"Alice", "Bob", "Charlie"};
        for (String user : users) {
            out.println("<p>" + user + "</p>");
        }
    %>
</body>
</html>
```

以下に、JSP の宣言部、式、アクションタグ、および Cookie について詳細に説明し、表を用いてわかりやすく可視化しました。

第 09 回

JSP(2)

宣言部

JSP の宣言部は、Java のメソッドや変数を宣言するために使用される。宣言部に記述されたコードは、サーブレットクラスのメンバとして定義される。

宣言部の書式

```
<%! Javaコード %>
```

宣言部の例

```
<%!
    private String greet() {
        return "Hello, World!";
    }
%>
```

特徴	説明
目的	JSP ページにメソッドや変数を定義する
使用場所	JSP ページ全体で使用可能
書式	`<%! ... %>`

式(Expression)

JSP の式は、Java の値を評価し、その結果を HTML ページに直接出力するために使用される。式は、ページがリクエストされたときに実行される。

式の書式

```
<%= 式 %>
```

式の例

```
<html>
<body>
    <h1>現在の日時: <%= new java.util.Date() %></h1>
</body>
</html>
```

特徴	説明
目的	JSP ページに Java の値を埋め込む
使用場所	HTML タグ内で使用可能

特徴	説明
書式	`<%= ... %>`

アクションタグ

JSP のアクションタグは、JavaBeans の操作、他のリソースのインクルード、転送などのアクションを実行するために使用される。

主なアクションタグ

アクションタグ	説明	例
`<jsp:useBean>`	JavaBean のインスタンスを生成する	`<jsp:useBean id="user" class="com.example.User" />`
`<jsp:setProperty>`	JavaBean のプロパティを設定する	`<jsp:setProperty name="user" property="name" value="John" />`
`<jsp:getProperty>`	JavaBean のプロパティを取得する	`<jsp:getProperty name="user" property="name" />`
`<jsp:include>`	他のリソースをインクルードする	`<jsp:include page="header.jsp" />`
`<jsp:forward>`	リクエストを他のリソースに転送する	`<jsp:forward page="login.jsp" />`

Cookie

1. Cookie の概念

Cookie は、ユーザーのブラウザに保存される小さなデータの断片です。サーバは、ユーザーの再訪問時に Cookie を読み取ることで、ユーザーの状態を維持しする。

特徴	説明
目的	ユーザーの状態や設定を保存する
保存場所	ユーザーのブラウザ
有効期限	設定可能（セッション終了まで、特定の期間など）

2. Cookie 内容設定

JSP で Cookie を設定する方法は以下の通りだ。

```
<%
    Cookie cookie = new Cookie("username", "JohnDoe");
    cookie.setMaxAge(60*60*24); // 1日間有効
```

```
    response.addCookie(cookie);
%>
```

メソッド	説明
`new Cookie(String name, String value)`	Cookie を作成
`setMaxAge(int expiry)`	Cookie の有効期限を設定（秒単位）
`response.addCookie(Cookie cookie)`	Cookie をレスポンスに追加

3. Cookie の利用例

JSP で Cookie を読み取る方法は以下の通りである。

```
<%
    Cookie[] cookies = request.getCookies();
    if (cookies != null) {
        for (Cookie cookie : cookies) {
            if (cookie.getName().equals("username")) {
                String username = cookie.getValue();
                out.println("Welcome back, " + username);
            }
        }
    }
%>
```

手順	説明
`request.getCookies()`	クライアントから送信されたすべての Cookie を取得
`cookie.getName()`	Cookie の名前を取得
`cookie.getValue()`	Cookie の値を取得

まとめ

宣言部を使用してメソッドや変数を定義し、式を使用して動的に Java の値を出力し、アクションタグを使用して JavaBeans の操作やリソースのインクルードを行い、Cookie を使用してユーザーの状態を保存することができる。これにより、よりダイナミックでインタラクティブな Web アプリケーションを構築することができる。

以下に、JSP(3)に関する内容と歴史を含めた説明を、表やソースコードを使用して初心者でもわかりやすく学べるように記述する。

第 10 回

JSP(3)

JDBC (Java Database Connectivity)

JDBC は、Java アプリケーションからデータベースに接続して操作を行うための API である。これにより、データベースのクエリ実行やデータの更新が可能となる。

JDBC の歴史

- **1996 年**: Java 1.1 で JDBC が導入された。これにより、Java アプリケーションからデータベースに接続する標準的な方法が提供された。
- **現在**: JDBC は多くのデータベースでサポートされ、Java EE 環境で広く使用されている。

JDBC の主要なクラス

クラス	説明
`DriverManager`	データベースへの接続を管理するクラス
`Connection`	データベース接続を表すクラス
`Statement`	SQL クエリを実行するためのクラス
`ResultSet`	SQL クエリの結果を格納するクラス

JDBC の使用例

```
// JDBCを使用してデータベースに接続し、データを取得する例
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class JDBCExample {
    public static void main(String[] args) {
        // データベースのURL、ユーザー名、パスワードを指定
        String url = "jdbc:mysql://localhost:3306/mydatabase";
        String user = "root";
        String password = "password";

        try {
            // データベースに接続
            Connection connection = DriverManager.getConnection(url, user,
password);

            // SQLステートメントを作成
            Statement statement = connection.createStatement();

            // SQLクエリを実行して結果を取得
            ResultSet resultSet = statement.executeQuery("SELECT * FROM
users");

            // 結果を表示
            while (resultSet.next()) {
                System.out.println("User ID: " + resultSet.getInt("id"));
```

```
                System.out.println("Username: " +
resultSet.getString("username"));
            }

            // リソースをクローズ
            resultSet.close();
            statement.close();
            connection.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

第 11 回

servlet

サーブレットと JSP

サーブレットは Java で書かれたサーバサイドのプログラムで、クライアントからのリクエストを処理してレスポンスを生成する。JSP は HTML の中に Java コードを埋め込んで動的なウェブページを生成する技術で、両者は連携して使用されることが多い。

比較項目	サーブレット	JSP
コード記述	Java コードが中心	HTML 中心で Java コードを埋め込む
目的	ビジネスロジックの処理	プレゼンテーションロジックの処理
使用方法	クラスとして定義	ページとして定義

アプリケーションクラス

アプリケーションクラスは、サーブレットのライフサイクル全体で共有されるデータや設定を管理するためのクラスである。

メソッドハンドラ

メソッドハンドラは、HTTP リクエストのメソッド (GET, POST など) に応じて適切な処理を行うメソッドを指す。

インスタンス変数

サーブレット内のインスタンス変数は、サーブレットの全インスタンス間で共有される。複数のリクエストが同時に処理される場合、スレッドセーフである必要がある。

JSP 連携

サーブレットと JSP は連携して使用される。サーブレットはビジネスロジックを処理し、JSP はその結果を表示する。

JavaBeans

JavaBeans は再利用可能なコンポーネントで、プロパティ、メソッド、イベントを持つ。JSP から JavaBeans を利用することで、データの受け渡しや処理が容易になる。

第 12 回

servlet(2)

JavaBeans のスコープ

JavaBeans のスコープは、オブジェクトの有効範囲を示す。主なスコープは以下の通り。

スコープ	説明	有効範囲
`page`	同一ページ内	JSP ページ内
`request`	同一リクエスト内	サーブレットと JSP 間
`session`	同一セッション内	ユーザーセッション間
`application`	アプリケーション全体	全てのユーザーとセッション間

JavaBeans の受け渡し

JavaBeans はサーブレットから JSP へデータを受け渡すために使用される。

```
<%-- サーブレットで設定されたJavaBeansを受け取る --%>
<jsp:useBean id="user" class="com.example.User" scope="request" />
<jsp:getProperty name="user" property="username" />
```

MVC モデル

MVC (Model-View-Controller) モデルは、アプリケーションの構造を整理するためのデザインパターンである。

コンポーネント	役割	例
Model	データとビジネスロジックの管理	JavaBeans, JDBC
View	ユーザーインターフェースの表示	JSP
Controller	リクエストの処理とレスポンスの生成	サーブレット

まとめ

JSP とサーブレットは、動的なウェブアプリケーションを構築するための強力なツールである。JDBC を使用してデータベースと連携し、サーブレットと JSP を連携させることで、ビジネスロジックとプレゼンテーションロジックを分離し、効率的な開発が可能となる。また、JavaBeans を使用することでデータの受け渡しが容易になり、MVC モデルを採用することでアプリケーションの構造を整理することができる。以下に、各技術の重要なポイントをまとめた。

- **JDBC**: データベース接続と操作
- **サーブレット**: ビジネスロジックの処理
- **JSP**: プレゼンテーションロジックの処理
- **JavaBeans**: データの受け渡しと再利用可能なコンポーネント
- **MVC モデル**: アプリケーション構造の整理

これらの技術を理解し、効果的に組み合わせることで、高品質なウェブアプリケーションを構築することができる。

第 13 回

まとめ

第 14 回

サーバシステムプログラミングと演習 100 分間テスト

問題 1 Linux について (20 点)

1. Linux とは何か、簡潔に説明しなさい。(5 点)
2. Linux ディストリビューションとは何か、具体例を挙げて説明しなさい。(5 点)
3. Ubuntu とは何か、その特徴を説明しなさい。(5 点)
4. シェルとは何か、その役割と機能を説明しなさい。(5 点)

問題 2 LAMP について (20 点)

1. LAMP とは何か、構成要素とその役割を説明しなさい。(10 点)
2. HTTP とは何か、その概要とレイヤ構成を説明しなさい。(5 点)
3. Apache の役割と、静的 Web ページと動的 Web ページの違いを説明しなさい。(5 点)

問題 3 JavaScript について (20 点)

1. JavaScript とは何か、その概要と Web ブラウザにおける動作を説明しなさい。(5 点)
2. JavaScript でできることを 4 つ挙げなさい。(5 点)
3. HTML ファイル中で JavaScript を記述する方法を説明しなさい。(5 点)
4. JavaScript における変数と if 文の使用方法を説明しなさい。(5 点)

問題 4 PHP について (20 点)

1. PHP とは何か、その概要と特徴を説明しなさい。(5 点)
2. PHP における変数、文字列の連結、連想配列の使用方法を説明しなさい。(5 点)
3. PHP で Web ページを動的に生成する方法を説明しなさい。(5 点)

4. PHP の定義済み変数「スーパーグローバル」について、その役割と具体例を挙げて説明しなさい。(5 点)

問題 5 JSP とサーブレットについて (20 点)

1. JSP と Java サーブレットの違いを説明しなさい。(5 点)
2. JSP におけるディレクティブとスクリプレットの役割を説明しなさい。(5 点)
3. JDBC とは何か、その役割と構成を説明しなさい。(5 点)
4. サーブレットコンテナの役割と、Tomcat におけるファイル構成を説明しなさい。(5 点)

シラバス到達目標

- 学習した技術を統合して実践的な Web 開発を行う能力を養成すること。
- システム開発の進め方を理解し、システムエンジニアとしての基本能力を身につけること。

第 01 回

講義概要

- Web アプリケーションの概念形成と、実践的なシステム構築。
- 複数の言語、複数の部品を組み合わせ、統合する能力の習得。
- 実践的な応用力の養成。

問題

1. **Web アプリケーションの概念**
    - Web アプリケーションとは何かを説明せよ。
2. **システム開発の工程**
    - システムエンジニアが関わる主要な開発工程を列挙せよ。

第 02 回

Xrdp と Apache

- xrdp とは何か、またその主要な機能を述べよ。
- Apache とは何か、またその特徴と用途を説明せよ。

Linux 基本コマンドと Bash Script

1. **基本コマンド**
    - `ls`, `cd`, `cp`, `mv`, `rm`, `chmod`, `chown`, `ps`, `grep`, `find`の各コマンドの機能を説明せよ。
2. **Bash Script**
    - 以下の Bash Script の意味を説明せよ。

```
#!/bin/bash
echo "Hello, World!"
```

Unicode と UTF-8 の違い

1. **Unicode**
    - Unicode とは何かを説明せよ。
2. **UTF-8**
    - UTF-8 とは何かを説明せよ。

第 03 回

LAMP

1. **LAMP とは何か**
    - LAMP の構成要素を説明せよ。
2. **LAMP 全体の動作**
    - LAMP の全体の動作を順を追って説明せよ。

HTTP と HTTPS の階層

1. **階層構造**
    - HTTP と HTTPS の階層構造を図を用いて説明せよ。

通信プロトコル

1. **通信プロトコルの基本**
    - 通信プロトコルとは何かを説明し、代表的なプロトコルを 3 つ挙げよ。

第 04 回

JavaScript

1. **変数**
    - `let`, `const`, `var`の違いを説明せよ。
2. **if 文**
    - 以下の if 文の動作を説明せよ。

```

```

```
let score = 75;
if (score >= 60) {
  console.log("合格");
} else {
  console.log("不合格");
}
```

3. **関数定義**
    - 以下の関数定義の意味を説明せよ。

```
function greet(name) {
  return "Hello, " + name;
}
let message = greet("Taro");
console.log(message); // "Hello, Taro"
```

4. **オブジェクト型の例**
    - 以下のコードの意味を説明せよ。

```
let person = {
  name: "Yamada",
  age: 30,
};
console.log(person.name); // "Yamada"
```

静的・動的 Web ページ

1. **静的 Web ページ**
    - 静的 Web ページとは何かを説明せよ。
2. **動的 Web ページ**
    - 動的 Web ページとは何かを説明せよ。

クライアントサイド技術

1. **クライアントサイド技術**
    - クライアントサイド技術の代表例を 3 つ挙げ、それぞれを説明せよ。

問題

1. **四則演算**
    - 以下の JavaScript コードを実行した結果を答えよ。

```
let a = 10;
let b = 20;
console.log(a + b); // ?
console.log(a - b); // ?
console.log(a * b); // ?
console.log(a / b); // ?
```

2. **`if`, `else`文**
    - `if`および`else`文の役割を説明せよ。
3. **`this`キーワード**
    - 以下のコードの`this`キーワードの意味を説明せよ。

```
function Person(name) {
  this.name = name;
}
let p = new Person("Yamada");
console.log(p.name); // "Yamada"
```

第 05 回

PHP

1. **CGI プログラム**
    - CGI プログラムとは何か、またその条件を説明せよ。
2. **文字コードの指定**
    - PHP で日本語を生成する際に必要なヘッダーを示し、その意味を説明せよ。
3. **スクリプトの作成・実行方法**
    - 以下の PHP スクリプトの作成と実行方法を説明せよ。

```
<?php
echo date("Y/m/d");
?>
```

変数

1. **PHP の組み込み関数**

- `floor()`, `ceil()`, `sqrt()`, `array_fill()`, `isset()`, `unset()`, `array_keys()`の機能を説明せよ。

2. **Java との違い**
   - PHP と Java の変数宣言とオブジェクト操作の違いを説明せよ。

連想配列

1. **連想配列とは何か**
   - 連想配列の概念を説明せよ。
2. **連想配列の操作**
   - 以下の連想配列操作の結果を答えよ。

```
$ages = array(
    "Peter" => 35,
    "Ben" => 37,
    "Joe" => 43
);
echo $ages["Peter"]; // ?
```

3. **連想配列の使用例**
   - フォームデータの処理に連想配列を使用する方法を説明せよ。

第 06 回

HTTP

1. **HTTP の概要**
   - HTTP の基本的な特徴を説明せよ。
2. **リクエストとレスポンスの構成**
   - HTTP リクエストとレスポンスの構成要素を説明せよ。
3. **HTTP メソッド**
   - 主な HTTP メソッドの一覧を示し、それぞれの役割を説明せよ。
4. **URL パラメータ**
   - 以下の URL パラメータの意味を説明せよ。

```
http://www.example.com/page?name=John&age=30
```

第 07 回

MySQL

1. **MySQL 概要**
    - MySQL とは何かを説明せよ。
2. **データベース操作**
    - データベースの作成、選択、削除の方法を説明せよ。
3. **テーブル操作**
    - テーブルの作成、削除、表示、構造表示の方法を説明せよ。
4. **レコード操作**
    - レコードの挿入、選択、更新、削除の方法を説明せよ。
5. **PHP からのデータベースアクセス**
    - PHP から MySQL データベースに接続し、データの挿入、選択、更新、削除を行う方法を説明せよ。

第 08 回

Java サーブレットと JSP

1. **サーブレットとは何か**
    - Java サーブレットの役割を説明せよ。
2. **サーブレットコンテナ**
    - サーブレットコンテナとは何かを説明せよ。
3. **Apache Tomcat**
    - Apache Tomcat の構成要素を説明せよ。
4. **JSP の基本**
    - JSP とは何か、その基本的な機能を説明せよ。
5. **ディレクティブ**
    - JSP のディレクティブを列挙し、それぞれの役割を説明せよ。
6. **スクリプトレット**
    - JSP のスクリプトレットの役割を説明せよ。

第 09 回

JSP(2)

1. **宣言部**
    - JSP の宣言部の書式と使用方法を説明せよ。
2. **式 (Expression)**
    - JSP の式の書式と使用方法を説明せ

よ。

3. **アクションタグ**
    - JSP のアクションタグを列挙し、それぞれの役割を説明せよ。
4. **Cookie**
    - Cookie の概念と使用方法を説明せよ。

第 10 回

JDBC

1. **JDBC の概要**
    - JDBC とは何か、その歴史を含めて説明せよ。
2. **主要なクラス**
    - JDBC の主要なクラスを列挙し、それぞれの役割を説明せよ。
3. **使用例**
    - JDBC を使用してデータベースに接続し、データを取得する方法を具体例を用いて説明せよ。

第 11 回

サーブレット

1. **サーブレットと JSP**
    - サーブレットと JSP の違いと連携方法を説明せよ。
2. **アプリケーションクラス**
    - アプリケーションクラスの役割を説明せよ。
3. **メソッドハンドラ**
    - メソッドハンドラの役割を説明せよ。

4. **インスタンス変数**
    - サーブレット内のインスタンス変数のスレッドセーフ性について説明せよ。
5. **JSP 連携**
    - サーブレットと JSP の連携方法を説明せよ。
6. **JavaBeans**
    - JavaBeans の役割と使用方法を説明せよ。

第 12 回

JavaBeans と MVC モデル

1. **JavaBeans のスコープ**
    - JavaBeans のスコープの種類とそれぞれの有効範囲を説明せよ。
2. **JavaBeans の受け渡し**
    - サーブレットから JSP へ JavaBeans を受け渡す方法を説明せよ。
3. **MVC モデル**
    - MVC モデルとは何か、その構成要素と役割を説明せよ。

以下に、提供されたテスト問題の模範解答を示す。

サーバシステムテスト模範解答

第 01 回

講義概要

1. **Web アプリケーションの概念**
    - Web アプリケーションとは、インターネットを介してブラウザ上で動作するアプリケーションソフトウェアのこと。クライアント側（ユーザーのブラウザ）とサーバ側（アプリケーションサーバ）で動作し、データの送受信や処理を行う。
2. **システム開発の工程**
    - 要件定義
    - 設計
    - 実装
    - テスト
    - デプロイ
    - 保守運用

第 02 回

Xrdp と Apache

1. **xrdp とは**
    - xrdp は、Microsoft RDP プロトコルのフリーかつオープンソースのサーバで、主に Linux で利用可能。RDP によるグラフィカルログインを提供する。
2. **Apache とは**
    - Apache は、オープンソースの Web サーバソフトウェアで、HTTP サーバとして広く利用されており、高いカスタマイズ性と拡張性を持つ。

Linux 基本コマンドと Bash Script

1. **基本コマンド**
    - `ls`: ディレクトリの内容を一覧表示
    - `cd`: ディレクトリの移動
    - `cp`: ファイルやディレクトリをコピー
    - `mv`: ファイルやディレクトリを移動または名前を変更
    - `rm`: ファイルやディレクトリを削除
    - `chmod`: ファイルの権限を変更
    - `chown`: ファイルの所有者を変更
    - `ps`: 実行中のプロセスを表示
    - `grep`: 文字列検索
    - `find`: ファイル検索
2. **Bash Script**
    - 以下の Bash Script は、「Hello, World!」というメッセージを表示する。

```
#!/bin/bash
echo "Hello, World!"
```

Unicode と UTF-8 の違い

1. **Unicode**
    - Unicode は、世界中の文字を一意に表すための標準文字コード。各文字に対して一意の番号（コードポイント）を割り当てる。
2. **UTF-8**
    - UTF-8 は、Unicode をエンコードするための可変長文字エンコーディング方式。1〜4 バイトで 1 つの文字を表す。

第 03 回

LAMP

1. **LAMP とは何か**
    - LAMP は、Linux + Apache + MySQL + P 言語 (php, perl, python 等) で構成されたオープンソースソフトウェアの組み合わせで、Web サービスの構築環境を指す。
2. **LAMP 全体の動作**
    - 1. HTTP リクエストを Linux が Apache に渡す。(HTTP GET)
    - 2. Apache は必要に応じてスクリプトを実行。
    - 3. スクリプトはデータベースを読み込んだりして動的ページを作成。
    - 4. Apache 経由で動的ページを返却。(HTTP 200 OK)

HTTP と HTTPS の階層

1. **階層構造**
    - HTTP は TCP 上のプロトコル
    - HTTPS は HTTP と TCP の間にセキュア通信の SSL が入る

層	**HTTP**	**HTTPS**
アプリケーション層	HTTP	HTTP
		SSL
トランスポート層	TCP	TCP
IP 層	IP	IP
ネットワーク層	ネットワーク	ネットワーク

通信プロトコル

1. **通信プロトコルの基本**
    - 通信プロトコルとは、データ通信を行うためのルールセット。
    - 代表的なプロトコル：HTTP, FTP, SMTP, TCP/IP, UDP

第 04 回

JavaScript

1. **変数**
    - `let`: 再代入は可能だが、再宣言は不可。ブロックスコープ。
    - `const`: 再代入も再宣言も不可。ブロックスコープ。
    - `var`: 再代入も再宣言も可能。関数スコープ。
2. **if 文**

- 以下の if 文は、score が 60 以上の場合「合格」を表示し、そうでない場合は「不合格」を表示する。

```
let score = 75;
if (score >= 60) {
  console.log("合格");
} else {
  console.log("不合格");
}
```

3. **関数定義**
   - 以下の関数定義は、name を引数に取り、「Hello, name」というメッセージを返す。

```
function greet(name) {
  return "Hello, " + name;
}
let message = greet("Taro");
console.log(message); // "Hello, Taro"
```

4. **オブジェクト型の例**
   - 以下のコードは、name と age のプロパティを持つ person オブジェクトを作成し、name プロパティの値を表示する。

```
let person = {
  name: "Yamada",
  age: 30,
};
console.log(person.name); // "Yamada"
```

静的・動的 Web ページ

1. **静的 Web ページ**
   - 静的 Web ページとは、既に出来上がっている HTML ファイルを送り返す方式。
2. **動的 Web ページ**
   - 動的 Web ページとは、リクエストの都度、ダイナミックに HTML ファイルを作り出す方式。

クライアントサイド技術

1. **クライアントサイド技術**
   - HTML: ウェブページの構造を記述するマークアップ言語。
   - CSS: ウェブページのスタイルを指定するスタイルシート言語。

- JavaScript: ウェブページの動作を制御するスクリプト言語。

問題

1. **四則演算**
   - 以下の JavaScript コードを実行した結果は次の通り。

```
let a = 10;
let b = 20;
console.log(a + b); // 30
console.log(a - b); // -10
console.log(a * b); // 200
console.log(a / b); // 0.5
```

2. **`if`, `else`文**
   - `if`文は条件が真の場合に実行されるブロックを指定し、`else`文は条件が偽の場合に実行されるブロックを指定する。

3. **`this`キーワード**
   - `this`キーワードは、現在のオブジェクトを指す。以下のコードでは、`this.name`は`Person`オブジェクトの`name`プロパティを指す。

```
function Person(name) {
  this.name = name;
}
let p = new Person("Yamada");
console.log(p.name); // "Yamada"
```

第 05 回

PHP

1. **CGI プログラム**
   - CGI (Common Gateway Interface) プログラムは、Web サーバシステムプログラムを実行させる仕掛けである。Web サーバプログラムからの呼び出しが簡単に行えること、文字列の扱いが容易であることが条件となる。

2. **文字コードの指定**
   - PHP で日本語を生成する際には、以下のヘッダーを含める。

```
header('Content-Type: text/html; charset=UTF-8');
```

3. **スクリプトの作成・実行方法**
    - 以下の PHP スクリプトは、現在の日付を表示する。

```php
<?php
echo date("Y/m/d");
?>
```

    - 実行方法は、`php date.php`をターミナルで実行する。

変数

1. **PHP の組み込み関数**
    - `floor()`: 少数以下を切り捨てる。
    - `ceil()`: 少数を切り上げる。
    - `sqrt()`: 平方根を求

める。

- `array_fill()`: 配列を指定した値で埋める。
- `isset()`: 変数がセットされているかを調べる。
- `unset()`: 変数のセットを解除する。
- `array_keys()`: 配列のキーを返す。

2. **Java との違い**
    - PHP はクラスがなくても動作し、変数を宣言なしで使用できる。型は推論で決まる。オブジェクトのメソッドやメンバへのアクセスは「->」で行い、文字列の連結は「.」で行う。

連想配列

1. **連想配列とは何か**
    - 連想配列は、キーと値のペアでデータを管理する配列。
2. **連想配列の操作**
    - 以下の連想配列操作の結果は次の通り。

```php
$ages = array(
    "Peter" => 35,
    "Ben" => 37,
    "Joe" => 43
);
echo $ages["Peter"]; // 35
```

3. **連想配列の使用例**

- フォームデータの処理に連想配列を使用する方法。

```php
// 連想配列でフォームデータを処理する
$formData = array(
    "name" => "John Doe",
    "email" => "john@example.com",
    "age" => 25
);

echo "Name: " . $formData["name"] . "\n";
echo "Email: " . $formData["email"] . "\n";
echo "Age: " . $formData["age"] . "\n";
```

第 06 回

HTTP

1. **HTTP の概要**
   - HTTP はアプリケーションレイヤの通信プロトコルで、ウェブブラウザとウェブサーバ間のデータ転送に使用される。ステートレスであり、セッション管理は Cookie などを使用する。主にテキスト形式のデータをやり取りし、通常は暗号化されていない（HTTPS で暗号化可能）。クライアントがサーバに要求する動作（メソッド）が規定されており、通信の結果はステータスコードで示される。
2. **リクエストとレスポンスの構成**
   - **HTTP リクエスト**
     - メソッド、URL、バージョン、ヘッダー、ボディで構成される。

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html
```

   - **HTTP レスポンス**
     - バージョン、ステータスコード、ステータスメッセージ、ヘッダー、ボディで構成される。

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 123

<html>
```

```
<body>Hello, World!</body>
</html>
```

3. **HTTP メソッド**

   - 主な HTTP メソッドの一覧。

メソッド	説明
GET	リソースの取得
POST	データの送信 (新規作成)
PUT	データの更新
DELETE	リソースの削除
HEAD	ヘッダー情報の取得 (ボディなし)
OPTIONS	利用可能なメソッドの取得
TRACE	リクエストのループバックテスト
CONNECT	トンネルを確立するためのプロキシ使用

4. **URL パラメータ**

   - 以下の URL パラメータは、name=John, age=30 のデータを含む。

```
http://www.example.com/page?name=John&age=30
```

第 07 回

MySQL

1. **MySQL 概要**

   - MySQL は、オープンソースのリレーショナルデータベース管理システム (RDBMS) であり、ウェブアプリケーションのデータベースとして広く使用される。SQL (Structured Query Language) を使用してデータの操作や管理を行う。

2. **データベース操作**

   - データベースの作成。

```
CREATE DATABASE mydatabase;
```

   - データベースの選択。

```
```

```
USE mydatabase;
```

- データベースの削除。

```
DROP DATABASE mydatabase;
```

3. **テーブル操作**

- テーブルの作成。

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100)
);
```

- テーブルの削除。

```
DROP TABLE users;
```

- テーブルの表示。

```
SHOW TABLES;
```

- テーブルの構造表示。

```
DESCRIBE users;
```

4. **レコード操作**

- レコードの挿入。

```
INSERT INTO users (name, email) VALUES ('John Doe',
'john@example.com');
```

- レコードの選択。

```
SELECT * FROM users;
```

- レコードの更新。

```
UPDATE users SET email='newemail@example.com' WHERE id=1;
```

- レコードの削除。

```
DELETE FROM users WHERE id=1;
```

5. **PHP からのデータベースアクセス**
   - PHP から MySQL データベースに接続し、データの挿入、選択、更新、削除を行う方法。

```
<?php
$servername = "localhost";
$username = "root";
$password = "password";
$dbname = "mydatabase";

// 接続を作成する
$conn = new mysqli($servername, $username, $password, $dbname);

// 接続をチェックする
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";

// データの挿入
$sql = "INSERT INTO users (name, email) VALUES ('John Doe',
'john@example.com')";
if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

// データの選択
$sql = "SELECT id, name, email FROM users";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["name"]. " -
Email: " . $row["email"]. "<br>";
    }
} else {
    echo "0 results";
}
```

```php
// データの更新
$sql = "UPDATE users SET email='newemail@example.com' WHERE
id=1";
if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}

// データの削除
$sql = "DELETE FROM users WHERE id=1";
if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}

// 接続の終了
$conn->close();
?>
```

第 08 回

Java サーブレットと JSP

1. **サーブレットとは何か**
    - サーブレットはサーバ上で動作する Java プログラムで、動的 Web アプリケーションを構築するためのサーバサイド技術。
2. **サーブレットコンテナ**
    - サーブレットコンテナは、サーブレットの実行環境を提供するコンポーネントである。
3. **Apache Tomcat**
    - Apache Tomcat は、Apache Software Foundation が開発しているオープンソースのサーブレットコンテナで、コンテナ (Catalina) と HTTP サーバ (Coyote) から構成される。
4. **JSP の基本**
    - JSP は、Java Server Pages の略で、動的 Web アプリケーションを構築するためのサーバサイドの技術。HTML の中に断片的な Java プログラムを埋め込む簡易なサーバサイド処理技術。
5. **ディレクティブ**
    - JSP のディレクティブは、JSP ページの処理方式をコンテナに伝える。代表的なディレクティブには`@page`, `@include`, `@taglib`がある。
6. **スクリプトレット**

- JSP のスクリプトレットは、HTML に埋め込む小さなプログラムである。`<% ... %>`の形式で記述し、命令文はセミコロン`;`で終わる。

第 09 回

宣言部

1. **宣言部**
   - 宣言部は、Java のメソッドや変数を宣言するために使用される。書式は`<%! ... %>`である。

```
<%!
    private String greet() {
        return "Hello, World!";
    }
%>
```

式 (Expression)

2. **式**
   - 式は、Java の値を評価し、その結果を HTML ページに直接出力するために使用される。書式は`<%= ... %>`である。

```
<html>
<body>
    <h1>現在の日時: <%= new java.util.Date() %></h1>
</body>
</html>
```

アクションタグ

3. **アクションタグ**
   - アクションタグは、JavaBeans の操作、他のリソースのインクルード、転送などのアクションを実行するために使用される。主なアクションタグには`<jsp:useBean>`, `<jsp:setProperty>`, `<jsp:getProperty>`, `<jsp:include>`, `<jsp:forward>`がある。

Cookie

4. **Cookie**
   - Cookie は、ユーザーのブラウザに保存される小さなデータの断片で、ユーザーの状態や設定を保存するために使用される。
   - **Cookie の設定方法**

```
<%
    Cookie cookie = new Cookie("username", "JohnDoe");
    cookie.setMaxAge(60*60*24); // 1日間有効
    response.addCookie(cookie);
%>
```

- **Cookie の読み取り方法**

```
<%
    Cookie[] cookies = request.getCookies();
    if (cookies != null) {
        for (Cookie cookie : cookies) {
            if (cookie.getName().equals("username")) {
                String username = cookie.getValue();
                out.println("Welcome back, " + username);
            }
        }
    }
%>
```

第 10 回

JDBC

1. **JDBC の概要**
   - JDBC は、Java アプリケーションからデータベースに接続して操作を行うための API である。1996 年に Java 1.1 で導入され、現在は多くのデータベースでサポートされている。
2. **主要なクラス**
   - `DriverManager`: データベースへの接続を管理するクラス
   - `Connection`: データベース接続を表すクラス
   - `Statement`: SQL クエリを実行するためのクラス
   - `ResultSet`: SQL クエリの結果を格納するクラス
3. **使用例**
   - JDBC を使用してデータベースに接続し、データを取得する方法。

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class JDBCExample {
    public static void main(String[] args) {
```

```
        String url = "jdbc:mysql://localhost:3306/mydatabase";
        String user = "root";
        String password = "password";

        try {
            Connection connection =
DriverManager.getConnection(url, user, password);
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery("SELECT
* FROM users");

            while (resultSet.next()) {
                System.out.println("User ID: " +
resultSet.getInt("id"));
                System.out.println("Username: " +
resultSet.getString("username"));
            }

            resultSet.close();
            statement.close();
            connection.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

第 11 回

サーブレット

1. **サーブレットと JSP**
    - サーブレットはビジネスロジックを処理し、JSP はプレゼンテーションロジックを処理する。両者は連携して使用され、サーブレットが処理した結果を JSP で表示する。
2. **アプリケーションクラス**
    - アプリケーションクラスは、サーブレットのライフサイクル全体で共有されるデータや設定を管理するクラスである。
3. **メソッドハンドラ**
    - メソッドハンドラは、HTTP リクエストのメソッド (GET, POST など) に応じて適切な処理を行うメソッドである。
4. **インスタンス変数**
    - サーブレット内のインスタンス変数は、サーブレットの全インスタンス間で共有されるため、スレッドセーフである必要がある。

5. **JSP 連携**
    - サーブレットと JSP の連携方法。

```
// サーブレット内のコード例
request.setAttribute("user", user);
RequestDispatcher dispatcher =
request.getRequestDispatcher("user.jsp");
dispatcher.forward(request, response);
```

6. **JavaBeans**
    - JavaBeans は再利用可能なコンポーネントで、プロパティ、メソッド、イベントを持つ。JSP から JavaBeans を利用することで、データの受け渡しや処理が容易になる。

第 12 回

JavaBeans と MVC モデル

1. **JavaBeans のスコープ**
    - JavaBeans のスコープの種類とそれぞれの有効範囲。

スコープ	説明	有効範囲
`page`	同一ページ内	JSP ページ内
`request`	同一リクエスト内	サーブレットと JSP 間
`session`	同一セッション内	ユーザーセッション間
`application`	アプリケーション全体	全てのユーザーとセッション間

2. **JavaBeans の受け渡し**
    - サーブレットから JSP へ JavaBeans を受け渡す方法。

```
<jsp:useBean id="user" class="com.example.User" scope="request"
/>
<jsp:getProperty name="user" property="username" />
```

3. **MVC モデル**
    - MVC モデルとは、Model-View-Controller の略で、アプリケーションの構造を整理するためのデザインパターン。Model がデータとビジネスロジックの管理、View がユーザーインターフェースの表示、Controller がリクエストの処理とレスポンスの生成を行う。